# Diversified Ensembling: An Experiment in Crowdsourced Machine Learning

Ira Globus-Harris
University of Pennsylvania
Philadelphia, USA
Amazon Web Services Artificial Intelligence
Pasadena, USA

Declan Harrison
University of Pennsylvania
Philadelphia, USA
Amazon Web Services Artificial Intelligence
Pasadena, USA

Michael Kearns
University of Pennsylvania
Philadelphia, USA
Amazon Web Services Artificial Intelligence
Pasadena, USA

Pietro Perona
California Institute of Technology
Pasadena, USA
Amazon Web Services Artificial Intelligence
Pasadena, USA

Aaron Roth
University of Pennsylvania
Philadelphia, USA
Amazon Web Services Artificial Intelligence
Pasadena, USA

## ABSTRACT

Crowdsourced machine learning on competition platforms such as Kaggle is a popular and often effective method for generating accurate models. Typically, teams vie for the most accurate model, as measured by overall error on a holdout set, and it is common towards the end of such competitions for teams at the top of the leaderboard to ensemble or average their models outside the platform mechanism to get the final, best global model. In [12], the authors developed an alternative crowdsourcing framework in the context of fair machine learning, in order to integrate community feedback into models when subgroup unfairness is present and identifiable. There, unlike in classical crowdsourced ML, participants deliberately *specialize* their efforts by working on subproblems, such as demographic subgroups in the service of fairness. Here, we take a broader perspective on this work: we note that within this framework, participants may both specialize in the service of fairness and simply to cater to their particular expertise (e.g., focusing on identifying bird species in an image classification task). Unlike traditional crowdsourcing, this allows for the diversification of participants' efforts and may provide a participation mechanism to a larger range of individuals (e.g. a machine learning novice who has insight into a specific fairness concern). We present the first medium-scale experimental evaluation of this framework, with 46 participating teams attempting to generate models to predict income from American Community Survey data. We provide an empirical analysis of teams' approaches, and discuss the novel system architecture we developed. From here, we give concrete guidance for how best to deploy such a framework.

## CCS CONCEPTS

• **Human-centered computing → Collaborative and social computing**; • **Computing methodologies → Machine learning**; **Ensemble methods**.

## KEYWORDS

Crowdsourcing, Ensembling Methods, Fairness

**ACM Reference Format:**
Ira Globus-Harris, Declan Harrison, Michael Kearns, Pietro Perona, and Aaron Roth. 2024. Diversified Ensembling: An Experiment in Crowdsourced Machine Learning. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24), June 03–06, 2024, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3630106.3658923

## 1 INTRODUCTION

Competition platforms are a popular framework for generating accurate machine learning models through communal efforts. Kaggle is the most popular of these "crowdsourced" machine learning platforms, boasting fifteen million user accounts and thousands of competitions to date.[1] Companies and non-profits use the platform to publicly host competitions for learning tasks, often with rewards for the team with the highest performing model. One benefit of crowdsourcing models is that it gives a wider community access to the model development process: Kaggle, for instance, has been considered a mechanism for the "democratization" of data science to a broader audience, particularly in the context of crowdsourced models for tasks with societal utility [5]. However, due to the standard structure of these competition frameworks, they do not truly leverage the expertise of all the competitors, and fail to explicitly align improvements in model fairness with competition success. Here, we implement and provide an empirical analysis of an alternate framework which provides such mechanisms.

In [12], the authors provide an alternative algorithmic framework for crowdsourcing machine learning models, which we implement here. Their framework was specifically designed for contexts where unfairness, in the form of disparate accuracy of models across

[1]https://www.kaggle.com/

identifiable subgroups of the distribution, is of concern, and where a model would be considered “fair" if the model’s error on each group is close to the Bayes optimal error on that group.[2] In this framework, competitors compete against a global model $f$. At each round, they submit a function defining a group $g$ and a model $h$ which they claim has improved error compared to $f$ when restricted to the group $g$ (although $h$ need not improve on $f$ overall). If it does, as verified on a holdout set, then this pair $g$ and $h$ are incorporated via a natural ensembling technique into the model $f$, and this updated $f$ is used in subsequent rounds of the competition. The framework has attractive theoretical guarantees: each update decreases the overall error of the model, and so it is guaranteed to quickly converge to a state such that *either* $f$ is Bayes optimal, or else no competitor can distinguish it from Bayes optimal using any $(g, h)$ pair. Moreover, the updates can be made in a way so that error is monotonically decreasing not just overall, but simultaneously on all of the groups $g$ identified in the competition so far[3]. Note that this approach is more general than the standard “Kaggle” design, which corresponds to a competition where teams always submit $(g, h)$-pairs where $g(x) \equiv 1$.

This approach addresses two ways in which Kaggle-style crowdsourcing platforms fail to optimally direct the participants’ efforts. In standard competitions, the final model is reflective of individual teams’ efforts rather than a communal goal: one team to unilaterally “wins” the competition by proposing the most accurate model. It may well be that there are subregions of the dataset on which the best model is *not* the winning model, but another competitors’. Thus, Kaggle does not truly leveraging each competitors strengths, nor does it provide a mechanism for competitors to specialize on specific subtasks. This is particularly a failure from the perspective of democratizing data science, as we wish to provide mechanisms for individuals who have specific real-world insights or expertise relevant to the particular machine learning task to contribute to model development. For instance, if a learning task consisted of data on individuals throughout the world, individuals belonging to particular communities or who are experts on particular subareas may be able to better design models for those particular subgroups through data engineering. Ensembling the different competitors’ models into one final model is a natural partial solution to this shortcoming of standard crowdsourcing, as it allows for specialization in model development. In practice, winning teams in crowdsourcing competitions often do such model ensembling internally in an ad hoc way in order to leverage the strengths of different models: anecdotally, of the last eight Kaggle competitions with monetary prizes where models were published post-competition, five explicitly used some form of model ensembling ([1, 3, 6, 7, 11, 13, 14, 17, 20, 25]). The framework of [12] explicitly builds this ensembling into the competition format, rather than relying on ad hoc ensembling outside of the competition framework.

[2]Note that this is *not* a constrained optimization style notion of fairness, where, e.g., false positives are constrained to be equal across groups, and instead corresponds to group minimax fairness [9, 18]. While in some cases a constraint-based approach to fairness may be more appropriate than a minimax style approach, we note that in many contexts, one might argue that when the Bayes optimal model is substantially different across groups, (un)fairness is now a question of data engineering and perhaps different features and/or more data should be collected to mitigate performance differences across the subgroups of interest.

[3]For formal statements, see Theorems 10, 12, and 14 in [12].

A second motivation is that this ensembling method provides a mechanism to identify issues of unfairness and bias, and allows competitors to specialize. In the Kaggle-style framework, the only objective of interest is overall model accuracy. This reduces participants’ incentives to focus on small regions of the distribution where the model performance is sub-optimal: efforts explicitly identifying and correcting bias on small groups only pay out if your model wins the competition. Companies need reporting (and reward) mechanisms for when individuals identify cases where their models underperform on groups of interest. In the framework of [12], such improvements are rewarded, and the goal of overall accuracy is explicitly aligned with optimal performance on subgroups.

**Results** We provide an empirical case study of the general framework for crowdsourced machine learning proposed by [12]. In a real competition between 46 teams consisting of students at a major American university, we find that the final model outperforms all competitors’ models individually (which is only possible because of the explicit ensembling in the competition design), and that competitors leveraged the ability to specialize. This specialization was done through a combination of algorithmic and manual data engineering approaches, and most teams attempted to use contextual knowledge of the task (measuring income) in order to make improvements to their models. We describe the novel platform infrastructure we implemented in order to host the competition and discuss the nuances of constructing such an architecture in a scalable manner. We discuss practical challenges and lessons learned from hosting such a system, from denial of service attacks to setting acceptance criteria in order to properly incentivizing later engagement in the competition, as well as usability challenges.

**Related Work** The original model ensembling method in [12] was designed as a “bias bounties" competition, and was framed specifically as a method to combat unfairness. Here, we consider their framework more generally as a method to do crowdsourced machine learning while leveraging the strengths of all teams’ models, rather than purely a method to combat unfairness. While the work of [12] contained some preliminary experimental results in a very simplified framework, they did not include any true cross-team experiments in which a global model is built using multiple teams’ contributions. Here, we provide the software architecture necessary to run such a crowdsourcing competition.

