

FAT Forensics: A Python Toolbox for Algorithmic Fairness, Accountability and Transparency

Kacper Sokol
K.Sokol@bristol.ac.uk
Department of Computer Science,
University of Bristol
Bristol, United Kingdom

Raul Santos-Rodriguez
enrsr@bristol.ac.uk
Department of Engineering
Mathematics, University of Bristol
Bristol, United Kingdom

Peter Flach
Peter.Flach@bristol.ac.uk
Department of Computer Science,
University of Bristol
Bristol, United Kingdom

ABSTRACT

Machine learning algorithms can take important, and sometimes legally binding, decisions about our everyday life. In many cases, however, these systems and their actions are neither regulated nor certified. Given the potential harm that such algorithms can cause, their fairness, accountability and transparency are of paramount importance. Recent literature suggested voluntary self-reporting on these aspects of predictive systems – e.g., “datasheets for datasets” – but their scope is often limited to a single component of a machine learning pipeline and their composition requires manual labour. To resolve this impasse and ensure high-quality, fair, transparent and reliable data-driven models, we developed an open source toolbox that can analyse selected fairness, accountability and transparency characteristics of these systems to automatically and objectively report them to relevant stakeholders. The software provides functionality for inspecting the aforementioned properties of all aspects of the machine learning process: data (and their features), models and predictions. In this paper we describe the design, scope and usage examples of this Python package, which is published under the BSD 3-Clause open source licence.

KEYWORDS

Fairness, Accountability, Transparency, Python, Software, Toolbox

1 INTRODUCTION

Open source software is the backbone of reproducible research, especially so for Artificial Intelligence (AI) and Machine Learning (ML) algorithms for which changing the seed of a random number generator may cause a state-of-the-art solution to become a sub-par predictor. Despite numerous efforts to ensure that publications are accompanied by code, both fields struggle with a reproducibility crisis [18]. This may be due to poor reporting, a requirement to keep trade secrets or simply a desire to gain an edge over competitors. One way to tackle this problem is to promote publishing high-quality software used for scientific experiments under an open source licence or enforce it as a part of the publishing and peer-review process, which has been advocated for a long time [38]. Despite their importance, implementations are nonetheless commonly treated just as a research by-product and are often abandoned after publishing the findings based upon them. We call this phenomenon *paperware* – a piece of code whose main purpose is to see a

paper towards publication rather than implement any particular concept with thorough software engineering practice. Such an 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.

Some researchers have recognised the widespread reliability issues with machine learning systems and proposed a range of unified frameworks to assess and document them. For example, multiple scholars have suggested approaches to characterise data sets [13, 17] to ensure their high quality and reliability. A similar strategy has been used with machine learning systems that are provided as services [4] accessed through an Application Programming Interface (API). Such efforts are laudable, however they suffer from limited scope and a labour-intensive creation process, which may slow down the ML research and development cycle. Furthermore, self-reporting – and lack of external audits – means that some of their aspects may be subjective, hence misrepresent the true behaviour of the investigated system, whether done intentionally or not. Certification, on the other hand, creates a need for external bodies, which seems difficult to achieve for all the machine learning systems that somehow affect humans.

To help address such shortcomings in the fields of AI and ML Fairness, Accountability and Transparency (FAT), we developed an open source Python package called FAT Forensics [36]. It is designed as an interoperable framework to *implement, test and deploy* novel algorithms proposed by the FAT research community as well as facilitate evaluation and comparison of such approaches against state-of-the-art methods, therefore democratising access to these techniques. To fully support research in this space, the toolbox is capable of analysing all facets of the machine learning process – data, models and predictions – by tackling their fairness, accountability (robustness, safety, security and privacy) and transparency (interpretability and explainability). The common interface layer of the framework (see Section 2.2 for more details) enables several *modes of operation*. A *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 prototyping and exploratory analysis. This mode is intended for FAT researchers who could use it to propose new fairness metrics, compare them with the existing ones or use them to inspect a new predictive system or data set. A *deployment mode*, characterised by “data in – data out”, on the other hand, offers to incorporate the package into a data processing pipeline to provide a (numerical)

The source code of the FAT Forensics [36] package is hosted on GitHub and available to download at <https://github.com/fat-forensics/fat-forensics>, with its documentation accessible at <https://fat-forensics.org>.

FAT analytics, hence support any kind of automated reporting or dashboarding. This mode is intended for ML engineers who may use it to monitor or evaluate a data-driven system during development and deployment.

Our main contribution is the design and implementation of a software package – FAT Forensics – that collates fairness, accountability and transparency algorithms for the entire predictive pipeline: data (raw and their features), models and predictions. The toolbox is supported by a thorough and beginner-friendly documentation, which spans tutorials, examples, how-to guides and a user guide. The framework is flexible enough to support workflows typical of academics and practitioners alike since it has been designed with research and deployment modes in mind. We look forward to our package being adopted by the FAT community, who will contribute their approaches here instead of releasing them as standalone code given the firm foundations of FAT Forensics. Another, minor, contribution of our work is a modular implementation of *local surrogate explanations* [37], discussed in Section 3.3, which shows the breath of transparency algorithms that can be built with our toolbox by simply combining some of the components that it implements.

In the following section we introduce our software, describe its architecture and list algorithms implemented in its latest release (version 0.1.0). Next, in Section 3, we describe a number of possible use cases and benefits of having various FAT algorithms under a common roof. Then, in Section 4, we survey the landscape of relevant reporting tools as well as commercial and freely available software that can be used to assess security, privacy, fairness, interpretability and explainability of data processing pipelines, namely: (raw) data, their features, predictive models and algorithmic decisions. In the final section we conclude the paper with a discussion and the envisaged long-term benefits of FAT Forensics.

2 INSPECTING FAT OF PREDICTIVE SYSTEMS

Depending on the maturity of a research field, proposing, adopting and developing a common software infrastructure for implementing novel algorithms and comparing them with pre-existing approaches may be difficult. Relatively young and still developing fields, such as algorithmic fairness, accountability and transparency, usually lack this type of software solutions. Within the past few years we have seen an increasing number of novel FAT algorithms implemented in different programming languages, each one with distinct requirements and API, making them difficult to use and compare in a systematic fashion.

