Towards Understanding Fairness and its Composition in Ensemble Machine Learning

Usman Gohar
Dept. of Computer Science
Iowa State University
Ames, IA, USA
ugohar@iastate.edu

Sumon Biswas
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
sumonb@cs.cmu.edu

Hridesh Rajan
Dept. of Computer Science
Iowa State University
Ames, IA, USA
hridesh@iastate.edu

***Abstract*— Machine Learning (ML) software has been widely adopted in modern society, with reported fairness implications for minority groups based on race, sex, age, etc. Many recent works have proposed methods to measure and mitigate algorithmic bias in ML models. The existing approaches focus on single classifier-based ML models. However, real-world ML models are often composed of multiple independent or dependent learners in an ensemble (e.g., Random Forest), where the fairness composes in a non-trivial way. How does fairness compose in ensembles? What are the fairness impacts of the learners on the ultimate fairness of the ensemble? Can fair learners result in an unfair ensemble? Furthermore, studies have shown that hyperparameters influence the fairness of ML models. Ensemble hyperparameters are more complex since they affect how learners are combined in different categories of ensembles. Understanding the impact of ensemble hyperparameters on fairness will help programmers design fair ensembles. Today, we do not understand these fully for different ensemble algorithms. In this paper, we comprehensively study popular real-world ensembles: Bagging, Boosting, Stacking, and Voting. We have developed a benchmark of 168 ensemble models collected from Kaggle on four popular fairness datasets. We use existing fairness metrics to understand the composition of fairness. Our results show that ensembles can be designed to be fairer without using mitigation techniques. We also identify the interplay between fairness composition and data characteristics to guide fair ensemble design. Finally, our benchmark can be leveraged for further research on fair ensembles. To the best of our knowledge, this is one of the first and largest studies on fairness composition in ensembles yet presented in the literature.**

***Index Terms*—fairness, ensemble, machine learning, models**

I. Introduction

Machine learning (ML) is ubiquitous in modern software today. Due to the black-box [1] nature of ML algorithms and its applications in critical decision-making [2, 3], fairness in ML software has become a huge concern. Measuring ML fairness [4–7] and mitigating the discrimination [5, 8, 9] has been studied extensively. Recent work in software engineering has shown the need to produce fair software and detect bias in complex ML software environments [10–13].

Prior research has mostly focused on fairness in standalone classifiers (e.g., *Logistic Regression, SVM*) [1, 14, 15]. However, a class of ML models called *ensemble models* are becoming increasingly important in practice today due to their superior performance across a multitude of ML & real-life challenges [16–20], and better generalization on unseen data, especially in smaller datasets [18, 21, 22]. Ensemble models combine the predictions of multiple base learners to make the final prediction, e.g., Random Forest uses a large number of decision trees, with the majority class being the final output. Ensemble models are the most mentioned ML algorithms on Kaggle [23], and in previous SE works on fairness, ensemble models comprise more than 80% of the total models [12, 13]. Like traditional ML models, ensemble models can also suffer from unfairness problem that discriminates against population subgroups based on race, gender, etc. Although many fairness mitigation techniques [24, 25] exist, they do not always generalize well [26–28]. Therefore, if we better understand the fairness composition in ensembles, we can design fair ensemble models without applying mitigation techniques. In this paper, we have conducted an empirical study to understand the composition of fairness in ensembles and the interplay of their properties with fairness.

Recently, multiple works have shown that ensembles can be leveraged to enhance fairness and mitigate bias in ML models [28–30]. Grgic-Hlaca *et al.* first explored fairness properties of random selection ensemble, only theoretically [31]. Bower *et al.* explored how fairness propagates through a multi-stage decision process like hiring [15]. Similarly, Dwork *et al.* introduced a framework to understand the composition of fairness in ensembles that *only* utilize AND, OR operators to make a decision, e.g., two credit bureaus' (AND) report a score to determine loan eligibility [4]. Feffer *et al.* studied how ensembles and bias mitigators can be combined using modularity to improve stability in bias mitigation [28]. Therefore, it is evident that fairness in ensembles and their composition is non-trivial. Moreover, prior works in SE have shown the impact of training processes such as hyperparameter optimization, data transformation, etc., on the fairness of ML software [12, 25, 32]. We postulate that ensemble hyperparameters also impact unfairness in ensembles, and failure to study them can amplify bias. However, ensemble hyperparameters are different than typical ML model hyperparameters as they dictate the design of the ensemble, e.g., number of learners, learning method, etc. However, no empirical study has been conducted to understand fairness composition in ensembles and the effect of their hyperparameter space on fairness. To this end, we have created a benchmark of 168 real-world ensemble models from

Kaggle and designed experiments to measure their fairness. We analyze fairness composition in ensemble criteria such as parallel and sequential ensembles, homogeneity of models, and different ensemble methods such as bagging, boosting, voting, stacking, etc., and all the ensemble classifiers available in the popular Scikit-learn [33]. Specifically, we answer the following overarching research questions:

- **RQ1:** *What are the fairness measures of various ensemble techniques?*
- **RQ2:** *How does fairness compose in ensemble models?*
- **RQ3:** *Can ensemble-related hyperparameters be chosen to design fair ensemble models?*

To the best of our knowledge, this is the first work to experimentally evaluate the fairness composition in popular ensembles and elicit fair ensemble design considerations. Our results show that fairness in ensembles composes in the base learners, and fair ensemble models can be built by carefully considering the composition. The analyses also identify learners that cause fairness problems which software developers can leverage to develop frameworks to measure fairness in base learners and encourage transparency. We also identify and explore ensemble-related hyperparameters to design fair ML models for each ensemble type. Lastly, we provide a comprehensive review of fairness composition in ensembles that will help direct future research in the area. Overall, the following are the key contributions of this paper:

- Explored fairness composition and its interplay with data characteristics and individual learners to mitigate bias.
- Empirically evaluate fairness patterns of popular ML ensemble models.
- We identified ensemble design considerations and hyperparameters that would guide developers in fair ensemble design and mitigate inherent unfairness effectively.
- A comprehensive fairness benchmark of popular ensembles that can be leveraged for further research on building fairness-aware ensembles. The benchmark, code, and experimental results are available: https://github.com/UsmanGohar/FairEnsemble

The rest of the paper is organized as follows: §II describes the motivation of our work and the background on ensembles. In §III, we discuss the methodology for our study, including benchmark collection, datasets and fairness and accuracy measures used, and finally, the experiment setup & design. In §IV, we discuss the state of fairness in ensembles **(RQ1)** and how it composes **(RQ2)**, §V discusses the design criteria to improve fairness in ensembles **(RQ3)**. Finally, we discuss the implications of our work in §VI, threats to validity in §VII, related works in §VIII and then present the conclusion in §IX.

II. Motivation and Background

In this section, we use a motivating example to illustrate the complexity of fairness composition in ensembles and the need to study bias induced by certain ensemble parameters.

A. Motivating example

Ensemble models are widely deployed to win competitions in online communities like Kaggle due to their superior performances [16–20, 23]. In prior SE works on fairness [12, 13], more than 80% of the models were ensemble based. However, those works did not consider fairness composition of individual learners, its effect on the fairness of ensembles, and the inherent bias in ensemble methods, which is non-trivial. Hence, not studying fairness composition in ensembles fails to capture the complete fairness of an ML pipeline. Consider the code snippet below of a top-performing model (Voting ensemble) from Kaggle, which is used to predict the income of an individual (*German Credit* dataset).

```
1  models = []
2  models.append(('LGR', LogisticRegression()))
3  models.append(('LDA', LinearDiscriminantAnalysis()))
4  models.append(('KNN', KNeighborsClassifier()))
5  models.append(('CART', DecisionTreeClassifier()))
6  models.append(('NB', GaussianNB()))
7  models.append(('RF', RandomForestClassifier()))
8  models.append(('SVM', SVC(gamma='auto'))
9  models.append(('XGBM', XGBClassifier()))
10 models.append(('LGBM', LGBMClassifier()))
11 model = VotingClassifier(estimators=models, voting='soft')
12 model.fit(X_train, y_train)
13 y_pred = model.predict(X_test)
```

A voting ensemble is a type of *heterogeneous* ensemble that combines the predictions of dissimilar learners. It comprises multiple learners (lines 2-10) and uses a voting mechanism (line 11) to make the prediction. In *soft* voting, the class label (1 or 0) with the higher average probability from the learners is chosen as the final prediction. We found that this ensemble is biased towards female applicants (Protected attribute: *Sex*) in terms of statistical parity difference (SPD:-0.203). In this example, before training the ensemble, a developer must decide the number of learners, select which learners to use, and the voting type (*soft/hard*). However, we found that ML libraries do not provide any fairness recommendations for building ensembles. Do these learners introduce unfairness in the predictions? How does the number of learners impact the fairness of the ensemble? More importantly, we observed that individual learners have their own fairness measures when analyzed in isolation but might result in an unfair model when used in an ensemble. For instance, our analysis shows that dropping XGBClassifier and LGBMClassifier (lines 9-10) can improve fairness by 27% (SPD:-0.148). Interestingly, we discuss later how these two learners are inherently fair themselves and not responsible for the unfairness.