The “bias bounty" idea used in [12] dates at least to a 2018 Vice editorial [21], and versions of the idea have been put into practice. In 2021, X (previously known as Twitter) released a bias bounties competition on their image cropping algorithm at DEFCON [8]. This competition had a substantially different framework than what is suggested here: a small number of competitors submitted written proposals, which were judged by a panel. Here, we consider larger-scale competitions where such empirical judging may be intractable (or legally fraught). Additionally, both the algorithmic framework of [12] and the system design discussed here tackle adversarial behavior by competitors in crowdsourcing competitions. Other work along these lines in more traditional crowdsourced machine learning include work on exploiting data leakage (and prevention mechanisms) in crowdsourcing by [19] and [15], and mechanisms for preventing overfitting in leaderboard-based competitions by [2], among others. Finally, the algorithmic framework proposed by [12] itself was independently and concurrently developed by [24].

Another similar idea to bias bounties is the "red teaming" of generative models to identify failure modes (e.g. poor performance on subgroups, leakage of confidential information, or code injection), which has recently been popularized: notably DEFCON 31 had a large redteaming event [23]. The goal of this form of redteaming is to identify model vulnerabilities, but does not aim to fix them. In comparison, our approach offers a principled way to incorporate fixes into a model, with formal optimality guarantees about the resulting ameliorated model.

**Limitations** The primary limitation of the crowdsourcing framework itself is that it gives no guidance for how to find improvements to the model. This is expected of any crowdsourcing framework: the competitors' goal is to identify improvements, and as the competition progresses, finding these improvements becomes more challenging. We note that while competitors may specialize and focus on such subgroups, they could instead consider improvements on the entire model. As such, this framework is only more general than the standard crowdsourcing framework: when useful, competitors may specialize, but otherwise can focus on model-wide improvements.

We emphasize that our framework is a proof-of-concept and that the recommendations we draw for future competitions are empirical in nature: we run a single competition, so are unable to make any statistical conclusions. Our competitors were given a relatively straightforward learning task on a tabular dataset where subgroups could be easily identified; in more complex tasks such as image identification, identifying subgroups may be more challenging.

Competitors were also students, not machine learning experts, and sometimes made seemingly counter-intuitive choices in their approaches, which we discuss in greater detail in Appendix D. We hope these observations may be more generally useful: one primary challenge of crowdsourcing as a means for democratizing data science is in the accessibility and usability of tools to those without computer science expertise [5]. We hope our observations may be used in subsequent crowdsourcing competitions and platforms in order to improve access and to avoid usability pitfalls.

## 2 PRELIMINARIES AND BACKGROUND

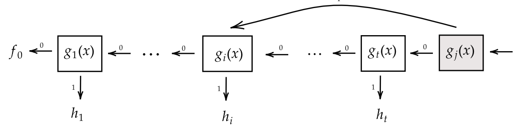


**Figure 1: Model ensembling procedure for submitted models. In gray, a repair node which is created when the $(g_t, h_t)$ update increases error on group $g_j$. The best previous version of model for $g_j$ has been tracked, and the model is repaired to point to this.**

For the purposes of our empirical study, we consider a competition for a regression problem. Formally, we consider a prediction task over a distribution of labelled examples $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ are features and $\mathcal{Y} \in \mathbb{R}$ are real-valued labels that will be predicted by a model $f$. Let $\mathcal{D} \in \Delta Z$ be the joint distribution over features and labels, and let $D \sim \mathcal{D}^n$ denote a finite set of $n$ labeled samples drawn i.i.d. from the distribution $\mathcal{D}$. We measure performance of the model $f$ by its mean squared error over the distribution, where the error for a single prediction $f(x)$ is defined as $\ell : \mathcal{X} \times Y \to \mathbb{R}$; $\ell(f(x), y)) = (f(x) - y)^2$, and average loss is denoted $\mathcal{L}(\mathcal{D}, f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)]$.

In [12], the authors propose an algorithmic framework termed a "bias bounty" for iteratively patching a predictor $f$ when regions with provably suboptimal performance are identified by auditors (See also [24] for a very similar proposed algorithm that they call "Prepend"). Formally, auditors are tasked with constructing $(g, h)$-pairs of functions: a *group* indicator function $g : \mathcal{X} \to \{0, 1\}$ and a *hypothesis* predictor $h : \mathcal{X} \to \mathcal{Y}$. We define the group loss of a predictor $f$ on $g$ as $\mathcal{L}(\mathcal{D}, f, g) = \mathbb{E}_{(x,y)\sim D}[(f(x) - y)^2 | g(x) = 1]$. We let $w = \mathbb{E}_{\mathcal{D}}[g(x)]$ be the *weight* of group $g$ over $\mathcal{D}$. If a competitor is able to construct a $(g, h)$-pair such that $w(\mathcal{L}(\mathcal{D}, f, g) - \mathcal{L}(\mathcal{D}, h, g)) > \alpha$, then the pair is *accepted* and the model $f$ is *updated.*

The algorithm for ensembling the model $f$ with the new model $h$ constructs a type of decision list with base node $f_0$, as shown in Figure 1: when a $(g, h)$ pair is accepted, a new decision node is prepended to the structure with group inclusion as the test on the prepended node. Then, the updated model $f'$ will predict $h(x)$ for all instances such that $g(x) = 1$, and $f(x)$ otherwise. We can iterate this process, allowing competitors to search for new $(g, h)$-pairs to reduce error on $f'$. Let $(g_i, h_i)$ denote the $i^{th}$ accepted pair to the model, and let $f_i$ denote the model after it has prepended $(g_i, h_i)$.

Since the proposed groups may not be disjoint, it is possible than an update $(g_k, h_k)$ may improve performance on group $g_k$ while decreasing performance on an earlier group $g_i$ due to group intersections. In order to avoid this, the model is iteratively patched after updates: for each $g_i$ with $i \leq j < k$, the model $f_j$ which performs best on $g_i$ tracked. After each update, if the model's error on any previously identified group has gotten worse, then the model is *repaired* by prepending a node with test for group inclusion of $g_i$ which "points" to $f_j$ if $g_i = 1$ and to $f_k$ otherwise. In Figure 1, the gray node is one such repair. This implies that once a $(g, h)$ pair is accepted into the overall model, the error rate on $g$ is non-increasing. We denote this as the "repair" procedure, and can make the following formal guarantee:

Theorem 2.1 ([12]). *Let $w_i = \mathbb{E}[g_i(x)]$ and let $\Delta_i = \mathcal{L}(\mathcal{D}, h_i, g_i) - \mathcal{L}(\mathcal{D}, f_i, g_i)$. If all accepted $(g_i, h_i)$ satisfy $w_i \Delta_i > \alpha$, then at most $1/\alpha$ submissions may be accepted, including repairs. Furthermore, if a group is introduced at round i, $\{\mathcal{L}(\mathcal{D}, h_i, g_i) - \mathcal{L}(\mathcal{D}, f_j, g_i); j > i\}$ is monotononically decreasing.*

In practice, we cannot verify improvements or track the group losses necessary for repairs over distributional loss. Instead, a sample of validation data, not accessible by competitors, is used to calculate all losses, and the above Theorem 2.1 is generalized for in-sample use.[4].

[4]See for example Theorems 12 and 14 in [12]

## 3 EMPIRICAL STUDY

We describe an empirical study conducted in an elective computer science course with a focus on algorithmic fairness at a prominent American research university during the spring semester of 2023. The study was deemed IRB exempt, and all students whose work is included in the subsequent analysis signed the consent form in Appendix F. The assignment instructions are given in Appendix G. Students were not monetarily rewarded, and the assignments were graded without knowledge of who had agreed to have their work included in the study. Identifying markers for participants have been removed.

### 3.1 Data Set and Prediction Task

Competitors worked on a regression task predicting annual income for individuals in Southern US states earning between \$0 and \$100,000, based on ACS PUMS data derived from the Python Folktables [10] package. The dataset contains 485,906 instances with twenty-one features, including sensitive attributes such as sex, race, age, and disability status. The full list of features included, along with their ACS encoding, are listed in Appendix C. Training, validation, and test splits were created with 70%, 15%, and 15% weights respectively. Training data was distributed to students, validation data was used to determine acceptance for model updates, and test data was used for post hoc model evaluation. Teams had access to the training data, the current global models' training errors, and the current global models' training predictions. They did not have access to these quantities for the validation data that was used to accept predictions or for the holdout data. While the raw dataset is publicly available online, teams were instructed to not access it, and reverse-engineering our exact learning task would have required effort. Additionally, this form of cheating would be easily detectable as we had access to all of the submitted code.

### 3.2 Competition Framework

The study contained a total of one hundred thirty-nine graduate and undergraduate students from various STEM disciplines, predominantly computer science and data science. Participants were split into forty-five teams of three to four students for the project, which students had slightly over a month to complete. The competition was structured to have teams compete in two ways.