2.1 FAT Forensics

To address these inconsistencies – while the community is still young and flexible enough to adopt it – we developed an open source Python framework for evaluating, comparing and deploying FAT algorithms. We chose Python – compatible with Python 3.5 and higher – because of its prevalence among AI and ML research communities and its overall simplicity. We opted for a minimal (required) dependency on NumPy (1.10.0 or higher) and SciPy (0.13.3 or higher) to facilitate easy deployment in a variety of settings. An optional dependency on Matplotlib (3.0.0 or higher) and scikit-learn

(0.19.2 or higher) enables access to basic visualisations and ML algorithms. The toolbox is hosted on GitHub¹ to encourage community contributions and it is released under the 3-clause BSD License, which opens 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 & integration testing, maintaining high code coverage;
- continuous integration;
- detailed technical API specification that includes function-level, module-level and functional documentation;
- tutorials that walk the users step-by-step through the main functionality of the package; and
- code examples that can be used as a reference material for more advanced users.

The toolbox implements a number of popular fairness, accountability and transparency algorithm – with many more to come – under a coherent API, reusing many functional components across FAT tools and making them readily accessible to the community. For example, grouping data based on values of a selected feature can be used for both: evaluating group-based fairness (i.e., disparate impact) such as *demographic parity* [16], and uncovering *systematic performance bias* of a predictive model. The initial development is focused heavily on tabular data and well-established predictive algorithms (scikit-learn [27]). Once a certain level of maturity is reached, the development will move towards techniques capable of handling sensory data (images, audio) and neural networks (TensorFlow [1], PyTorch [26]). 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 avoid re-implementing them. We will also encourage researchers and practitioners to contribute their novel approaches to our software package or make them compatible with it, therefore improving the overall reproducibility of FAT algorithms by exposing them to the wider community in a robust and sustainable environment.

We identify two main application areas of our toolbox. The first one is directed toward FAT research communities in AI and ML. We provide them with a platform to develop, test, compare and evaluate their novel algorithms without the burden of setting up a software engineering workflow (see Section 3 for an example of this application). This in turn will ensure that our framework contains (or is compatible with) implementations of cutting-edge algorithms, encouraging its use for monitoring and auditing of data-driven systems – the second intended application domain. Our package should appeal to the latter user group since its members can access a low-level API that may be used for FAT reporting and certification (see Section 3 for examples that could contribute to these tasks). Both of these application areas give ML researchers and practitioners a tool to inspect the quality, accountability and robustness of their systems in a transparent and reproducible manner.

2.2 Design and Architecture

A considerable portion of the FAT software is developed with the intention to support research outputs. This approach often results in unnecessary dependencies, data sets and (interactive) visualisations

¹<https://github.com/fat-forensics/fat-forensics>

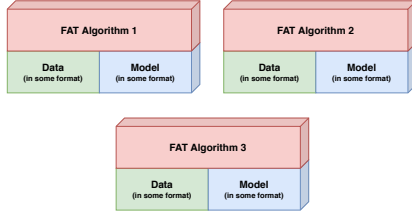


Figure 1: Typical landscape of academic software consists of standalone code bases without standardised interfaces.

being distributed with the code base, which itself uses non-standard API – see Figure 1. To mitigate these issues, FAT Forensics decouples the core functionality of an FAT algorithm from its possible presentation to the user (e.g., visualisations) and dependencies that are pertinent only to experiments (e.g., data sets and predictive algorithms) – see Figure 2. Since visualisations are a vital part of the first application mode that we advocate (research), we provide a basic plotting module within the package, however its functionality is conditioned on an *optional* Matplotlib software dependency. This abstraction of the FAT software infrastructure is achieved by making minimal assumptions about the operational settings of these algorithms, therefore providing a common interface layer for key FAT functionality, focusing only on the interactions between data, models, predictions and people.

Similarly, a predictive model is assumed to be a plain Python object that has `fit`, `predict` and, optionally, `predict_proba` methods, therefore making it compatible with scikit-learn [27] – the most popular Python machine learning toolbox – without introducing a hard dependency on it. Additionally, this approach enables our package to easily support an arbitrary black-box predictive model, e.g., TensorFlow, PyTorch or even one hosted on the Internet and accessible via a web API, by representing it as a Python object equipped with the aforementioned methods. Furthermore, model-specific transparency (as well as fairness and accountability) techniques for glass-box predictive algorithms (decision trees, linear models, etc.) implemented by standard machine learning libraries will be incorporated into the package over time to improve its versatility. Finally, a data set is assumed to be a two-dimensional NumPy array: either a classic or a structured array, with the latter being a welcome addition given that some of the features may be categorical (string-based).

In addition to relaxed input requirements, all of the techniques incorporated into the package are decomposed into atomic blocks that can be easily reused to create new functionality. The FAT methods implemented in the latest release of the package – version 0.1.0 – are shown in Table 1. To ground the idea of atomic-level decomposition and show the reusability of such building blocks, even across FAT borders, we provide three examples.

Fairness. All of the: sample-size disparity, sub-population fairness (e.g., group unaware, equal opportunity, equal accuracy, demographic parity [16]), sub-population predictive performance disparity and summary statistics can be based on a function that partitions a data set with respect to unique values of a chosen feature – a functionality implemented as one of the core components of the

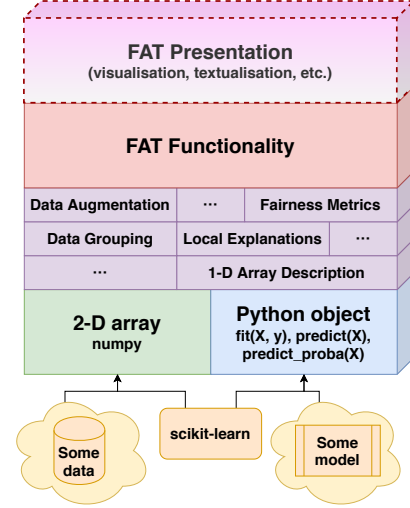


Figure 2: Modular architecture of FAT Forensics. The input requirements – data sets and predictive models – are lax. The FAT tools are built from blocks of atomic functionality, therefore making the process of creating new algorithm as easy as connecting the right blocks.

package. This grouping can be then coupled with any standard (predictive) performance metric to assess group-based fairness. With the addition of a module that fits a threshold for different groups (given data points ranking), a variety of different fairness criteria, not limited to the ones implemented in the package itself, can be derived. To this end, the user just needs to provide a function that measures some notion of performance based only on predicted and true labels. Additionally, the grouping functionality can help the user to evaluate the predictive performance, number of data points and feature distribution across different (possibly underrepresented and adversely affected) sub-populations.

