Predicting Fairness of ML Software Configurations

Salvador Robles Herrera
University of Texas at El Paso
El Paso, USA
sroblesher1@miners.utep.edu

Verya Monjezi
University of Texas at El Paso
El Paso, USA
vmonjezi@miners.utep.edu

Vladik Kreinovich
University of Texas at El Paso
El Paso, USA
vladik@utep.edu

Ashutosh Trivedi
University of Colorado Boulder
Boulder, USA
ashutosh.trivedi@colorado.edu

Saeid Tizpaz-Niari
University of Texas at El Paso
El Paso, USA
saeid@utep.edu

ABSTRACT

This paper investigates the relationships between hyperparameters of machine learning and fairness. Data-driven solutions are increasingly used in critical socio-technical applications where ensuring fairness is important. Rather than explicitly encoding decision logic via control and data structures, the ML developers provide input data, perform some pre-processing, choose ML algorithms, and tune hyperparameters (HPs) to infer a program that encodes the decision logic. Prior works report that the selection of HPs can significantly influence fairness. However, tuning HPs to find an ideal trade-off between accuracy, precision, and fairness has remained an expensive and tedious task. *Can we predict the fairness of HP configuration for a given dataset? Are the predictions robust to distribution shifts?*

We focus on group fairness notions and investigate the HP space of 5 training algorithms. We first find that tree regressors and XGBoots significantly outperformed deep neural networks and support vector machines in accurately predicting the fairness of HPs. When predicting the fairness of ML hyperparameters under temporal distribution shift, the tree regressors outperform the other algorithms with reasonable accuracy. However, the precision depends on the ML training algorithm, dataset, and protected attributes. For example, the tree regressor model was robust for training data shift from 2014 to 2018 on logistic regression and discriminant analysis HPs with sex as the protected attribute; but not for race and other training algorithms. Our method provides a sound framework to efficiently perform fine-tuning of ML training algorithms and understand the relationships between HPs and fairness.

CCS CONCEPTS

• **Software and its engineering → Extra-functional properties**.

KEYWORDS

ML Hyperparameters, Fairness, distribution shifts

PROMISE '24, July 16, 2024, Porto de Galinhas, Brazil

ACM ISBN 979-8-4007-0675-2/24/07
https://doi.org/10.1145/3663533.3664040

ACM Reference Format:
Salvador Robles Herrera, Verya Monjezi, Vladik Kreinovich, Ashutosh Trivedi, and Saeid Tizpaz-Niari. 2024. Predicting Fairness of ML Software Configurations. In *Proceedings of the 20th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE '24), July 16, 2024, Porto de Galinhas, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3663533.3664040

1 INTRODUCTION

Artificial intelligence (AI) has become an integral part of modern software solutions that assist in socio-economic and legal-critical decision-making processes such as releasing patients [24], identifying loan defaults [20], and detecting tax evasion [38]. Since AI-driven software solutions are derived from prior experiences, it is not surprising that they encode social biases in their decision logic that may discriminate against vulnerable communities and violate ethical and legal requirements.

Unfortunately, discriminatory software is a major threat to the trustworthiness of automated decision-support systems, and fairness defects are plentiful in real systems. Parole decision-making software was found to harm black and Hispanic defendants by falsely predicting a higher risk of recidivism than for non-Hispanic white defendants [23]; Amazon's hiring algorithm disproportionately rejected more female applicants than male applicants [22]; and data-driven auditing algorithm selected black taxpayers with earned income tax credit claims (EITC) at much higher rates than other racial groups for an audit [38, 41]. As evidenced by these examples, resulting software may particularly disadvantage minorities and protected groups and be found non-compliant with law such as the US Civil Rights Act [6]. Therefore, helping programmers detect and mitigate fairness bugs in social-critical data-driven software systems is crucial to ensure inclusion in our modern, increasingly digital society.

Recently, the software engineering community has made concentrated efforts to improve fairness of data-driven software [5, 8, 9, 19, 39, 40, 45]. One area of focus is on establishing best practices for fairness of training data-driven software that involves *pre-processing* [5], *feature selection* [45], and *hyperparameter selection* [9, 30, 40]. Specifically, hyperparameter tuning (or optimization) is the process of tweaking the configuration space of ML training algorithms to infer an ML model, based on some validation dataset, that provides an ideal trade-off between complexity and performance. Some prominent examples of hyperparameters

include l1 vs. l2 loss function in support vector machines, the maximum depth of a decision tree, and the number of layers/neurons in neural networks.

Chakraborty et al. [9] were the first to observe the influence of hyperparameter (HP) configurations and fairness. Fairway [8] searched the space of HPs to mitigate software discrimination. Parfait-ML [40] found and localized discriminatory HPs in the prevalent ML algorithms. While the critical role of HPs on fairness is evident, exploring the space of HPs to develop fair ML software is time-consuming and expensive due to the need to train the entire model for any given configuration. *This paper investigates whether a relationship between HPs of different ML algorithms and fairness can be learned via ML regression methods.* If feasible, the models can be used to predict whether a HP meets fairness requirements before performing a (full) training cycle. Thus, it can significantly reduce the required computations of hyperparameter tuning.

As new data is typically acquired on an annual or periodic basis and ML system evolves over time, it becomes imperative to monitor how the fairness of a specific HP configuration changes over time. Hence, this paper also analyzes the robustness of fair configuration of HPs under temporal distribution shifts. We emphasize on time distribution shifts and investigate when and how robustness is affected or not. We pose the following research questions:

- *Can we accurately predict group fairness of ML hyperparameters for a given training dataset and protected attribute? what is the performance of neural methods, support vectors, and trees as the prediction methods?*
- *Can we predict the fairness of ML hyperparameters under the temporal distribution shift of test data? which types of ML algorithms are more robust to the shift?*

To answer these research questions, we perform experiments over the HP space of five popular ML algorithms (Decision Tree Classifier, Support Vector Machine, Logistic Regression Classifier, Random Forest, and Discriminant Analysis) trained over four socially-critical datasets (Adult Census, Compas Recidivism, Default Credit, and Bank Marketing). To measure fairness, we use the average odd difference (AOD) which is the average of differences between the true positive rates and the false positive rates of two protected groups (e.g., male vs. female).

We first use an evolutionary search algorithm to explore various HP configurations and record the fairness of resulting models (over some validation datasets) [40]. Then, we feed the collected datasets of HPs (features) and fairness (outcome) to four ML regression methods (Deep Neural Network, Support Vector Regressor, Tree Regressors, and XGBoost) to learn a function from HPs to fairness. To measure the accuracy of prediction models, we used four metrics including the coefficient of determination (known as R^2 score).

Over a fixed training dataset, we observe that Tree Regressors and XGBoost outperform other algorithms in accurately predicting AOD fairness of ML hyperparameters for all five algorithms. In 40% of cases, they achieved $R^2 \geq 0.95$, and only in 6.7% of cases, the algorithms do not achieve a reliable accuracy (an $R^2 \leq 0.5$). Under the temporal distribution shifts, we observe that the precision depends on the training algorithm, dataset, and the protected attribute. When there is one year shift (e.g., trained over the income census 2014 and predicting for 2015), Tree Regressor and XGBoost achieved high accuracy in 20% of benchmarks. We observed that these cases are related to the HP space of Logistic Regression and Discriminant Analysis with sex as the protected attribute; and the precision is significantly degraded for other training algorithms and protected attributes like race.

In summary, our experiences provide a sound and robust framework to systematically examine the influence of hyperparameters over fairness. Our vision of usage describes how collecting and training fairness characteristics of hyperparameters can help reduce the cost of training data-driven software solutions by avoiding biased configurations and leveraging promising hyperparameters. We also observe the difficulty of making such predictions in general, and point out circumstances for successful usages and challenges for future research.

2 BACKGROUND

We consider *regression problem* tasks where the *target* variable is a group fairness metric, and *features* are the hyperparameter variables of ML training algorithms. The values of hyperparameter variables and target fairness variable are collected after running an evolutionary hyperparameter optimization algorithm.

Fairness Notion. Fairness definitions are either concerned with individual fairness or group fairness. Examples of individual fairness is fairness through awareness (FTA) [15] that requires two *individuals* with similar non-protected attributes are treated similarly, regardless of their background. *Group fairness* requires the statistics of ML outcomes for different *protected groups* remain similar [21]. There are multiple metrics to measure group fairness. Among them, *equal opportunity difference* (EOD) measures the difference between the true positive rates (TPR) of two protected groups. Similarly, *average odd difference* (AOD) is the average of differences between the true positive rates (TPR) and the false positive rates (FPR) of two protected groups [3, 8, 40, 45].