The first way teams competed exactly mirrors the crowdsourcing framework described above: teams all worked to update a central, constantly up-to-date model, which we will call the *global* model. This model initially began as a relatively low-error gradient-boosting regressor trained by the course staff. Teams were given the initial training predictions from the global model, and tasked with constructing $(g, h)$-pairs to improve (subgroup) accuracy. If a team's $(g, h)$-pair was accepted, they were rewarded with points proportional to the reduction in the model's overall validation error. The model was updated to incorporate the $(g, h)$-pair, and a notification was sent to all participants containing the reduction in error as well as the global models' updated training predictions.

Every time a team submitted a $(g, h)$ pair to this global competition, that pair was also submitted to a *local* version of the competition specific to that particular group, which built an ensembled model using only that single teams' submissions. The initial local model for each team was a depth one decision tree fit on the training data, essentially predicting the mean label for all instances. Teams' local models were evaluated based on the validation error rate, and a leaderboard was displayed throughout the competition for teams to view the relative performance of their local model against others.

Of the 6914 $(g, h)$-pairs that the forty-five teams submitted, 3137 of them were submitted by Team 7, who automated a brute force approach. We discuss their approach briefly in 4.2, but omit their submissions in our analysis in order to give a clearer picture of the other teams' efforts.

### 3.3 Analysis of the Competition

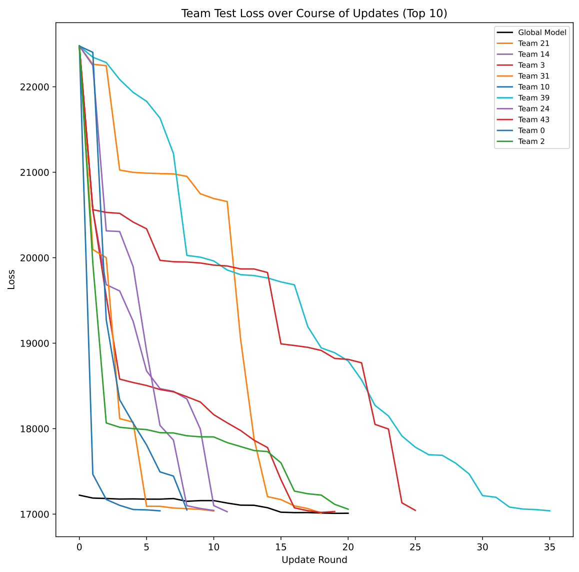


**Figure 2: Tracking test error of the top ten teams' local models per accepted update vs. global model. In the legend, the teams are ordered by the performance of their local model top-to-bottom; see Table 1 for final rankings of teams' models.**

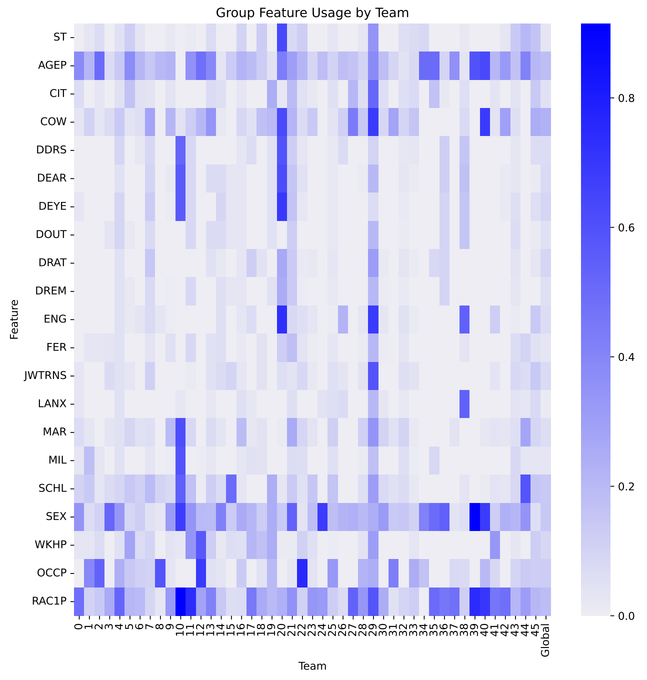


**Figure 3: Frequency of feature usage in group submissions by team. The features listed on the $y$-axis are explained in more depth in Appendix C. This plot is an analysis of all $(g, h)$-pairs submitted, not only those which were accepted.**

We provide empirical observations about the approaches employed by students in the competition. Based on these observations, we conclude with a concrete set of suggestions for how to create competition designs which are maximally usable for competitors.

**The global model outperformed all local models:** Ensembling the crowdsourced models produced a better model than individual teams' efforts. Eleven teams (including Team 7) managed to submit a total of twenty accepted updates to the global model, which outperformed all teams' local models at the completion of the study. Specifically, the best team had an overall squared error of 17012.32 on the holdout test data, while the global model had squared error of 17010.49–a slight improvement but an improvement nonetheless. This is possible only because the global model is explicitly ensembling contributions from multiple teams. We discuss performance of the global model on its accepted groups and its overall performance, and include more detailed comparisons to the best teams' local models in Appendix 8.

**Specializing models to subgroups helped:** Teams could choose to compete using only traditional Kaggle-style updates where the submitted $g$ represents the entire dataset. One team (team 10) did so: each of their six successful updates fully replaced their previous models. [5] Their final model performed well, 5th out of all groups, as shown in Figure 2. Thus, treating the competition as purely a Kaggle-style exercise was a relatively competitive strategy, but was not *the* most competitive strategy among our participants. This is further demonstrated in Figure 8 in Appendix B; which shows that accepted updates usually somewhat specialized. Furthermore, while the global model did accept some later updates over the entire dataset, they each triggered the repairs described in Section 2. In other words, there were models submitted earlier in the competition that performed better on their associated subgroup than later updates, and thus replacing the entire model with a later one naively would have caused an increase in error on those groups — even as it led to decreased overall error. The automated repair method corrects for this, which involves *re-ensembling* the submitted model with previous submissions, even when the most recently accepted submission corresponds to a trivial "group" corresponding to the entire dataset. Thus *even when competitors who choose to ignore the ability to target subgoups $g$ and instead submit Kaggle-style updates have their updates accepted, the resulting model is still an ensemble, which is accuracy-improving.*

[5]They did also initially try smaller subgroups, so this is not seen in Figure 4 which shows group weights of all groups including those not accepted. They then changed their approach to only focus on improvements to the whole model.

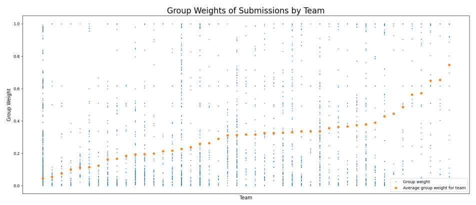


**Figure 4: Distribution of weights ($\mathbb{E}_{\mathcal{D}}[g(x)]$) of all groups submitted by teams (including rejected submissions), with teams sorted by average group weight. Each vertical line corresponds to a single team, and the blue dots are the weights of the groups they submitted. The orange dots are the average weight of groups submitted by the teams.**

**Most teams specialized, and did so differently:** Over the course of the competition, a total of 896 updates were made across the 46 teams, ranging from 5 to 42 updates per team. As seen in Figure 4, the sizes of groups submitted on average differed significantly between teams, indicating that teams broke the problem up differently. More explicitly, in Figure 3, we see the distribution over submissions of which features teams used to define their groups $g$, measured by frequency. A complete mapping of the feature acronyms to their meanings is provided in Appendix C; in the figure while features such as race (RAC1P), binary sex (SEX), and age (AGEP) were commonly targeted by a majority of teams, we see subsets of teams focusing on other features such as education level (SCHL) and disability status (primarily DDRS, DEAR and DEYE). We discuss this specialization in greater detail in Appendix B.

**Most teams employed a combination of manual, automated, and learned approaches:** Teams had multiple approaches for identifying groups of datapoints to make improvements on. They could *manually* identify regions where the model performed poorly, e.g. by conditioning on different features to find regions the model might be specified to perform poorly on. Or, they could use an *algorithmic* approach, e.g. specifying some class $\mathcal{G}$ of possible groups and then trying to learn the group $g$ in $\mathcal{G}$ where the current model performed worst. Teams were given a file with potential algorithmic approaches they could attempt and were encouraged to explore their own methods.[6] Purely algorithmic approaches such as learning clusters of datapoints where performance was suboptimal were largely less successful: only three of the 19 $(g, h)$-pairs accepted to the global model were automated, and in general automated updates had an acceptance rate of 13.6 percent, as opposed to the 25.8 percent acceptance rate for manual updates. Automated approaches were also less popular in terms of overall number of submissions: only 21.8 percent of the submitted updates were automated approaches from 37 unique teams.