Accountability. Estimating density of a region in which a data point of interest is located (based on the distribution of training data) can provide important clues about the robustness of its prediction. To this end, a density score can be treated as a proxy measurement of the confidence of a predictive model [28]. In addition to engendering trust in ML predictions, a density estimate can help to compute realistic counterfactual explanations [29]. While optimising for counterfactuals, a scoring function can discount the ones sourced from sparse regions since such data points will usually be impossible to realise in the real life; for example, consider a person who is 200 years old.

Transparency. A counterfactual explainer can be used to generate an explicit (of a selected class) or implicit (of any class other than the one of the given instance) hypothetical instance, i.e., what-if scenario. By restricting the set of features that a counterfactual can be conditioned on – e.g., by choosing only the protected attributes² – such an explanation can be used to assess individual fairness through *disparate treatment*. Another possible use case is

²Data features that can be used for discriminatory treatment, e.g., gender.

	Fairness	Accountability	Transparency
Data & Features	<ul style="list-style-type: none"> • Systemic Bias. • Sub-population Representation. 	<ul style="list-style-type: none"> • Sampling Bias. • Data Density Checker. 	<ul style="list-style-type: none"> • Data Description. • Summary Statistics.
Models	<ul style="list-style-type: none"> • Group-based Fairness. 	<ul style="list-style-type: none"> • Group-based Performance Metrics. • Systematic Performance Bias. 	<ul style="list-style-type: none"> • Global Surrogates (bLIMEy). • Partial Dependence.
Predictions	<ul style="list-style-type: none"> • Counterfactual Fairness. 	<ul style="list-style-type: none"> • Prediction Confidence. 	<ul style="list-style-type: none"> • Model-agnostic Counterfactuals. • Local Surrogates (bLIMEy). • LIME (bLIMEy implementation). • Individual Conditional Expectation.

Table 1: FAT functionality implemented in the latest release – version 0.1.0 – of FAT Forensics.

discovering feature variations that do not affect the prediction of a given data point (akin to the anchors explainer [32]).

Surrogate explainers [9, 31, 37] also exhibit a high level of modularity. bLIMEy [37] – a flexible meta-algorithm for building bespoke surrogate explainers – consists of the following atomic components, all of which are a part of the FAT Forensics package³: (i) interpretable representation composition; (ii) data sampling; and (iii) explanation generation. For example, an interpretable representation of tabular data can be constructed with quartile-based discretisation or a feature space partition extracted from a decision tree; data can be augmented with Gaussian or Mixup [42] sampling (the latter guarantees a diverse and local sample); and an explanation can be generated with a decision tree (e.g., a root-to-leaf path) or a linear model (e.g., feature influence). 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.

2.3 Discussion

Sharing a common functional base between algorithmic implementations of fairness, accountability and transparency tools is one of many advantages of a combined FAT software package. This versatility of the toolbox makes it appealing to academics and industrial researchers since it allows them to investigate all societal aspects of an entire predictive pipeline: data, models and predictions. This in turn ought to encourage them to contribute their own algorithms and bug fixes to the package considering their best interest. Furthermore, having a software framework whose ownership is outside of a single lab, company or research group ensures its longevity – the contributors are not limited to the package creators and designers of the algorithms implemented therein – and the tools are not biased towards implementations originating from a single group. With all of that in mind, a development of such software becomes a community effort driving it towards a common goal.

Since contributions to the package will go through a community review process before being incorporated into the toolkit, we can easily avoid common pitfalls – such as subpar code quality and unnecessary dependencies – that academic software is particularly vulnerable to. This approach will also help to gear the package

towards real-world use cases, as opposed to just enabling reproducibility claims when publishing research. Having said that, we do not aim to simply “wrap” all of the relevant packages under a common API. If at all, we will only do so for good quality code to avoid perpetuating issues of these packages. Microsoft’s Interpret [25] and Oracle’s Skater [21], for example, “wrap” a number of explainability packages, risking users’ trust as they are prone to errors introduced therein. LIME [31], which is part of both these libraries, has recently been shown to have issues with locality and coherence of its explanations [22, 37], which affects both Interpret and Skater. In a long term, therefore, we want to re-implement popular algorithms from the grounds up, which should be possible given the common functional base of the package. In doing so we will be able to enforce high-quality code that is easy to manage and maintain since it is fully under the control of package contributors.

The major development challenges of FAT Forensics were designing the architecture – package structure, function versatility, testing, error legibility and input validation – and preparing the documentation – code examples, tutorials, API specification and how-to guides. Usually, the main barrier, especially for a lay audience, to understand the functionality of and adopt a software package is the lack of appropriate training materials. Many tools in the FAT space are supported by just two types of documentation: a technical *API reference*, which is only suitable for (proficient) users who are already familiar with the package and its structure; and *code examples*, often delivered as Jupyter Notebooks, which immerse new users into elaborate use cases instead of gradually easing them into the code functionality, therefore discouraging further exploration of the package. These approaches are the most popular since they usually do not require additional effort: the former can be generated automatically from the source code and the latter tends to be an artefact of research experiments, however neither of them is designed with an end user in mind. FAT Forensics overcomes these issues and evens out the learning curve by basing its documentation⁴ on four main pillars, which together build up the user’s confidence in working with the package:

- narrative-driven *tutorials* are designated for new users, guiding them step by step through practical use cases of all the main aspects of the package;

