FairCORELS, an Open-Source Library for Learning Fair Rule Lists

Ulrich Aïvodji
UQAM
Montréal, Canada
aivodji.ulrich@uqam.ca

Julien Ferry*
LAAS-CNRS, Université de Toulouse, CNRS
Toulouse, France
jferry@laas.fr

Sébastien Gambs
UQAM
Montréal, Canada
gambs.sebastien@uqam.ca

Marie-José Huguet
LAAS-CNRS, Université de Toulouse, CNRS, INSA
Toulouse, France
huguet@laas.fr

Mohamed Siala
LAAS-CNRS, Université de Toulouse, CNRS, INSA
Toulouse, France
msiala@laas.fr

ABSTRACT

`FairCORELS` is an open-source Python module for building fair rule lists. It is a multi-objective variant of `CORELS`, a branch-and-bound algorithm to learn certifiably optimal rule lists. `FairCORELS` supports six statistical fairness metrics, proposes several exploration parameters and leverages on the fairness constraints to prune the search space efficiently. It can easily generate sets of accuracy-fairness trade-offs. The models learnt are interpretable by design and a sparsity parameter can be used to control their length.

CCS CONCEPTS

• **Software and its engineering** → *Software libraries and repositories*; • **Computing methodologies** → **Rule learning**; **Supervised learning**; *Machine learning algorithms.*

KEYWORDS

Machine Learning, Fairness, Interpretability

ACM Reference Format:
Ulrich Aïvodji, Julien Ferry, Sébastien Gambs, Marie-José Huguet, and Mohamed Siala. 2021. FairCORELS, an Open-Source Library for Learning Fair Rule Lists. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3481965

1 INTRODUCTION

As machine learning techniques are being increasingly used in high stake decision-making systems, fairness and interpretability of such systems have emerged as important ethical issues to address.

More precisely, a major concern is to ensure that a model's predictions are fair, even if it is trained on possibly biased data. Existing fairness-enhancing techniques can be categorized into three main families, mitigating biases at different stages of the machine learning pipeline: *pre-processing*, *post-processing* and *algorithmic modification techniques* [6, 13]. Our approach falls into the last category, as it consists of directly modifying the learning algorithm to enforce fairness.

Another major concern is about facilitating the understanding of a model. While some approaches attempt to explain black-box models or their predictions [14], it has been shown that such explanations may not be trustworthy (*e.g.*, they could be manipulated by the model provider) [1, 19, 20]. For this reason, building models that are interpretable by design (such as *rule lists of reasonable size*) is an interesting approach for avoiding the *a posteriori* explanation of black-box models [19]. In particular, rule-based models exhibit interesting properties for interpretability [12] and have been shown to be more expressive than decision trees of equivalent size [18]. While state-of-the-art tools for fair learning exist, such as `fairlearn`[1], such alternatives are not specifically designed for learning rule lists. In addition, providing optimality guarantees is important, as there can be societal implications for a lack of optimality [4], especially when addressing fairness.

To address jointly these concerns, we propose `FairCORELS` [2], an open-source Python library to build rule lists models that are both fair and accurate. `FairCORELS` leverages on recent advances, provided by `CORELS`, for learning optimal rule lists [3], by adapting them to also take into account fairness constraints. Given a statistical notion of fairness and a sensitive attribute that could lead to discrimination, our algorithm searches for the rule list optimizing the decrease of both unfairness and misclassification error.

The paper is organized as follows. In the next section, we describe our method for learning fair rule lists. Afterwards in Section 3, we illustrate the use of the proposed Python library for mitigating unfairness in several well-known classification tasks. Finally, we conclude in Section 4.

*First author

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3481965

[1] https://fairlearn.org/

2 ALGORITHM FOR LEARNING FAIR RULE LISTS

In this section, we briefly review the fairness notions supported by our library before introducing the optimization problem as well as the FairCORELS algorithm.