Furthermore, prior research has shown the impact of hyperparameters on fairness [12, 25, 32]. Ensemble hyperparameters dictate how ensembles combine learners for final prediction. In this example, if a developer used *"hard"* voting (line 11), the fairness of the ensemble would improve (SPD: -0.195). Similarly, some of these hyperparameters also affect the design properties of the learners, which impacts fairness. *XGBoost* (line 9) is another example of an ensemble (boosting). Boosting builds an ensemble of trees (learners) using various methods. What properties of these trees (e.g., tree depth, number of features, etc.) and the learning method impact the overall

TABLE I: Types of ensemble models used in our experiments

Categories	Ensemble Types	Algorithms	Composition	Classifiers
Sequential	Boosting	Construct n homogeneous estimators sequentially that improve predictions based on the previous estimator's incorrect predictions	Homogeneous Homogeneous Homogeneous	XGBoost AdaBoost Gradient Boosting
Parallel	Bagging	Construct n parallel homogeneous models that are aggregated using averaging	Homogeneous Homogeneous Homogeneous	Random Forest ExtraTrees Bagging Classifier
	Voting	Construct a list of n heterogeneous user-specified weighted classifiers that are aggregated using majority voting or argmax	Heterogeneous	Voting Classifier
	Stacking	Construct a list of n heterogeneous classifiers as base learners and a meta-classifier to decide weights for each learner	Heterogeneous	Stacking Classifier

fairness of a boosting ensemble? Exploring these parameters will help developers understand how to design fair ensembles. Therefore, in addition to understanding fairness composition in the learners, it is equally important to understand how the design of ensembles using these parameters impacts fairness.

B. Ensemble learning in ML software

Ensemble models are a class of ML classifiers where the predictions from different learners (models) are pooled using a combination method (voting, average, random, etc.) to make the final predictions. In the motivational example above, we only discussed a single type of ensemble. Categories of ensembles are based on homogeneity, learning technique, and ensemble types. All the ensemble types covered in our study and the corresponding classifiers are given in Table I. There are mainly two categories of ensembles: Sequential and Parallel.

Sequential Ensembles. These ensembles sequentially generate base learners. Each learner in this ensemble depends on the previous learners in the sequence because the next learner attempts to correct the wrong predictions from the previous learner and so on [34]. AdaBoost is an example of a sequential model where it reweighs (higher) misclassified examples.

Parallel Ensembles. Parallel ensembles train individual base learners in parallel and independently of each other. These learners are combined using techniques such as bagging (a random sample of data with replacement) or voting, which encourages improved variance [34] e.g., *Random Forest.*

Homogeneity of ensembles. Ensemble methods that use single-type base learners are called *homogeneous* models [35]. These individual learners are combined to generate the final result, e.g., *XGBoost* and *AdaBoost* use decision trees. By contrast, *heterogeneous* ensembles combine the predictions of dissimilar individual learners [35]. A popular heterogeneous ensemble method is *Voting*. Finally, ensemble method types are divided into *Boosting, Bagging, Voting*, & *Stacking* [33].

III. Methodology

In this section, we discuss the benchmark collection process, the datasets, and fairness and accuracy measures. Finally, we describe the experimental design and setup.

A. Benchmark Collection

For our experiments, we collected ensemble models from Kaggle [36] for datasets that have been used in prior fairness literature [12, 28, 37]. Unlike these works, we only collect ensemble-based models for evaluation. Specifically, we collect all ensemble classifiers available via the popular scikit-learn library [33]. We follow a similar benchmark collection process as in [13]. Table II summarizes the datasets and the classifiers in the benchmark.

TABLE II: Summary of the datasets and the number of models collected for each in the benchmark

Datasets	PA	Size	#XGB	#ADB	#GBC	#RF	#ET	#STK	#VT	Total
Adult Census	sex	32561	6	6	6	6	6	1	2	33
Titanic ML	sex	891	6	6	6	6	6	6	6	42
Bank Marketing	age	41118	6	6	6	6	2	1	2	29
German Credit	sex	1000	6	1	1	6	1	1	1	17

XGB: XGBoost, ADB: AdaBoost, GBC: Gradient Boosting , RF: Random Forest, ET: Extra Trees, STK: Stacking, VT: Voting

For each dataset, we collected Kaggle kernels for each ensemble type in Table I and classifiers available in scikit-learn. We filter these kernels based on four-step filtering criteria: 1) it should contain the predictive model (some kernels focus on data exploration only), 2) protected attribute is included in the training data, 3) at least five up-votes, and 4) ranked by up-votes. We used Kaggle API to collect these models and pass them through the filtering criteria. Finally, we select the top 6 models for each ensemble classifier from each dataset. In total, we created a benchmark of 168 ensembles across four datasets. We could not find certain classifiers on Kaggle for datasets like German Credit. To handle those, we use default models from scikit-learn to ensure we can evaluate across different datasets. The number of models mined is shown in Table II. Next, we present an overview of the datasets used in our benchmark.

Adult Census. The dataset contains income and personal information about individuals [38]. We used *sex* as the protected attribute and *male* as the privileged class. The classification task predicts if a person makes over $50,000 in annual income.

Titanic. The dataset contains passenger data, such as gender, cabin class, etc., and is pre-split into train & test sets; however, the test set does not contain any instance of a male passenger surviving [39]. Hence, we only use the training set, with *gender* as the protected attribute and *Female* as the privileged class. The prediction task is whether a passenger survives.

Bank Marketing. The dataset contains bank customers' personal information such as age, job type, etc. [40]. The protected attribute is *age*, where $age > 25$ is considered privileged class and $age < 25$ as unprivileged [13]. The prediction task determines whether a client will subscribe to

a term deposit.

German Credit. This dataset contains personal and financial information about individuals who apply for loans at a bank [41]. We used the processed dataset [42] since most models in our benchmark used it. This version has nine attributes, such as sex, credit amount, etc. We choose *sex* as the protected attribute and *male* as the privileged class. The prediction task is whether an individual is a credit risk.

B. Measures

Multiple quantitative fairness and accuracy measures are available to evaluate a model. We use measures that have been previously used in literature [11, 13]. Let $D = (T, S, Y)$ be a dataset where T is the training set, S is the protected attribute ($S = 1$, if privileged group (*p*), else $S = 0$ (*up*)) and Y is the classification label ($Y = 1$ if favorable label, else $Y = 0$). Let $\hat{Y}$ denote the prediction of an ML model. Next, we define our measures in terms of these notations.

1) Accuracy Metrics: We evaluate the performance of the models using accuracy and F1 metrics as defined below:

$$Accuracy = (true\ positive + true\ negative)/total$$
$$F1 = 2 * (precision * recall)/(precision + recall)$$

where recall: $TP/(TP+FN)$, precision: $TP/(TP+FP)$

2) Fairness Measures: Broadly, fairness metrics are divided into three categories [43]. We have selected a subset of these metrics representing the three categories without being exhaustive. Furthermore, we have followed the recommendations of Friedler *et al.* [13] in terms of metrics selection.

Group fairness metrics: *Group fairness* means similar predictive outcomes for protected attributes, e.g., race (Asian/White) on a group level.

Equal Opportunity Difference (EOD): This is defined as the difference of true-positive rates (TPR) between privileged and unprivileged groups [44].

$$EOD = TPR_{up} - TPR_p$$

where TPR: $TP/(TP + FN)$, FPR: $FP/(FP + TN)$

Average Odds Difference (AOD): This is defined as the mean of false-positive rate (FPR) difference and true-positive rate difference between unprivileged and privileged groups [27].

$$AOD = [(FPR_{up} - FPR_p) + (TPR_{up} - TPR_p)]/2$$

Disparate Impact (DI): This is defined as the ratio of the probability of unprivileged group vs. privileged group getting a favorable prediction [7]

$$DI = P[\hat{S} = 1|Y = 0]/P[\hat{S} = 1|Y = 1]$$

We convert Disparate Impact (DI) to log scale to improve readability compared with other metrics.