³https://fat-forensics.org/how_to/transparency/tabular-surrogates.html

⁴<https://fat-forensics.org>

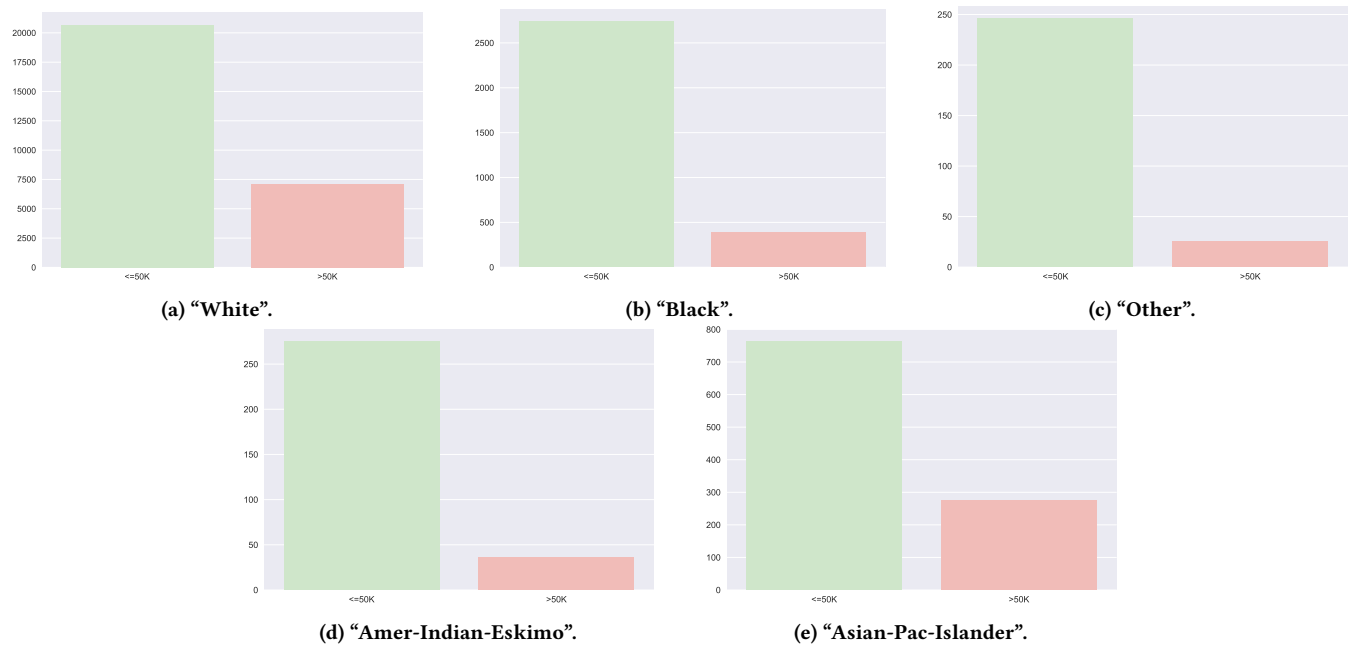


Figure 3: Income distribution for each unique value of the “race” feature in the Adult data set.

- *how-to guides* are created for relatively new users of the package, showcasing the flexibility of the software and demonstrating how to use it to solve particular FAT challenges, e.g., how to build a bespoke local surrogate explainer by pairing a data generator, an interpretable representation and a transparent surrogate model;
- *API documentation* describes functional aspects of the algorithms implemented in the package and is aimed at a technical audience, serving as a reference material complemented by task-focused *code examples* that place the code base in a context; and
- a *user guide* discusses theoretical aspects of the algorithms implemented in the package, e.g., their restrictions, caveats as well as computational time and memory complexity.

3 FAT FORENSICS USE CASES

To demonstrate how FAT Forensics can be applied to analyse FAT aspects of real data and illustrate how the common infrastructure of the package facilitates its broad functionality we present three distinct use cases. To this end, we employ the UCI Census Income (Adult) data set⁵, which is commonly used in algorithmic fairness and transparency research. The data analysis presented in this section is inspired by the tutorials that are a vital part of the FAT Forensics documentation⁶. The results shown in this section can be reproduced with a dedicated Jupyter Notebook, which is hosted on GitHub⁷. All of the examples included below are representative of the FAT Forensics *research mode*. To demonstrate the

deployment mode we present an interactive dashboard built using Plotly’s Dash, which facilitates interactive analysis of the same data set using FAT Forensics as the back-end⁸.

3.1 Grouping for FAT

One of the basic building blocks of FAT Forensics is grouping data based on (sets of) unique values for categorical features and threshold-based binning for numerical attributes. This algorithmic concept proves to be useful for a number of fairness, accountability and transparency tools. Below, we present its three possible applications within the scope of FAT Forensics.

Data Transparency. When analysing a data set prior to any sort of modelling it is usually advised to inspect the ground truth distribution to uncover whether the target classes are balanced. While this in itself is a very important step of a data modelling pipeline, asking the same question for each (protected⁹) group can help to prevent modelling biases and systematic under-performance. With FAT Forensics it is easy to inspect the class distribution for each protected sub-population, for example, the “race” attribute in the Adult data set – see Figure 3. The plots show that while the classes are imbalanced for all of the partitions, the strongest disproportion can be observed for the “Other”, “Amer-Indian-Eskimo” and “Black” sub-populations.

Model Fairness. Grouping can also be used to compute (pairwise) group-based fairness metrics to investigate disparate impact of a model [16]. Since some of these metrics are known to be mutually incompatible [24], it is usually a good idea to compare them side by side – see Figure 4. Based on these plots we can easily see that

⁵<http://archive.ics.uci.edu/ml/datasets/Census+Income>

⁶<https://fat-forensics.org/tutorials/index.html>

⁷https://github.com/fat-forensics/resources/blob/master/fat_forensics_overview/FAT_Forensics.ipynb

⁸<https://fatf.herokuapp.com>. Please allow 15–30 seconds for the dashboard to load.

⁹A sub-population derived by conditioning on a protected feature.

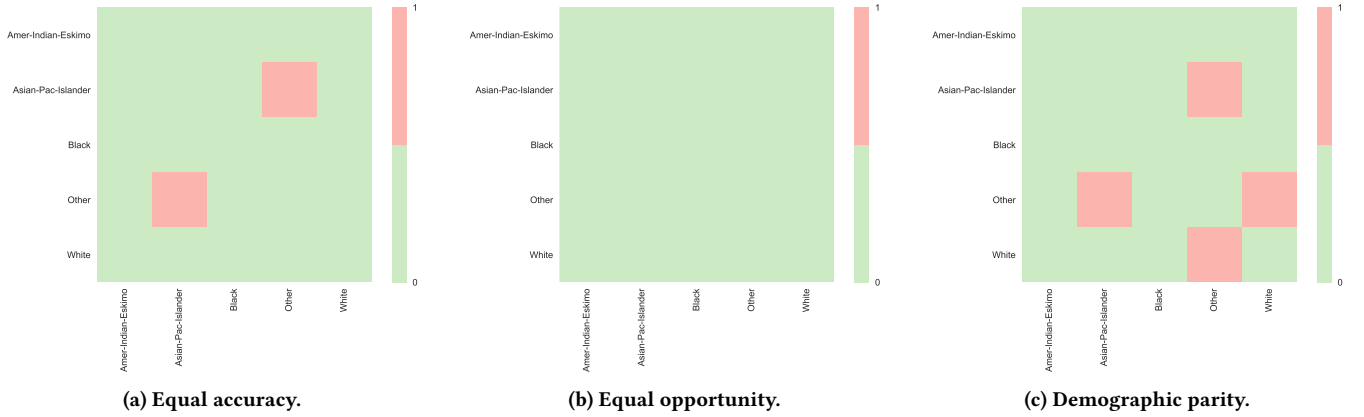


Figure 4: Pairwise group-based fairness for unique values of the “race” feature in the Adult data set. Red (1) indicates disparate impact for a given pair of sub-populations and green (0) shows that both groups are treated comparably.

there is a fairness disparity between the “Asian-Pac-Islander” and “Other” groups when *equal accuracy* and *demographic parity* are considered. Additionally, the “Other” and “White” sub-populations are also treated unfairly with respect to each other according to the *demographic parity* metric. Notably, the *equal opportunity* metric does not show any signs of disparate impact for any pair of the protected sub-populations.