ML training process. In the data-driven paradigm, the ML programmers often provide input data and build an ML model using a programming interface [42]. The interface interacts with the core ML algorithms and builds different ML models. At the heart of the training process, tweaking HPs is particularly challenging since they cannot be estimated from the input data, and there is no analytical formula to calculate its values [25]. Examples of HPs are tolerance of optimization in SVMs, maximum features to search in random forests, and minimum samples in leaf nodes of decision trees. We distinguish ML training HPs from *model parameters* that are inferred automatically after training.

Related Work. There are multiple works that studied the influence of HPs on fairness [8, 9, 40]. Evolutionary algorithms have been significantly used to optimize the HP search in the training process [28, 32, 37, 44]. However, there are a few tools to perform Pareto-optimal search in the space of HPs to infer an idea trade-off between fairness (e.g., EOD and AOD) and accuracy (e.g., F1 socre). Parfait-ML [40] is a gray-box evolutionary search algorithm that explores the training space of ML algorithms to manifest HPs that minimize/maximize fairness within an acceptable range of accuracy requirements. Given a training algorithm and a fairness-sensitive dataset, Parfait-ML [40] generates a set of HPs that characterize the Pareto-dominant curves of fairness and unfairness based on a group

fairness metric such as EOD and AOD. All of these works focus on either testing the space of HPs to find biased configurations or mitigating biased ML models by carefully picking HPs. Instead, we focus on learning the fairness characteristics of ML HPs in order to improve the efficiency and fairness of ML software development.

3 FAIRNESS OF ML HYPERPARAMETERS

While the relationships between ML hyperparameters and functional metrics such as overall accuracy, precision, and F1 scores have been well studied, the presence of fairness makes the hyperparameter tuning challenging. Our goal is to study whether a relationship between hyperparameters and fairness can be learned via regression ML methods.

The ML Paradigm. Machine learning software often deploys mature, off-the-shelf ML libraries to learn various models from training data. We can succinctly view a training problem as the problem of identifying a mapping $M : \mathcal{X} \rightarrow \mathcal{Y}$ from a set $\mathcal{X}$ of inputs to a set $\mathcal{Y}$ of outputs by learning from a fixed dataset $\mathcal{D} = \{(\mathbf{x_i}, \mathbf{y_i})\}_{i=1}^{N}$ so that M generalizes well to previously unseen situations.

The training process includes the set $\mathcal{H}$ of hyperparameters that let the users define the hypothesis class for the learning tasks. The data-driven programs sift through the given dataset $\mathcal{D}$ to learn an "optimal" value $\theta \in \Theta_h$ and thus compute the learning model $M_h(\theta|\mathcal{D}) : \mathcal{X} \rightarrow \mathcal{Y}$ automatically. When $\mathcal{D}$ and θ are clear from the context, we write M_h for the resulting model.

The fitness of a hyperparameter $h \in \mathcal{H}$ is evaluated by computing the accuracy (ratio of correct results) of the model M_h on a validation dataset $\mathcal{D}_*$. We denote the accuracy of a model M over $\mathcal{D}_*$ as ACC^M. The dataset $\mathcal{D}_*$ is typically distinct from $\mathcal{D}$ but assumed to be sampled from the same distribution. In the fairness-sensitive applications, we also have fairness requirements. Let predicate $\pi : \mathcal{X} \rightarrow \{0, 1\}$ be the input variables characterizing the protected status of a data point $\mathbf{x}$ (e.g., race, sex, or age). Without loss of generality, we assume there are only two protected groups: a group with $\pi(\mathbf{x}) = 0$ and a group with $\pi(\mathbf{x}) = 1$. We also assume that the predicate $\phi : \mathcal{Y} \rightarrow \{0, 1\}$ over the output variables characterizes a favorable outcome (e.g., low re-offend risk) with $\phi(\mathbf{y}) = 1$.

Given $\mathcal{D}_*$ and $M : \mathcal{X} \rightarrow \mathcal{Y}$, we define true-positive rate (TPR) and false-positive rate (FPR) for the protected group $i \in \{0, 1\}$ as

$$TPR^M(i) = \frac{\left|\{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_* : \pi(\mathbf{x}) = i, \phi(M(\mathbf{x})) = 1, \phi(\mathbf{y}) = 1\}\right|}{\left|\{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_* : \pi(\mathbf{x}) = i\}\right|}$$

$$FPR^M(i) = \frac{\left|\{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_* : \pi(\mathbf{x}) = i, \phi(M(\mathbf{x})) = 1, \phi(\mathbf{y}) = 0\}\right|}{\left|\{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_* : \pi(\mathbf{x}) = i\}\right|}.$$

We use a prevalent notions of group fairness [3, 8, 45]: *equal opportunity difference* (EOD) of M against $\mathcal{D}_*$ between two groups is $EOD^M = \left|TPR^M(0) - TPR^M(1)\right|$, and *average odd difference* (AOD) of M against $\mathcal{D}_*$ between two groups is

$$AOD^M = \frac{|TPR^M(0) - TPR^M(1)| + |FPR^M(0) - FPR^M(1)|}{2}.$$

We notice that $0 \leq EOD^M \in \mathbb{R} \leq 1$ and $0 \leq AOD^M \in \mathbb{R} \leq 1$, and higher values of EOD^M and AOD^M indicate low fairness.

The key idea behind hyperparameter optimization methods [8, 40] for fairness and accuracy is to search the space of hyperparameters and find hyperparameters that maximize fairness, while making sure that the accuracy does not degrade below a given threshold. To understand the effects of hyperparameters on fairness, these methods also allow us to find hyperparameters that minimize fairness. The both sets of hyperparameters that manifest maximum and minimum fairness values provide us with representative samples to systematically investigate the relationships between hyperparameters and fairness.

A *fairness trace* $\mathcal{F}$ of an ML training algorithm M is characterized by the tuple $(\mathcal{D}, \mathcal{H}, \Pi, F)$ where $\mathcal{D}$ is the training dataset, $\mathcal{H}$ is the hyperparameter, Π is a predicate over protected attributes, and $F \in \{EOD, AOD\}$ is the group fairness metric of ML model M_h^D when trained over a dataset $D \in \mathcal{D}$ with the hyperparameter $h \in \mathcal{H}$ and the protected attribute predicate $p \in \Pi$.

> Given a set of fairness training set $\mathcal{F}$, our goal is to infer a function $A(\omega|D, p) : \mathcal{H} \rightarrow [0, 1]$ with different hypothesis classes of ω (e.g., linear, tree, and neural network regressors) over fairness training traces $\mathcal{F}$ that generalizes well to the testing traces $\mathcal{F}_*$, i.e., learn fairness as a function of hyperparameters for a fixed dataset $D \in \mathcal{D}$ and protected predicate $p \in \Pi$. We call A and A_w HP prediction algorithm and model, respectively.

From the practical point of view, it is not only important that fairness of hyperparameters of ML training algorithms can be predicted for a specific task at a specific time; but also whether a similar relationship can be learned under distribution shifts (e.g., under covariate shift [26, 43] when the feature distributions are different). One specific type of shifts is temporal distribution shifts where our goal is to evaluate whether the prediction model is robust to the temporal changes. In other words, we study the accuracy of $A_{D^0 \rightarrow D^k}$, that is to evaluate the ML prediction algorithm $A(\omega|D^0, p)$, trained with the base dataset at the time unit 0, with the dataset D^k, sampled after k units of time (e.g, after k year).

4 LEARNING FAIRNESS FUNCTIONS

Our goal is to infer ω of HP prediction algorithms A to accurately predict the fairness of ML algorithm configurations. There are many different ML algorithms that can be used to solve the inference problem. After some initial experimentation, we focus on four types of ω with the following prediction algorithms:

1) Deep Neural Network (DNN). We primarily work with feedforward neural networks with ReLU activation units. A feedforward neural network is characterized by its number of hidden layers, the input and output dimensions, and the number of neurons in each layer. Each hidden layer i implements an affine mapping T_i that takes outputs of the previous layer and applies a linear function using its weights along with ReLU activation function (σ). The function $f_{\mathcal{N}} : \mathbb{R}^{w_0} \rightarrow \mathbb{R}^{w_{N+1}}$ implemented by DNN $\mathcal{N}$ is $A_\omega = T_{k+1} \circ \sigma \circ T_k \circ \sigma \circ \cdots T_2 \circ \sigma \circ T_1$.

2) Support Vector Regression (SVR). SVR follows the same principles as a support vector machine (SVM) that aims to infer the parameters of a hyperplane that separate data points of different class labels with an error margin. Since A is a regression task, we change the objective to infer a hyperplane that can predict target outcomes within a tolerance margin [11], i.e., to minimize $\frac{1}{2}||\omega^2||$ subject to $|F - \omega.\mathcal{H} - b| \leq \epsilon$.