**Groups were often chosen using contextual knowledge:** One approach many (roughly 85 percent) of the teams had was to condition across the feature space using their knowledge of the context of the prediction task. For instance, many teams chose to examine subgroups related to gender or race. In particular, of the 821 submissions that used race as a predicate for the group, 327 (39%) subsetted over African Americans, and when binarized sex was a predicate, it was overwhelmingly (96% of the time) used to subset for female-identified individuals. In their write-ups, students stated that they believed that these subgroups might, due to systemic bias or discrimination, have disparate pay, and hence these features might be leveraged to improve the model. The efficacy of this approach is arguable, as discussed in greater detail in Appendix D: specialization to narrow subgroups did help teams find updates, but often this required careful examination of the data.

**Finding Later Updates and Identifying Promising Subgroups is Hard** As in any competition, we expected teams to struggle to find improvements to the model as the competition wore on. This prediction was correct: at the end of the month only eleven of forty-five teams managed to have updates accepted on the global model. One primary challenge that students wrote about was how to balance specialization with generalization: if the subgroups of interest are too small, then restricting training to that subgroup will likely overfit. Specialization only helps if you have reason to believe that this subgroup has some generalizable structure that the larger model has yet to find, and knowing when this might occur is challenging.

**Final Models Perform Similarly While Making Different Predictions** While the model desired from a large scale competition would be crowdsourced from all competitors, we constructed local models to reflect efforts from individual teams for purposes of grading. However, these local models produced an interesting phenomena; the leaderboard for the most accurate local models shows a narrow margin compared to the crowdsourced global model yet these models make substantially different predictions. In Figure 9, we plot the absolute difference in predictions between the global model and the top five local models to see a non-trivial density of instances for which the models disagree.

## 4 PLATFORM DESIGN

In this section, we introduce a platform for hosting the competitions. The traditional method for authenticated user interaction with a server is to develop a *full stack* solution; a comprehensive system covering web development, database management, back end software, and more. Naturally, these systems require expertise to build and have serious implications when incorrectly constructed. Context management services, such as *Wordpress*, reduce these requirements but have monthly fees and frequently contain vulnerabilities.[7] Instead, our platform leverages GitHub to host competitions, gaining the security and web interface inherent to GitHub. The open-source package and detailed installation instruction are available for download at https://github.com/Declancharrison/Bias-Bounties-Template.

### 4.1 Platform Design

In our construction, competitions are hosted on private GitHub repository cloned by the organizer from our package template. During the package installation, the organizer connects a backend computation source, e.g. an AWS EC2 instance or personal Linux machine, to the repository using a continuous integration and deployment tool, *GitHub Actions*. By utilizing *Actions*, we "solve" two major components of a secure web-server stack: data base management and a front-end interface.

Since competitions are run inside *private* repositories, an organizer may easily add and remove participants through the GitHub GUI, avoiding the expertise needed to manage an SQL database. Additionally, the competition platform gains GitHub's user authentication protocols; accessing a competition implies repository read permissions and there are currently no known vulnerabilities for reading private repositories without appropriate permissions.

Due to the expected background of participants in ML competitions, we assume basic familiarity and likely comfort with GitHub. One challenge of a competition is the global model needs to be accessible to competitors, be constantly updated, and be able to accept competitors $(g, h)$-pair submissions. GitHub trivially solves these problems through *push* and *pull* requests to a repository. Our platform stores necessary competition information such as the current global model's training predictions and the leaderboard in such a repo, and participants interact with the repository to submit $(g, h)$ pairs through pull requests. The detailed format of these pull requests is described in Appendix E, but, at its core, participants upload models to a cloud provider (like Google Drive or AWS S3) and provide links to each models in their request. Feedback is provided to participants via comments on pull requests and any successful update forces a push to the repository with updated model information. The submission-feedback loop with backend protocol is shown in Figure 5.

As an added bonus factor, GitHub allows users to create websites from HTML and Markdown code in their repository, which allows us to easily integrate a public leaderboard and general competition information in a digestible format for users.

### 4.2 Security Protections & Vulnerability Concerns

In order to evaluate whether or not a submitted $(g, h)$-pair ought to be integrated into the model, a competitors' code has to be run on the server on the validation data. As a result, there is a risk of an adversarial competitor's malicious code being run on the host machine. While this risk cannot be entirely mitigated, we describe

[6]Algorithmic methods file is listed in code repository listed in Appendix.

[7]https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=wordpress

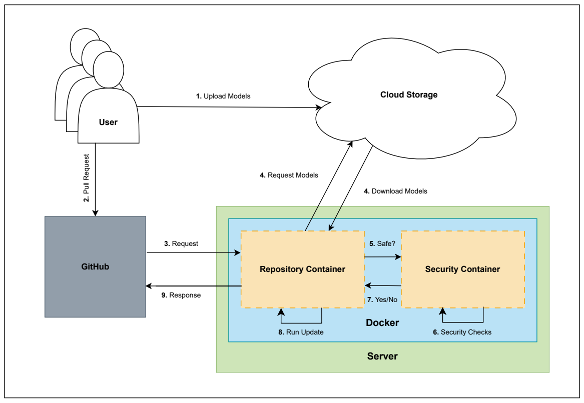


**Figure 5: Protocol for $(g, h)$ pair submissions from participants to a competition repository.**

baseline security measure implementations and describe future methods for securing systems.

First, GitHub is used to authenticate all teams. Thus, the primary layer of security is that the competitors are verified. Within a classroom context this worked well: if a student had submitted malicious code, we would have been able to trace it back to the student and their grade would have been impacted, which was a suitable disincentive. In deployment outside this setting, we suggest hosts require users to sign a legal agreement prior to participation as is standard practice in ML competitions.

Secondly, the competition is run within a user-mode Docker container with limited kernel privileges. When a (g,h) pair is submitted from a participant, the files are downloaded into the Docker container. The models are passed to a second Docker container with no internet access to the run the security checks from the following paragraph to determine if malicious code is identified. This protection ensures the host machine is not affected from code run in the competition but does not protect the repository files.

Lastly, models are checked on load for malicious behavior to include loading or importing unnecessary packages e.g. *sys* or *os*. Since models are saved and loaded as serialized byte streams, evaluating model intentions is a difficult task as it is done at the opcode level via a disassembler. We intend to increase the breadth of checks done in this section through updates to our package.

In practice, the primary security issue we had was an inadvertent denial of service attack by the team who automated submissions. This led to a long queue for other teams' pairs to be verified. In the future, we would implement submission limits per team (per day and overall), to avoid this.

## 5 LESSONS LEARNED

We provide a general overview of lessons learned from the framework's deployment, both from a systems perspective and for optimizing competitors' engagement with the platform.

**Distribute Environment Files.** A major factor in this competition is being able to pass ML models between participants and the server. Distribution of a makefile that constructs a virtual environment or a Docker container for participants to work helps this process run smoothly. While this is a relatively standard practice for running code across platforms, we emphasize its' importance: using different versions of packages (or Python) caused major frustrations for participants as it often resulted in denial of submissions for "security" errors.

**Limit daily submissions.** As discussed in greater detail in Section 4.2, Team 7 took an automated, brute-force approach that caused a Denial of Service. In order to prevent this, as well as to incentivize competitors to only send in submissions that they truly believe are competitive, it would be wise to limit the number of daily submissions. Additionally, this reduces overfitting to the validation set, as it prevents hillclimbing.

**Prime competitors to think critically about group identification.** Since competitors were students in a class which focused largely on fairness in machine learning, they were primed to think about the effect of societal factors. As discussed in Section 3.3, this led students to, e.g., almost exclusively condition on binarized sex being female when considering sex as a predicate. While competitors should be encouraged to use their general knowledge to identify regions of disparate performance, doing so effectively may require careful data engineering, and looking for general performance improvements is also useful.

**Setting the $\alpha$ threshold for acceptance is a tricky, data-dependent task.** The competition designers set a threshold $\alpha$ which determines how much of an improvement an update must incur in order to be accepted. While we do not have the counterfactual of setting different values for our $\alpha$, re-simulating the competition with higher or lower values led to lower performance and overfitting respectively. Before beginning a competition, hosts should try various threshold values for their given task and loss function with trusted users to observe what value gives the best generalization/performance balance.

**Alter reward system to scale over time.** Competitors received points based on the amount of validation error decreased by an accepted update on the global model. Intuitively, teams that participated earlier in the study were able to make more updates to the global model: notably, Team 12 made the first seven updates to global model. However, as the competition progressed, the problem became more difficult and teams weren't as able to decrease the error of the model. In order to bolster effort throughout the duration of the competition, rewards could scale by both the amount of error reduced as well as the number of previous updates made or time since start of the competition.

