MCGrad: Multicalibration at Web Scale

Niek Tax
Meta Platforms, Inc.
London, United Kingdom
niek@meta.com

Lorenzo Perini
Meta Platforms, Inc.
London, United Kingdom
lorenzoperini@meta.com

Fridolin Linder
Meta Platforms, Inc.
Munich, Germany
flinder@meta.com

Daniel Haimovich
Meta Platforms, Inc.
London, United Kingdom
danielha@meta.com

Dima Karamshuk
Meta Platforms, Inc.
London, United Kingdom
karamshuk@meta.com

Nastaran Okati
Max Planck Institute for Software Systems
Kaiserslautern, Germany
nastaran@mpi-sws.org

Milan Vojnovic
Meta Platforms, Inc.
London, United Kingdom
The London School of Economics and Political Science
London, United Kingdom
m.vojnovic@lse.ac.uk

Pavlos Athanasios Apostolopoulos
Meta Platforms, Inc.
Menlo Park, California, USA
pavlosapost@meta.com

Abstract

We propose MCGrad, a novel and scalable multicalibration algorithm. Multicalibration - calibration in subgroups of the data - is an important property for the performance of machine learning-based systems. Existing multicalibration methods have thus far received limited traction in industry. We argue that this is because existing methods (1) require such subgroups to be manually specified, which ML practitioners often struggle with, (2) are not scalable, or (3) may harm other notions of model performance such as log loss and Area Under the Precision-Recall Curve (PRAUC). MCGrad does not require explicit specification of protected groups, is scalable, and often improves other ML evaluation metrics instead of harming them. MCGrad has been in production at **Meta**, and is now part of hundreds of production models. We present results from these deployments as well as results on public datasets. We provide an open source implementation of MCGrad at https://github.com/facebookincubator/MCGrad.

CCS Concepts

• **Computing methodologies → Machine learning**.

Keywords

Multicalibration, Calibration, Uncertainty Quantification, Fairness

ACM Reference Format:
Niek Tax, Lorenzo Perini, Fridolin Linder, Daniel Haimovich, Dima Karamshuk, Nastaran Okati, Milan Vojnovic, and Pavlos Athanasios Apostolopoulos. 2026. MCGrad: Multicalibration at Web Scale. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD 2026), August 9–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770854.3783954

1 Introduction

A machine learning model is said to be *calibrated* when its predictions match the true outcome frequencies [2, 39, 56]. The importance of calibration to ML-based systems has been widely recognized [24, 34, 38], including in web applications such as content ranking [40, 48], recommender systems [11, 23, 57], digital advertising [7, 9, 17] and content moderation [36, 41, 59]. Calibration is necessary for such systems to make optimal decisions without under- or overestimating risk or opportunity.

Multicalibration is a more powerful property that extends the concept of calibration to ensure that predictors are simultaneously calibrated on various (potentially overlapping) groups [29, 32, 46, 49]. Often, multicalibration is applied as a post-processing step to fix a base classifier's predictions. While it originated in the study of algorithmic fairness, its significance extends far beyond it. A growing body of work shows that multicalibration can increase model performance and robustness in many circumstances, ranging from out-of-distribution prediction to confidence scoring of LLMs [14, 16, 22, 35, 60, 61].

Despite its potential, multicalibration has received limited traction in industry. We argue that this is due to three main reasons. **First**, existing multicalibration methods require *protected groups* to be *manually defined* [31]: not only do users need to specify the covariates that require protection (e.g., `user age` and `user country`), but also they must define concrete protected groups through binary membership indicators, such as "`is an adult in the US?`". This poses a significant challenge to real-world application of multicalibration because practitioners (a) may not have established the precise set of protected groups, (b) may be subject to changes to the definition of protected groups over time thus requiring ongoing effort and change management (e.g., due to changes related to legal context, policy frameworks, ethical considerations, or otherwise), or (c) may only seek an overall performance improvement without prioritizing specific groups for protection. The difficulties defining protected groups become even more pronounced when the set of features that we want to protect is large. **Second**, existing methods lack the ability to scale to large datasets and/or many protected groups, which makes them hard to deploy in production. For instance, Hebert-Johnson et al. [29]'s algorithm scales at least linearly in time and memory with the number of groups, being potentially inefficient when a large set of groups is specified. To be deployed at web scale, multicalibration methods must be highly optimized so as to not introduce significant computational requirements at training or inference time. **Third**, existing multicalibration methods lack guardrails to be safely deployed, and risk harming model performance (e.g., due to overfitting), which may prevent practitioners from deploying them altogether.

In this paper, we introduce MCGrad (MultiCalibration Gradient Boosting) a novel multicalibration algorithm deployed in production that is lightweight and safe to use. MCGrad requires only the specification of a set of protected features, rather than predefined protected groups. It then identifies miscalibrated regions within this feature space, and calibrates a base classifier's predictions over all groups that can be defined based on these features. On a high level, MCGrad uses a Gradient Boosted Decision Tree (GBDT) algorithm recursively, such that the result converges to a multicalibrated predictor (Section 3.1). By using a highly optimized GBDT implementation, like LightGBM [33], MCGrad inherits scalability and regularization (Section 3.2). Finally, it employs an early stopping procedure to avoid overfitting, ensuring that model performance is not harmed (Section 3.3).