3) Tree Regressor (TR). The algorithm uses an ensemble method where it trains a number of regression trees in parallel [27]. At the inference time, the output is the mean of predicted outcomes from each regression tree.
4) XGBoost (XGB). This algorithm is similar to the Random Forest Regressor, but rather than an ensemble method, it uses tree boosting techniques [17] that start from a single leaf and iteratively add branches to the tree until the performance is not improving. XGBoost [10] is a special tree boosting algorithm that scales the training process via sparse data analysis and approximating training. XGBoost has been shown to significantly outperform other tree boosting techniques [4].

Our goal is to compare the performance of different hypothesis classes of HP prediction algorithms. If feasible, the models can be used as a part of data-driven software developments to predict the fairness of a selected HP without requiring a full training cycle. *In summary, our goal is to study the feasibility of training regression models to predict the fairness of ML hyperparameters in two levels: i) fixed dataset over a protected attribute and ii) temporal distribution shifts between training and test data.*

5 EXPERIMENTS

We first give background on ML training algorithms and their hyperparameters. Then, we overview training datasets used to infer ML models and evaluate their fairness. Subsequently, we introduce ML prediction algorithms (regressors) that are used to learn the fairness models of hyperparameters. Finally, we discuss research questions and perform experiments to evaluate the performance of different algorithms under different circumstances.

ML Training Algorithms. We consider 5 different ML training algorithms provided by *scikit-learn*, a popular ML library. The following is an overview of these algorithms:
A) Decision Tree (DT) algorithm [33] is a supervised learning algorithm to infer tree classifiers. During training, the DT algorithms use information theoretic measures, known as impurity, to decide which feature (node) and values (edges) to split upon. The final level of the tree contains leaf nodes that decide the outcome labels. We consider the following HPs: max_depth, min_samples_split, min_samples_leaf, and min_weight_fraction_leaf (**numerical**); and Criterion, splitter, and max_features (**Categorical**).
B) Logistic Regression (LR) infers the parameters of sigmoid functions to map data into a likelihood score of receiving a particular label. We use an implementation from scikit-learn [35] that has the following hyperparameters: tol, C, intercept_scaling, max_iteration, and l1_ratio (**numerical**); and solver, penalty norm, dual_prime formulation, fit_intercept, and multi_class (**Categorical**).
C) Support Vector Machine (SVM) learns a hyperplane of lower dimensions to separate the data of high-dimensional spaces on the basis of their class labels. The scikit-learn implementations of SVM algorithms [36] include the following hyperparameters: tol, C, and intercept_scaling (**numerical**); and penalty, loss, degree, fit_intercept, and class_weight (**Categorical**).
D) Random Forest (RF) is an ensemble learning algorithm that fits a number of trees and takes the expected outcomes of these trees to make a prediction. We used an implementation from scikit-learn with the following hyperparameters: max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, n_estimators, and max_samples (**numerical**); and criterion, max_features, oob_score, and warm_start (**Categorical**).
E) Discriminant Analysis (DA) defines a smooth classifier where the data fits a Gaussian prior to inferring a posterior for each class label [34]. The scikit-learn implementations of DA algorithms include the following hyperparameters: tol, and reg_param (**numerical**); and linear(0)_quadratic(1), solve_Linear, Shrinkage_Linear, component, store_covariance, and type_dataset (**Categorical**).

Fairness-Sensitive Training Datasets. We consider 4 fairness-critical datasets from the literature of ML fairness to infer ML classifiers using the ML training algorithms [1, 8, 18, 40]. Table 1 shows these datasets with different protected attributes that form 6 training tasks. Adult Census Income [12] (*Ce*) is a binary classification dataset that predicts whether an individual has an income over $50K$ a year. Sex and race are two protected attributes for this dataset. COMPAS Software (*Co*) [31] is a dataset in the criminal justice domain that classifies defendants into high- or low-risks for re-offending (i.e., risk of recidivism). We consider sex and race as the protected attributes. German credit data (*Cr*) [13] predicts whether an individual might have bad credit in assessing a loan default. Sex is the protected attribute. Bank Marketing [14] is a dataset in the advertisement applications to predict the likelihood that an individual will be a subscriber.

Training ML prediction algorithms. We use Parfait-ML [40] to generate a dataset of HP values (as feature variables) and AOD fairness values (as the target variable). Given a ML training algorithm (e.g., decision tree) and a fairness-sensitive dataset (e.g., Adult Census); we run Parfait-ML for four hours and record all the HPs as well as the AOD fairness of the corresponding ML model. Given the HP traces of each ML training algorithm and fairness-sensitive training datasets; we leverage the ML prediction algorithms to train ML regressors that can accurately predict the fairness of a HP configuration. To train those models, we did standard hyperparamter tuning to obtain the following predictive models:
1. Deep Neural Network (DNN). We consider prevalent deep neural network architecture from the fairness literature with 4 hidden layers, each with 32 neurons [16, 29, 47]. We implemented DNN using the TensorFlow library, version 2.14.0, where we used "mean squared error" (MSE) as the loss function and Adam for optimization. We infer the parameter of DNN after 50 epochs (batch size of 64). *2. Support Vector Regression* (SVR). We consider different classes of SVR for learning fairness functions. After initial experimentation and fine-tuning over SVR, LinearSVR, and NuSVR algorithms; we consider NuSVR as the core SVR algorithm from scikit-learn, version 1.2.2, with the following configuration: "rbf" as the kernel, "auto" as gamma for kernel coefficient, and 10,000 for the maximum number of iterations.
3. Tree Regressor (TR). We consider the implementation of random tree regressors from scikit-learn, version 1.2.2, where after initial experiments, we set the algorithm with 100 tree estimators each with a maximum depth of 35 nodes throughout the experiments.
4. XGBoost (XGB) We consider the implementations from XGBoost library [10], using version 2.0.2. After initial exploration, we set the max. depth of each tree to 30.

Table 1: Fairness-sensitive datasets used as the training of ML algorithms.

Dataset	|Instances|	|Features|	Protected Groups		Outcome Label	
			Group1	*Group2*	*Label 1*	*Label 0*
Adult *Census* Income (Ce)	48, 842	14	Sex-Male	Sex-Female	High Income	Low Income
			Race-White	Race-Non White		
Compas Software (Co)	7, 214	28	Sex-Male	Sex-Female	Did not Reoffend	Reoffend
			Race-Caucasian	Race-Non Caucasian		
German *Credit* (Cr)	1, 000	20	Sex-Male	Sex-Female	Good Credit	Bad Credit
Bank Marketing (Ba)	45, 211	17	Age-Young	Age-Old	Subscriber	Non-subscriber

Evaluation of the HP Prediction Algorithms. Our next goal is to evaluate the accuracy of the model in predicting the AOD bias of HPs. The first metric is the loss function, which calculates the fitness of our predictions compared to the actual AODs. This loss function is commonly Mean Square Error (MSE).

To evaluate the accuracy of our models over the testing data, we use three primary metrics: the Root Mean Square Error (RMSE), Relative Root Mean Square Error (Relative RMSE), and Coefficient of Determination (R^2). Since the metrics are highly correlated, we will focus on R^2 as our main metric. Let n be the number of (testing) samples, $\hat{AOD}_i$ be the predicted fairness from the ML prediction algorithm, AOD_i be the observable (ground truth) fairness for the i-th data point, and $\overline{AOD} = \frac{\sum_{i=1}^{n}(AOD_i)}{n}$ be the average of the AOD values.

- **Relative RMSE.** The Relative Root Mean Square Error is equal to the RMSE divided by the average $\bar{AOD}$ value of our dataset (lower values show a better fit):

$$Relative\ RMSE = \frac{1}{\overline{AOD}} * \sqrt{\frac{\sum_{i=1}^{n}(\hat{AOD}_i - AOD_i)^2}{n}}$$

- **The coefficient of determination** (R^2). This is the main evaluation metric in our experiments. This statistic measures how well the data fits our trained ML prediction model. The following is the formula for the R^2 (higher values show a better fit):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(AOD_i - \hat{AOD}_i)^2}{\sum_{i=1}^{n}(AOD_i - \overline{AOD})^2}$$

  This value can normally range from 0 to 1: if the R^2 value is 1, then we have perfectly predicted the output values whereas a lower value casts doubts on the validity of the trained models. In many practical situations, we depend on a single threshold that informs us if our R^2 value is reasonable. We report R^2 values above 0.95, 0.8, and 0.5; and consider values below 0.0 as an invalid prediction model.

Baseline Model. We also consider a basic simple prediction model to understand the precision of models. In particular, we use the average AOD of all samples, i.e., $\overline{AOD}$ to predict AOD of any HP values. We expected that our trained ML prediction models perform (significantly) better than this naive predictor.