Statistical Parity Difference (SPD): This is defined similar to DI but uses the difference between the probabilities. [45].

$$SPD = P[\hat{S} = 1|Y = 0] - P[\hat{S} = 1|Y = 1]$$

Individual fairness metrics:

Theil Index (TI): It measures both the group and individual fairness [6]. It is defined using the following equation:

$$TI = \sum_{i=1}^{n} \frac{b_i}{\bar{a}} ln \frac{b_i}{\bar{a}}, \text{ where } b_i = \hat{s}_i - s_i + 1.$$

C. Experiment Design & Setup

Each ensemble model has specific requirements for training (e.g., *XGBoost* can handle Null values, but *Random Forest* cannot *etc.*) that we need to handle before we can evaluate them. We used the same preprocessing steps across all the kernels and datasets to ensure consistent comparison. Next, we evaluated the accuracy and fairness of base learners and the final ensemble level and analyzed the results.

For our data preprocessing, we start by converting all non-numerical features to categorical data, i.e., *Binary* or *Ordinal* (e.g., *male: 1*,*female: 0* or non-binary levels, like *Marital-Status* to *Divorced: 0, Married: 1, Single: 2* etc.). Next, we remove missing values from our datasets and convert continuous sensitive attributes to categorical (e.g., age > 25: 1, age < 25 :0 corresponding to old and young, respectively). These preprocessing steps are necessary for most ensembles and the AIF360 toolkit. We denote the privileged and unprivileged groups and the favorable label for each dataset separately. For example, in *Titanic* dataset, *male* is the unprivileged group, and the favorable label is *Survivied: 1* i.e., the individual survived the titanic crash. The groups and the labels have been chosen as seen before in literature [13, 46]. Finally, the dataset is shuffled and split into train and test sets using a $70\% - 30\%$ split. For each dataset, we have selected the top 6 kernels by upvotes. We run the preprocessing steps discussed before training the model to evaluate based on accuracy and fairness metrics. We use five fairness metrics and two accuracy measures to generate results for each model. These experiments are repeated ten times, and the mean is reported [11]. We used the *IBM AIF 360 Fairness Toolkit* to calculate the fairness metrics. Finally, a non-zero value for fairness metrics suggests a bias in the model. A positive value of a fairness metric suggests the model is biased against the privileged group and vice-versa.

IV. Fairness in Ensembles and its composition

In this section, we explore the state of fairness in ensembles and its composition in all popular ensemble methods.

A. State of fairness in ensemble models

Before understanding the composition of fairness in ensembles, we first investigate how different ensemble techniques impact fairness **(RQ1)**. Are certain ensemble classifiers more unfair? Does the architecture of an ensemble method (stacking, boosting, etc.) contribute to fairness? Does any particular ensemble classifier exhibit a better fairness-accuracy trade-off? To answer these questions, we experiment to evaluate the

TABLE III: Fairness and accuracy comparison of all ensemble ML classifiers across the datasets in our benchmark. The ranks were calculated using the Scott-Knott test [47]. Each cell depicts the median score; Darker, lighter, light, lightest and white colored cell denotes the first, the second, the third, fourth, and lowest rank, respectively. The rank ranges from 1 to 5.

Dataset	Protected Attribute	Ensemble Classifiers	Ensemble Type	Accuracy (+)	F1 (+)	SPD (-)	EOD (-)	AOD (-)	DI (-)	TI (-)	Mean Accuracy Rank (-)	Mean Fairness Rank (-)
Titanic	Sex	TM-XGB	Boosting	0.82	0.75	-0.65	-0.50	0.43	-1.83	0.14	2	1.4
		TM-ADB		0.81	0.75	-0.81	-0.77	0.70	-2.56	0.15	3.5	4.4
		TM-GBC		0.82	0.74	-0.71	-0.57	0.54	-2.09	0.14	3.5	2.2
		TM-RF	Bagging	0.81	0.73	-0.68	-0.58	0.52	-2.30	0.16	5	3
		TM-ET		0.82	0.75	-0.80	-0.75	0.68	-2.76	0.15	2	4
		TM-VT	Voting	0.83	0.77	-0.74	-0.51	0.54	-2.10	0.12	1	2.8
		TM-STK	Stacking	0.82	0.76	-0.76	-0.63	0.58	-2.40	0.13	3	2.6
Adult	Sex	AC-XGB	Boosting	0.87	0.71	-0.18	-0.08	0.08	-1.14	0.11	1	1.6
		AC-ADB		0.86	0.66	-0.20	-0.15	0.14	-1.32	0.12	2.5	4.4
		AC-GBC		0.86	0.68	-0.19	-0.14	0.11	-1.25	0.12	2	3
		AC-RF	Bagging	0.85	0.67	-0.18	-0.13	0.11	-1.26	0.12	5	3.4
		AC-ET		0.84	0.65	-0.19	-0.10	0.10	-1.11	0.13	4	2.8
		AC-VT	Voting	0.85	0.66	-0.17	-0.13	0.09	-1.29	0.12	4.5	3.4
		AC-STK	Stacking	0.86	0.68	-0.18	-0.11	0.09	-1.28	0.11	2.5	3.6
Bank Marketing	Age	BM-XGB	Boosting	0.93	0.70	0.15	0.08	0.08	0.77	0.05	1.5	2.2
		BM-ADB		0.88	0.49	0.15	0.18	0.12	1.04	0.11	4.5	4.4
		BM-GBC		0.89	0.48	0.14	0.12	0.09	1.09	0.10	4.5	3.8
		BM-RF	Bagging	0.89	0.55	0.18	0.09	0.09	0.80	0.07	2.5	3
		BM-ET		0.91	0.54	0.14	0.06	0.06	0.82	0.07	2.5	2.2
		BM-VT	Voting	0.94	0.69	0.12	0.06	0.05	0.71	0.05	1	1.2
		BM-STK	Stacking	0.93	0.72	0.15	0.04	0.06	0.71	0.05	1.5	1.2
German Credit	Sex	GC-XGB	Boosting	0.72	0.65	-0.07	-0.02	0.08	-0.12	0.17	1.5	1.8
		GC-ADB		0.72	0.55	-0.11	-0.07	0.13	-0.19	0.16	3	4.4
		GC-GBC		0.72	0.56	-0.08	-0.06	0.10	-0.15	0.15	3	2.6
		GC-RF	Bagging	0.72	0.64	-0.09	-0.04	0.09	-0.13	0.15	2	2
		GC-ET		0.70	0.60	-0.11	-0.07	0.11	-0.14	0.16	3	3.4
		GC-VT	Voting	0.73	0.54	-0.08	-0.05	0.08	-0.16	0.16	2	2.6
		GC-STK	Stacking	0.73	0.55	-0.09	-0.06	0.10	-0.17	0.14	2	2.8

fairness of ensemble models using a diverse set of metrics. Table III shows the mean fairness for all ensembles. Figure 1 illustrates the cumulative fairness for all 168 models.

Our findings showed dataset-specific fairness patterns for ensemble models; however, some exhibited more unfairness than others. We used the Scott-Knott ranking test [47] to compare the fairness and accuracy of the ensemble types and determine if the differences are significant. The test assigns a rank to the classifiers based on their performance, with a higher rank indicating better results. In our experiments, the classifiers were ranked from 1st to 5th (some with the same rank) for each metric.

Finding 1: Among all the ensemble models, XGBoost exhibits the best accuracy-fairness trade-off.

Table III shows varying fairness performance among the ensemble classifiers across different datasets. Interestingly, we observe that fairness can be highly inconsistent even within the same ensemble type. For instance, *XGBoost* has the highest rank in 8 out of 10 fairness metrics for the highly biased *Titanic* and *Adult* datasets, with a mean fairness rank of 1.4 and 1.6, respectively. On the other hand, *AdaBoost* has the lowest rank in 13 out of 16 fairness metrics across all the datasets. Additionally, we observe that *XGBoost* stands out with high accuracy and fairness across all ensemble models, contrary to typical inverse behavior seen in ML models. For example, in the Titanic dataset, the average performance change for the XGBoost classifier in accuracy and f1 score is less than 0.01. However, their cumulative mean fairness is 14% more than the next most fair model (*GBC*) in Titanic. For the other datasets, we observe a similar pattern; however, the difference is lesser due to low unfairness in the dataset. Upon further investigation, we found that boosting method and base learner design is responsible for the fairer performance of *XGBoost*. Homogeneous ensembles use decision trees as the base learner, and the construction of these trees differs among them. For example, the depth of the decision tree in *AdaBoost*, *GradientBoosting*, *XGBoost* is one, three, and six, respectively. Lower tree depth means fewer features are selected, which has been shown to often increase unfairness [13, 48]. Importantly, we found that the fairness of an ensemble is determined by the composition of fairness within these base learners and the learning method (boosting, bagging, etc.). In the next section, we delve deeper into the properties of base learners to understand how to create fair ensembles.