Empirically, on benchmark datasets, we show that MCGrad improves the base predictor's outputs in terms of multicalibration, calibration, and even predictive performance (Section 4). We complement this analysis by describing the impact of MCGrad on real-world data: MCGrad has been used in production at **Meta** on hundreds of machine learning models, and in total, generates over a million multicalibrated real-time predictions per second (Section 5). MCGrad successfully addresses all three previous challenges and, to the best of our knowledge, this is the largest-scale adoption of multicalibration in production. Our deployment results show significant impact: on 24 of 27 models tested via A/B testing on our Looper platform [43], MCGrad significantly outperformed Platt scaling, leading to promotion of the MCGrad-calibrated variants to production. Additionally, across 120+ models on our internal ML platform, MCGrad improved log loss for 88.7% of models, PRAUC for 76.7%, and Expected Calibration Error for 86.0%. With these real-world results, we add to the growing body of evidence that shows that multicalibration can significantly contribute to model performance. We also provide valuable practical learnings from applying multicalibration in industry (Section 6), and link multicalibration to related areas (Section 7).

2 Background on (Multi)Calibration

Let $\mathcal{X} = \mathbb{R}^d$ be a d-dimensional input space and $\mathcal{Y} = \{0, 1\}$ the binary target space. Let $D = \{(x_i, y_i)\}_{i=1}^n \sim p(X, Y)$ be the dataset with n i.i.d. samples, where the random variables X, Y represent, respectively, a d-dimensional feature vector and the target. A probabilistic predictor f is a map $f\colon \mathcal{X} \to [0, 1]$, which assigns an instance $x \sim p(X)$ to an estimate of its class conditional probability $p(Y = 1 \mid X = x)$. For any probabilistic predictor f, we denote with F its logit (referred to as the predictor), i.e., $f(x) = \mu(F(x))$, where $\mu(F) = 1/(1 + e^{-F})$ is the sigmoid function. We assume a base probabilistic predictor f_0 is given; this predictor is the one targeted for calibration via a post-processing procedure.

Let $\mathcal{L}\colon \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ be the log loss (negative log-likelihood), i.e., for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and any predictor F,

$$\mathcal{L}(F(x), y) = -[y \log(\mu(F(x))) + (1 - y) \log(1 - \mu(F(x)))].$$

We use the symbol $\mathbb{E}$ for the expectation with respect to an arbitrary distribution that is clear from the context, and $\mathbb{E}_D$ for the sample mean over D.

Calibration. A predictor f is perfectly calibrated if and only if $\mathbb{P}(Y = 1 \mid f(X) = p) = p$, where p is the true underlying probability. Intuitively, for all input pairs $(x, y) \in D$, if f predicts 0.8, we expect that 80% of them have 1 as label. Recently, Tygert [58] introduced the Estimated Cumulative Calibration Error (ECCE), a novel parameter-free metric for measuring calibration that computes the maximum difference between the means of labels and predictions over any score interval:

$$\mathrm{ECCE}(f) = \frac{1}{n} \max_{1 \le i \le j \le n} \Big| \sum_{k=i}^{j} (y_{(k)} - f(x_{(k)})) \Big|, \tag{1}$$

where $(x_{(1)}, y_{(i)}), \ldots, (x_{(n)}, y_{(n)})$ are the points $\{(x_i, y_i)\}_{i \in D}$ ordered such that $f(x_{(1)}) \le \cdots \le f(x_{(n)})$.

This metric enjoys useful statistical properties. In particular, it can be standardized by dividing it by the following scaling statistic:[1]

$$\sigma(f) = \sqrt{\frac{1}{n^2} \sum_{k=1}^{n} f(x_k)(1 - f(x_k))}.$$

This gives rise to two ways of interpreting the metric. The absolute scale $\mathrm{ECCE}(f)$ is designed to measure the magnitude of miscalibration. When the data size increases, $n \to \infty$, $\mathrm{ECCE}(f)$ converges to a true magnitude of miscalibration (0 if f is calibrated and a positive value otherwise). The sigma scale ${}^{\mathrm{ECCE}(f)}/_{\sigma(f)}$ however converges to a distribution when f is calibrated, and diverges to infinity when it is miscalibrated. The two scales are used to answer different questions. ECCE is used to answer the question "is the miscalibration large?". ${}^{\mathrm{ECCE}(f)}/_{\sigma(f)}$ is used to answer the question "is there statistical evidence of miscalibration?". For example, we might have that $\mathrm{ECCE}(f_1) = 0.05 = 20\sigma(f_1)$ and $\mathrm{ECCE}(f_2) = 0.05 = 2\sigma(f_2)$. This would tell us that while the magnitude of miscalibration on the data appears to be similar for f_1, f_2, the result is statistically significant for f_1 but not for f_2, which could simply be due to random sampling. See [1, 58] for further discussion.

Multicalibration. Unlike calibration, multicalibration considers the intersection of the *score interval* and the *protected groups*. Let $\mathcal{G} \subset \{g : [0, 1] \to \{0, 1\}\}$ be the set of *interval membership functions*, i.e. $g(v) = \mathbb{1}_I(v)$ for $v \in [0, 1]$, and any interval $I \subseteq [0, 1]$.

[1]Notice that $\sigma(f)$ is the standard deviation of the sample mean of prediction labels with prediction probabilities according to f. For a calibrated predictor f, the expected value of ${}^{\mathrm{ECCE}(f)}/_{\sigma(f)}$ scales as c, for large data size n, with $c = 2\sqrt{2/\pi} \approx 1.6$ [1].