Let $\mathcal{D} = (X, Y, A)$ denotes a dataset, in which each example $e_i = (x_i, y_i, a_i) \in \mathcal{D}$ with $x_i \in \{0,1\}^n$ a set of n unprotected features, $y_i \in \{0,1\}$ a label and $a_i = (a_{i,p}, a_{i,u}) \in \{0,1\}^2$ with $a_{i,p} + a_{i,u} \leq 1$ indicating protected and unprotected group memberships. The goal of a supervised learning algorithm is to learn a classifier whose predictions $\hat{Y}$ match as accurately as possible the true labels Y.

2.1 Quantifying Fairness

Several fairness notions have been proposed in the machine learning literature, which can be grouped into three types of techniques: *individual*, *causal* and *statistical* fairness [21]. This work is designed to mitigate statistical fairness, also called *group* fairness, which aims at equalizing some values between distinct *protected* groups of instances, defined with respect to the value of some sensitive attributes. The core idea is that individuals should be treated similarly, whatever *protected* group they belong to, because such groups only differ by the value of some features that should not (for ethical, legal or philosophical reasons) be used for decision making.

Depending on the measure being equalized across protected groups, several metrics have been proposed. Our work implements six metrics, namely statistical parity [9], predictive parity [8], predictive equality [8], equal opportunity [15], equalized odds [15] and conditional use accuracy equality [7]. These metrics are summarized in Table 1, along with the statistical measure to be equalized across protected groups.

2.2 Problem Formulation

FairCORELS solves a bi-objective optimization problem minimizing both prediction error and unfairness and it is based on a ϵ-constraint approach. Formally, FairCORELS [2] explores the space of rule lists $\mathcal{R}$ to find the rule list minimizing the objective function f_{obj} while having fairness at least ϵ (*ie* exhibiting at most unfairness $1 - \epsilon$).

Function `misc(.)` denotes the misclassification error, and `unf(.)` measures unfairness according to the selected metric. In the provided implementation of FairCORELS, `unf(.)` can be either UNF_{SP}, UNF_{PP}, UNF_{PE}, UNF_{EO}, $\text{UNF}_{\text{EOdds}}$, UNF_{CUAE}, as defined in Table 1. In this setting, the rule list r^* outputted by FairCORELS is the solution of the following problem, in which K_r is the length of rule list r and λ is a regularization parameter controlling the accuracy/sparsity trade-off:

$$\begin{aligned} \underset{r \in \mathcal{R}}{\arg\min} \quad & f_{obj} = \texttt{misc}(r, X, Y) + \lambda . K_r \\ \text{s.t.} \quad & \texttt{unf}(r, X, Y, A) \leq 1 - \epsilon \end{aligned}$$

2.3 Algorithm Overview

Prior to the use of FairCORELS is the pre-computation of rules, which are given as inputs to FairCORELS. Rules can be any combination of attributes (or even attributes themselves) that evaluate to either 0 or 1 for all instances.

Based on the CORELS algorithm [3], FairCORELS represents the space of rule lists $\mathcal{R}$ using a prefix tree, in which each node is a rule and each path from the root is a potential solution. Starting from a root node, the exploration frontier is contained in a priority queue Q, which can be ordered following several strategies: *depth-first* search, *breadth-first* search or different *best-first* searches. We also implement several custom *breadth-first* search heuristics, among which the `BFS obj.-aware` orders rule lists of equal lengths depending on their objective function values. At each iteration, the prefix r at the top of Q is selected. For each possible extension r' of it, f_{obj} is computed. Whenever its value is improved and the solution meets the fairness constraint, the current best solution is updated. All extensions r' that do not violate any of the bounds are pushed into Q.

FairCORELS exploits several bounds to efficiently prune $\mathcal{R}$, leveraging both the objective function and the fairness constraint. It returns theoretically certifiable optimal rule lists (*i.e.*, given a set of training instances and rules, it provably returns the rule list with smaller f_{obj} among those meeting the fairness constraints). However, as the prefix tree size grows exponentially with the number of rules, a parameter n_{iter} can be used to stop the exploration whenever the number of nodes evaluated in the prefix tree reaches n_{iter}. This parameter is typically used to limit the memory footprint of the program or to quickly obtain some solution.