Finding 2: Fairness measures show more instability compared to accuracy metrics.

Prior works have shown that ensembles improve the stability of accuracy metrics by aggregating multiple learners trained

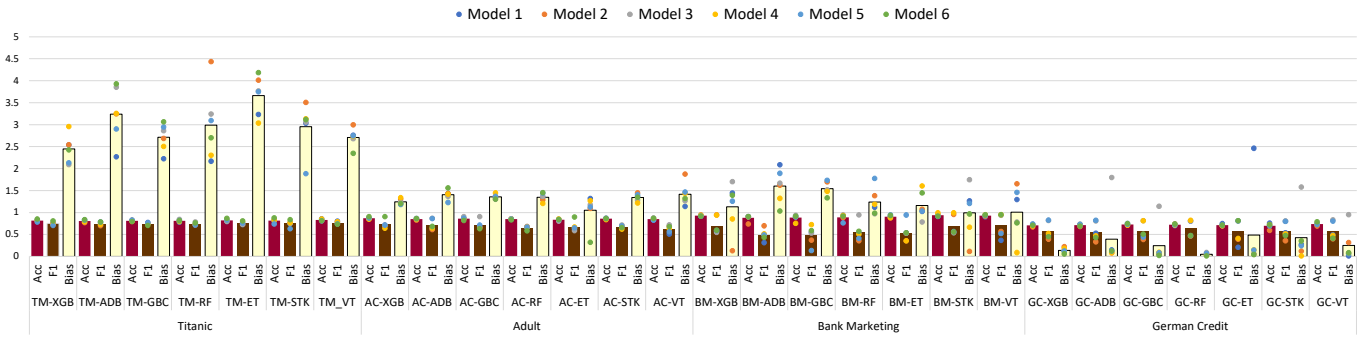

Fig. 1: Cumulative bias and performance of all ensembles. The bars represent mean values, and the dots the models

on subsets of data (bootstrap sampling) [44, 49]. Intuitively, we explore the variance exhibited by fairness metrics in ensembles. Interestingly, despite aggregating multiple learners, the stability of fairness measures in ensembles still suffers, especially in smaller datasets. This is attributed to the change in data distribution after random train/test splits in smaller datasets [11, 13]. For larger datasets (*Adult, Bank Marketing*), the standard deviation for all fairness measures is less than 0.02. For smaller datasets, the average standard deviation of the metrics is shown in Figure 2. Firstly, we observe that the stability of fairness metrics remains consistent between all the ensembles for a specific dataset. Furthermore, we observed that group fairness measures exhibit higher variability than individual fairness measures (TI). Surprisingly, heterogeneous models also exhibit instability despite using dissimilar learners to reduce variance. From Figure 3, we also see that the volatility in fairness metrics is greater than in accuracy metrics for homogeneous models. Given a random train/test split, it might cause the model to seem fairer than it is. Hence, even with improved stability in fairness compared to non-ensemble methods, developers should evaluate the training set and repeat training over multiple runs in ensembles.

Finding 3: Libraries do not provide API support to measure fairness of base learners in ensembles

Biswas and Rajan [13] discussed that hyperparameter optimization goals induce unfairness. In the case of heterogeneous ensemble models, the developer must carefully choose the number and type of individual base learners. Libraries do not provide any recommendations to developers, who try to select a diverse set of learners to improve accuracy. However, this might not always result in a fair ensemble. For instance, removing a GaussianNB learner from the *BM-STK3* model improved its Statistical Parity Difference (SPD) from 0.13 to 0.11 while also increasing accuracy. Heterogeneous ensemble models, such as *Voting* models that use weighted voting and *Stacking* models that use a meta-learner to determine the best weighing configuration of learners, can be challenging to train fairly since libraries do not provide API support to measure the fairness of base learners, especially in combinations with other learners at the ensemble level. Hence, developers have little

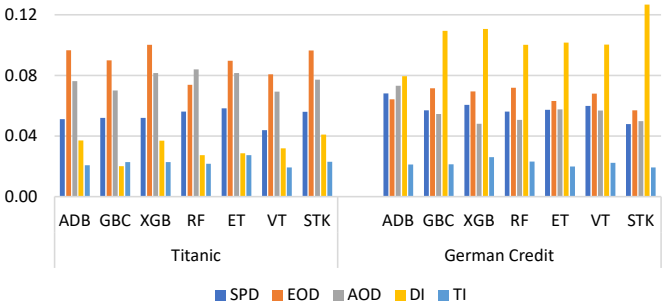

Fig. 2: Standard deviation of fairness metrics over multiple experiments. Other datasets have very low standard deviation.

information on how to weigh and select individual learners, which can lead to unfair ensembles. Similarly, understanding fairness composition in base learners of homogeneous models can help the developer identify fairness issues such as bias in specific features (e.g., decision tree learners in random forest randomly select features). Therefore, API support to measure fairness in base learners can help developers better understand & detect unfairness in ensembles.

B. How does fairness compose in ensembles?

In this section, we investigate the composition of fairness in ensembles. We posit that the underlying unfairness of ensembles is a product of the composition of fairness in base learners and the learning method. All homogeneous models use a decision tree as the base learner, whereas heterogeneous models can be constructed with any ML classifier. We investigate how fairness composes in these base learners and how it is propagated by the learning techniques **(RQ2)**. In general, our findings show that the complexity of base learners significantly impacts the fairness of ensembles and that more research is needed to develop fair learning techniques in ensembles.

Finding 4: The unfairness of homogeneous ensembles is caused by the complexity of the base learner and dataset characteristics.

In Figure 3, we plotted the most biased and the least biased homogeneous ensembles in our benchmark. We see how the

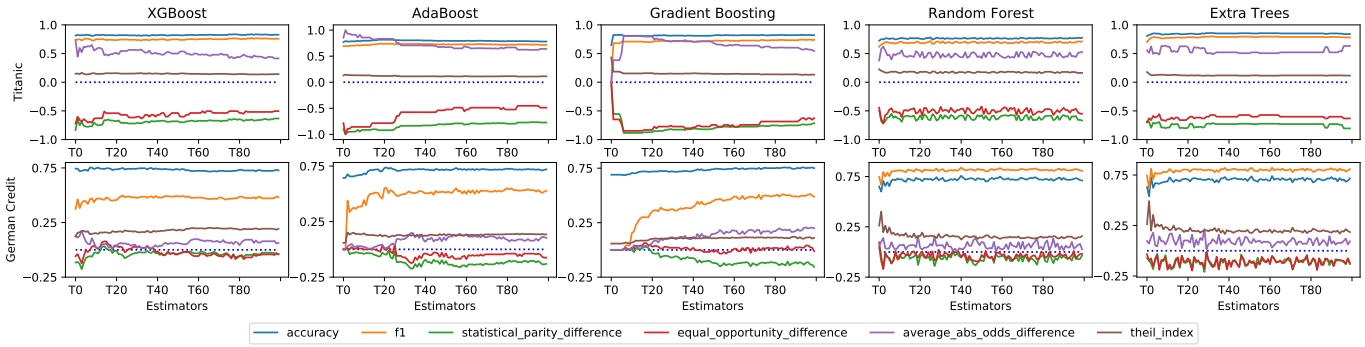

Fig. 3: Composition of fairness in homogeneous ensembles with respect to base learners. Default no. of trees: T100

fairness of the base learners directly contributes to the fairness of ensembles during training. However, we observe different fairness composition patterns in terms of the datasets. For example, in *Titanic*, the fairness of the boosting ensembles improves, while the opposite is observed for some fairness metrics in *German Credit*. The variation in fairness patterns is also seen in specific classifiers. From Figure 1, we observe that all *TM-XGB* models show similar bias except *TM-XGB4*. We investigate the difference in unfairness by comparing all the parameters of the base learner decision tree with the other *XGB* models and found that *TM-XGB4* uses a shallow decision tree with max_depth : 2, which is causing the unfairness to amplify. The model construct is shown below:

```
model = XGBClassifier(n_estimators= 500,max_depth=2,
    subsample=0.5, learning_rate=0.1)
```