Similarly, let $\mathcal{H} \subseteq \{h : \mathcal{X} \to \{0,1\}\}$ be a set of *group membership functions* over $\mathcal{X}$, such that a point $x \in \mathcal{X}$ belongs to a group $h \in \mathcal{H}$ if and only if $h(x) = 1$. The groups can be overlapping, i.e., some point $x \in \mathcal{X}$ may belong to multiple groups.

Multicalibration is defined in various ways across the literature [3, 13, 22, 26, 29, 53]. Here, we employ an operational definition of multicalibration:

Definition 2.1 (Multicalibration). A probabilistic predictor f is α-multicalibrated (α-MC) with respect to $\mathcal{H}$ if, for the MC-deviation given as $\Delta_{h,g}(f) \coloneqq \left|\mathbb{E}\left[h(X)g(f(X))(Y - f(X))\right]\right|$ and a given scale parameter $\tau_h(f)$:

$$\Delta_{h,g}(f) \le \alpha\tau_h(f) \quad \forall h \in \mathcal{H}, g \in \mathcal{G}.$$

We say that f is *multicalibrated* if $\alpha = 0$.

A common choice is to use a uniform bound on the MC-deviation, which is accommodated by choosing the scale parameter $\tau_h(f)$ to be a constant equal to 1. In Appendix A, we show that Definition 2.1, with $\tau_h(f)^2 = \mathbb{E}\left[h(X)f(X)(1 - f(X))\right]$, naturally matches with Multicalibration Error (MCE) by Guy et al. [25], which quantifies multicalibration error by computing the maximum ECCE over the protected groups with a suitable normalization:

$$\mathrm{MCE}(f) = \max_{h \in \mathcal{H}} \frac{\mathrm{ECCE}_h(f)}{\sigma_h(f)} \tag{2}$$

where $\mathrm{ECCE}_h(f), \sigma_h(f)$ are defined as $\mathrm{ECCE}(f), \sigma(f)$ above but restricted to the group h. Roughly speaking, MCE measures the strongest statistical evidence of miscalibration across protected groups. The unit of the MCE is the same as for ECCE/σ. It can be re-scaled to an absolute measure by multiplying with $\sigma(f)$.

Using a multicalibration metric that is equivalent to the definition of α-MC can help ensure that empirical work is aligned with theory. In Section 4 we use MCE to measure multicalibration, thereby directly estimating the minimal α for which a model is α-MC. To our knowledge, this is the first empirical multicalibration study that achieves this. See Appendix A for further discussion.

3 MCGrad: A Practical Algorithm for Multicalibration

This paper tackles the following problem:

Given a labeled dataset D, an initial probabilistic predictor f_0;
Return a probabilistic predictor f that is α-MC w.r.t. $\mathcal{H}$.

Designing an algorithm for multicalibration without specifying a set of groups is challenging for four reasons. First, the absence of groups requires the multicalibration model to automatically determine the regions of the feature space where the base model is highly miscalibrated. Second, adjusting the model's output to improve its calibration in a specific region might harm the predictions over other regions, thus increasing the multicalibration error. Third, a post-processing algorithm has to be fast and lightweight to prevent large memory consumption or increased latency of model inference. Fourth, while existing algorithms claim to achieve multicalibration on the training set, they are subject to overfitting issues.

Our proposed algorithm MCGrad, presented in Algorithm 1, is a lightweight algorithm that multicalibrates a base model without requiring the specification of the groups on which the model has to be calibrated. MCGrad relies on a key observation: by including the base model's predictions as a feature, the loss function of the multicalibration algorithm captures the feature values that correspond to miscalibrated initial predictions. In addition, decreasing the loss function implies correcting the base model's predictions for some regions of the feature space. Because corrections in some regions may negatively impact others, MCGrad runs multiple rounds, using the previous round's processed predictions as input. Since GBDTs are regularized using shrinkage (or a step size < 1), a simple rescaling step reduces the number of trees and rounds required, while having a negligible impact on overfitting. Finally, MCGrad employs early stopping to avoid overfitting.

Algorithm 1 MCGrad

```
Input: a probabilistic predictor $f_0$, a dataset $D$;
  $D_{train}, D_{valid}$ = train_validation_split($D$)
  $F_0 = \mu^{-1}(f_0)$ # inverse sigmoid transformation
  $\varepsilon_{-1} = +\infty$, $\varepsilon_0 = \mathbb{E}_{D_{valid}}\left[\mathcal{L}(F_0(X), Y)\right]$ # initialize errors
  $t = 1$ # initialize number of rounds for early stopping
  while $\varepsilon_{t-1} - \varepsilon_t > 0$ do
      $h_t(x, f_{t-1}(x))$ = fit GBDT on $\{((x_i, f_{t-1}(x_i)), y_i)\}_{i \in D_{train}}$
      $\theta_t = \arg\min_\theta \mathbb{E}_{D_{train}}\left[\mathcal{L}(\theta \cdot (F_{t-1}(X) + h_t(X, f_{t-1}(X))), Y)\right]$
      $F_t(x) = \theta_t \cdot (F_{t-1}(x) + h_t(x, f_{t-1}(x)))$
      $f_t(x) = \mu(F_t(x))$
      $\varepsilon_{t+1} = \mathbb{E}_{D_{valid}}\left[\mathcal{L}(F_t(X), Y)\right]$
      $t \leftarrow t + 1$
  end while
  # Found $t - 1$ to be the best number of rounds
  for $s$ in $[1, \ldots, t-1]$ do
      $h_s(x, f_{s-1})$ = fit GBDT on $\{((x_i, f_{s-1}(x_i)), y_i)\}_{i \in D}$
      $\theta_s = \arg\min_\theta \mathbb{E}_D\left[\mathcal{L}(\theta \cdot (F_{s-1}(X) + h_s(X, f_{s-1}(X))), Y)\right]$
      $F_s(x) = \theta_s \cdot (F_{s-1}(x) + h_s(x, f_{s-1}(x)))$
      $f_s(x) = \mu(F_s(x))$
  end for
  $f(x) = \mu(F_{t-1}(x))$
Output: $f$;
```