Model Performance Disparity. Grouping can also be used to inspect systematic bias of a predictive model, i.e., check whether it under-performs for any selected sub-population. For this experiment we will also use the “race” feature, measuring predictive performance with *accuracy* and *true negative rate* – Figure 5. Unsurprisingly, the first metric (Figure 5a) yields the same result as group-based fairness analysis under *equal accuracy* (Figure 4a). Using the *true negative rate*, on the other hand, reveals that four different pairs of sub-populations composed based on the “race” feature exhibit significant performance differences, with the “Other” group suffering from the worst pairwise performance disparity, spanning all the other sub-populations but “Amer-Indian-Eskimo”.

3.2 Data Density for FAT Robustness

FAT Forensics can also help to investigate robustness of a predictive model and assess quality of exemplar explanations. The latter applies, for example, to counterfactuals, both when they are used as a transparency and fairness technique for an individual data point.

Prediction Robustness. FAT Forensics comes with a bespoke neighbour-based density estimator. This algorithm can be used to validate robustness of a prediction as dense data regions (with respect to the corresponding training set) should enable accurate predictive modelling in these spaces. To see how this could be used, we estimate the density of Adult based on its first 1,000 instances and select four data points – two from a sparse and two from a dense region – to assess robustness of their predictions. Note that the density scores computed with this method are between 0 and 1, where high values characterise instances that lie in a sparse region since their n^{th} neighbour (a parameter set by the user) is relatively distant. The two data points sourced from dense regions

have density scores of 0; both are of the “ $\leq 50K$ ” class and are predicted correctly. The two instances selected from sparse regions are assigned density scores of 1; one is of the “ $\leq 50K$ ” class and is predicted correctly, however the other is of the “ $> 50K$ ” class and is misclassified by our model. Upon closer inspection we notice that this data point has a relatively high value of the “*fnlwgt*” feature (1,226,583), which is in the 99.99th percentile of the data set – a clue to its high density score and incorrect prediction.

Feasibility of Counterfactual Explanations. A similar approach can be taken to assess trustworthiness of exemplar explanations. A high density score (with respect to the training data) of a counterfactual instance, which serves as an explanation, may indicate that such a data point is unlikely to occur or outright impossible in the real life. For example, imagine a counterfactual explanation where the foil states that the age of a person would have to be 142 to achieve the desired prediction. Explaining the aforementioned misclassified data point (with a high density score / residing in a sparse region) yielded multiple counterfactual instances such as:

- had you had “*capital-gain*” of 25,000 instead of 0, you would have been predicted as “ $> 50K$ ” (density score of 1 / sparse region); and
- had you had “*capital-loss*” of 4,000 instead of 0 and “*fnlwgt*” of 430,985 instead of 1,226,583, you would have been predicted as “ $> 50K$ ” (density score of 0.02 / dense region).

Clearly, having 25,000 of “*capital-gain*” could be a property of a high-income person, nonetheless preserving the unusually high value of the “*fnlwgt*” feature makes this explanation unlikely, i.e., this exemplar resides in a sparse data region. The second explanation, on the other hand, decreases the value of the “*fnlwgt*” attribute – therefore moving the exemplar to a dense region – and also shows that even with 4,000 of “*capital-loss*” this person would still be classified as a high-income individual, casting even more suspicion on the unusual original value of the former feature.

Counterfactual Fairness. Counterfactual explanations can also be used to assess individual fairness by forcing their foils to include at least one protected attribute. Doing so for the same data point shows us that its prediction is fair as our counterfactual explainer

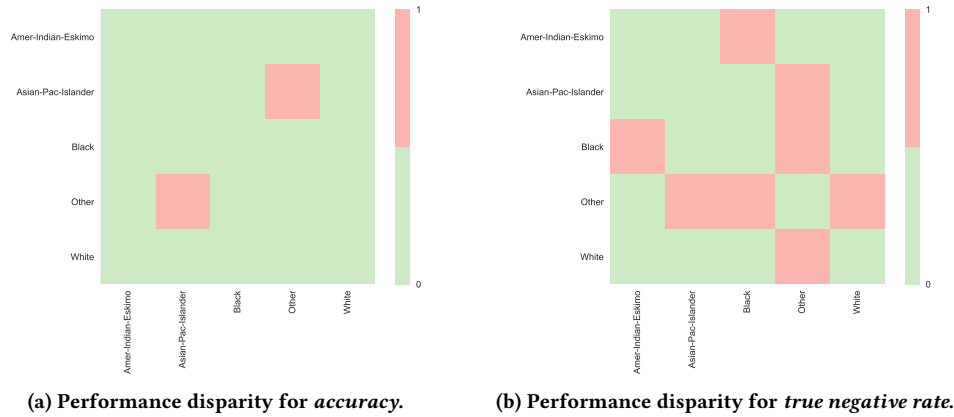


Figure 5: Pairwise group-based performance disparity for unique values of the “race” feature in the Adult data set. Red (1) indicates disparate performance for a given pair of sub-populations and green (0) shows that the model performs comparably for both groups.

could not find any explanation conditioned on at least one of the protected features.

3.3 Modular Surrogate Explainers

Local surrogate explainers are a popular technique for improving interpretability of black-box predictions [31]. These tools construct a transparent model in the neighbourhood of the explained instance to approximate and explain the predictive behaviour of the underlying black box in this region. Given the high modularity of surrogate explainers [37], FAT Forensics implements their core building blocks and allows the user to compose a bespoke explainer from these components. Depending on the use case, one type of a surrogate model may have an advantage over another; therefore, supporting their straightforward construction process enables considerable improves to the quality and faithfulness of their explanations [34, 35]. For example, consider the two local surrogates depicted in Figure 6, where a tree-based explainer is better able to approximate the decision boundary close to the selected instance.