Research Questions. We pose the following research questions:

- **RQ1.** Can we accurately predict the fairness of ML hyperparameters? What is the performance of different classes of predictors?
- **RQ2.** Can we predict the fairness of ML hyperparameters under temporal distribution shift of dataset?

Technical Details. We implemented our experiments using a cloud server machine with Intel(R) Xeon(R) CPU @ 2.20ghz. The system has 12.7 GB RAM, and the disk space is 225.8 GB. Throughout this paper, we repeated experiments 10 times each with different random seeds. We report both the average and the standard deviation over 10 repeated experiments. **The replication package is available at** link.

RQ1: the relationship between hyperparameters and fairness (fixed dataset)

To study the RQ1, we use 30 different benchmarks: ML hyperparameters of 5 training algorithms, trained over 6 different fairness-sensitive tasks (e.g., Decision Tree over Census dataset with Sex; Decision Tree over Census with Race, etc.).

Table 2 presents the performance of various ML prediction algorithms (DNN, SVR, TR, and XGB). These algorithms are assessed based on their ability to learn the relationship between hyperparameters and fairness across the 30 scenarios. For each ML prediction algorithm, we evaluate its performance using three key metrics: Root Mean Square Error (RMSE), Relative RMSE, and the coefficient of determination (R^2). Additionally, we compare these results with baseline performance, specifically in terms of Relative RMSE. It's important to note that an ML prediction algorithm is not deemed competitive if its Relative RMSE exceeds that of the baseline. Furthermore, in our assessment criteria, any R^2 values below 0.5 are not considered competitive.

In Table 2,we present the results of our analysis, which include both the average values and standard deviations from 10 repeated experiments for each of the 30 benchmarks. In our experiments, we use unique splits of training and testing fairness traces with a ratio of 80% training and 20% testing ratio. The results are reported, with the standard deviations enclosed in parentheses alongside the averages for each benchmark and ML prediction algorithm. We compare the results based on these averages and the range of two standard deviations; and deem a method outperforms another if the average is not within two standard deviations.

We highlight the best R^2 result as well as those within 2 standard deviations of the best result. For example, in the Random Forest case, trained over the Credit dataset with Sex as the protected attribute, the Tree Regressor algorithm has outperformed all other methods in predicting the AOD fairness of Random Forest hyperparameters whereas in the benchmark with the SVM algorithm trained over Census with Sex; DNN, Tree Regressor, and XGBoost have all achieved similar performance to predict AOD fairness of SVM

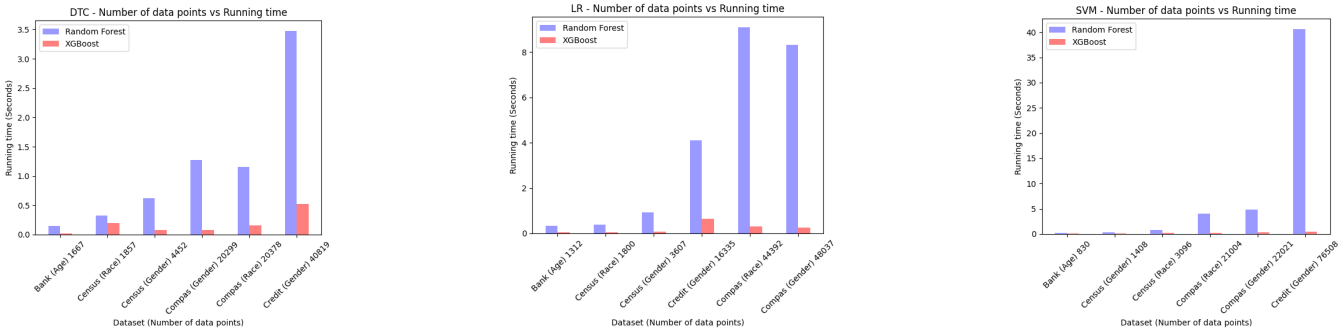

Figure 1: Computation Complexities of Tree Regressor vs. XGBoost. Each plot shows one ML training algorithm (Decision Tree Classifier in the left, Logistic Regression in the center, and Support Vector Machine in the right).

Table 2: (RQ1) Comparisons between different ML prediction algorithms (DNN, SVR, TR, and XGB) in learning fairness of hyperparameters for a given ML training algorithm, dataset, and protected attribute. The numbers show the average of 10 repeated experiments wherein the standard deviations are reported in the parenthesis (NV shows $R^2 \leq 0.0$).

Algorithm	Dataset	Prot.	Baseline	Deep Neural Network		Support Vector Regression		Tree Regressor		XGBoost	
				Rel. RMSE	R^2	Rel. RMSE	R^2	Rel. RMSE	R^2	Rel. RMSE	R^2
Decision Tree	Census	sex	0.444 (0.013)	0.366 (0.018)	0.316 (0.057)	0.387 (0.012)	0.237 (0.034)	0.157 (0.014)	**0.875 (0.018)**	0.164 (0.012)	**0.863 (0.016)**
		race	0.610 (0.029)	0.607 (0.032)	0.003 (0.099)	0.624 (0.037)	NV	0.233 (0.033)	**0.849 (0.045)**	0.230 (0.029)	**0.853 (0.042)**
	Compas	sex	0.273 (0.004)	0.115 (0.014)	0.820 (0.050)	0.239 (0.035)	0.219 (0.234)	0.057 (0.003)	**0.956 (0.005)**	0.058 (0.003)	**0.954 (0.005)**
		race	0.347 (0.003)	0.141 (0.013)	0.832 (0.031)	0.278 (0.012)	0.358 (0.055)	0.068 (0.004)	**0.962 (0.005)**	0.066 (0.004)	**0.964 (0.005)**
	Credit	sex	2.219 (0.036)	1.260 (0.077)	0.676 (0.045)	2.317 (0.185)	NV	0.694 (0.026)	**0.902 (0.008)**	0.712 (0.031)	**0.897 (0.010)**
	Bank	age	4.008 (0.658)	4.034 (0.621)	NV	3.848 (0.675)	0.076 (0.073)	2.168 (0.533)	**0.706 (0.070)**	2.039 (0.466)	**0.734 (0.084)**
Logistic Regression	Census	sex	0.432 (0.008)	0.317 (0.072)	0.428 (0.304)	0.217 (0.008)	0.747 (0.016)	0.053 (0.008)	**0.984 (0.005)**	0.053 (0.008)	**0.985 (0.005)**
		race	0.197 (0.010)	0.163 (0.037)	0.285 (0.333)	0.130 (0.016)	0.560 (0.086)	0.044 (0.023)	**0.940 (0.059)**	0.051 (0.026)	**0.920 (0.076)**
	Compas	sex	0.377 (0.002)	0.072 (0.016)	0.962 (0.019)	0.266 (0.005)	0.502 (0.018)	0.008 (0.001)	**1.000 (0.000)**	0.011 (0.001)	0.999 (0.000)
		race	0.372 (0.004)	0.128 (0.039)	0.871 (0.084)	0.318 (0.008)	0.267 (0.045)	0.007 (0.002)	**1.000 (0.000)**	0.011 (0.002)	0.999 (0.000)
	Credit	sex	0.400 (0.001)	0.203 (0.022)	0.741 (0.064)	0.238 (0.005)	0.645 (0.017)	0.113 (0.004)	**0.920 (0.006)**	0.116 (0.004)	**0.916 (0.006)**
	Bank	age	0.679 (0.014)	0.469 (0.049)	0.515 (0.097)	0.402 (0.021)	0.646 (0.043)	0.172 (0.018)	**0.935 (0.014)**	0.182 (0.024)	**0.926 (0.020)**
Random Forest	Census	sex	0.220 (0.011)	1.375 (1.335)	NV	0.210 (0.013)	0.081 (0.054)	0.055 (0.006)	**0.935 (0.015)**	0.057 (0.005)	**0.931 (0.012)**
		race	0.160 (0.012)	2.547 (1.381)	NV	0.162 (0.012)	NV	0.093 (0.007)	**0.657 (0.055)**	0.095 (0.003)	**0.637 (0.066)**
	Compas	sex	0.046 (0.017)	0.074 (0.022)	NV	0.050 (0.009)	NV	0.026 (0.007)	**0.567 (0.282)**	0.022 (0.008)	**0.708 (0.159)**
		race	0.042 (0.007)	0.130 (0.203)	NV	0.049 (0.005)	NV	0.019 (0.004)	**0.807 (0.036)**	0.021 (0.004)	**0.739 (0.106)**
	Credit	sex	0.625 (0.007)	0.536 (0.086)	0.246 (0.261)	0.614 (0.008)	0.034 (0.018)	0.317 (0.004)	**0.743 (0.006)**	0.338 (0.004)	0.708 (0.008)
	Bank	age	0.662 (0.118)	27.331 (25.2)	NV	0.518 (0.061)	0.251 (0.466)	0.331 (0.054)	**0.710 (0.126)**	0.350 (0.064)	**0.677 (0.139)**
SVM	Census	sex	0.828 (0.009)	0.275 (0.022)	**0.889 (0.018)**	0.293 (0.023)	0.874 (0.022)	0.244 (0.027)	**0.912 (0.020)**	0.267 (0.029)	**0.895 (0.025)**
		race	0.733 (0.004)	0.378 (0.021)	**0.733 (0.029)**	0.399 (0.025)	0.702 (0.038)	0.355 (0.021)	**0.764 (0.028)**	0.381 (0.023)	**0.729 (0.032)**
	Compas	sex	0.178 (0.001)	0.088 (0.016)	0.751 (0.106)	0.162 (0.029)	0.151 (0.314)	0.066 (0.002)	**0.864 (0.007)**	0.069 (0.002)	**0.850 (0.008)**
		race	0.069 (0.003)	0.063 (0.004)	0.161 (0.072)	0.129 (0.031)	NV	0.049 (0.003)	0.496 (0.044)	0.052 (0.003)	0.431 (0.050)
	Credit	sex	0.449 (0.001)	0.043 (0.005)	0.991 (0.003)	0.181 (0.016)	0.835 (0.028)	0.018 (0.001)	**0.998 (0.000)**	0.021 (0.002)	**0.998 (0.000)**
	Bank	age	0.782 (0.052)	0.684 (0.035)	0.223 (0.089)	0.645 (0.039)	0.312 (0.070)	0.571 (0.050)	0.462 (0.062)	0.615 (0.039)	0.373 (0.071)
Discriminant Analysis	Census	sex	0.760 (0.005)	0.052 (0.009)	0.995 (0.002)	0.144 (0.006)	0.964 (0.003)	0.002 (0.000)	**1.000 (0.000)**	0.004 (0.000)	**1.000 (0.000)**
		race	0.325 (0.012)	0.031 (0.007)	0.991 (0.004)	0.153 (0.015)	0.777 (0.028)	0.003 (0.001)	**1.000 (0.000)**	0.006 (0.001)	**1.000 (0.000)**
	Compas	sex	0.106 (0.002)	0.023 (0.013)	0.937 (0.090)	0.022 (0.002)	0.955 (0.008)	0.001 (0.001)	**1.000 (0.000)**	0.002 (0.000)	0.999 (0.000)
		race	0.139 (0.004)	0.024 (0.006)	0.969 (0.014)	0.023 (0.002)	0.972 (0.006)	0.001 (0.001)	**1.000 (0.000)**	0.002 (0.000)	**1.000 (0.000)**
	Credit	sex	0.944 (0.005)	0.097 (0.009)	0.989 (0.002)	0.288 (0.007)	0.907 (0.005)	0.006 (0.003)	**1.000 (0.000)**	0.005 (0.003)	**1.000 (0.000)**
	Bank	age	0.369 (0.016)	0.186 (0.026)	0.742 (0.062)	0.260 (0.019)	0.499 (0.066)	0.023 (0.005)	**0.996 (0.002)**	0.027 (0.004)	**0.994 (0.001)**