MaxDepth sets the maximum depth of the decision tree. The depth of the tree is defined as the number of splits (nodes), where the feature to be split is chosen based on the highest information gain among the features. Deeper trees are more complex and reduce errors [50]. For *XGBoost* models, the default depth is 6. Our analysis showed that the protected attribute (*Sex*) has the highest information gain among all the features in the *Titanic* dataset. Therefore, the protected attribute is the most important feature to split on at the tree's root, resulting in a high degree of unfairness in *TM-XGB4*. Base learners in all boosting models use the best feature to split, which improves accuracy. However, it has been shown that unfairness is encoded in specific features [51]. If these features are also among the best features of a dataset, a shallower ensemble will be more unfair due to a reduced number of features. This corroborates similar observations in the literature [12, 48]. We observe the same pattern for all boosting models. For example, *AdaBoost* and *GradientBoosting* exhibit more bias than *XGBoost* because of shallower base learners (1 and 3, respectively). In Figure 1, *TM-GBC1* is fairer because the learner is deeper (depth:5).

Finally, further analysis of the properties of a decision tree suggests that regularization parameters like min samples leaf and max-leaf nodes also impact tree depth, hence affecting the fairness of the ensemble. Therefore, it is important to carefully balance the complexity of the tree-based base learners for

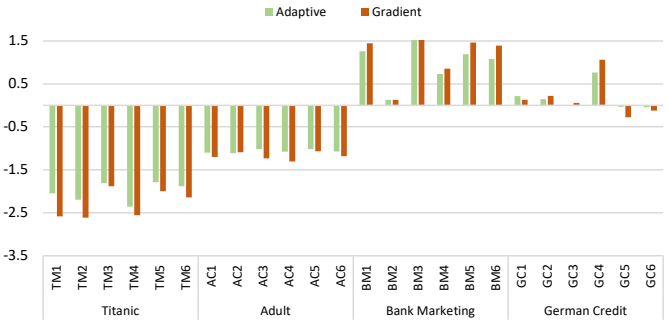

Fig. 4: Cumulative fairness in adaptive vs. gradient learning

homogeneous models with the fairness outcomes, especially with the underlying properties of the data.

Finding 5: *Gradient-based* composition propagates more unfairness compared to *Adaptive* boosting models.

We have established that base learners and the underlying data properties influence the unfairness of homogeneous models. However, boosting models also use a learning technique to improve the model's predictions sequentially. *XGBoost* and *GradientBoosting* models use gradient-based optimization and *AdaBoost* uses an adaptive weighting technique. We compare these techniques by training the boosting models on the same base learner decision tree (depth:6). We only use *XGBoost* and choose this depth in our experiment since our analysis (Table III) showed that it is the most fair boosting model. The results are shown in Figure 4. For all the models except GC1, we see that adaptive learning is fairer than gradient optimization. We use the Scott-Knott rank test to test statistical significance. Accordingly, we observed that adaptive learning outperformed gradient optimization in all datasets except German Credit, where the difference was not statistically significant. Consequently, we can see that adaptive learning propagates less bias in highly biased datasets. Our analysis should help guide further research into designing fair learning techniques for boosting ensembles.

TABLE IV: Ensemble-related hyperparameters (HP) that can affect the design of fair ensembles

HP	Values	Default	ADB	GBC	XGB	RF	ET	Voting	Stacking
n_estimators	Total number of trees/boosting rounds	100	✓	✓	✓	✓	✓		
booster	Booster type: gbtree, gblinear, dart	gbtree			✓				
bootstrap	Data sampling with replacement	RF: True, Others: False		✓	✓	✓	✓		
voting	Voting Type: soft, hard	hard						✓	
estimators	Base learners for the ensemble	ADB: DecisionTree, Others: none	✓					✓	✓
final_estimator	Meta-learner to combine learner predictions	Logistic Regression							✓

V. Fair Ensembles Design

In §IV, we found that base learners of ensembles propagate bias. Many bias mitigation techniques applied during model training (*inprocess*) have been successful [24, 25, 32]. Techniques applied during the model training phase can assist developers in improving the fairness of ML software. These works have also established the role of hyperparameters in mitigating and amplifying fairness bugs (unfairness) in ML software. If we understand and identify what ensemble parameters and design choices affect the fairness, we can mitigate inherent bias in ensembles. Moreover, it will help developers, and libraries better explain fairness bugs in the ensemble hyperparameter space. This section explores the hyperparameter design space for ensembles to boost fairness performance. We have found that some hyperparameters directly affect the fairness of ensembles. Specifically, we evaluate how ensembles can be designed to be fair using ensemble hyperparameters summarized in Table IV. We use the Scott-Knott test to determine the significance of our results. Our findings provide a comprehensive review of all ensemble hyperparameters.

Finding 6: Developers should carefully choose dropout regularization to balance fairness and overfitting.

Our analysis shows that dropout impacts fairness in relation to the underlying data properties. Vinayak and Gilad-Bachrach [52] proposed DART, a dropout technique derived from deep neural networks, for boosted trees. An ensemble of boosted regression trees suffers from over-specification, i.e., the trees added at the end have little contribution to the final result [52]. *DART* alleviates this by constructing the next tree from the residuals of a random sample of the previous trees. In *XGBoost*, the *rate-drop* ([0-1]) parameter controls this sampling rate. No trees are dropped on the lower end of this rate, while on the higher extreme, all trees are dropped. We investigate the efficacy of DART with *ratedrop = 0.5*, in reducing unfairness in boosting models by comparing it with the default *XGBoost* booster *gbtree*. We analyze the change in performance and fairness of dropout in Figure 5.

From Figure 5, we can see that dropout can impact the fairness of boosting models. For example, in *Adult* dataset, initial trees exhibit less unfairness than the latter. Using dropout, the subsequent trees only learn on a random sample of initial trees, which in this case are fairer. This improves the fairness of the models. The opposite is observed in *Titanic* dataset. In both scenarios, the change in accuracy is less than 0.1, but a significant impact is seen on fairness. Therefore, developers should be cautious about the effect of regularization on fairness. More research is needed to understand the fairness-overfitting trade-off and develop fair regularization.

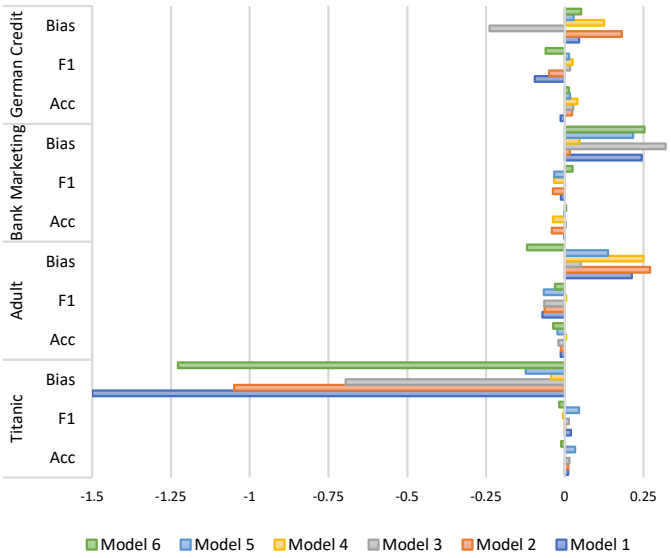

Fig. 5: Change in accuracy and total bias when using DART. A negative value denotes lower fairness & accuracy

Finding 7: Randomness in feature splitting does not improve fairness in bagging models.

Random Forest improves the variance of the model by introducing randomness to the process of model building by randomly selecting features. *Extra Trees* introduces additional randomness by randomly finding the splits for each feature and then selecting the best split from them, i.e., independent of the target variable. In contrast, *Random Forest* finds the best split for each feature which has been shown to improve accuracy [53]. However, no work has studied its effect on fairness. Here, we ask whether randomness at the feature splitting level causes bagging models to be unfair.