3.1 Achieving Multicalibration with T Rounds

Our intuition builds on the following two key insights. First, GBDT returns a solution f that approximately sets the gradient of the loss function $\mathcal{L}$ to zero:

$$\mathbb{E}_D\left[h(X)(Y - f(X))\right] = 0 \qquad \forall h \in \mathcal{H},$$

where $\mathcal{H}$ is the set of all regression trees over the feature space.

Second, if we augment the feature space $\mathcal{X}$ with a single additional feature $f_0(x)$, then the GBDT approximately achieves

$$\mathbb{E}_D\left[h(X, f_0(X))(Y - f_1(X))\right] = 0 \qquad \forall h \in \mathcal{H}' \tag{3}$$

where $\mathcal{H}' \subset \{h : \mathcal{X} \times [0,1] \to \mathbb{R}\}$ is the space of trees with an extra real-valued input, and $f_1(x) = \mu(F_0(x) + h_1(x, f_0(x)))$. This creates a bridge between (a) group and interval membership functions and (b) regression trees over the augmented feature space. Strictly speaking, for any $h \in \mathcal{H}, g \in \mathcal{G}$ there exists $h' \in \mathcal{H}'$ such that $h(x)g(f(x)) = h'(x, f(x))$ for $x \in \mathcal{X}$.

Perhaps surprisingly, Eq. (3) is equivalent to 0-multicalibration according to Definition 2.1 under the condition $f_1 \equiv f_0$. However, this condition may not always hold: *correcting f_0 for some regions* *identified by $h(x, f_0(x))$ returns a predictor f_1 that might be miscalibrated on regions identified by its own predictions $h(x, f_1(x))$.* To resolve this, we create a loop over multiple rounds: in each round t, we train a GBDT model f_t using the features x and the previous round's predictions f_{t-1}, such that, after T rounds, we have

$$\mathbb{E}_D\left[h(X, f_{T-1}(X))(Y - f_T(X))\right] = 0 \qquad \forall h \in \mathcal{H}^*$$

where $f_T(x) = \mu(F_{T-1}(x) + h_T(x, f_{T-1}(x)))$.

Intuitively, if for some probabilistic predictor f^*, $f_T \to f^*$ as $T \to \infty$, then for all $h \in \mathcal{H}^*$, $\mathbb{E}_D\left[h(X, f_{T-1}(X))(Y - f_T(X))\right] \to \mathbb{E}_D\left[h(X, f^*(X))(Y - f^*(X))\right] = 0$ as $T \to \infty$, which is our desired condition. This suggests that, for sufficiently large T, the probabilistic predictor f_T should be approximately multicalibrated. See Appendix B for further intuitions and theoretical results.

3.2 Fast Training and Prediction

To be deployed at web scale, there are strict requirements for an algorithm to be efficient both at training and inference time. Several design choices ensure that MCGrad meets these requirements.

Efficient gradient boosting. First, the algorithm has been designed to rely on a relatively small number of calls to a GBDT, delegating the most compute intensive steps to one of many highly optimized GBDT implementations (e.g. [8, 33]). This differentiates it from existing implementations such as [29]. In our implementation, we use LightGBM.

Rescaling the logits. GBDTs are regularized in multiple ways, including using a step size smaller than 1 for scaling each additional tree. While this helps to avoid overfitting, it results in a predictor which can be improved by rescaling it by a factor slightly greater than 1. As a result, the next round will attempt to apply this rescaling. Since linear rescaling is not easily expressed by decision trees, this may require many trees, ending up requiring unnecessarily many trees to achieve multicalibration.

We introduce a simple rescaling after every round:

$$\theta_t = \arg\min_{\theta} \mathbb{E}_D\left[\mathcal{L}(\theta \cdot (F_{t-1}(X) + h_t(X, f_{t-1}(X))), Y)\right]$$

where h_t is obtained with LightGBM. We call the round t's learned predictor (on the logits) as $F_t = \theta_t(F_{t-1} + h_t)$.

Note that this constant is typically very close to 1 and has a very limited detrimental effect on regularization. Rescaling is not the same as using a step size of 1: it affects the whole sum of trees (i.e., the whole predictor), while the step size only targets each tree sequentially (i.e., each weak learner).

3.3 Preventing Overfitting

Modern GBDT algorithms support various methods for regularization, such as limiting the growth of the trees, the leaf splits, and the number of trees. MCGrad's recursive structure can give rise to additional overfitting beyond standard GBDTs, which we address in the following ways.

Early stopping. While multiple rounds are required for convergence, they also increase the capacity of the model. With T rounds and M trees in each round, the MCGrad model is at least as expressive as a tree ensemble with $T \cdot M$ trees. Since the model capacity of tree ensembles is Ω(number of trees) (see [54]), overfitting is more likely for a large number of rounds.