hyperparameters. Overall, we highlight TR algorithms in 93.3% (28 out of 30 cases); XGB in 80% (24 out of 30); DNN in 6.6% (2 out of 30); and SVR in 0.0% (0 out of 30). Note that in some cases the R^2 measure is the same on both algorithms (e.g., see Discriminant Analysis benchmark with Compas dataset and Race); but if we look at the Relative RMSE measure we can observe that a smaller value is better. Therefore we can differentiate the best algorithm per dataset. To precisely compare different ML prediction algorithms, we report the many cases when we have R^2 values greater than a specific threshold.

How many R^2 values do we have more than 0.95? Both Tree Regressor and XGBoost have 12/30 (40%) values exceeding this threshold noting that the datasets in which R^2 values exceed 0.95 are the same datasets for both algorithms. For the Deep Neural Network model, we have 6/30 (20%) cases exceeding this threshold including half of the previous datasets in TR and XGB. Finally, the SVR method only has 3/30 (10%) values that have the R^2 greater than 0.95.

How many R^2 values do we have more than 0.80? Tree Regressor exceeds this threshold in 22/30 (73%) datasets, and XGBoost is very close to it with 21/30 (70%) datasets meeting the 0.8 threshold. For the Deep Neural Network, 11/30 (37%), close to half of the previous datasets meet the R^2 threshold. Finally, the SVR method has 6/30 (20%) values that are greater than 0.8.

Table 3: (RQ1) The performance of Tree Regressors when the EOD fairness, instead of AOD, is used. The red fonts show benchmarks when R^2 is degraded whereas the bold fonts show ones that R^2 is increased.

Algorithm	Dataset	Prot.	Base.	Tree Regressor
				R^2
Decision Tree	Census	sex	0.518 (0.018)	0.882 (0.015)
		race	0.691 (0.034)	0.868 (0.038)
	Compas	sex	11.910 (1.950)	0.674 (0.100)
		race	13.114 (2.590)	0.560 (0.109)
	Credit	sex	2.309 (0.042)	0.897 (0.008)
	Bank	age	4.083 (0.696)	0.689 (0.074)
Logistic Regression	Census	sex	0.480 (0.013)	0.983 (0.007)
		race	0.191 (0.009)	0.930 (0.070)
	Compas	sex	1.198 (0.009)	0.999 (0.000)
		race	1.913 (0.038)	0.999 (0.000)
	Credit	sex	0.489 (0.002)	0.925 (0.006)
	Bank	age	0.735 (0.013)	0.933 (0.014)
Random Forest	Census	sex	0.322 (0.014)	**0.955 (0.010)**
		race	0.211 (0.014)	**0.711 (0.044)**
	Compas	sex	18.002 (7.909)	0.553 (0.339)
		race	19.452 (5.010)	0.823 (0.047)
	Credit	sex	0.748 (0.011)	0.608 (0.012)
	Bank	age	0.761 (0.123)	0.692 (0.104)
SVM	Census	sex	0.744 (0.013)	0.869 (0.018)
		race	0.705 (0.015)	0.602 (0.048)
	Compas	sex	1.185 (0.018)	0.813 (0.011)
		race	1.708 (0.071)	**0.539 (0.035)**
	Credit	sex	0.460 (0.001)	0.998 (0.000)
	Bank	age	0.909 (0.058)	**0.495 (0.067)**
Discriminant Analysis	Census	sex	0.889 (0.007)	1.000 (0.000)
		race	0.338 (0.010)	1.000 (0.000)
	Compas	sex	11.644 (1.304)	0.980 (0.013)
		race	13.242 (1.015)	0.992 (0.015)
	Credit	sex	1.000 (0.008)	1.000 (0.000)
	Bank	age	0.519 (0.026)	0.997 (0.001)

Comparing the ML prediction algorithms using a different fairness notion. We also perform experiments to understand the efficacy of algorithms to predict a different notion of fairness than AOD. Specifically, we consider the equal opportunity difference (EOD), the true positive rate differences between two protected groups, as the target variable of our training tasks. Since Tree Regressor has achieved the highest performance; we are only training Tree Regressors models to predict EOD fairness.

Table 3 shows the results of experiments. In 19/30 cases (63.3%); we observe that TR shows a similar performance when used to predict EOD, instead of AOD. We observe that in 7/30 cases (23.3%), the performance of TR (based on R^2) is degraded: TR predicts AOD more accurately than EOD. When we closely examine these cases, we observe that 4 cases out of 7 involve the COMPAS dataset. We revisited the training datasets for COMPAS and noted that the difference in true positive rates between two groups are zero for many examples, while this is not the case for false positive rates that contribute to AOD. We conjecture that this is the main reason behind the difficulty in learning EOD, compared to AOD. In 4/30 cases (13.3%), TR predicts EOD more accurately than AOD.

Comparing the computational complexity of ML prediction algorithms. While we observe that Tree Regressor slightly outperforms XGBoost in the accuracy of predicting fairness of ML hyperparameters; we observed that XGBoost is significantly faster, meaning that it trains ML prediction models in relatively less running time. Figure 1 shows the computation times of training with TR vs. XGB over three ML training algorithms (DT, LR, and SVM), noting that the results are similar for the other two algorithms (RF and DA). In each plot (corresponds to one ML training algorithm), we have different datasets with a protected attribute and the size of fairness traces used to train the regression models. As the size grows, the XGBoost algorithm significantly scales better.