Linear Surrogate. Figure 6a shows a surrogate explanation of the marked data point (the black dot) based on a linear model. Even though the decision boundary can be easily approximated with a linear model – an almost vertical line crossing the x-axis around 0.25 – the local decision boundary is tilted because of the data distribution. This issue can be partially overcome by weighting the sampled data based on their proximity to the explained instance, however finding a robust approach to generate these weights is an open research question [37].

Tree-based Surrogate. A better approximation of the local decision boundary can be generated with a tree-based surrogate [34, 35] – an approach enabled by FAT Forensics’ modular implementation of bLIMEy [37]. Figure 6b shows this improvement; note that the local model aligns well with the gap between the two clouds of points close to the explained instance, hence offers a faithful approximation of the underlying black box in this neighbourhood. Furthermore, tree-based explainers deliver a range of insights such

as logical conditions describing a local approximation of the black-box decision boundary – an improvement over linear surrogates, which can only provide feature influence explanations [34]. Additionally, the structure of a surrogate tree can be used to compose a faithful local interpretable representation [35].

4 RELATED WORK

Systematic evaluation and comparison of artificial intelligence and machine learning techniques is an active area of research across many different communities. In this section we review these efforts as well as discuss software packages and reporting approaches designed to assess fairness, accountability and transparency of AI and ML systems.

4.1 Predictive Algorithms

In well-established research communities, e.g., supervised or reinforcement learning, we can observe convergence towards commonly accepted performance metrics and evaluation software. For predictive performance of supervised learning algorithms these can be, for example, accuracy, F1 score or AUC, which are fundamental components of any such software package – see scikit-learn’s `sklearn.metrics`¹⁰ [8] and TensorFlow’s `tf.metrics`¹¹ [1] modules. Given the independence of these metrics from the implementation of the underlying predictive algorithm, some software – e.g., *PyCM*¹² [15] – is focused exclusively on calculating them. In other research communities, e.g., reinforcement learning, we can also observe common software platforms used to systematically evaluate novel approaches, making relevant research results easier to reproduce and compare. Examples of these are Project Malmö¹³ [19] and OpenAI Gym¹⁴ [7] (which includes the MuJoCo environment [39]). Alternatively, projects such as *cookiecutter*¹⁵ allow researchers to

¹⁰<https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics>

¹¹https://www.tensorflow.org/api_docs/python/tf/metrics

¹²<https://github.com/sebandhaghighi/pycm>

¹³<https://www.microsoft.com/en-us/research/project/project-malmo>

¹⁴<https://gym.openai.com>

¹⁵<https://github.com/audreyt/cookiecutter>

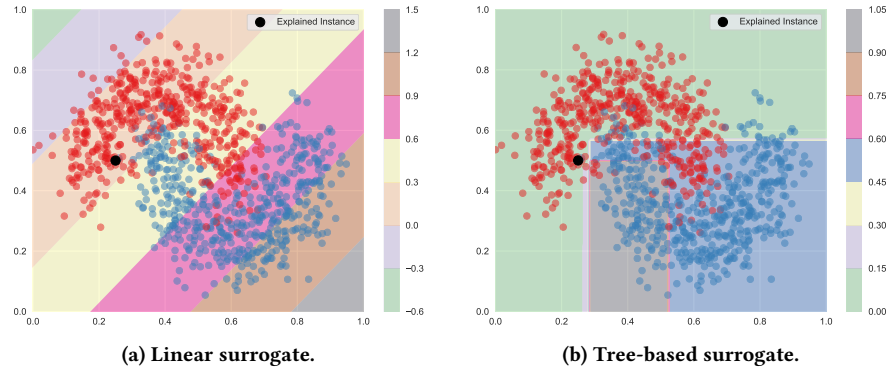


Figure 6: Linear (LIME-equivalent) – Panel (a) – and tree-based – Panel (b) – local surrogate explainer without an interpretable representation built with the bLIMEy meta-algorithms for the Two Moons toy data set. Using a simple two-dimensional data set and forgoing an interpretable representation enables direct visualisation of the surrogate models.

create software packages that adhere to a common structure, hence make them easier to adopt and integrate. With the availability of all these packages, it is clear that a common software platform for inspecting FAT aspects of AI and ML systems (data sets, models and predictions) would be a welcome addition.

4.2 FAT Software

Many academics, companies and developers joined the forces to make AI systems more transparent and socially acceptable¹⁶, however the landscape of FAT research software is relatively incoherent when compared to mature fields such as supervised learning. A recent attempt to create a common framework for FAT algorithms is the What-If tool¹⁷, which implements group-based model fairness evaluation and counterfactual prediction explainability (transparency). While its goal is similar to FAT Forensics, the What-If tool is intended primarily for visual analytics. A different tool originating from the TensorFlow community is *TensorFlow Extended*¹⁸ – a platform that facilitates analysis of TensorFlow models by measuring, for example, their predictive performance for multiple sub-populations identified in a data set.

In addition to general frameworks such as the What-If tool, we can also find implementations of individual interpretability and explainability algorithms published in the literature. Examples of these are: LIME¹⁹ [31], Anchor²⁰ [32], shap²¹ [23] and PyCE-box²² [14]. Many of these have been collected and built into algorithmic *transparency* packages with the most prominent ones being:

- Skater²³ [21];
- eli5²⁴;

- Microsoft’s Interpret²⁵ [25];
- Yellowbrick²⁶ [6]; and
- AI Explainability 360²⁷.

The open source software landscape for AI and ML *fairness* is even more irregular. Relevant packages use different programming languages, often lack a licence or documentation and vary in code quality. The most important ones are:

- AI Fairness 360²⁸ [5];
- BlackBoxAuditing²⁹ [2, 11];
- fairness-comparison³⁰ [12];
- fairtest³¹ [40];
- fairml³²; and
- fairlearn³³ [3].

Finally, open source software for AI and ML *accountability* (security and privacy) is the most scarce. Packages prominent in this space are: TensorFlow Privacy³⁴, OpenMined’s Grid³⁵ (a part of PyTorch [26]) and DeepGame³⁶ (a deep neural network verification tool). A research and software development direction complementary to accountability is *robustness* of predictive systems against adversarial attacks. Packages available in this space are: FoolBox³⁷, CleverHans³⁸ and IBM’s adversarial robustness toolbox³⁹.

Despite this lack of coherence in the open source world, relevant commercial products start to emerge in this space. The most prominent one is IBM’s cloud offering called Watson OpenScale⁴⁰, which allows the users to measure disparate impact within data and for

¹⁶For example, TuringBox [10] – <https://turingbox.mit.edu> – which is an online platform to automatically benchmark and evaluate AI systems with respect to chosen metrics, e.g., accuracy and fairness.