3 FAIRCORELS PYTHON LIBRARY: EXAMPLE USE

The source code and documentation of FairCORELS is available online[2], and FairCORELS can also be easily installed using Pypi[3], with `pip install faircorels`. A complete description of the algorithm's parameters is provided, along with example code and binarized data for two well-known biased datasets. FairCORELS is based on the Python binding of CORELS[4]. The core of the algorithm is implemented in C++ and called from the Python wrapper using Cython. Along with the classifier object, FairCORELS also contains some tools for auditing a model's fairness. A wrapper object for fair ensemble learning using the Bootstrap Aggregating (Bagging) method [23] is also provided.

In this paper we propose an illustration of the use of FairCORELS, for the Statistical Parity metric, on four well-known biased classification tasks:

- The *Adult* dataset [11] in which the objective is to predict whether someone earns more than 50,000$ per year. We consider gender as the sensitive attribute.
- The *COMPAS* dataset [5] in which the objective is to predict whether criminal offenders will recidivate within the next two years. We consider race as the sensitive attribute.
- The *Bank Marketing* dataset [16] in which the objective is to predict whether a customer will subscribe to a term deposit. We use age as the sensitive attribute.
- The *Default of Credit Card* dataset [22], in which the objective is to predict whether a person will default in payment. We consider gender as the sensitive attribute.

[2] https://github.com/ferryjul/fairCORELS
[3] https://pypi.org/project/faircorels/
[4] https://github.com/corels/pycorels

Table 1: Summary of statistical fairness metrics supported by `FairCORELS`, along with the related statistical measure to be equalized, and the associated mathematical expression.

Fairness Metric	Statistical Measure	Expression
Statistical parity (SP)	Probability of being assigned to the positive class	$\text{UNF}_{\text{SP}} = \lvert P(\hat{Y}=1\lvert A=0) - P(\hat{Y}=1\lvert A=1)\rvert$
Predictive parity (PP)	Positive predictive value (PPV)	$\text{UNF}_{\text{PP}} = \Delta_{PPV} = \lvert P(Y=1\lvert\hat{Y}=1,A=0) - P(Y=1\lvert\hat{Y}=1,A=1)\rvert$
Predictive equality (PE)	False positive rate (FPR)	$\text{UNF}_{\text{PE}} = \Delta_{FPR} = \lvert P(\hat{Y}=1\lvert Y=0,A=0) - P(\hat{Y}=1\lvert Y=0,A=1)\rvert$
Equal opportunity (EOpp)	False negative rate (FNR)	$\text{UNF}_{\text{EOpp}} = \Delta_{FNR} = \lvert P(\hat{Y}=0\lvert Y=1,A=0) - P(\hat{Y}=0\lvert Y=1,A=1)\rvert$
Equalized odds (EOdds)	FNR and FPR	$\text{UNF}_{\text{EOdds}} = \max(\Delta_{FNR}, \Delta_{FPR})$
Conditional Use Accuracy Equality (CUAE)	Positive predictive value and Negative predictive value	$\text{UNF}_{\text{CUAE}} = \max(\Delta_{PPV}, \Delta_{NPV})$ with $\Delta_{NPV} = \lvert P(Y=0\lvert\hat{Y}=0,A=0) - P(Y=0\lvert\hat{Y}=0,A=1)\rvert$