We solve this problem with a standard early stopping procedure on the number of rounds T. Specifically, we split i.i.d. the dataset D into training (D_{train}) and validation (D_{valid}) sets, and determine the number of rounds T by taking the last round before the expected loss over the validation set increases. That is,

$$T = \min\{t : \mathbb{E}_{D_{valid}}\left[\mathcal{L}(F_{t+1}(X), Y) - \mathcal{L}(F_t(X), Y)\right] > 0\} - 1,$$

where the models f_t are obtained using D_{train} instead of D. Using early stopping in a real-world deployed system has a relevant consequence: *MCGrad does not harm the base model's prediction*, and, instead, would select $T = 0$ as optimal number of rounds if the first step decreased the initial performance:

$$\mathbb{E}_{D_{valid}}\left[\mathcal{L}(F_1(X), Y) - \mathcal{L}(F_0(X), Y)\right] > 0 \implies T = 0$$

which means that $f_T = f_0$.

Regularizing through the min sum Hessian in leaf. Augmenting the data with the previous round's model necessarily gives rise to regions of the augmented feature space that are particularly prone to overfitting. Consider a leaf of a partially constructed tree in GBDT. The leaf splitting algorithm can choose to split the leaf on values of the previous predictor $f_t(x)$. In this leaf, the left tail $f_t(x) < a$ for some small a will contain only negative labels even if the true distribution may assign a positive probability to a positive label (and analogously for the right tail). In that case the new model can improve the likelihood simply by assigning a very low probability to this tail, as low as zero. Common regularization strategies, like setting the minimum number of samples per leaf, are insufficient to address this scenario. We solve this problem by using a more targeted form of regularization that limits the minimum total Hessian in a leaf, which is offered as one of multiple regularization techniques in LightGBM. The Hessian in a leaf S equals $\sum_{(x,y)\in S} f_t(x)(1 - f_t(x))$ and, as such, is a refined version of the simple sample size rule $|S|$. As the predicted probabilities become close to 0 or 1, the total Hessian becomes smaller and split in multiple leaves can no longer be considered. As shown in our ablation studies (Section 4) this regularization reduces overfitting and improves performance.

4 Benchmark Experimental Analysis

We evaluate MCGrad on both public and production data. In this section, we present experiments on benchmark datasets and compare MCGrad against state-of-the-art baselines. In Section 5 we present results of MCGrad in production. Here, we focus on five research questions:[2]

- Q1. How does MCGrad compare to *existing baselines* on unspecified groups?
- Q2. Existing methods require manual specification of protected groups. Does MCGrad still protect those specified groups, even though they are not specified in MCGrad?
- Q3. Does MCGrad benefit from running multiple rounds?
- Q4. What is the effect of *rescaling the logits* and the *regularization through min sum Hessian in leaf* on MCGrad?

[2]Code: https://github.com/facebookresearch/mcgrad_multicalibration_at_web_scale.

Q5. How does MCGrad's computational time compare to existing multicalibration methods?

4.1 Experimental Setup

Data. Our experimental analysis uses 11 datasets (see Table 1). Six binarized prediction tasks are derived from the American Community Survey (ACS) [15, 18] using the folktable package, plus a California-specific income task (ACSIncomeCA). The remaining five datasets are: UCI Bank Marketing [44] (term deposit prediction), UCI Default of Credit Card Clients [62] (debt default), Home Mortgage Disclosure Act (HMDA) [10] (mortgage acceptance), and Medical Expenditure Panel Survey (MEPS) [55] (medical visits). For all datasets we use the same protected groups as in [28]. Protected attributes are selected following standard fairness benchmarks.

Baselines. We compare MCGrad against two multicalibration baselines: Discretization-Free MultiCalibration (DFMC) [31] and HKRR Hebert-Johnson et al. [29]. DFMC shares some similarities with MCGrad: It fits a single GBDT with a fixed maximum depth of 2 for each weak learner. Protected groups are explicitly specified by the user, rather than learned like in MCGrad. Protected groups are provided to the algorithm as binary features that encode group membership. HKRR is the boosting procedure proposed by Hebert-Johnson et al. [29], for which we use Hansen et al. [28]'s implementation[3]. We also include Logistic Regression as BasePred, the base predictor f_0 shared by all methods, and Isotonic [63] (Isotonic Regression), a widely adopted calibration algorithm.

Metrics. We evaluate MCGrad relative to baselines on three dimensions: the effect on 1) *predictive performance*, 2) *calibration*, and 3) *multicalibration*.

1) *Predictive performance*. We employ the Area Under Precision-Recall Curve (PRAUC) [12] and the log loss.

2) *Calibration*. Various calibration metrics have been proposed in the literature. The *Expected Calibration Error (ECE)* [45] is the most commonly used calibration measure in machine learning, which bins the model scores and computes the deviation from perfect calibration, i.e., the absolute difference between the model's predicted accuracy and its empirical accuracy [45]. Binning-based calibration metrics, including ECE, vary significantly based on the choice of bins. This is a well-documented shortcoming [1, 51]. Alternative metrics that aim to circumvent the drawbacks of binning include *smooth ECE (smECE)* which relies on kernel smoothing [4], but merely replaces sensitivity to the arbitrary choice of binning with sensitivity to an arbitrary choice of kernel [1]. The *Estimated Cumulative Calibration Error (ECCE)* [1] is a calibration metric that is rooted in cumulative statistics that does not require making an arbitrary choice in bin or kernel, which we covered in more detail in Section 2. *Brier score* is a proper scoring rule that is commonly used to evaluate probabilistic forecasts that is essentially just the mean squared error of predicted probability. Brier score decomposes into calibration and the model's ability to separate the positive and negative class [5]. In this paper, we quantify calibration error using ECCE, with exceptions in Section 5, where for pragmatic reasons we resort to ECE and Brier score whenever ECCE was not measured in the production system.