To investigate this, we compare *Random Forest* and *Extra Tree* models in our benchmark. We keep the rest of the parameters and data split the same. Each model is run ten times, and the mean is reported in Table V. For all datasets except *German Credit*, the test showed that *Extra Tree* models with random splits were more biased compared to optimal splits (*RF*). This is a key finding because this suggests that

TABLE V: Mean total fairness in Random Forest (RF) and ExtraTrees (ET) models. * denotes the top rank based on the Scott-Knott significance rank test for *each* dataset.

	Titanic		Adult		Bank Marketing		German Credit	
	RF*	ET	RF*	ET	RF*	ET	RF	ET
Model1	-2.04	-1.96	-1.33	-1.32	1.16	1.35	0.03	0.02
Model2	-4.43	-5.37	-1.29	-1.30	1.37	1.41	0.43	0.29
Model3	-3.38	-4.13	-1.32	-1.45	0.96	1.07	0.02	-0.04
Model4	-2.42	-2.46	-1.21	-1.29	1.36	1.35	-0.16	-0.09
Model5	-2.98	-3.38	-1.48	-1.51	1.61	1.51	0.00	-0.04
Model6	-2.61	-2.61	-1.47	-1.41	0.98	1.26	-0.04	-0.28

a split chosen independently of the target is still more unfair than an optimal split. However, in the fairer dataset (*German* Credit), we observe no difference in fairness. Regarding bias mitigation methods, our results suggest randomness in feature split-point might not be an effective way to tackle bias in decision tree-based models.

Finding 8: The uncertainty in classifiers can have a large impact on fairness in voting classifiers.

A *Voting* classifier is an ensemble method where the prediction is based on the probabilities of each base learner within the ensemble. Voting classifiers are of two types, *Soft* and *Hard* Voting. In hard Voting, the label with the majority of votes from the base learners is the final prediction, whereas, in soft voting, it is based on the average of the probabilities of each output class. If the average probability of a class is less than 0.5, class 0 is predicted, and vice-versa. We investigate the effect of the voting type on fairness and found that the uncertainty in the model prediction can have a large impact on fairness. For instance, *AC-VT5* uses soft voting with *Logistic Regression (LR), Random Forest (RF), KNN*, and *Decision Tree (DT)* as base classifiers. As shown in Table VI, *DT* introduces significant unfairness when used in soft voting compared to hard. We found that *DT* has an output class probability of either 1 or 0 while other classifiers are in the range [0,1].

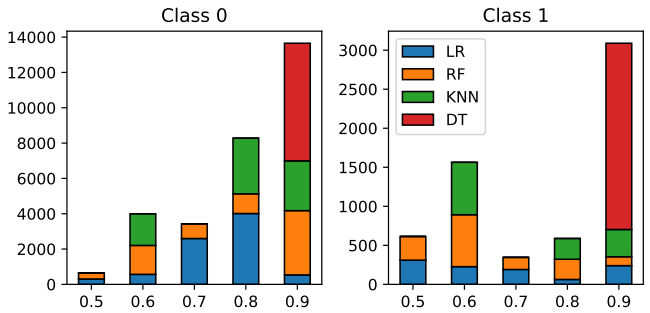

Fig. 6: Frequency of output class probabilities for base learners in AC-VT5

Figure 6 shows the output probabilities for AC-VT5. We observe that DT has higher extreme probabilities compared to others. In this case, the average is skewed by the extreme

TABLE VI: Soft vs Hard Voting in *AC-VT5*

	Soft					Hard				
	Voting	LR	RF	KNN	DT	Voting	LR	RF	KNN	DT
Acc	0.83	0.78	0.81	0.81	0.83	0.81	0.78	0.81	0.81	0.81
F1	0.56	0.41	0.44	0.43	0.56	0.44	0.41	0.41	0.45	0.44
SPD	-0.14	-0.06	-0.08	-0.07	-0.14	-0.07	-0.06	-0.06	-0.08	-0.07
EOD	-0.14	0	0	0	-0.14	-0.03	0	-0.03	-0.04	-0.03
AOD	0.1	0.01	0.02	0.02	0.1	0.02	0.01	0.02	0.03	0.02
DI	-1.43	-0.64	-1.13	-1.02	-1.43	-1.08	-0.64	-1.11	-1.02	-1.08
TI	0.17	0.21	0.2	0.2	0.17	0.2	0.22	0.21	0.2	0.2

probabilities of *DT*. This changes the prediction for 558 out of 9049 test samples. And since *DT* is less fair than other classifiers, the overall unfairness also increases. For hard voting, equal weight is given to each classifier. In that case, the other three classifiers, which are fairer, won the majority vote. For some models, soft voting was fairer than hard, e.g., *AC-VT3*, which shows that base learners' uncertainties can impact fairness in both voting types. This suggests the need to develop frameworks to measure model uncertainties and their fairness at a component level to aid developers in designing fair voting ensembles. Our analysis should also encourage further research in fairness-aware weighting techniques to handle fairness issues arising from model uncertainties.

Finding 9: Two-layer stacking can significantly reduce unfairness.

All of the *Titanic* ML stacking models shown in Figure 1 exhibit similar bias except *TM-STK5*, which is the least biased model for all fairness metrics except Thiel Index (TI). On closer inspection, we found out that *TM-STK5* uses a two-layered stacking approach where a second layer of base learners act as the meta-learner, which causes the model to be fairer. The model construct is shown below:

```
1 layer_one_estimators = [('rf_1', RandomForestClassifier(
      n_estimators=40, random_state=42)),('knn_1',
      KNeighborsClassifier(n_neighbors=6))]
2 layer_two_estimators = [('rf_2', RandomForestClassifier(
      n_estimators=40, random_state=42)),('xg_2',
      XGBClassifier(objective = 'binary:logistic',
      colsample_bytree = 0.8, learning_rate = 0.3,
      max_depth = 7, min_child_weight = 3, n_estimators =
      100, subsample = 0.6))]
3 layer_two = StackingClassifier(estimators=
      layer_two_estimators, final_estimator=XGBClassifier(
      n_estimators = 100))
4 model = StackingClassifier(estimators=
      layer_one_estimators,final_estimator=layer_two)}
```

We validate our finding by training all stacking models in our benchmark using this two-layered nested stacking approach. To ensure consistency, we did not change the kernel's feature set or any preprocessing method. The results are shown in Figure 7. For all stacking models in our benchmark, every model significantly improved in all fairness measures except *Thiel Index*, which is typical as previous works [8, 54] have shown that achieving fairness in terms of all fairness metrics is often difficult. Moreover, accuracy measures did not degrade significantly. For example, *TM-STK6* improved *SPD* scores by 28% while accuracy dropped only 4.68%. Overall across all

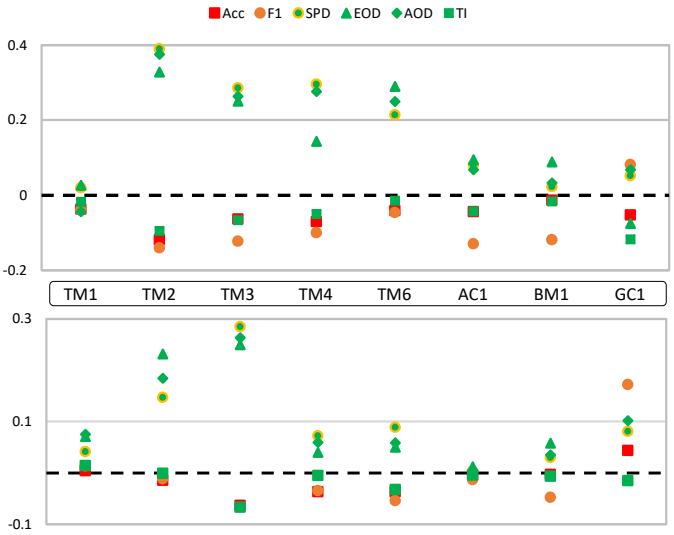

Fig. 7: Performance & fairness changes after using two-layered stacking (Top) and XGB as meta-learner (Bottom). A positive change indicates performance/fairness increase and vice-versa

datasets, the accuracy score dropped by 6.50% while the *SPD* improved by 31.8%.

In Stacking, the first layer uses a list of base learners to generate a set of predictions and a meta-learner to learn from them. However, instead of using a single classifier as the meta-learner, the predictions are fed into another set of base learners in a two-layered approach. This ensures that the outcome is not based on a single meta-learner. Therefore, the second layer of learners creates a new set of predictions, which are then fed into the second layer's final meta-classifier. We compare this approach to simply using an ensemble model (*XGB*) as the meta-learner in default Stacking models and found similar patterns in fairness measures (Figure 7 bottom). Our results did not significantly vary between using other ensembles as the meta-learner. This supports observations that ensembles are fairer than standalone classifiers. Therefore, developers should use an ensemble as a meta-learner or the two-layer approach to improve the fairness of Stacking models.

