The Journal of Open Source Software

FAT Forensics: A Python Toolbox for Implementing and Deploying Fairness, Accountability and Transparency Algorithms in Predictive Systems

Kacper Sokol[1], **Alexander Hepburn**[2], **Rafael Poyiadzi**[2], **Matthew Clifford**[2], **Raul Santos-Rodriguez**[2], **and Peter Flach**[1]

1 Department of Computer Science, University of Bristol **2** Department of Engineering Mathematics, University of Bristol

DOI: 10.21105/joss.01904

Software

- Review
- Repository
- Archive

Editor: Ariel Rokem

Reviewers:

- @bernease
- @osolari

Submitted: 12 September 2019
Published: 29 January 2020

Background

Predictive systems, in particular machine learning algorithms, can take important, and sometimes legally binding, decisions about our everyday life. In most cases, however, these systems and decisions are neither regulated nor certified. Given the potential harm that these algorithms can cause, their qualities such as **fairness**, **accountability** and **transparency** (FAT) are of paramount importance. To ensure high-quality, fair, transparent and reliable predictive systems, we developed an open source Python package called *FAT Forensics*. It can inspect important fairness, accountability and transparency aspects of predictive algorithms to automatically and objectively report them back to engineers and users of such systems. Our toolbox can evaluate all elements of a predictive pipeline: data (and their features), models and predictions. Published under the BSD 3-Clause open source licence, *FAT Forensics* is opened up for personal and commercial usage.

Summary

FAT Forensics is designed as an interoperable framework for *implementing*, *testing* and *deploying* novel algorithms devised by the FAT research community. It facilitates their evaluation and comparison against the state of the art, thereby democratising access to these techniques. In addition to supporting research in this space, the toolbox is capable of analysing all components of a predictive pipeline – data, models and predictions – by considering their fairness, accountability (robustness, security, safety and privacy) and transparency (interpretability and explainability).

FAT Forensics collates all of these diverse tools and algorithms under a common application programming interface. This is achieved with a modular design that allows to share and reuse a collection of core algorithmic components – see Figure 1. This architecture makes the process of creating new algorithms as easy as connecting the right blocks, therefore supporting a range of diverse use cases.

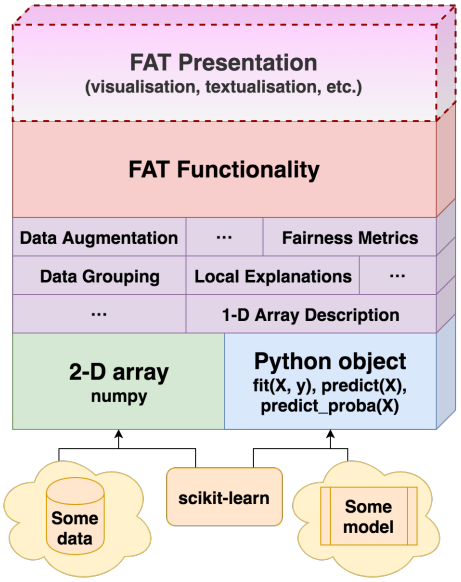

Figure 1: Modular architecture of FAT Forensics.

The format requirements for data sets and predictive models are kept to a minimum, lowering any barriers for adoption of *FAT Forensics* in new and already well-established projects. In this abstraction a data set is assumed to be a two-dimensional NumPy array: either a classic or a structured array. The latter is a welcome addition given that some of the features may be categorical (string-based). A predictive model is assumed to be a plain Python object that has `fit`, `predict` and, optionally, `predict_proba` methods. This flexibility makes our package compatible with scikit-learn – the most popular Python machine learning toolbox – without introducing additional dependencies. Moreover, this approach makes *FAT Forensics* compatible with other packages for predictive modelling since their predictive functions can be easily wrapped inside a Python object with all the required methods.

Our package improves over existing solutions as it collates algorithms across the FAT domains, taking advantage of their shared functional building blocks. The common interface layer of the toolbox supports several *modes of operation*. The **research mode** (data in – visualisation out), where the tool can be loaded into an interactive Python session, e.g., a Jupyter Notebook, supports prototyping and exploratory analysis. This mode is intended for FAT researchers who may use it to propose new fairness metrics, compare them with the existing ones or use them to inspect a new system or a data set. The **deployment mode** (data in – data out) can be used as a part of a data processing pipeline to provide a (numerical) FAT analytics, supporting automated reporting and dashboarding. This mode is intended for machine learning engineers and data scientists who may use it to monitor or evaluate a predictive system during its development and deployment.