3) *Multicalibration*. Błasiok et al. [3] proposed the *maximum group-wise smECE* metric, which was used for empirical evaluations in Hansen et al. [28]. Guy et al. [25] proposed an extension of *ECCE* to quantify multicalibration, called *Multicalibration Error (MCE)*, which we defined in Eq. (2). *Maximum group-wise smECE* and *MCE* have in common that they both are defined as a maximum of some quantity over groups, but differ in the calibration quantity calculated in each group, where they respectively use smECE and *ECCE*. In this paper, we use MCE to quantify multicalibration. In the online supplement[4], we additionally report group-wise smECE, thereby providing results that are in-line with [28]. We can measure any multicalibration metric with respect to either of two sets of groups. *Prespecified Groups*: refers to a small (less than 15 across all datasets) set of protected groups that are formed using various features of the datasets. For this, we use the groups specifications that were used in Hansen et al. [28]. *Unspecified Groups*: covers the case where the user does not specify which groups to protect against. In that case, the scores are expected to be multicalibrated with respect to all possible groups, and to calculate this, an extensive set of combinatorially generated groups are used.

Hyperparameters. MCGrad uses LightGBM as GBDT implementation. To ensure seamless adoption by ML engineers, we set default hyperparameters so that MCGrad works out-of-the-box, without requiring application-specific tuning in the majority of use-cases. We determined the default hyperparameters by performing a grid search over 35 company-internal datasets and selecting the configuration that never degraded the base model's log loss or PRAUC, and minimized the average MCE across datasets. The resulting hyperparameters are: learning_rate= 0.02873, max_depth= 5, min_child_samples = 160, n_estimators= 94, num_leaves= 5, lambda_l2= 0.00913, min_gain_to_split= 0.15. All other parameters are set as for LightGBM defaults.

For HKRR we follow Hansen et al. [28] and pick the best hyperparameters from a set of four specifications using a held-out validation set. For DFMC we use the default hyperparameters of LightGBM since the paper does not suggest any specific hyperparameters.

4.2 Experimental Results

Q1. MCGrad vs existing baselines on *unspecified* groups. Figure 1 (left-hand side) shows the Multicalibration Error (MCE) for all compared methods using all available features. Overall, MCGrad outperforms all baselines on 10 out of the 11 datasets by reducing the base predictor's MCE between 4% (Credit) and 93% (ACSEmploy), and obtaining an average reduction of 56.1%. In contrast, DFMC, Isotonic, and HKRR only achieve an average reduction of, respectively, 11.9%, 9.6%, and 1.4%. In addition, MCGrad never harms the base predictor's performance: even on MEPS, where there is low evidence of BasePred's miscalibration (3.9 MCE), it slightly improves the MCE by 2.8%. On the other hand, Isotonic, DFMC, and HKRR increase (worsen) the MCE of BasePred on 3, 3 and 4 datasets respectively.

[3]https://github.com/dutchhansen/empirical-multicalibration

[4]https://github.com/facebookresearch/mcgrad_multicalibration_at_web_scale

Dataset	# Samples n	# Features d	Class Distr.	Protected Attributes
MEPS [55]	11079	139	0.169	Race, Gender, Age, Income, Insurance Status
Credit [62]	30000	118	0.221	Gender, Age, Education, Marital Status
Marketing [44]	45211	41	0.116	Age, Marital Status, Education, Occupation
HMDA [10]	114185	89	0.752	Race, Gender, Ethnicity, Age
ACSIncomeCA [15, 18]	195665	10	0.410	Race, Gender, Age, Education, Income, Employment
ACSMobility [15, 18]	616207	23	0.735	Race, Gender, Age, Education, Income, Employment
ACSPublic [15, 18]	1123374	24	0.293	Race, Gender, Age, Education, Income, Employment
ACSTravel [15, 18]	1458542	19	0.437	Race, Gender, Age, Education, Income, Employment
ACSIncome [15, 18]	1655429	10	0.370	Race, Gender, Age, Education, Income, Employment
ACSEmploy [15, 18]	3207990	20	0.456	Race, Gender, Age, Education, Income, Employment
ACSHealth [15, 18]	3207990	31	0.150	Race, Gender, Age, Education, Income, Employment

Table 1: Number of samples, features, class distribution ($p(Y = 1)$), and protected attributes for each benchmark dataset.