The procedure is identical for the four datasets. The first step towards the use of `FairCORELS` is the binarization of the dataset. Then, given the set of binarized features, a rule mining procedure searches for rules formed by single- and two-clause antecedents, only retaining rules that capture at least 0.05% of the training instances. In our experiments, the sensitive attribute is not used to prevent disparate treatment. We illustrate the use of `FairCORELS` to enforce the Statistical Parity metric, but similar results can be obtained for any of the six supported group fairness metrics. For all experiments, the maximum size of the prefix tree is fixed to 25.10^5, the `BFS obj.-aware` heuristic is used to order the queue, and the regularization parameter $\lambda = 10^{-3}$. The corresponding model creation is shown in Code snipet 1.

```
# Create the classifier object
clf = FairCorelsClassifier(
        n_iter=2.5*1e6, # Max. size of the prefix tree
        c=10e-3,  # Regularization parameter Lambda
        max_card=2, # Rule mining: max. rules cardinality
        min_support = 0.05, # Rule mining: min.support
        policy="bfs", # Exploration heuristic
        bfs_mode=2, # BFS obj. -aware
        fairness=1, # 1 for Statistical Parity
        epsilon=epsilon, # The fairness constraint
        maj_vect=A_train_u, # Binary vector (unprotected group membership)
        min_vect=A_train_p, # Binary vector (protected group membership)
        useUnfairnessLB=True) # Use the advanced filtering
# Train it
clf.fit(
    X_train,
    y_train,
    features=featuresNames,
    prediction_name="high_income")
```

Code snipet 1: Example to train a rule list with statistical parity constraint.

Note that rule mining can either be performed as pre-processing or using the built-in options. We use the built-in options in Code snipet 1, mining rules that are conjunctions of 2 attributes (or their negations) capturing at least 5% of the training instances. For our experiments, we performed rule mining as pre-processing to control the number of generated rules before training a model. Note that min_{vect} and maj_{vect} are used to define the protected (respectively unprotected) groups and measure unfairness while training the model. They can correspond to a particular feature, but can also be any binary vectors, as long as they do not intersect. Our Python package was designed to meet `scikit-learn`[5] naming conventions and can easily be integrated in any piece of code originally written for `scikit-learn` models.

Our experiments were conducted on an Intel Xeon Processor E3-1271 v3 (3.60 GHz) with 32GB of RAM. Presented values (accuracy, fairness) are averaged using 5-folds cross-validation. The reported model's statistics are measured on their respective train/test split. Recall that $unfairness = 1 - fairness$.

Generating Single Rule Lists. Rule list 2 has been learned maximizing accuracy only (hence setting the fairness constraint $\epsilon = 0$). It has high accuracy but exhibits considerable unfairness. As we prevent the use of the sensitive attribute in the model's decisions, the *unfair* nature of this interpretable model may not be obvious. Indeed, some of the non-sensitive attributes (or combinations of them) may be correlated with the sensitive one. In this case, statistical fairness measures allow for measuring disparate impact, which happens when a model's outcomes differ significantly depending on individuals' membership to some protected group, even without explicit knowledge of such membership.

```
if [capitalGain>=5095.5] then [high]
else if [1881.5<=capitalLoss<=1978.5] then [high]
else if [education:hs_grad AND capitalLoss<=1534.0] then [low]
else if [occupation:whiteCollar AND hoursPerWeek>=40.5] then [high]
else [low]
```

Rule list 2: Example of a rule list found on the Adult Income dataset, for the Statistical Parity metric, maximizing accuracy only ($\epsilon = 0$). Test accuracy on the associated train/test split is 0.817**, and unfairness is** 0.073**. For these running parameters, test accuracy (averaged across the 5 folds) is** 0.817**, and unfairness is** 0.066**.**

Rule list 3 corresponds to $\epsilon = 1.0$. We want to point out that in this case the only rule list able to achieve perfect fairness is a constant (trivial) classifier. As the used dataset is class-unbalanced, constantly predicting the majority class leads to an acceptable accuracy (better than random guess), without exhibiting unfairness at all.

```
if 1 then [low]
```

Rule list 3: Example of a rule list found on the Adult Income dataset, for the Statistical Parity metric, with fairness constraint $\epsilon = 1.00$. Test accuracy (averaged across the 5 folds) is 0.752**, and unfairness is** 0.00**.**

Between these two extremes, we can generate many trade-offs by applying different constraints ϵ. By doing so, we are able to significantly reduce unfairness without reaching trivial accuracy.