Performance of fairness models across multiple releases of the same dataset. We aim to evaluate the performance of our fairness models across various releases of the same dataset. Specifically, we focus on the Census dataset, collecting its 2014, 2015, and 2018 versions. Each release is treated as an individual dataset, and we apply the experimental setup from RQ1 to these experiments.

The results of this analysis, detailed in Table 4 follow the same structure as in Table 2. When assessing the efficacy of predicting AOD from hyperparameters, the Tree Regressor exhibited better performance, achieving the best performance in 100% cases (30 out of 30 cases). XGB also demonstrated similar performance with 97% in 29 out of 30 cases. Conversely, both DNN and SVR failed to achieve notable performance in any instance. The results suggest that DNN, SVR, TR, and XGB could achieve a R^2 score over 0.95 in 10% (3 out of 30), 10% (3 out of 30), 47% (14 out of 30 cases), and 40% (12 out of 30) respectively while considering cases with R^2 0.8 and higher, these percentages are 33% (11 out of 30), 30% (9 out of 30), 93% (28 out of 30 cases), and 93% (28 out of 30).

Comparing the performance on the protected attributes Sex and Race, a clear preference for the Sex attribute emerges in our analysis. Specifically, 60% (9 out of 15) of the datasets showed better performance with Sex as the protected attribute. Breaking it down by algorithm: all three Decision Tree datasets (2014, 2015, and 2018) performed better with Race, while Logistic Regression and Discriminant Analysis datasets uniformly favored Sex (3/3 in both cases). In the Random Forest datasets, one out of three showed better performance with Sex, and in the SVM datasets, two out of three favored Sex. This trend indicates a higher value for the Sex attribute in the majority of the cases.

Answer RQ1: Our experiences show that Tree Regressor and XGBoost achieve the highest accuracy in predicting AOD fairness of HPs, with Tree Regressor slightly outperforming XGBoost. However, XGBoost scales better to larger datasets, and learns the fairness functions faster. When a different fairness notion is used, we observe that Tree Regressor achieved similar or better performance. We also find that the results are consistent in modeling the fairness of HPs across multiple releases of the same dataset where TR and XGB accurately model fairness in 100% and 97% cases.

RQ2: the relationship between hyperparameters and fairness over temporal data distribution shift

In this study, we shift our focus to understanding the robustness of our predictive models in predicting the fairness of HPs under temporal distribution shifts. This analysis helps us to validate if our predictive models, which estimate the fairness of ML systems based on specific hyperparameter configurations, can maintain their accuracy over time as data characteristics change and ML system evolves. It is essential to assess whether the fairness of these algorithms remains consistent over time, and identify if there are systematic biases introduced due to changes in the underlying

Table 4: Different versions of income census datasets that are released in 2014, 2015, and 2018.

Algorithm	Dataset	Prot.	Baseline	DNN	SVM	TR	XGB
				R^2	R^2	R^2	R^2
Decision Tree	2014	Sex	0.672 (0.030)	0.372 (0.043)	0.287 (0.033)	**0.845 (0.043)**	**0.835 (0.045)**
	2015		0.651 (0.022)	0.649 (0.041)	0.079 (0.020)	**0.933 (0.015)**	**0.923 (0.019)**
	2018		0.979 (0.049)	0.657 (0.033)	0.279 (0.033)	**0.932 (0.015)**	**0.932 (0.015)**
	2014	Race	0.374 (0.007)	0.386 (0.081)	0.108 (0.043)	**0.940 (0.015)**	**0.936 (0.011)**
	2015		0.481 (0.008)	0.664 (0.038)	0.611 (0.021)	**0.940 (0.012)**	**0.937 (0.018)**
	2018		0.365 (0.015)	0.454 (0.084)	0.166 (0.026)	**0.936 (0.012)**	0.907 (0.017)
Logistic Regression	2014	Sex	0.706 (0.015)	0.946 (0.032)	0.930 (0.008)	**0.988 (0.006)**	**0.985 (0.007)**
	2015		0.616 (0.008)	0.946 (0.017)	0.933 (0.014)	**0.992 (0.002)**	**0.992 (0.002)**
	2018		0.813 (0.036)	0.910 (0.038)	0.801 (0.038)	**0.989 (0.008)**	**0.990 (0.007)**
	2014	Race	0.144 (0.004)	NV	0.371 (0.062)	**0.830 (0.016)**	**0.807 (0.027)**
	2015		0.136 (0.008)	0.382 (0.221)	0.688 (0.070)	**0.890 (0.031)**	**0.898 (0.026)**
	2018		0.161 (0.005)	0.265 (0.171)	0.445 (0.048)	**0.719 (0.044)**	**0.665 (0.055)**
Random Forest	2014	Sex	0.098 (0.008)	NV	0.435 (0.100)	**0.865 (0.028)**	**0.862 (0.038)**
	2015		0.197 (0.006)	0.105 (0.425)	0.708 (0.046)	**0.967 (0.020)**	**0.966 (0.014)**
	2018		0.188 (0.006)	0.102 (0.227)	0.631 (0.059)	**0.918 (0.041)**	**0.925 (0.040)**
	2014	Race	0.170 (0.011)	NV	0.688 (0.040)	**0.951 (0.012)**	**0.940 (0.025)**
	2015		0.238 (0.009)	0.449 (0.133)	0.588 (0.087)	**0.968 (0.004)**	**0.966 (0.009)**
	2018		0.151 (0.016)	NV	0.064 (0.367)	**0.873 (0.082)**	**0.863 (0.109)**
SVM	2014	Sex	0.632 (0.019)	0.813 (0.027)	0.770 (0.031)	**0.962 (0.008)**	**0.954 (0.009)**
	2015		0.546 (0.014)	0.807 (0.029)	0.792 (0.031)	**0.956 (0.010)**	**0.945 (0.016)**
	2018		0.937 (0.026)	0.654 (0.024)	0.570 (0.027)	**0.813 (0.020)**	**0.800 (0.037)**
	2014	Race	0.206 (0.003)	0.537 (0.048)	0.554 (0.039)	**0.750 (0.032)**	**0.707 (0.041)**
	2015		0.257 (0.007)	0.714 (0.035)	0.700 (0.054)	**0.870 (0.027)**	**0.853 (0.041)**
	2018		0.691 (0.017)	0.699 (0.042)	0.694 (0.052)	**0.841 (0.026)**	**0.830 (0.029)**
Discriminant Analysis	2014	Sex	0.730 (0.005)	0.996 (0.001)	0.972 (0.004)	**1.000 (0.001)**	**1.000 (0.001)**
	2015		0.741 (0.007)	0.995 (0.002)	0.984 (0.002)	**1.000 (0.000)**	**1.000 (0.000)**
	2018		0.706 (0.006)	0.995 (0.003)	0.968 (0.009)	**0.999 (0.001)**	**1.000 (0.001)**
	2014	Race	0.102 (0.003)	0.865 (0.024)	0.879 (0.007)	**0.997 (0.002)**	**0.996 (0.001)**
	2015		0.131 (0.002)	0.919 (0.012)	0.913 (0.009)	**0.999 (0.000)**	**0.999 (0.000)**
	2018		0.112 (0.005)	0.868 (0.041)	0.889 (0.012)	**0.999 (0.000)**	**0.999 (0.000)**

Table 5: One-year data shift: training with income census 2014 and testing with 2015 one.