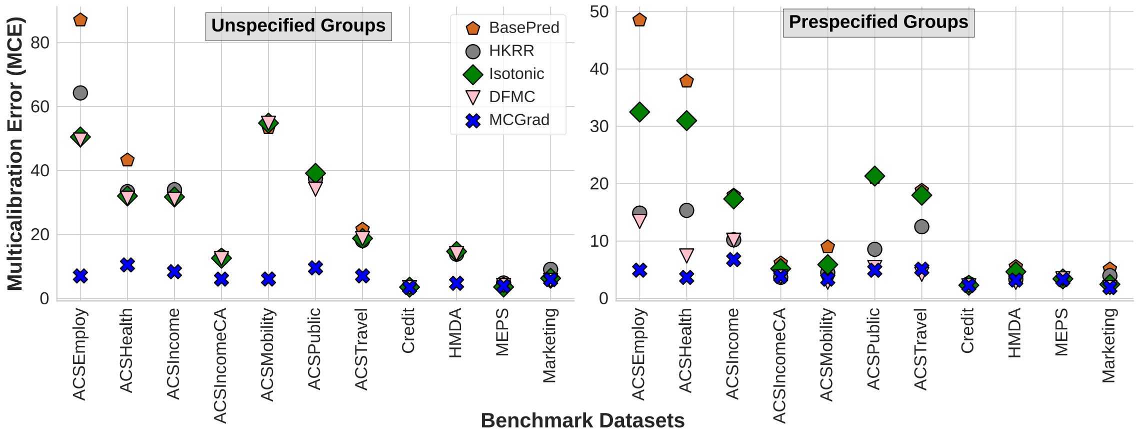

Figure 1: Multicalibration Error computed using *unspecified* groups (left) or the manually *prespecified* groups (right) for all compared methods on each benchmark dataset. Overall, MCGrad achieves a better (lower) error for 10 out of 11 datasets when tested on unspecified groups, and for 5 out of 11 datasets when tested on prespecified groups.

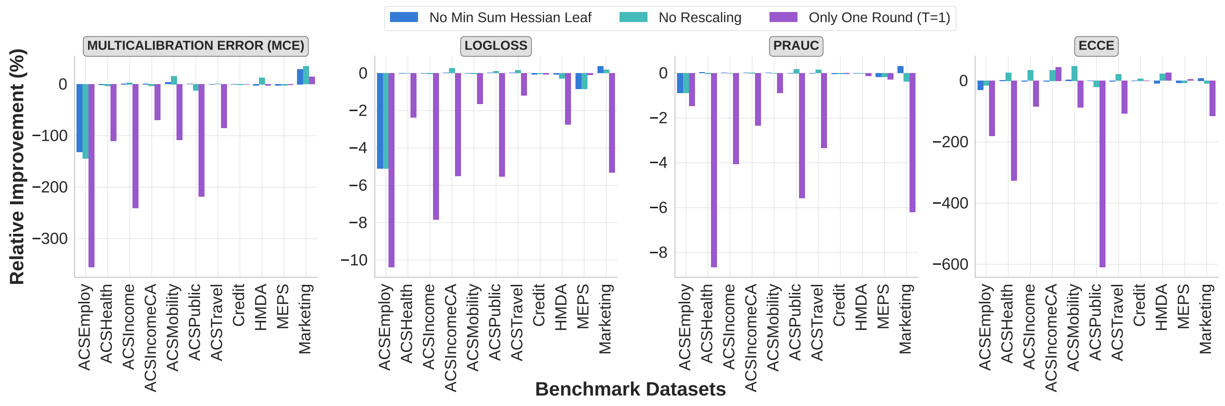

Figure 2: Improvement (in %) of MCGrad's variants relative to the original version. While setting $T = 1$ yields a significant drop in performance, the effect of the rescaling factor and min sum Hessian in leaf are mild due to the datasets limited size.

Table 2 reports the average performance over the datasets. MCGrad achieves by far the best average log loss improvement (10.4%) and PRAUC improvement (8.1%) over the base predictor. Besides these average improvements over datasets, MCGrad never harms performance, which helps build the necessary trust among ML engineers to deploy it in production environments. Other methods don't have this guarantee. For instance, while DFMC improves model performance metrics by a lower 1.8% and 0.76%, it harms PRAUC and log loss on two datasets. Isotonic and HKRR even

Baseline	MCE		log loss		PRAUC		ECCE	
	Avg	Rank	Avg	Rank	Avg	Rank	Avg	Rank
MCGrad	**6.60**	**1.18**	**0.375**	**1.00**	**0.714**	**1.00**	1.97	2.27
Isotonic	24.00	2.91	0.410	3.27	0.662	3.91	**1.53**	**1.54**
DFMC	24.95	3.00	0.408	2.27	0.671	2.18	2.20	2.91
HKRR	25.74	3.73	0.414	4.00	0.637	5.00	3.29	3.54
BasePred	29.07	4.18	0.414	4.45	0.668	2.91	4.66	4.73

Table 2: Average value (Avg) and rank (Rank) over the datasets, for various metrics. MCGrad has the best average log loss and PRAUC, and second-best average ECCE.

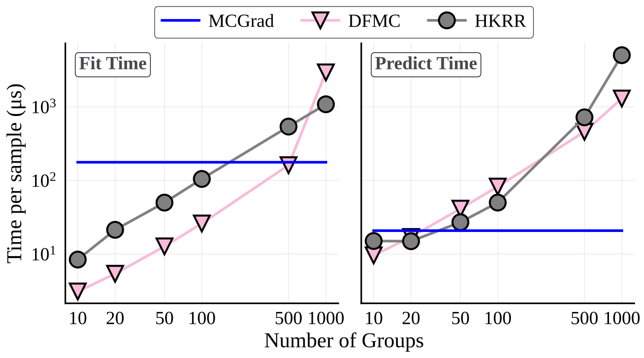

Figure 3: Computational time of multicalibration algorithms with varying numbers of groups. (Left) Fit time and (Right) predict time in seconds as a function of the number of groups on a log-log scale. Runtime for DFMC and HKRR increases with the number of groups, while MCGrad is constant.

decrease the PRAUC on average by, respectively, 1% and 5% with a negligible change in log loss. For each metric, we rank all compared methods from 1 (best) to 5 (worst) and report the average ranking position in Table 2. A lower average rank indicates how often the method is preferred, as opposed to the average performance, which reflects the magnitude. MCGrad consistently ranks as #1 baseline on log loss and PRAUC, while only ranking third for MCE on the MEPS dataset. Finally, MCGrad ranks second best method in terms of global calibration (ECCE), losing against Isotonic Regression, which is expected because it is trained to directly optimize the global calibration of the base predictor.