VI. Discussion

In this study, we undertook the important task of understanding the composition of fairness in ensemble machine learning. Fairness of ML algorithms has been extensively studied, starting from empirical evaluation and identification [12, 13, 48] to mitigation [25, 37, 55, 56] and testing [1, 14, 57, 58]. However, no work has explored the composition of fairness in ensemble models, although ensembles cover the majority of prior SE works and open-source [12, 13, 23]. We showed that considering ensemble models as monolithic classifiers leaves the opportunity to identify the root cause of unfairness. Consequently, our work has shown that fair ensembles can be designed without using bias mitigation techniques. Our research also identifies root causes of unfairness in different ensembles and their interplay with the input space in the pipeline [59], which would guide fairness bug localization and repair in ensemble learning. For example, we report fairness patterns in individual learners that can induce bias in ensembles such as tree depth, minimum leaf node samples, etc. These can also be leveraged for fairness-improving interventions such as feature selection, data preprocessing, etc. Overall, our result would draw attention to the fairness of ensembles which are popular learning algorithms but mostly overlooked by the community.

Moreover, research in SE showed the impact of hyperparameters on fairness and their role in helping developers mitigate bias during model training [12, 25, 32]. We extend that to explore the hyperparameter space for ensembles to guide developers to design fair ensembles using currently available compositions and configurations. Our findings also made direct design suggestions for enhancing specific ensemble library APIs for feature splitting, dropout regularization, and fairness-accuracy trade-offs. This should encourage the development of fairness-aware regularization techniques and investigate the trade-off between fairness mitigation and overfitting. We found that many ensemble models do not have library support to monitor the fairness of individual learners. Finally, our work would encourage the development of tools and API support to improve the transparency of ML software to address fairness concerns.

VII. Threats to Validity

Benchmark: We ensure the quality of the benchmark by collecting only high-quality kernels from Kaggle (at least five upvotes). Additionally, we only consider runnable models, include the protected attribute in training, and have an accuracy greater than 65%, similar to [13]. Finally, we select the top 6 (upvotes) models for each ensemble type.

Sampling Bias: To the best of our knowledge, this is the most extensive review of popular ensembles. Moreover, conclusions are supported by statistical tests across four datasets. However, they may change slightly if other datasets are used.

Generalizability: To avoid the threat of non-generalized findings, we conduct experiments on four different datasets for each ensemble type and compare across multiple ensemble algorithms for both boosting and bagging. Moreover, we use multiple fairness metrics and verify our results by running the experiment multiple (ten) times and using the mean of the values.

VIII. Related Work

a) Fairness in ML classification: The ML community has proposed multiple methods to measure [2, 4, 5, 8, 27, 60] and mitigate unfairness in ML models [5, 8, 9, 24, 61]. However, most of these works have focused on the theoretical evaluation of fairness. Recently, the SE community has increasingly shown interest in fairness in ML software [10]. Empirical studies have investigated the characteristics of biased models and unfairness in ML pipelines, compared mitigation strategies and developer concerns about fairness [11–13, 62]. Some research in SE has focused on fairness

testing and verification and uncovering fairness violations [1, 14, 57, 58, 63]. Finally, a body of work has identified unfairness in data and proposed appropriate mitigation techniques [25, 37, 55, 64].

b) Ensemble Fairness: Grgic-Hlaca *et al.* [31] investigated the impact of fairness in the random-selection-based ensemble. They showed theoretically that its fairness at the ensemble level is always fairer than its components. Wang *et al.* [65] studied the composition of fairness in multi-component recommender systems and presented conditions under which individual components compose fairness. AdaFair [30] proposed a fairness-aware AdaBoost model where unfairly classified instances were up-weighted. A recent work [29] analyzed and compared seven ML models to show that ensembles were fairer than individual classifiers. Feffer *et al.* [28] conducted an empirical study to analyze modular ensembles. They developed a library to find the best configuration using any combination of ensembles and mitigators. In Fair Pipelines [15], the authors explored the propagation of fairness in multi-stage pipelines where a set of decisions impacts the final result, e.g., the hiring process. MAAT [56] proposes an ensemble approach to improve fairness performance by separately combining models optimized for fairness and accuracy. Finally, Tizpaz-Niari *et al.* studied the parameter space of ML algorithms and its impact on fairness [32]. This work is the closest to our study; however, it proposed a testing approach to tune the parameters for achieving fairness and did not consider ensembles (except *random forest*). Our work has focused on comprehensively evaluating fairness composition in all popular ensemble models and how the different algorithmic design configurations (parameters) impact fairness.

IX. Conclusion

Ensembles are widely used for predictive tasks due to superior performance. However, most approaches to measuring fairness and mitigation focus on single classifiers. In this paper, we conduct an empirical study to evaluate the composition of fairness in popular ensemble techniques. The results showed that base learners induce bias in ensembles and that we can mitigate inherent bias in ensembles by using certain base learner configurations and appropriate parameters. Lastly, works have shown the need to support developers during model training in mitigating bias. Our analysis of the hyper-parameter space should help developers build fairness-aware ensembles and automated tools to detect bias in ensembles.

Acknowledgment

This work was supported in part by US NSF grants CCF-19-34884, CCF-22-23812, and CNS-21-20448. We also thank the anonymous reviewers for their insightful comments. All opinions are of the authors and do not reflect the view of sponsors.

References

[1] A. Aggarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, "Black box fairness testing of machine learning models," in *ESEC/FSE 2019*. Association for Computing Machinery, 2019, p. 625–635.

[2] L. Dixon, J. Li, J. Sorensen, N. Thain, and L. Vasserman, "Measuring and mitigating unintended bias in text classification," in *AAAI/ACM*, ser. AIES '18, 2018, p. 67–73.

[3] J. Larson, L. Kirchner, S. Mattu, and J. Angwin, "Machine Bias." [Online]. Available: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[4] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 214–226. [Online]. Available: https://doi.org/10.1145/2090236.2090255

[5] T. Calders and S. Verwer, "Three naive bayes approaches for discrimination-free classification," *Data Mining and Knowledge Discovery*, vol. 21, pp. 277–292, 2010.

[6] T. Speicher, H. Heidari, N. Grgic-Hlaca, K. P. Gummadi, A. Singla, A. Weller, and M. B. Zafar, "A Unified Approach to Quantifying Algorithmic Unfairness: Measuring Individual &;Group Unfairness via Inequality Indices," ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3219819.3220046

[7] M. B. Zafar, I. Valera, M. Gomez-Rodriguez, and K. P. Gummadi, "Fairness constraints: A flexible approach for fair classification," *Journal of Machine Learning Research*, vol. 20, no. 75, pp. 1–42, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-262.html

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

[9] G. Goh, A. Cotter, M. Gupta, and M. P. Friedlander, "Satisfying real-world goals with dataset constraints," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/dc4c44f624d600aa568390f1f1104aa0-Paper.pdf

[10] Y. Brun and A. Meliou, "Software fairness," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 754–759. [Online]. Available: https://doi.org/10.1145/3236024.3264838

[11] S. A. Friedler, C. Scheidegger, S. Venkatasubramanian, S. Choudhary, E. P. Hamilton, and D. Roth, "A comparative study of fairness-enhancing interventions in machine learning," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, ser. FAT* '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 329–338. [Online]. Available: https://doi.org/10.1145/3287560.3287589

[12] S. Biswas and H. Rajan, "Fair preprocessing: Towards understanding compositional fairness of data transformers in machine learning pipeline," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 981–993. [Online]. Available: https://doi.org/10.1145/3468264.3468536

[13] S. Biswas and H. Rajan, "Do the machine learning models on a crowd sourced platform exhibit bias? an empirical study on model fairness," ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 642–653. [Online]. Available: https://doi.org/10.1145/3368089.3409704

[14] S. Galhotra, Y. Brun, and A. Meliou, "Fairness testing: Testing software for discrimination," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 498–510. [Online]. Available: https://doi.org/10.1145/3106237.3106277

[15] A. Bower, S. N. Kitchen, L. Niss, M. J. Strauss, A. Vargas, and S. Venkatasubramanian, "Fair pipelines," 2017. [Online]. Available: https://arxiv.org/abs/1707.00391

[16] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 3133–3181, jan 2014.