To encourage long-term maintainability, sustainability and extensibility, *FAT Forensics* has been developed employing software engineering best practice such as unit testing, continuous integration, well-defined package structure and consistent code formatting. Furthermore, our toolbox is supported by a thorough and beginner-friendly documentation that is based on four main pillars, which together build up the user's confidence in using the package:

- narrative-driven **tutorials** designated for new users, which provide a step-by-step guidance through practical use cases of all the main aspects of the package;
- **how-to guides** created for relatively new users of the package, which showcase the flexibility of the toolbox and explain how to use it to solve user-specific FAT challenges; for example, how to build your own local surrogate explainer by pairing a data augmenter and an inherently transparent local model;

- **API documentation** describing functional aspects of the algorithms implemented in the package and designated for a technical audience as a reference material; it is complemented by task-focused *code examples* that put the functions, objects and methods in context; and
- a **user guide** discussing theoretical aspects of the algorithms implemented in the package such as their restrictions, caveats, computational time and memory complexity, among others.

We hope that this effort will encourage the FAT community to contribute their algorithms to *FAT Forensics*. We offer it as an attractive alternative to releasing yet more standalone packages, keeping the toolbox at the frontiers of algorithmic fairness, accountability and transparency research. For a more detailed description of *FAT Forensics*, we point the reader to its documentation[1] and the paper (Sokol, Santos-Rodriguez, & Flach, 2019) describing its design, scope and usage examples.

Related Work

A recent attempt to create a common framework for FAT algorithms is the *What-If* tool[2], which implements various fairness and explainability approaches. A number of Python packages collating multiple state-of-the-art algorithms for either fairness, accountability or transparency also exist. Available algorithmic *transparency* packages include:

- Skater[3] (Kramer et al., 2018),
- ELI5[4],
- Microsoft's Interpret[5] (Nori, Jenkins, Koch, & Caruana, 2019), and
- IBM's AI Explainability 360[6].

Packages implementing individual algorithms are also popular. For example, LIME[7] for Local Interpretable Model-agnostic Explanations (Ribeiro, Singh, & Guestrin, 2016) and PyCEbox[8] for Partial Dependence (Friedman, 2001) and Individual Conditional Expectation (Goldstein, Kapelner, Bleich, & Pitkin, 2015) plots.

Algorithmic *fairness* packages are also ubiquitous, for example: Microsoft's fairlearn[9] (Agarwal, Beygelzimer, Dudik, Langford, & Wallach, 2018) and IBM's AI Fairness 360[10] (Bellamy et al., 2018). However, *accountability* is relatively underexplored. The most prominent software in this space deals with robustness of predictive systems against adversarial attacks, for example:

- FoolBox[11],
- CleverHans[12] and
- IBM's Adversarial Robustness 360 Toolbox[13].

[1] https://fat-forensics.org
[2] https://pair-code.github.io/what-if-tool
[3] https://github.com/oracle/Skater
[4] https://github.com/TeamHG-Memex/eli5
[5] https://github.com/interpretml/interpret
[6] https://github.com/IBM/AIX360
[7] https://github.com/marcotcr/lime
[8] https://github.com/AustinRochford/PyCEbox
[9] https://github.com/fairlearn/fairlearn
[10] https://github.com/IBM/AIF360
[11] https://github.com/bethgelab/foolbox
[12] https://github.com/tensorflow/cleverhans
[13] https://github.com/IBM/adversarial-robustness-toolbox

FAT Forensics aims to bring together all of this functionality from across fairness, accountability and transparency domains with its modular implementation. This design principle enables the toolbox to support two modes of operation: research and deployment. Therefore, the package caters to a diverse audience and supports a range of tasks such as implementing, testing and deploying FAT solutions. Abstracting away from fixed data set and predictive model formats adds to its versatility. The development of the toolbox adheres to best practices for software engineering and the package is supported by a rich documentation, both of which make it stand out amongst its peers.

Acknowledgements

This work was financially supported by Thales, and is the result of a collaborative research agreement between Thales and the University of Bristol.

References

Agarwal, A., Beygelzimer, A., Dudik, M., Langford, J., & Wallach, H. (2018). A reductions approach to fair classification. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning*, Proceedings of machine learning research (Vol. 80, pp. 60–69). Stockholmsmässan, Stockholm Sweden: PMLR. Retrieved from http://proceedings.mlr.press/v80/agarwal18a.html

Bellamy, R. K. E., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., Lohia, P., et al. (2018). AI fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics*, 1189–1232.

Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, *24*(1), 44–65.

Kramer, A., Choudhary, P., silversurfer84, Dyke, B. V., Thai, A., Pasumarthy, N., Lemaitre, G., et al. (2018). *Datascienceinc/skater: 1.1.2*. Zenodo. doi:10.5281/zenodo.1423046

Nori, H., Jenkins, S., Koch, P., & Caruana, R. (2019). InterpretML: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*.

Ribeiro, M. T., Singh, S., & Guestrin, C. (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* (pp. 1135–1144).

Sokol, K., Santos-Rodriguez, R., & Flach, P. (2019). FAT forensics: A python toolbox for algorithmic fairness, accountability and transparency. *arXiv preprint arXiv:1909.05167*.