[5] https://scikit-learn.org/

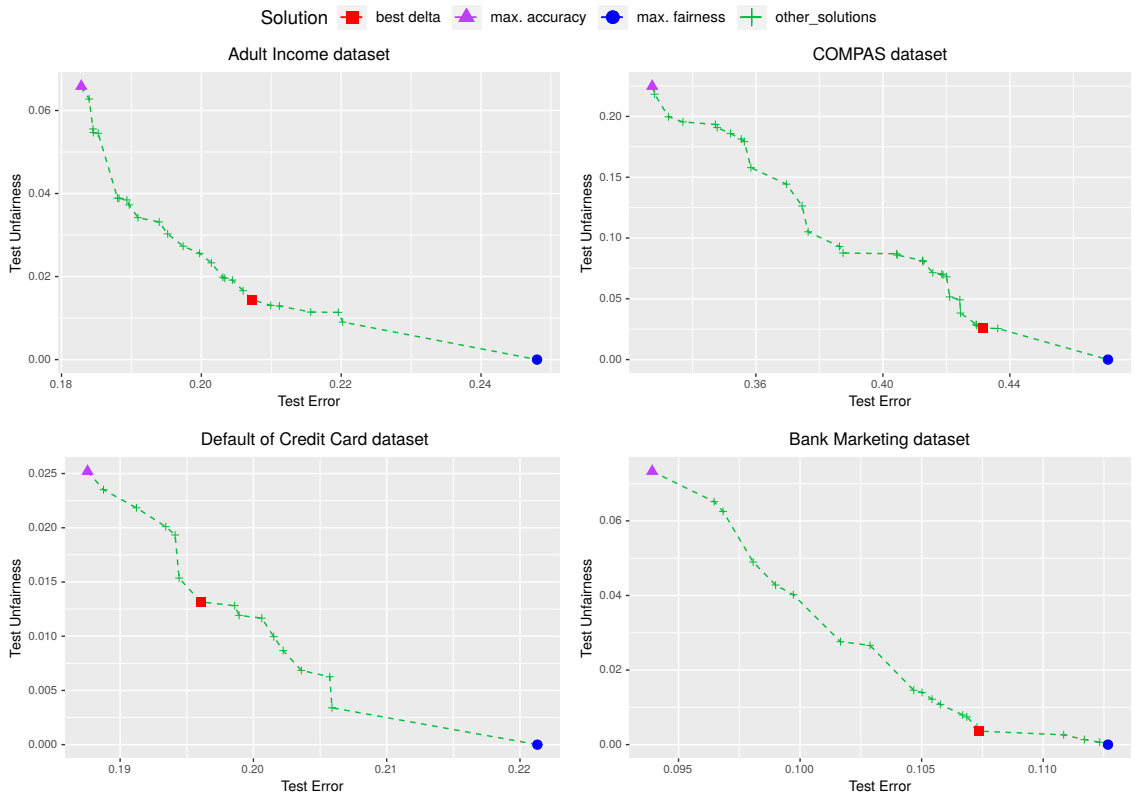

Figure 1: Pareto front on test set (unfairness/classification error tradeoffs) for statistical parity on four different datasets. Bottom-left (low unfairness, low error) is preferable.

For example, Rule list 4 has very low unfairness while maintaining good accuracy. A (possible) measure of the trade-off's quality has been proposed in [17]: $delta = accuracy - unfairness$. Among our generated solutions, Rule List 4 maximizes this criterion.

```
if [education:hs_grad AND hoursPerWeek>=40.5] then [low]
else if [35.5<=age<=61.5 AND occupation:professional] then [high]
else if [capitalGain>=7073.5] then [high]
else [low]
```

Rule list 4: Example of a rule list found on the Adult Income dataset, for the Statistical Parity metric, with fairness constraint $\epsilon = 0.984$. Test accuracy on the associated train/test split is 0.792**, and unfairness is** 0.0036**. For these running parameters, test accuracy (averaged across the 5 folds) is** 0.793 **and unfairness is** 0.014**.**