[17] N. C. Oza and S. Russell, *Online ensemble learning*. University of California, Berkeley, 2001.

[18] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, vol. 40, no. 2, pp. 139–157, 2000.

[19] E. Bauer and R. Kohavi, "An empirical comparison of voting classifi-

cation algorithms: Bagging, boosting, and variants," *Machine learning*, vol. 36, no. 1, pp. 105–139, 1999.

[20] M. Hosni, I. Abnane, A. Idri, J. M. Carrillo de Gea, and J. L. Fernández Alemán, "Reviewing ensemble classification methods in breast cancer," *Computer Methods and Programs in Biomedicine*, vol. 177, pp. 89–112, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260719301907

[21] N. Ueda and R. Nakano, "Generalization error of ensemble estimators," in *Proceedings of International Conference on Neural Networks (ICNN'96)*, vol. 1, 1996, pp. 90–95 vol.1.

[22] Y. Bian and H. Chen, "When does diversity help generalization in classification ensembles?" *IEEE Transactions on Cybernetics*, pp. 1–17, 2021.

[23] D. Hin, "Stackoverflow vs kaggle: A study of developer discussions about data science," *CoRR*, vol. abs/2006.08334, 2020. [Online]. Available: https://arxiv.org/abs/2006.08334

[24] B. H. Zhang, B. Lemoine, and M. Mitchell, "Mitigating unwanted biases with adversarial learning," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 335–340. [Online]. Available: https://doi.org/10.1145/3278721.3278779

[25] J. Chakraborty, S. Majumder, Z. Yu, and T. Menzies, *Fairway: A Way to Build Fair ML Software*. New York, NY, USA: Association for Computing Machinery, 2020, p. 654–665. [Online]. Available: https://doi.org/10.1145/3368089.3409697

[26] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 259–268. [Online]. Available: https://doi.org/10.1145/2783258.2783311

[27] M. Hardt, E. Price, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf

[28] M. Feffer, M. Hirzel, S. C. Hoffman, K. Kate, P. Ram, and A. Shinnar, "An empirical study of modular bias mitigators and ensembles," 2022. [Online]. Available: https://arxiv.org/abs/2202.00751

[29] P. J. Kenfack, A. M. Khan, S. A. Kazmi, R. Hussain, A. Oracevic, and A. M. Khattak, "Impact of model ensemble on the fairness of classifiers in machine learning," in *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, 2021, pp. 1–6.

[30] D. Bhaskaruni, H. Hu, and C. Lan, "Improving prediction fairness via model ensemble," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019, pp. 1810–1814.

[31] N. Grgic-Hlaca, M. Zafar, K. P. Gummadi, and A. Weller, "On fairness, diversity and randomness in algorithmic decision making," 06 2017.

[32] S. Tizpaz-Niari, A. Kumar, G. Tan, and A. Trivedi, "Fairness-aware configuration of machine learning libraries," ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 909–920. [Online]. Available: https://doi.org/10.1145/3510003.3510202

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] S. Butte, A. Prashanth, and S. Patil, "Machine learning based predictive maintenance strategy: A super learning approach with deep neural networks," in *2018 IEEE Workshop on Microelectronics and Electron Devices (WMED)*, 2018, pp. 1–5.

[35] M. O. Elish, T. Helmy, and M. I. Hussain, "Empirical study of homogeneous and heterogeneous ensemble models for software development effort estimation," *Mathematical Problems in Engineering*, vol. 2013, pp. 1–21, 2013.

[36] Kaggle.com., "Kaggle: Your machine learning and data science community." 2010. [Online]. Available: https://www.kaggle.com

[37] J. Chakraborty, S. Majumder, and T. Menzies, "Bias in machine learning software: Why? how? what to do?" in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 429–440. [Online]. Available: https://doi.org/10.1145/3468264.3468537

[38] "Uci ml adult census," 1994. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/adult

[39] "Kaggle titanic," 2018. [Online]. Available: https://www.kaggle.com/c/titanic

[40] "Uci ml bank marketing," 2012. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/bank+marketing

[41] M. Lichman, "Uci," 1994. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

[42] "Kaggle german credit," 2016. [Online]. Available: https://www.kaggle.com/uciml/german-credit

[43] R. Binns, "Fairness in machine learning: Lessons from political philosophy." [Online]. Available: https://arxiv.org/abs/1712.03586

[44] R. K. E. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. K. Lohia, J. Martino, S. Mehta, A. Mojsilovic, S. Nagar, K. N. Ramamurthy, J. T. Richards, D. Saha, P. Sattigeri, M. Singh, K. R. Varshney, and Y. Zhang, "Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias," *ArXiv*, vol. abs/1810.01943, 2018.

[45] T. Calders and S. Verwer, "Three naive bayes approaches for discrimination-free classification," *Data Mining and Knowledge Discovery*, vol. 21, pp. 277–292, 2010.

[46] F. Martínez-Plumed, C. Ferri, D. Nieves, and J. Hernández-Orallo, "Fairness and missing values," *arXiv preprint arXiv:1905.12728*, 2019.

[47] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, 2015, pp. 789–800.

[48] J. M. Zhang and M. Harman, ""ignorance and prejudice" in software fairness," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 1436–1447.

[49] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[50] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, p. 81–106, mar 1986. [Online]. Available: https://doi.org/10.1023/A:1022643204877

[51] N. Grgić-Hlača, M. B. Zafar, K. P. Gummadi, and A. Weller, "Beyond distributive fairness in algorithmic decision making: Feature selection for procedurally fair learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[52] K. V. Rashmi and R. Gilad-Bachrach, "Dart: Dropouts meet multiple additive regression trees," 2015.

[53] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[54] R. Berk, H. Heidari, S. Jabbari, M. Kearns, and A. Roth, "Fairness in criminal justice risk assessments: The state of the art," 2017. [Online]. Available: https://arxiv.org/abs/1703.09207

[55] J. Chakraborty, H. Tu, S. Majumder, and T. Menzies, "Can we achieve fairness using semi-supervised learning?" *CoRR*, vol. abs/2111.02038, 2021. [Online]. Available: https://arxiv.org/abs/2111.02038

[56] Z. Chen, J. M. Zhang, F. Sarro, and M. Harman, "Maat: A novel ensemble approach to addressing fairness and performance bugs for machine learning software," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 1122–1134. [Online]. Available: https://doi.org/10.1145/3540250.3549093

[57] H. Zheng, Z. Chen, T. Du, X. Zhang, Y. Cheng, S. Ti, J. Wang, Y. Yu, and J. Chen, "Neuronfair: Interpretable white-box fairness testing through biased neuron identification," in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, 2022, pp. 1519–1531.

[58] F. Tramer, V. Atlidakis, R. Geambasu, D. Hsu, J.-P. Hubaux, M. Humbert, A. Juels, and H. Lin, "Discovering unwarranted associations in data-driven applications with the fairtest testing toolkit," 10 2015.

[59] S. Biswas, M. Wardat, and H. Rajan, "The art and practice of data science pipelines: A comprehensive study of data science pipelines in theory, in-the-small, and in-the-large," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 2091–2103. [Online]. Available: https://doi.org/10.1145/3510003.3510057

[60] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, "Learning fair

representations," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 325–333. [Online]. Available: https://proceedings.mlr.press/v28/zemel13.html

[61] G. Pleiss, M. Raghavan, F. Wu, J. Kleinberg, and K. Q. Weinberger, "On fairness and calibration," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/b8b9c74ac526fffbeb2d39ab038d1cd7-Paper.pdf

[62] K. Holstein, J. Wortman Vaughan, H. Daumé, M. Dudik, and H. Wallach, "Improving fairness in machine learning systems: What do industry practitioners need?" in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–16.

[63] S. Biswas and H. Rajan, "Fairify: Fairness verification of neural networks," in *ICSE'2023: The 45th International Conference on Software Engineering*, May 14-May 20 2023.

[64] Y. Li, L. Meng, L. Chen, L. Yu, D. Wu, Y. Zhou, and B. Xu, "Training data debugging for the fairness of machine learning software," in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, 2022, pp. 2215–2227.

[65] X. Wang, N. Thain, A. A. Sinha, F. Prost, E. H. Chi, J. Chen, and A. Beutel, "Practical compositional fairness: Understanding fairness in multi-component recommender systems," in *WSDM 2021*, 2021. [Online]. Available: https://arxiv.org/pdf/1911.01916.pdf