## 6 DATASET

In the process of running the competition, we generated a database of nearly seven thousand $(g, h)$-pairs which competitors submitted, which may be of independent interest. The dataset consists of 6914 $(g, h)$-pairs of models which make predictions over the income task described in Section 3, where the $g$'s describe subsets of the distribution and the $h$'s are a variety of different models trained over those subsets.

In general, large datasets of machine learning models may be of academic interest. Kaggle itself has a Meta Kaggle[8] dataset, which is a public dataset of competitions and submissions on the platform

[8]https://www.kaggle.com/datasets/kaggle/meta-kaggle

and which has been widely used, e.g. in [16] and [22]. We believe that, in a similar spirit, our dataset is also of use. In particular, the fact that it provides a large collection of subgroups in addition to models offers multiple use cases.

First, the fairness literature often assumes fairness guarantees are desired with respect to some rich class of groups $\mathcal{G}$. In practice, these papers usually either contain no experimental results at all, or results with respect to extremely minimal and usually disjoint groups: e.g. race, binarized sex, or the two-way marginals of race and binarized sex. Here, we have a much richer collection of many thousands of groups which span many features, and suggest that these may be used for benchmarking fairness approaches.

Secondly, we note that the ensembling framework proposed in [12] was independently developed by [24] in the context of multi-group learning. There, they provide an additional algorithm that frames multi-group learning as a form of sleeping experts, where each $(g, h)$-pair corresponds to an expert, and each expert is “awake" when $g(x) = 1$. This formulation is beneficial, as the reduction of sleeping experts to the offline setting gives an algorithm leads to improved sample complexity compared to the decision-list style updates proposed in [12]. However, this work is theoretical in nature, and does not provide experimental guarantees: in particular, it assumes that $|\mathcal{G}|$ is finite and that $\ell \circ \mathcal{H}$ has bounded pseudodimension, and requires computations over a large number of $(g, h)$-pairs. It is not clear how much improvement this methodology would give in practical settings. Our dataset offers one way of measuring this.

Thus, the dataset of model-pairs and its associated training and test data, may be used as both a form of fairness benchmark and as a mechanism to evaluate ensembling methods and expert learning more generally.

## 7 ETHICAL CONSIDERATIONS STATEMENT

As discussed in Section 3 in greater detail, this work is considered IRB exempt. Participants were not monetarily compensated, and grades were given without knowledge of which students had agreed to have their work included. From a pedagogical perspective, we deemed this project to be a good use of student time: crowdsourcing-style projects provide a mechanism to get hands-on machine learning experience in a context that is more open-ended than a standard implementation assignment, and it gave students an opportunity to grapple with the nuances of bias and disparate performance of machine learning models. Throughout the semester, students also completed assignments related to other notions of fairness in machine learning. All identifying characteristics of student submissions have been omitted in the analysis.

## 8 ADVERSE IMPACT STATEMENT

Crowdsourcing is an abstract framework for model development, which might be used in a variety of different contexts, and it is up to the hosts of competitions to decide whether or not a particular model ought to be built or not. As a mechanism for achieving fairness with respect to subgroups, our framework assumes that these groups are identifiable from the feature space and are present in the dataset. It should not be assumed to be a universal fix for a model's disparate accuracy on any (potentially unidentifiable) subgroup, and in some use-cases, alternate fairness notions that are not targeted by our framework may be more appropriate.

## ACKNOWLEDGMENTS

We give warm thanks to Peter Hallinan for many helpful conversations.

## REFERENCES

[1] Benji Andrews, Hema Natarajan, Maggie, Ron Ellis, and Ryan Holbrook. 2023. Benetech - Making Graphs Accessible. https://kaggle.com/competitions/benetech-making-graphs-accessible

[2] Avrim Blum and Moritz Hardt. 2015. The Ladder: A Reliable Leaderboard for Machine Learning Competitions. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 1006–1014. https://proceedings.mlr.press/v37/blum15.html

[3] Aaron Carman, Alexander Heifler, Ashley Chow, and Ryan Holbrook. 2023. ICR - Identifying Age-Related Conditions. https://kaggle.com/competitions/icr-identify-age-related-conditions

[4] US Census. October 20, 2022. *2021 ACS PUMS Data Dictionary*. Data Dictionary. US Census. https://www2.census.gov/programs-surveys/acs/tech_docs/pums/data_dict/PUMS_Data_Dictionary_2021.pdf

[5] Sophie Chou, William Li, and Ramesh Sridharan. 2014. Democratizing data science. In *Proceedings of the KDD 2014 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA*. 24–27.

[6] Ashley Chow, Glenn Cameron, Manfred Georg, Mark Sherwood, Phil Culliton, Sam Sepah, Sohier Dane, and Thad Starner. 2023. Google - American Sign Language Fingerspelling Recognition. https://kaggle.com/competitions/asl-fingerspelling

[7] Ashley Chow, Eduard Trulls, Jevster, Kwang Moo Yi, Sohier Dane, Tanji Gou, and Weiwei Sun. 2023. Image Matching Challenge 2023. https://kaggle.com/competitions/image-matching-challenge-2023

[8] Rumman Chowdhury and Jutta Williams. 2021. Introducing Twitter's First Algorithmic Bias Bounty Challenge. https://blog.twitter.com/engineering/en_us/topics/insights/2021/algorithmic-bias-bounty-challenge

[9] Emily Diana, Wesley Gill, Michael Kearns, Krishnaram Kenthapadi, and Aaron Roth. 2021. Minimax group fairness: Algorithms and experiments. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 66–76.

[10] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. 2021. Retiring Adult: New Datasets for Fair Machine Learning. *Advances in Neural Information Processing Systems* 34 (2021).

[11] Alex Franklin, David Gagnon, Maggie, Meg Benner, Natalie Rambis, Perpetual Baffour, Phil Culliton, Scott Crossley, and Ulrich Boser. 2023. Predict Student Performance from Game Play. https://kaggle.com/competitions/predict-student-performance-from-game-play

[12] Ira Globus-Harris, Michael Kearns, and Aaron Roth. 2022. An algorithmic framework for bias bounties. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. 1106–1124.

[13] Addison Howard, Archit Agarwal, Ashley Chow, Rantig, Kellen J Gracey, Robert JC Brown, and Sohier Dane. 2022. GoDaddy - Microbusiness Density Forecasting. https://kaggle.com/competitions/godaddy-microbusiness-density-forecasting

[14] Addison Howard, Jevster, Katherine Gustilo, Katy Borner, Ryan Holbrook, and Yashvardhan Jain. 2023. HuBMAP - Hacking the Human Vasculature. https://kaggle.com/competitions/hubmap-hacking-the-human-vasculature

[15] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. 2012. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6, 4 (2012), 1–21.

[16] Dominik Kowald, Matthias Traub, Dieter Theiler, Heimo Gursch, Emanuel Lacic, Stefanie Lindstaedt, Roman Kern, and Elisabeth Lex. 2019. Using the Open Meta Kaggle Dataset to Evaluate Tripartite Recommendations in Data Markets. *arXiv preprint arXiv:1908.04017* (2019).

[17] Alex Lourenco, Brent Seales, Christy Chapman, Daniel Havir, Ian Janicki Janicki, JP Posma, Nat Friedman, Ryan Holbrook, Seth P., Stephen Parsons, and Will Cukierski. 2023. Vesuvius Challenge - Ink Detection. https://kaggle.com/competitions/vesuvius-challenge-ink-detection

[18] Natalia Martinez, Martin Bertran, and Guillermo Sapiro. 2020. Minimax pareto fairness: A multi objective perspective. In *International Conference on Machine Learning*. PMLR, 6755–6764.

[19] Arvind Narayanan, Elaine Shi, and Benjamin I. P. Rubinstein. 2011. Link prediction by de-anonymization: How We Won the Kaggle Social Network Challenge. In *The 2011 International Joint Conference on Neural Networks*. 1825–1834. https://doi.org/10.1109/IJCNN.2011.6033446

[20] Joe Ng, Carl Elkin, Aaron Sarna, Walter Reade, and Maggie Demkin. 2023. Google Research - Identify Contrails to Reduce Global Warming. https://kaggle.com/competitions/google-research-identify-contrails-reduce-global-warming

[21] Amit Elazari Bar On. 2018. We Need Bug Bounties for Bad Algorithms. https://www.vice.com/en/article/8xkyj3/we-need-bug-bounties-for-bad-algorithms

[22] Rebecca Roelofs, Vaishaal Shankar, Benjamin Recht, Sara Fridovich-Keil, Moritz Hardt, John Miller, and Ludwig Schmidt. 2019. A meta-analysis of overfitting in machine learning. *Advances in Neural Information Processing Systems* 32 (2019).