Generating a Set of Accuracy/Fairness Trade-offs. `FairCORELS` can also be applied to generate a set of accuracy/fairness tradeoffs instead of a single solution. This can be done easily by performing successive (possibly parallel) calls to `FairCORELS`, varying the fairness constraint parameter ϵ. Indeed, this allows domain experts to select the most appropriate solution, depending on the accuracy/fairness trade-off, but also possibly on the associated models themselves. The interpretable nature of the built models makes their analysis easier. Figure 1 shows a set of non-dominated solutions (in terms of error and unfairness, on their test sets) for the four datasets, in which Rule Lists 2 ("max. accuracy"), 3 ("max. fairness") and 4 ("best delta") are represented on the Adult Income dataset plot. We see that the set of possible trade-offs is large enough for the decision maker to select the appropriate one.

4 DISCUSSION

With `FairCORELS`, we propose an in-processing method for learning interpretable and fair models. One of the strengths of our approach is that it includes most of the standard statistical fairness metrics. In addition, our core method is metric-agnostic and thus integrating new fairness metrics should be straightforward.

Our formulation of the fair learning problem is particularly convenient for real-life fair learning problems, where particular unfairness values may correspond to legal requirements (e.g., the 80% rule for Statistical Parity [10]).

As shown in [2], `FairCORELS` is able to achieve interesting tradeoffs between accuracy and fairness, that are competitive with state-of-the-art interpretable and non-interpretable methods.

Additionally, `FairCORELS` is an exact method, that can be used to certify the optimality of the generated models. As demonstrated throughout this paper (and the related demonstration), `FairCORELS` scales fairly well and can be used to learn accurate and fair models using datasets of various sizes.

REFERENCES

[1] Ulrich Aivodji, Hiromi Arai, Olivier Fortineau, Sébastien Gambs, Satoshi Hara, and Alain Tapp. 2019. Fairwashing: the risk of rationalization. In *International Conference on Machine Learning*. 161–170.

[2] Ulrich Aïvodji, Julien Ferry, Sébastien Gambs, Marie-José Huguet, and Mohamed Siala. 2019. Learning Fair Rule Lists. *arXiv preprint arXiv:1909.03977* (2019).

[3] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. 2017. Learning certifiably optimal rule lists. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 35–44.

[4] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. 2018. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research* 18, 234 (2018), 1–78.

[5] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. Machine bias. *ProPublica, May* 23 (2016).

[6] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilović, et al. 2019. AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development* 63, 4/5 (2019), 4–1.

[7] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. 2018. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research* (2018).

[8] Alexandra Chouldechova. 2017. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data* 5, 2 (2017), 153–163.

[9] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*. ACM, 214–226.

[10] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 259–268.

[11] Andrew Frank and Arthur Asuncion. 2010. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California. *School of information and computer science* 213 (2010), 2–2.

[12] Alex A Freitas. 2014. Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter* 15, 1 (2014), 1–10.

[13] 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*. ACM, 329–338.

[14] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *Comput. Surveys* 51, 5 (2018), 93.

[15] Moritz Hardt, Eric Price, Nati Srebro, et al. 2016. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*. 3315–3323.

[16] Sérgio Moro, Paulo Cortez, and Paulo Rita. 2014. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems* 62 (2014), 22–31.

[17] Edward Raff, Jared Sylvester, and Steven Mills. 2018. Fair forests: Regularized tree induction to minimize model bias. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. ACM, 243–250.

[18] Ronald L Rivest. 1987. Learning decision lists. *Machine learning* 2, 3 (1987), 229–246.

[19] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 5 (2019), 206–215.

[20] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. 2020. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 180–186.

[21] Sahil Verma and Julia Rubin. 2018. Fairness definitions explained. In *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*. IEEE, 1–7.

[22] I-Cheng Yeh and Che-hui Lien. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications* 36, 2 (2009), 2473–2480.

[23] Zhi-Hua Zhou. 2019. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.