Algorithm	Dataset	Prot.	Baseline	DNN	SVM	TR	XGB
				R^2	R^2	R^2	R^2
Decision Tree	Census 2014 → Census 2015	Sex	1.306 (0.036)	NV	NV	NV	NV
	Census 2015 → Census 2014		0.565 (0.006)	NV	NV	NV	NV
	Census 2014 → Census 2015	Race	0.523 (0.008)	NV	NV	NV	NV
	Census 2015 → Census 2014		0.355 (0.005)	NV	NV	NV	NV
Logistic Regression	Census 2014 → Census 2015	Sex	0.690 (0.012)	0.933 (0.021)	0.861 (0.012)	**0.979 (0.002)**	**0.978 (0.002)**
	Census 2015 → Census 2014		0.645 (0.007)	0.912 (0.024)	0.802 (0.016)	**0.971 (0.010)**	**0.968 (0.009)**
	Census 2014 → Census 2015	Race	0.138 (0.008)	0.082 (0.323)	NV	**0.617 (0.037)**	0.523 (0.062)
	Census 2015 → Census 2014		0.143 (0.004)	NV	0.235 (0.055)	**0.519 (0.041)**	**0.510 (0.045)**
Random Forest	Census 2014 → Census 2015	Sex	0.335 (0.007)	NV	NV	NV	NV
	Census 2015 → Census 2014		0.196 (0.008)	NV	NV	NV	NV
	Census 2014 → Census 2015	Race	0.237 (0.010)	NV	NV	0.385 (0.066)	0.312 (0.076)
	Census 2015 → Census 2014		0.213 (0.015)	NV	NV	NV	NV
SVM	Census 2014 → Census 2015	Sex	0.639 (0.018)	**0.590 (0.092)**	0.422 (0.053)	**0.538 (0.067)**	**0.526 (0.071)**
	Census 2015 → Census 2014		0.568 (0.017)	0.497 (0.068)	0.332 (0.086)	**0.679 (0.036)**	**0.636 (0.051)**
	Census 2014 → Census 2015	Race	0.260 (0.007)	0.448 (0.044)	0.278 (0.086)	0.414 (0.052)	0.346 (0.060)
	Census 2015 → Census 2014		0.204 (0.003)	NV	NV	NV	NV
Discriminant Analysis	Census 2014 → Census 2015	Sex	0.714 (0.006)	0.983 (0.004)	0.971 (0.003)	**0.989 (0.000)**	**0.988 (0.000)**
	Census 2015 → Census 2014		0.760 (0.007)	0.975 (0.005)	0.959 (0.004)	**0.987 (0.001)**	**0.987 (0.001)**
	Census 2014 → Census 2015	Race	0.130 (0.002)	**0.513 (0.055)**	**0.613 (0.053)**	**0.583 (0.024)**	**0.585 (0.027)**
	Census 2015 → Census 2014		0.105 (0.003)	0.262 (0.079)	0.306 (0.044)	0.368 (0.032)	0.376 (0.032)

data distributions that degrade the performance of our predictive models. To do so, we focus on several Census dataset released in 2014, 2015, and 2018. This choice was motivated by the need to understand how the data may reflect social and demographic shifts over these years and how it can impact the predictability of fairness models based on a hyperparameter setting. We utilize three releases of the Census dataset (2014, 2015, and 2018) and two protected attributes (Race and Sex) to analyze different temporal shifts. Analyzing these scenarios aims to give insights into the robustness of fairness models facing distribution changes in the test data. To do so, we learn a fairness model on one distinct release of Census and then evaluate the performance of this model by testing it on another distinct release.

Evaluating the robustness of fairness models in one year shift. In this experiment, we consider two consequent years (2014, and 2015) of Census dataset releases to evaluate the robustness of fairness models in minimal distribution shifts. Table 5 shows the results of this analysis. The structure of this table is similar to Table 2 except for the column *dataset* that shows the scenario of temporal distribution in the dataset. For example, Census 2014 → Census 2015 in column *dataset* shows that we first train the fairness model of different hyperparameter configurations on the 2014 release of the Census dataset then the 2015 release is used as the prediction dataset to evaluate the performance of the fairness model. The results in Table 5 show that the Tree Regressor followed by XGBoost outperforms other algorithms. However, we observed a significant reduction in cases with R^2 greater than 0.95 (only 4 out of 20 cases while in 40% (8 out of 20 cases) the prediction models failed to achieve positive R^2 scores when training and testing data distributions differ.

The results also show that when sex is the protected attribute, the Tree Regressor and XGBoost fairness models trained on the fairness traces of Logistic Regression and Discriminant Analysis robust against a one-year distribution shift in data achieving a minimum performance of 97% in those cases.

Evaluating the robustness of fairness models for a longer temporal shifts. To assess the robustness of our fairness models under conditions of increased temporal distribution shift, our experiment extends the time period from one year to three and four years. We utilize three versions of the Census datasets from 2014, 2015, and 2018, focusing on sex and race as the protected attributes. The goal of this experiment is to predict the fairness of the 2018 Census dataset using fairness models that were trained on earlier versions, specifically the 2014 and 2015 Census datasets.

Table 6: Three to four years data shift: training with census income 2014/15 and testing with 2018.

Algorithm	Dataset	Prot.	Baseline	DNN	SVM	TR	XGB
				R^2	R^2	R^2	R^2
Decision Tree	Census 2014 → Census 2018	Sex	0.891 (0.032)	NV	NV	NV	NV
	Census 2015 → Census 2018		0.694 (0.004)	NV	NV	NV	NV
	Census 2014 → Census 2018	Race	0.370 (0.015)	NV	NV	NV	NV
	Census 2015 → Census 2018		0.348 (0.014)	NV	NV	NV	NV
Logistic Regression	Census 2014 → Census 2018	Sex	0.740 (0.019)	0.933 (0.031)	0.810 (0.027)	**0.961 (0.010)**	**0.958 (0.011)**
	Census 2015 → Census 2018		0.691 (0.017)	0.913 (0.027)	0.768 (0.036)	**0.954 (0.011)**	**0.952 (0.012)**
	Census 2014 → Census 2018	Race	0.305 (0.013)	NV	NV	NV	NV
	Census 2015 → Census 2018		0.292 (0.012)	NV	NV	NV	NV
Random Forest	Census 2014 → Census 2018	Sex	0.319 (0.018)	NV	NV	NV	NV
	Census 2015 → Census 2018		0.188 (0.006)	0.004 (0.181)	NV	**0.648 (0.094)**	**0.658 (0.061)**
	Census 2014 → Census 2018	Race	0.211 (0.013)	NV	NV	NV	NV
	Census 2015 → Census 2018		0.303 (0.014)	NV	NV	NV	NV
SVM	Census 2014 → Census 2018	Sex	0.919 (0.018)	NV	NV	NV	NV
	Census 2015 → Census 2018		0.817 (0.010)	NV	NV	0.069 (0.042)	0.103 (0.040)
	Census 2014 → Census 2018	Race	0.658 (0.013)	0.011 (0.052)	NV	0.026 (0.031)	0.108 (0.039)
	Census 2015 → Census 2018		0.651 (0.014)	0.085 (0.053)	0.076 (0.028)	0.024 (0.050)	NV
Discriminant Analysis	Census 2014 → Census 2018	Sex	0.713 (0.007)	0.986 (0.003)	0.968 (0.008)	**0.990 (0.001)**	**0.990 (0.001)**
	Census 2015 → Census 2018		0.743 (0.008)	0.970 (0.008)	0.960 (0.009)	**0.985 (0.002)**	**0.985 (0.002)**
	Census 2014 → Census 2018	Race	0.247 (0.005)	NV	NV	NV	NV
	Census 2015 → Census 2018		0.265 (0.006)	NV	NV	NV	NV

This approach allows us to evaluate how robust our models are against changes over time and maintain their accuracy in predicting fairness in different temporal contexts. Table 6 shows the results of these experiments, and the structure of the table is similar to Table 5. The results suggest that increasing the time interval of the distribution shift further increases the number of cases that failed to achieve positive R^2 scores with 60%(12 out of 20 cases) compared to 40% in one-year shifts. However, Tree Regressor and XGBoost fairness models showed strong robustness in Logistic Regression and Discriminant Analysis cases when sex is the protected attribute. Overall, Tree Regressor and XGBoost achieved the best performance in 4 out of 20 cases (20%) with the R^2 scores higher than 0.95. When comparing the results in Table 5 to the results of one-year shift in Table 5 over the Tree Regressor, Logistic Regression benchmark with Census 2014 → Census 2015 achieved R^2 of 0.979 while it achieved R^2 of 0.961 with only 0.018 loss in its performance under a four year shift (Census 2014 → Census 2018).

Answer RQ2: Our results show that temporal distribution shifts can reduce the performance of our prediction models. We observed that not all models are robust to shifts. However, in 20% of the cases, the HP prediction models remain robust against the temporal shift even on datasets with four years of gaps between training and test data. Specifically, we found that the HPs of Logistic Regression and Discriminant Analysis are more robust to the distribution shift when the protected attribute is sex.

6 DISCUSSION

Limitation. The input dataset is arguably the main source of discrimination in data-driven software, however, it is known that the training process may amplify or suppress the present discriminatory instances [46]. In this work, we focus on the training process and study relationships between ML hyperparameters and fairness. Our approach can serve as a mitigation strategy to avoid biased hyperparameter configurations and pick promising ones for full training. However, our approach alone cannot eliminate all possible fairness bugs. Our approach also requires a diverse set of hyperparameters and their fairness characteristics using an automated fine-tuning algorithm. We rely on Parfait-ML [40] to generate such traces, but Parfait-ML relies on heuristics to generate a diverse set of configurations and is not guaranteed to always find interesting hyperparameters in a given time limit (4 hours in this work). In addition, we only use two group fairness metrics (*AOD* and *EOD*). These metrics do not consider the distribution of different groups and may deem software fair, while it might still harm a vulnerable community or minority group. In general, coming up with a suitable fairness definition is an open challenge.