[23] Austin Carson Sven Cattell, Rumman Chowdhury. 2023. https://aivillage.org/generative%20red%20team/generative-red-team/

[24] Christopher J Tosh and Daniel Hsu. 2022. Simple and near-optimal algorithms for hidden stratification and multi-group learning. In *International Conference on Machine Learning*. PMLR, 21633–21657.

[25] Bojan Tunguz, Dieter, Karnika Kapoor, Parul Pandey, Paul Mooney, Phil Culliton, Rob Mulla, Sanyam Bhutani, and Will Cukierski. 2023. 2023 Kaggle AI Report. https://kaggle.com/competitions/2023-kaggle-ai-report

## A REPRODUCIBILITY

Code to reproduce primary results and figures may be located at https://github.com/Declancharrison/Diversified-Ensembling-Reproducibility.

## B ADDITIONAL ANALYSIS OF MODEL PERFORMANCE

We provide additional plots and discussion for the empirical analysis. As discussed in Section 3.3, the final global model outperformed all other teams' models when mean squared error was evaluated on the holdout test dataset. This, as well as a more detailed comparison of the global models' overall error versus those of the top 10 local teams is shown in Table 1. Here we see that the global model does indeed outperform the others on the holdout test loss. Additionally, we can inspect how much overall error is improved over rounds of updates, for both the global model and for the different teams' models. This is shown in Figure 6.

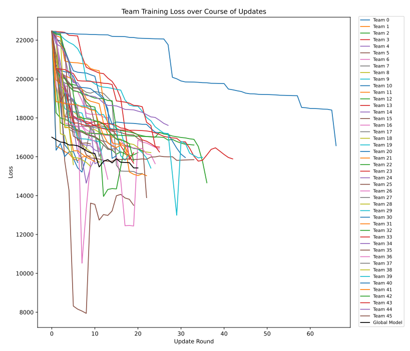
As this ensembling procedure is a mechanism for specialization and could be done in an effort at achieving good performance for subgroups of interest, it's also useful to inspect the performance of the global model on all of the subgroups which were accepted. In Figure 7, we show this. On the left, group loss is plotted with respect to the validation dataset. Because the validation data is used to determine the threshold for accepting groups and due to the repair procedure described in section 2 and formalized in Theorem 2.1, the group loss of a group after it has been introduced into the ensembled model is guaranteed to monotonically decrease. This is shown in Figure 7: once the lines turn solid, which corresponds to the group being introduced into the model ensemble, then the lines become non-increasing. On the right of Figure 7 is the group error on an entirely held-out test dataset, and we see that with the exception of group 20 and a slight change for group 1, monotonicity of group error improvement is largely preserved. The fact that it is not for group 1 and 20 is not overly surprising, as both of these groups made up less than 1 and 0.1 percent of the dataset respectively, and hence our generalization guarantees are weaker.

These figures also show the submitted groups overlap, as otherwise group errors wouldn't be impacted by later updates, and that despite the overall error improvements being quite small, some subgroups are quite impacted by the model changes. This supports the idea that allowing specialization in your crowdsourced framework can make meaningful impacts to performance on subgroups.

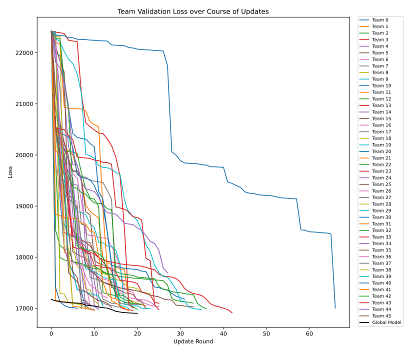
In Figures 8 and 9, we examine teams' specialization further. In Figure 8, the weights of groups who were accepted is plotted for both teams' local models (in blue) and the global model (in orange). Here, we see that some successful updates are over the entire dataset or over substantial portions of it, while others are much smaller, only using 1 to 20 percent of the dataset. In Figure 9, we visualize how the global model differs from the top 5 best local models. We see that despite the fact that the test loss on each of the 5 models is quite close, they do in fact make substantially different predictions for many points.

| Team | Training Loss | Validation Loss | Test Loss | Number of Updates |
|---|---|---|---|---:|
| Global Model | 15424.70 | 16900.12 | 17010.49 | 20 |
| Team 21 | 16052.41 | 16948.26 | 17012.32 | 18 |
| Team 14 | 15757.40 | 16977.12 | 17028.13 | 11 |
| Team 3 | 15667.19 | 16957.37 | 17030.65 | 19 |
| Team 31 | 15752.59 | 16959.99 | 17037.82 | 10 |
| Team 10 | 15706.57 | 16997.82 | 17038.68 | 6 |
| Team 39 | 15971.36 | 16968.34 | 17039.29 | 35 |
| Team 24 | 15907.08 | 16971.54 | 17043.18 | 10 |
| Team 43 | 16245.32 | 16972.86 | 17044.38 | 25 |
| Team 0 | 15987.90 | 16982.91 | 17049.55 | 8 |
| Team 2 | 16210.24 | 16979.03 | 17058.09 | 20 |

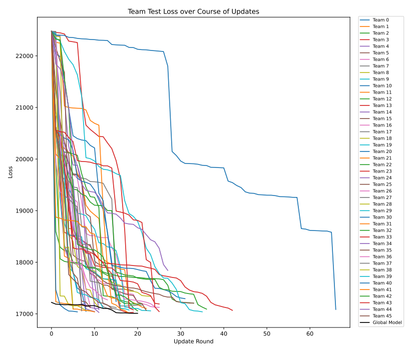
**Table 1: Final leaderboard of top ten participating teams sorted by increasing test error with number of updates made to each model.**



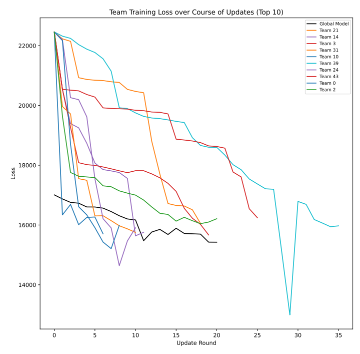
(a) train



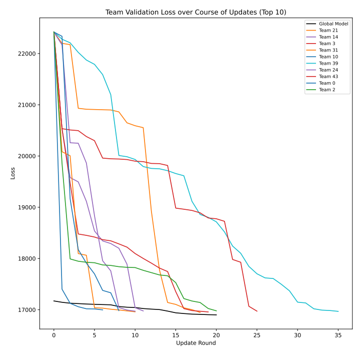
(b) validation



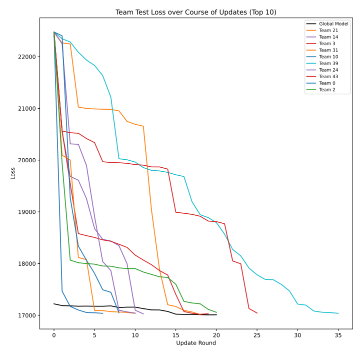
(c) test



(a) train (top ten)



(b) validation (top ten)



(c) test (top ten)

**Figure 6: Error of global and teams’ models measured at each accepted updated. Top row of subfigures consists of all teams while the bottom row displays the top ten teams and the global model. The $y$-axis is squared error, the $x$-axis is the round of accepted submission.**

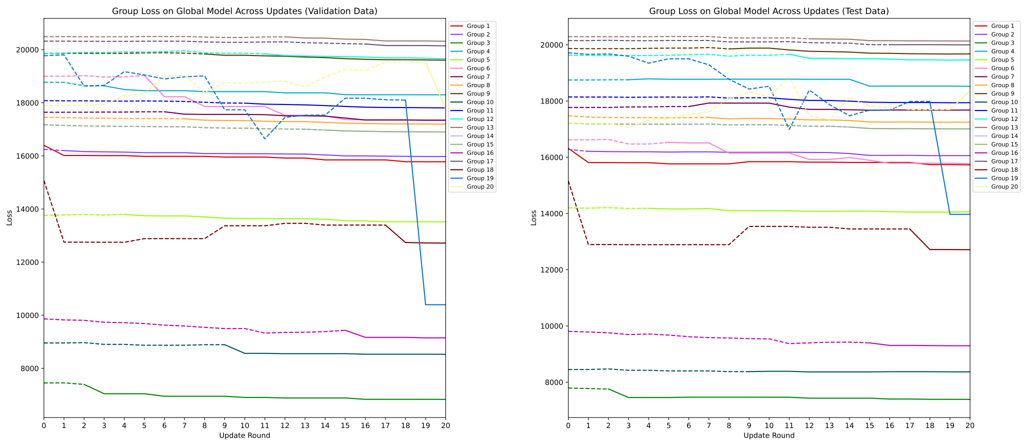


**Figure 7: Loss of the global model on introduced groups across rounds of updates, on the validation data used to determine acceptance of proposed models and on holdout test data. For each group introduced, the loss on that group is dotted until the round at which it was introduced, at which point the line becomes solid. Groups are ordered in the order in which they were introduced, e.g., Group 12 is the group that was specified in the 12th accepted update to the global model.**