¹⁷<https://pair-code.github.io/what-if-tool>

¹⁸<https://github.com/tensorflow/model-analysis>

¹⁹<https://github.com/marcotcr/lime>

²⁰<https://github.com/marcotcr/anchor>

²¹<https://github.com/slundberg/shap>

²²<https://github.com/AustinRochford/PyCEbox>

²³<https://github.com/oracle/Skater>

²⁴<https://github.com/TeamHG-Memex/eli5>

²⁵<https://github.com/microsoft/interpret>

²⁶<https://github.com/DistrictDataLabs/yellowbrick>

²⁷<https://github.com/IBM/AIX360>

²⁸<https://github.com/IBM/AIF360>

²⁹<https://github.com/algofairness/BlackBoxAuditing>

³⁰<https://github.com/algofairness/fairness-comparison>

³¹<https://github.com/columbia/fairtest>

³²<https://github.com/adebayoj/fairml>

³³<https://github.com/Microsoft/fairlearn>

³⁴<https://github.com/tensorflow/privacy>

³⁵<https://github.com/OpenMined/Grid>

³⁶<https://github.com/TrustAI/DeepGame>

³⁷<https://github.com/bethgelab/foolbox>

³⁸<https://github.com/tensorflow/cleverhans>

³⁹<https://github.com/IBM/adversarial-robustness-toolbox>

⁴⁰<https://www.ibm.com/cloud/watson-openscale>

predictive models⁴¹ (group-based fairness of ground truth labelling and predicted classes respectively) as well as apply selected model and prediction transparency approaches, in addition to the standard functionality for monitoring predictive performance.

4.3 FAT Reporting

Aside from evaluating FAT aspects of predictive systems with software, some researchers are advocating unified reports describing their quality, reliability and other properties of interest. For example, Gebru et al. [13] proposed “datasheets for datasets” that aim to provide standardised information about technical properties of a data set, its intended use and provenance. Holland et al. [17] have independently come up with a similar idea called “dataset nutrition labels” that mimics well-known food nutrition labels by providing basic details about a data set. Similarly, Kelley et al. [20] proposed “privacy labels” that inform users about the ways in which their data are collected, used and shared.

Other components of AI and ML workflows have also been addressed in like manner. Arnold et al. [4] proposed a similar solution for AI services called “supplier’s declarations of conformity”. Their goal is to provide developers and suppliers of AI products-as-a-service with a consistent way of reporting quality, security, interpretability and fairness of their products. Having rigorous and easily comparable reports for such products is of paramount importance given that their use does not require any prior AI or ML knowledge. Reisman et al. [30] came up with a similar idea of “algorithmic impact assessments”, which is a framework for systematic evaluation of automated decision-making systems to keep them accountable. A related concept was introduced by Yang et al. [41] who created “nutritional labels” for rankings. Additionally, Sokol and Flach [33] proposed “explainability fact sheets” intended for coherent reporting of a wide range of properties expected of AI explainability algorithms. Alternative solutions include “checklists”⁴² for AI systems, “data ethics workbooks”⁴³ and test-driven data analysis⁴⁴.

While all of these initiatives aim to improve transparency and accountability of various components found in predictive systems, the time and effort required to produce many of them may be prohibitive on a large scale, therefore hindering their uptake. Moreover, some of these solutions require an in-depth understanding of the system or data being assessed, which often means that they cannot be retrofitted. The toolbox described in this paper could be used to automatically generate metrics and insights – spanning all aspects of a machine learning system (data, models and predictions) – for FAT reports such as the aforementioned “report cards” and “fact sheets”, thus partially alleviating the issues with their manual, error-prone and subjective creation process. Furthermore, FAT Forensics has the potential to become a vital component of any ML pipeline development process; where continuous integration is used in software engineering to ensure high quality of the code, our toolbox could be used to evaluate FAT of any component of an ML pipeline during its development and deployment.

⁴¹<https://www.ibm.com/blogs/watson/2018/09/trust-transparency-ai>

⁴²<https://www.oreilly.com/ideas/of-oaths-and-checklists>

⁴³<https://www.gov.uk/government/publications/data-ethics-workbook/data-ethics-workbook>

⁴⁴<http://www.tdda.info>

5 CONCLUSIONS

While software is the primary driver of progress in AI and ML research, its quality is often found lacking. Some research fields, such as supervised and reinforcement learning, have reached a consensus on that matter and agreed on standardised metrics and software frameworks used to evaluate and compare novel algorithms. At the moment, the algorithmic fairness, accountability and transparency community lacks such a common software infrastructure and evaluation criteria to analyse, compare and communicate research results in a coherent fashion. This paper offers a possible solution by proposing a flexible and modular open source Python toolbox to facilitate the development, evaluation, comparison and deployment of FAT algorithms.

FAT Forensics has been released to the public under the BSD 3-Clause licence with a collection of popular FAT algorithms. Our toolbox has been implemented with two use cases in mind: *research* – intended for exploratory analysis; and *deployment* – designed for report generation, monitoring and certification. Since FAT Forensics is an open source initiative we envisage the research community to incorporate their code outputs into this software package, therefore making their contributions easily accessible, reproducible and attractive to FAT enthusiasts. We hope and expect that all the software engineering best practice followed during the initial development of FAT Forensics have helped us to create a sustainable software package that is easy to extend and contribute to, serving the community for a long time to come.

ACKNOWLEDGMENTS

This work was financially supported by Thales, and is the result of a collaborative research agreement between Thales and the University of Bristol. We would also like to acknowledge hard work of our student software engineers: Alexander Hepburn, Rafael Poyiadzi and Matthew Clifford.

REFERENCES

- [1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. 265–283.
- [2] Philip Adler, Casey Falk, Sorelle A Friedler, Tionney Nix, Gabriel Rybeck, Carlos Scheidegger, Brandon Smith, and Suresh Venkatasubramanian. 2018. Auditing black-box models for indirect influence. *Knowledge and Information Systems* 54, 1 (2018), 95–122.
- [3] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. 2018. A Reductions Approach to Fair Classification. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, Stockholm, Sweden, 60–69. <http://proceedings.mlr.press/v80/agarwal18a.html>
- [4] Matthew Arnold, Rachel KE Bellamy, Michael Hind, Stephanie Houde, Sameep Mehta, Aleksandra Mojsilovic, Ravi Nair, Karthikeyan Natesan Ramamurthy, Alexandra Olteanu, David Piorkowski, Darrell Reimer, John Richards, Jason Tsay, and Kush R Varshney. 2019. FactSheets: Increasing trust in AI services through supplier’s declarations of conformity. *IBM Journal of Research and Development* 63, 4/5 (July 2019), 6:1–6:13. <https://doi.org/10.1147/JRD.2019.2942288>
- [5] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. 2018. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. (2018).