Threat to Validity. To address the internal validity and ensure our finding does not lead to an invalid conclusion, we follow established guidelines [2] and repeat every experiment 10 times and report both the average and standard deviations. We deem a result significant if it does not overlap with any other methods within 2 standard deviations that imply 95% confidence in the results. However, this requirement might be conservative, instead, non-parametric methods and effect sizes can be used to alleviate the conservativeness of our comparisons [7].

To ensure that our results are generalizable and address external validity, we perform our experiments on the hyperparameter space of five popular ML training algorithm from scikit-learn library over six fairness-sensitive applications that have been widely used in the fairness literature. To learn the prediction model of ML hyperparameters for fairness, we used four algorithms from TensorFlow, XGBoost, and scikit-learn library. However, it is an open problem whether these libraries, algorithms, and applications are sufficiently representative and expressive to cover challenging fairness scenarios and train complex models.

Usage Vision. We envision software and AI engineers using our framework in the training process of data-driven (ML) software development. Specifically, in the cross-validation stage, the users can query the ML prediction models with a hyperparameter configuration and receive its prediction of fairness. This avoids the need for training the entire model with a biased hyperparameter, improves the efficiency of the training process, and saves computational resources. We left further study on the amounts of cost reductions as an interesting future work.

7 CONCLUSION

The ML hyperparameters play an influential and challenging role in the accuracy, robustness, and fairness of data-driven software. To understand and optimize tuning hyperparameters for fairness, we propose a data-driven framework based on regression methods to predict the fairness of ML hyperparameters. We showed that this task is feasible for a fixed dataset with a specific protected attribute; hence, it helps the developers save computational resources during the training by avoiding biased hyperparameters. We showed that

generalizations under (temporal) distribution shift are feasible, but such generalizations are limited to certain classes of ML algorithms, training datasets, and protected attributes.

There are multiple exciting future directions. One direction is to leverage multi-objective algorithms to learn the frontiers of accuracy vs. fairness in the ML hyperparameter space. Similarly, we can leverage such functions to learn the prediction models for multiple fairness metrics such as EOD, AOD, and statistical parity.

ACKNOWLEDGEMENTS

Tizpaz-Niari and Trivedi were partially supported by the NSF under grants CNS-2230060, CCF-2317206, and CCF-2317207.

REFERENCES

[1] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2019. Black Box Fairness Testing of Machine Learning Models *(ESEC/FSE 2019)*. 625–635. https://doi.org/10.1145/3338906.3338937

[2] Andrea Arcuri and Lionel Briand. 2014. A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verification and Reliability* (2014), 219–250. https://doi.org/10.1002/stvr.1486

[3] 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.

[4] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. 2021. A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review* 54 (2021), 1937–1967.

[5] Sumon Biswas and Hridesh Rajan. 2021. Fair Preprocessing: Towards Understanding Compositional Fairness of Data Transformers in Machine Learning Pipeline *(ESEC/FSE 2021)*. Association for Computing Machinery, New York, NY, USA, 981–993. https://doi.org/10.1145/3468264.3468536

[6] Ruth G Blumrosen. 1978. Wage discrimination, job segregation, and the title vii of the civil rights act of 1964. *U. Mich. JL Reform* 12 (1978), 397.

[7] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. 2021. Bias in machine learning software: Why? how? what to do? 429–440.

[8] Joymallya Chakraborty, Suvodeep Majumder, Zhe Yu, and Tim Menzies. 2020. Fairway: A Way to Build Fair ML Software *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 654–665. https://doi.org/10.1145/3368089.3409697

[9] Joymallya Chakraborty, Tianpei Xia, Fahmid M Fahid, and Tim Menzies. 2019. Software engineering for fairness: A case study with hyperparameter optimization. *arXiv preprint arXiv:1905.05786* (2019).

[10] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[11] Harris Drucker, Christopher J Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. 1996. Support vector regression machines. *Advances in neural information processing systems* 9 (1996).

[12] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/census+income

[13] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

[14] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/bank+marketing

[15] 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*. 214–226.

[16] Ming Fan, Wenying Wei, Wuxia Jin, Zijiang Yang, and Ting Liu. 2022. Explanation-Guided Fairness Testing through Genetic Algorithm *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 871–882. https://doi.org/10.1145/3510003.3510137

[17] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[18] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness Testing: Testing Software for Discrimination *(ESEC/FSE 2017)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3106237.3106277

[19] Usman Gohar, Sumon Biswas, and Hridesh Rajan. 2023. Towards Understanding Fairness and Its Composition in Ensemble Machine Learning *(ICSE '23)*. IEEE Press, 1533–1545. https://doi.org/10.1109/ICSE48619.2023.00133

[20] Home Credit Group. 2018. Home Credit Default Risk. https://www.kaggle.com/competitions/home-credit-default-risk/overview.

[21] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of Opportunity in Supervised Learning. In *NIPS*.

[22] David Ingold and Spencer Soper. 2016. Amazon Doesn't Consider the Race of Its Customers. Should It? https://www.bloomberg.com/graphics/2016-amazon-same-day/. Online.

[23] Surya Mattu Julia Angwin, Jeff Larson and Lauren Kirchne. 2021. Machine Bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Online.

[24] Arjun Kharpal. 2018. TECH Health care start-up says A.I. can diagnose patients better than humans can, doctors call that 'dubious'. *CNBC* (2018). https://www.cnbc.com/2018/06/28/babylon-claims-its-ai-can-diagnose-patients-better-than-doctors.html

[25] Max Kuhn, Kjell Johnson, et al. 2013. *Applied predictive modeling*. Vol. 26. Springer.

[26] Meelis Kull and Peter Flach. 2014. Patterns of dataset shift. In *First international workshop on learning over multiple contexts (LMCE) at ECML-PKDD*, Vol. 5.

[27] Wei-Yin Loh. 2011. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery* 1, 1 (2011), 14–23.

[28] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. 2019. Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing*. Elsevier, 293–312.

[29] Verya Monjezi, Ashutosh Trivedi, Gang Tan, and Saeid Tizpaz-Niari. 2023. Information-theoretic testing and debugging of fairness defects in deep neural networks. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1571–1582.

[30] Tho M Nguyen, Saeid Tizpaz-Niari, and Vladik Kreinovich. 2024. How to Make Machine Learning Financial Recommendations More Fair: Theoretical Explanation of Empirical Results. (2024).

[31] ProPublica. 2021. Compas Software Ananlysis. https://github.com/propublica/compas-analysis. Online.

[32] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. 2017. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*. PMLR, 2902–2911.

[33] scikit learn. 2021. Decision Tree Classifier. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. Online.

[34] scikit learn. 2021. Discriminant Analysis. https://scikit-learn.org/stable/modules/lda_qda.html. Online.

[35] scikit learn. 2021. Logistic Regression. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Online.

[36] scikit learn. 2021. Support Vector Machine. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html. Online.

[37] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.

[38] Jim Tankersley. 2023. Black Americans Are Much More Likely to Face Tax Audits, Study Finds. *The New York Times* (31 Jan 2023). https://www.nytimes.com/2023/01/31/us/politics/black-americans-irs-tax-audits.html

[39] Saeid Tizpaz-Niari, Olga Kosheleva, and Vladik Kreinovich. 2024. How to Gauge Inequality and Fairness: A Complete Description of All Decomposable Versions of Theil Index. (2024).

[40] Saeid Tizpaz-Niari, Ashish Kumar, Gang Tan, and Ashutosh Trivedi. 2022. Fairness-aware configuration of machine learning libraries. In *Proceedings of the 44th International Conference on Software Engineering*. 909–920.

[41] Saeid Tizpaz-Niari, Verya Monjezi, Morgan Wagner, Shiva Darian, Krystia Reed, and Ashutosh Trivedi. 2023. Metamorphic Testing and Debugging of Tax Preparation Software. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. 138–149. https://doi.org/10.1109/ICSE-SEIS58686.2023.00019

[42] Saeid Tizpaz-Niari, Pavol Černý, and Ashutosh Trivedi. 2020. Detecting and understanding real-world differential performance bugs in machine learning libraries. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2020)*. 189–199. https://doi.org/10.1145/3395363.3404540

[43] Kush R Varshney. 2021. Trustworthy machine learning. *Chappaqua, NY* (2021).

[44] Lingxi Xie and Alan Yuille. 2017. Genetic cnn. In *Proceedings of the IEEE international conference on computer vision*. 1379–1388.

[45] Jie M Zhang and Mark Harman. 2021. "Ignorance and Prejudice" in Software Fairness. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 1436–1447.

[46] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* 48, 1 (2020), 1–36.

[47] Peixin Zhang, Jingyi Wang, Jun Sun, Guoliang Dong, Xinyu Wang, Xingen Wang, Jin Song Dong, and Ting Dai. 2020. White-box fairness testing through adversarial sampling. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 949–960.