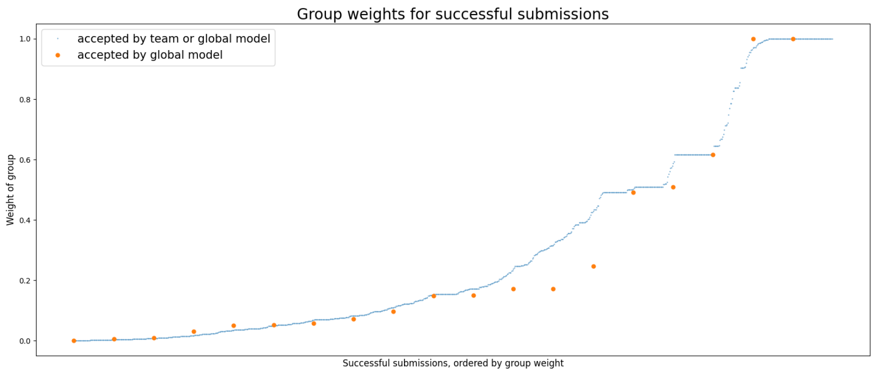


**Figure 8: Distribution of group weights for successful submissions. On the $y$-axis, weight of the group submitted, on the $x$-axis, individual submissions, as ordered by weight. The blue dots correspond to successful updates to either the global model or the team model, while orange dots correspond to submissions to the global model only.**

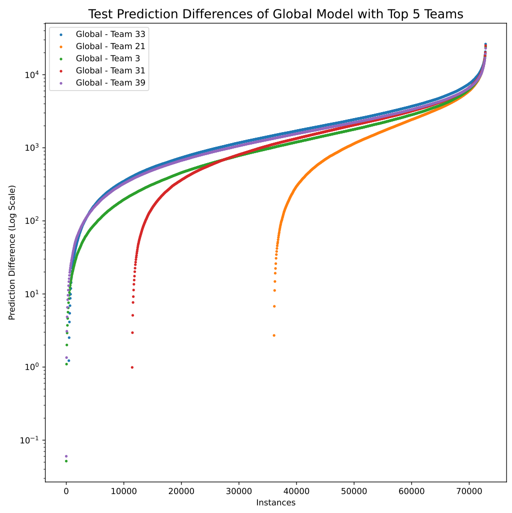


**Figure 9: Test prediction disagreement between the global model and top five local models. On the $x$-axis are individual datapoints, and on the $y$ axis the (absolute) difference in prediction of $y$ by the global model and an individual teams' model as measured on this point. The $y$-axis is log-scaled, and the disagreement is plotted for each of the top 5 teams. The datapoints are sorted on the $x$-axis so that the difference in prediction is monotonic. We see that while the top 5 teams had test loss that was relatively close to the global models' (see Table 1), they make substantially different predictions for many points.**

## C DATA FEATURES

In the following Table 2, the full list of features used in the income prediction task students were competing over are listed. The exact categories within each feature and the formal descriptions of categories listed here are from the PUMS data dictionary ([4]).

| Feature | Feature Description |
|---|---|
| ST | State and territory codes |
| AGEP | Age |
| CIT | Citizen status |
| COW | Class of worker |
| DDRS | Self care difficulty |
| DEAR | Hearing difficulty |
| DEYE | Vision difficulty |
| DOUT | Independent living difficulty |
| DRAT | Veteran service connected disability rating |
| DREM | Cognitive difficulty |
| ENG | Ability to speak English |
| FER | Gave birth to child within the past 12 months |
| JWTRNS | Means of transportation to work |
| LANX | Language other than English spoken at home |
| MAR | Marital status |
| MIL | Military service |
| SCHL | Educational attainment |
| SEX | Binary sex |
| WKHP | Usual hours worked per week past 12 months |
| OCCP | Occupation code |
| RAC1P | Race code |

**Table 2: Features from ACS PUMS survey used for the competitions' income prediction task.**

## D ADDITIONAL OBSERVATIONS

As our competitors were students and not machine learning experts, some of the choices they made are perhaps counter-intuitive. We list some of these choices here.

**Most teams never attempted to make global improvements to the model:** Students were primarily graded in terms of their improvements to their team model, as opposed to the global model, as we anticipated that updates to the global model would become rapidly difficult to find for many students. Since their team models were initially trained as decision tree stumps with marginally more predictive power than predicting the mean label, a valid strategy would have been to simply improve this model over the entire dataset. However, only 15 of the 45 teams attempted updates over the whole dataset.

**Teams may not have carefully examined their group functions:** For instance, out of the 222 updates which assigned $g$ to the entire dataset, 42 were complex logical combinations of many features, which also happened to include a disjunction of, e.g., binarized sex labels, and hence the entire dataset. This could have been due to mistakes manually conditioning over group membership (e.g. missing parentheses in the logical operations or a swap of an OR and an AND). Or it might be due to automated processes that teams used to identify groups, leading to a complex-looking group that can be much simplified. In either case, it might indicate that teams were not carefully engaging with what they were submitting.

Similarly, of the 95 submissions which considered the binary feature for presence of a hearing disability (which were submitted by 22 distinct teams), *all of them* specified groups considered exclusively the hearing loss feature marginalized on *not* having hearing loss. The presence of a disability might have predictive power that conceivably could be ignored by a larger model, since they make up a minority population (about 2.3 percent for hearing loss). However, predicating on *not* having hearing loss amounts to considering the majority of the dataset, and one might postulate that this is not a meaningful form of specialization. Students did not discuss this choice in their write up. However, the way ACS PUMS categorizes disability is a bit unusual: they are categorical variables with value 1 being presence of the disability, and value 2 being absence, rather than a binary encoding. Thus without carefully checking the categorization of values, students might have misunderstood the meaning of values, though we have no way of verifying this.

We expect that capping submissions per day would somewhat mitigate this, as it incentivizes teams to be relatively confident in their submission and hence inspect it closely.

**Teams conflated societal inequity with disparate model performance** As discussed in Section 3.3, teams often chose to specialize on groups who face systemic income inequity. While the performance of machine learning models may very well be disparate across such groups, it is not necessarily always the case. For instance, if there is a strong signal in the training data that income inequity between two sufficiently large groups exists, a model trained on the entire dataset will likely use this signal. The bigger issue here is often in the prediction task itself, the selection of features used for the task, or under-representation in the dataset. Finding disadvantaged groups where the existing training data *can* be leveraged to improve performance in a way that a model trained over the entire dataset wouldn't capture is potentially nontrivial, especially without access to external data sources.

## E PLATFORM DEEP DIVE

In this section, we expand on deeper technical details of the platform introduced in Section 4. The end goal of this section is to provide the reader an explanation for all steps listed in Figure 10 in an end-to-end platform loop from submitting $(g, h)$-pairs to receiving feedback.

*E.0.1 Submitting and testing $(g, h)$ pairs.* As previously mentioned in Section 4, the method for submitting a $(g, h)$-pair for update is through a pull request on the competition's repository. Once a participant has trained a $(g, h)$-pair of models, they upload them to a cloud storage space such as Google Drive or AWS S3 with unauthenticated accessibility. To submit their pair, participants create a pull request on the repository hosting the competition by inserting the links to their models in the file */competitors/request.yaml* next to the variables *g_url* and *h_url*. The platform infrastructure will download and test this pair and provide feedback to the participant through a comment on their pull request.

## E.1 Repository permissions and GitHub Actions integration

Repository permissions are set such that all members with access are able to instantiate a pull request but only workspace administrators may push to the repository, following GitHub's standard structure for making updates to an open-source repository. In order to automatically test updates from participants in a competition, the platform utilizes GitHub Actions to spawn a *workflow* script anytime a user creates a pull request. Our specific workflow first checks for any file changes made outside of the */competitors/request.yaml* file used for submissions, and runs the backend procedure for testing $(g, h)$-pair updates on models.