- arXiv:1810.01943
- [6] Benjamin Bengfort, Rebecca Bilbro, and Kristen McIntyre. 2019. Yellowbrick v1.0.1. <https://doi.org/10.5281/zenodo.3474252>
 - [7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. (2016). arXiv:1606.01540
 - [8] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: Experiences from the scikit-learn project. In *ECML PKDD Workshop on Languages for Data Mining and Machine Learning*. 108–122.
 - [9] Mark Craven and Jude W Shavlik. 1996. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*. 24–30.
 - [10] Ziv Epstein, Blakeley H Payne, Judy Hanwen Shen, Casey Jisoo Hong, Bjarke Felbo, Abhimanyu Dubey, Matthew Groh, Nick Obradovich, Manuel Cebrian, and Iyad Rahwan. 2018. TuringBox: An Experimental Platform for the Evaluation of AI Systems. In *IJCAI*. 5826–5828.
 - [11] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 259–268.
 - [12] Sorelle A. Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P. Hamilton, and Derek Roth. 2019. A Comparative Study of Fairness-enhancing Interventions in Machine Learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* '19)*. ACM, New York, NY, USA, 329–338. <https://doi.org/10.1145/3287560.3287589>
 - [13] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2018. Datasheets for Datasets. *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
 - [14] Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. 2015. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics* 24, 1 (2015), 44–65.
 - [15] Sepand Haghighi, Masoomeh Jasemi, Shaahin Hessabi, and Alireza Zolanvari. 2018. PyCM: Multiclass confusion matrix library in Python. *Journal of Open Source Software* 3, 25 (2018), 729. <https://doi.org/10.21105/joss.00729>
 - [16] Moritz Hardt, Eric Price, and Nathan Srebro. 2016. Equality of Opportunity in Supervised Learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*. Curran Associates Inc., USA, 3323–3331. <http://dl.acm.org/citation.cfm?id=3157382.3157469>
 - [17] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. 2018. The Dataset Nutrition Label: A Framework To Drive Higher Data Quality Standards. (2018). arXiv:1805.03677
 - [18] Matthew Hutson. 2018. Artificial intelligence faces reproducibility crisis. *Science* 359, 6377 (2018), 725–726. <https://doi.org/10.1126/science.359.6377.725>
 - [19] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. 2016. The Malmo Platform for Artificial Intelligence Experimentation. In *IJCAI*. 4246–4247.
 - [20] Patrick Gage Kelley, Joanna Bresee, Lorrie Faith Cranor, and Robert W Reeder. 2009. A nutrition label for privacy. In *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM, 4.
 - [21] Aaron Kramer, Prami Choudhary, silversurfer84, Ben Van Dyke, Alvin Thai, Nitin Pasumathy, Guillaume Lemaitre, Dave Thompson, and Ben Cook. 2018. datascienceinc/Skater: 1.1.2. <https://doi.org/10.5281/zenodo.1423046>
 - [22] Thibault Laugel, Xavier Renard, Marie-Jeanne Lesot, Christophe Marsala, and Marcin Detyniecki. 2018. Defining locality for surrogates in post-hoc interpretability. *3rd Workshop on Human Interpretability in Machine Learning (WHI 2018) at the 35th International Conference on Machine Learning (ICML 2018), Stockholm, Sweden* (2018). arXiv:1806.07498
 - [23] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4765–4774. <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>
 - [24] Thomas Miconi. 2017. The impossibility of “fairness”: A generalized impossibility result for decisions. (2017). arXiv:1707.01195
 - [25] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. 2019. InterpretML: A Unified Framework for Machine Learning Interpretability. (2019). arXiv:1909.09223
 - [26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8026–8037. <http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
 - [27] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
 - [28] Miquel Perello-Nieto, E Silva Telmo De Menezes Filho, Meelis Kull, and Peter Flach. 2016. Background Check: A general technique to build more reliable and versatile classifiers. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1143–1148.
 - [29] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. 2020. FACE: Feasible and Actionable Counterfactual Explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 344–350.
 - [30] Dillon Reisman, Jason Schultz, Kate Crawford, and Meredith Whittaker. 2018. Algorithmic impact assessments: A practical framework for public agency accountability. *AI Now Institute* (2018).
 - [31] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 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, San Francisco, CA, USA, August 13-17, 2016*. 1135–1144.
 - [32] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
 - [33] Kacper Sokol and Peter Flach. 2020. Explainability fact sheets: A framework for systematic assessment of explainable approaches. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 56–67.
 - [34] Kacper Sokol and Peter Flach. 2020. LIMETree: Interactively customisable explanations based on local surrogate multi-output regression trees. (2020). arXiv:2005.01427
 - [35] Kacper Sokol and Peter Flach. 2020. Towards Faithful and Meaningful Interpretable Representations. (2020). arXiv:2008.07007
 - [36] Kacper Sokol, Alexander Hepburn, Rafael Poyiadzi, Matthew Clifford, Raul Santos-Rodriguez, and Peter Flach. 2020. FAT Forensics: A Python toolbox for implementing and deploying fairness, accountability and transparency algorithms in predictive systems. *Journal of Open Source Software* 5, 49 (2020), 1904.
 - [37] Kacper Sokol, Alexander Hepburn, Raul Santos-Rodriguez, and Peter Flach. 2019. bLIMEy: Surrogate Prediction Explanations Beyond LIME. *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
 - [38] Sören Sonnenburg, Mikio L Braun, Cheng Soon Ong, Samy Bengio, Leon Bottou, Geoffrey Holmes, Yann LeCun, Klaus-Robert Müller, Fernando Pereira, Carl Edward Rasmussen, et al. 2007. The need for open source software in machine learning. *Journal of Machine Learning Research* 8, Oct (2007), 2443–2466.
 - [39] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 5026–5033.
 - [40] Florian Tramer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. 2017. FairTest: Discovering unwarranted associations in data-driven applications. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 401–416.
 - [41] Ke Yang, Julia Stoyanovich, Abolfazl Asudeh, Bill Howe, HV Jagadish, and Gerome Miklau. 2018. A Nutritional Label for Rankings. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 1773–1776.
 - [42] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. *International Conference on Learning Representations* (2018). <https://openreview.net/forum?id=r1Ddp1-Rb>