*E.1.1 Docker Integration.* An important implementation detail not discussed in Section 4 is the usage of Docker in our platform. In order to protect the host machine and standardize the operating environment, the server back-end computations are run inside docker containers. The *repository* container maintains the file system for teams models, interacts with *GitHub Actions* when participants submit requests, and pushes information changes to GitHub when updates are made. The *security* container's sole purpose is to run forward passes of submitted models to indicate to the repository container if the model is safe or potentially unsafe. The security container runs a primitive security check on models, scanning for malicious keywords and opcodes. The security container is not connected to the internet in order to reduce potential risks to the host machine and competition results.

*E.1.2 End-to-End Submission Workflow.* Using Figure 10 as reference, we expand on the steps in the platform feedback loop for submitting pairs to a competition repository.

(1) User saves and uploads trained g,h models to cloud storage with publicly accessible URLs.
(2) User submits pull request to GitHub repository editing *competitors/request.yaml* file to include g,h model URLS.
(3) GitHub Actions notifies Repository Container of pull request from user.
(4) Repository Container downloads changed files and downloads models from URLs.
(5) Repository Container notifies Security Container of new models and requests forward pass.
(6) Security Container runs primitive security checks on models and ensures outputs are correctly formatted.
(7) Security Container informs Repository Container of safe/unsafe models
(8) If models are safe, Repository Container loads team and global models into memory and attempts updates. If unsafe, skip step.
(9) Repository Container closes/makes comments on Pull Request and deletes branch. If updates are accepted, changes are pushed to GitHub.

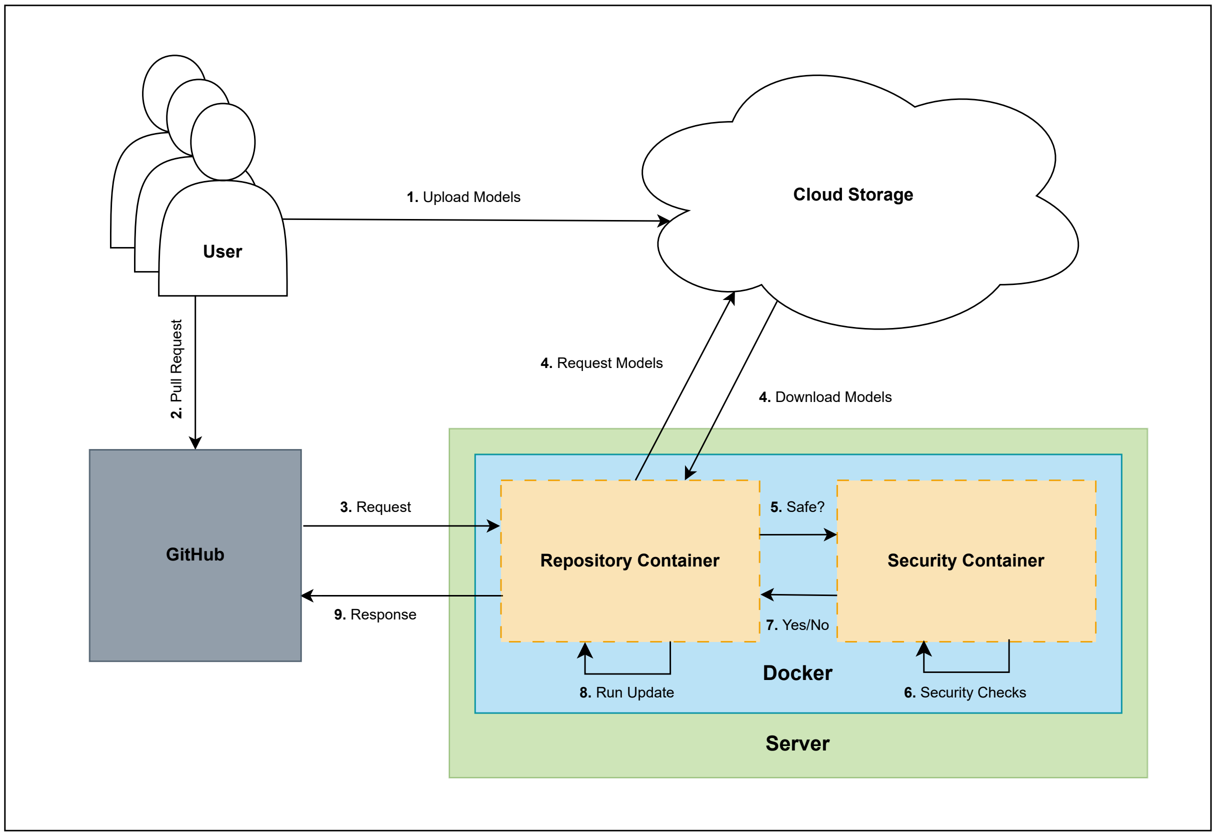


**Figure 10: Protocol for (g,h) pair submissions from participants to a competition repository.**

## F CONSENT FORM AND IRB EXEMPTION

This study was deemed IRB Exempt under category 1;

Below is the consent form that students signed for the purposes of this study.

**Project Title:** Bias Bounties Analysis

### F.1 Summary and Purpose of the study

This research is being conducted by (omitted for anonymization) at (omitted for anonymization). The purpose of this research is to understand the heuristic approaches and methodologies used to isolate regions where machine learning models perform poorly. Your participation will not require any work additional to your work in the course and will have no risks or benefits to you directly, but may lead to better understanding of how to improve machine learning models in the future. Your participation is voluntary and will not affect your grade in the course. Your work will be de-identified, and as there is no personal information present in your machine learning models, re-identification is unlikely. The data will be stored, and could be used for future research.

### F.2 Procedures

We will analyze your machine learning models submitted as part of the final course project for (omitted for anonymization). We may also look at your (de-identified) source code and final project write-ups to make qualitative conclusions about your approaches.

### F.3 Confidentiality

While personally identifiable information in this context is extremely minimal, your names will not be associated with your models in the context of the research study, and no information about individuals will be accessible from the published work.

### F.4 Benefit and Risks to Participants

There are no specific benefits or risks to you in particular if you join this study. Whether or not you do participate will not affect your grade in this class.

### F.5 Right to Withdraw

Your participation in this research is completely voluntary. You may refuse to participate or withdraw at any time before the end of the semester.

### F.6 Contact information for questions

(Omitted for anonymization)

## G PROJECT DESCRIPTION

Here we include the project description that students received at the outset of the assignment.
**Bias Bounties Project** *Due date: April 21st, 2023*

### TASK

In this project, your team will act as “Bias Bounty Hunters" for a regression model trained with sole intent of minimizing empirical loss (*root mean squared error*). The model accepts US Census data from the Folktables package with predefined features and predicts the annual income of an individual. Your team is tasked with finding *certificates of suboptimality* in order to reduce the correctable bias in the model. These certificates take the form $(g, h)$, where $g$ is a group indicator function, $h$ is a regression model, and $h$ performs strictly better on $g$ than the current model.

### CLASS SEMANTICS + NOTATION

Since we expect the communal model to quickly converge to approximate Bayes-Optimal, each team will also have a *team model*. We will refer to the communal predictive model as the *global model* and each team's model as their *private model*. Both of these models are Pointer Decision Lists which follow the update procedure we discussed in class. Each time a team submits a potential update to the global model, the update will also be attempted on the team's private model. This is to incentivize all teams to continue searching for updates since you will be graded on your private model's accuracy (and given extra credit based your teams change to the global model's accuracy).

### SUBMITTING CERTIFICATES OF SUBOPTIMALITY

In order to submit potential $(g, h)$ pair updates, you will create a pull request on the GitHub repository with the file /competitors/request.yaml altered to reflect the URLs of your models. If any other file is altered, the pull request will immediately be denied. For explicit, detailed instructions, please refer to the GitHub README file.

### DELIVERABLES

Each team will deliver one report with the information in the template populated. The template contains formatting for your report as well as descriptions on how to fill out each section. Any amount of information which goes over the section's word limit will not be used when grading. While a Bias Bounty Competition is generally focused around individual teams' updates to the global model's accuracy, this course is not expecting to produce expert ML practicioners. Your private model's accuracy will be included in the overall grading scheme but we are much more interested in *how* your team attempted to find updates. Furthermore, while we want your team to find working methods, **we also care about what methods you tried that did not work since this is a heavily understudied field.** You should document your methods as you work on the project!

### CHEATING POLICY

The (university ommitted) Code of Academic Integrity is in effect and attempted malicious actions will not be taken lightly. If you or a member of your team are caught cheating or acting maliciously, your team will be removed from the project and will receive a score of 0.