Q2. MCGrad vs existing baselines on *prespecified* groups. To verify that good performance on the MCE metric for all groups implies also protection for prespecified groups, we report results just for these groups in Figure 1 (right-hand side). Overall, MCGrad outperforms all baselines on 5 out of 11 datasets achieving a 54% average MCE improvement, and is followed by DFMC that is the best baseline on three datasets (ACSMobility, HDMA, ACSTravel) and achieves an average MCE improvement of 51%. HKRR obtains an average improvement in MCE of 36%, while Isotonic only achieves an improvement of 16%. Note that HKRR and DFMC have direct access to these groups, while MCGrad only has access to the features used to define the groups.

Q3. The impact of multiple rounds on MCGrad. Because MCGrad recurrently improves its own predictions by using T rounds, one relevant question is whether $T > 1$ is really necessary in practice. For this goal, we compare MCGrad with its variant Only One Round that forces the number of rounds $T = 1$.[5] Figure 2 (purple line) shows the relative improvement of Only One Round for the four considered metrics and all benchmark datasets. In the majority of cases, it is evident that Only One Round deteriorates MCGrad's performance: on the MCE and ECCE axes, it more than doubles its value (relative improvement < -100) for, respectively, 6 and 5 datasets, while, on the log loss and PRAUC axes, it worsens the performance by approximately 2% or more on, respectively, 8 and 6 out of 11 datasets. Interestingly, the largest deterioration of performance corresponds to MCGrad running for the highest number of rounds: the optimal T for ACSEmploy, ACSIncome, ACSPublic, and ACSMobility is, respectively, 20, 35, 27 and 22.

Q4. Impact of *rescaling the logits* and the *regularization through min sum Hessian in leaf*. *Rescaling the logits* of all predictions by learning a constant speeds up MCGrad training by reducing rounds, especially on very large datasets. As a downside, it might affect MCGrad's performance. Figure 2 (light blue) shows the relative deterioration after removing rescaling. Effects are mild and balanced: removal worsens MCE, log loss, and PRAUC on 6, 6, and 7 datasets respectively, with average drops of 9%, 0.5%, 0.03%.

LightGBM's default *min sum Hessian leaf* (MSHL) is 0.001 with a min leaf sample size of 20. We claim that higher MHL values prevent probabilities from pushing to extremes, reducing overfitting (Section 3.3). We compare MCGrad with MHL= 0.001 versus our default of 20. Figure 2 (dark blue bar) shows mild deterioration using 0.001, likely due to limited dataset sizes. On average, lower MHL slightly harms performance, but this becomes evident on ACSEmploy (the largest dataset), where the MHL variant runs only 2 rounds (vs. 10) because our regularization mechanism stops the training earlier to avoid overfitting.

Q5. Computational time of MCGrad vs multicalibration baselines as the number of groups scales. Scalability can be a relevant bottleneck for multicalibration deployment in production environments. To evaluate this, we measure the training (fit) and inference (predict) times of MCGrad, HKRR, and DFMC as the number of protected groups increases from 10 to 1000, using $\approx 1M$ training and $\approx 300k$ test samples from the ACSEmploy dataset.

Figure 3 shows the methods' computational time per sample (in microseconds) as a function of the number of groups. Notably, the gap between MCGrad and each baseline becomes larger as group count increases: at 1000 groups, MCGrad maintains nearly constant inference time ($\sim 20\mu s$) while HKRR and DFMC require, respectively, 5019 and 1292 μs, meaning MCGrad achieves less than 2% of their latency overhead. This gap becomes even larger in production settings where models with 100s of features can generate tens of thousands of potential protected groups.

More importantly, these seemingly small latency differences have strong business impact. Production models typically serve online inference where even millisecond-scale increases in response

[5]Early stopping can possibly return $T = 0$.

Model	PRAUC (↑)		Brier Score (↓)	
	Platt Scaling	MCGrad	Platt scaling	MCGrad
#1	0.5295	**0.6565** (+23.9%)	Not available	Not available
#2	0.1824	**0.2433** (+33.4%)	0.0117	**0.0116** (-0.45%)
#3	0.2161	**0.2300** (+6.4%)	0.0642	**0.0632** (-1.56%)
#4	0.6268	**0.6354** (+1.4%)	0.0394	**0.0365** (-7.24%)

Table 3: Online results for four models on Looper, where either Platt Scaling or MCGrad is used for post-processing.

	First Time Period				Second Time Period			
Percentile	Log loss	PRAUC	AUROC	ECE	Log loss	PRAUC	AUROC	ECE
10th	-26.50%	+3.58%	+3.15%	-93.88%	-16.34%	+6.31%	+1.97%	-90.84%
25th	-10.35%	+0.93%	+0.52%	-87.45%	-3.43%	+1.92%	+1.10%	-83.07%
50th	-1.39%	+0.32%	+0.17%	-53.83%	-0.86%	+0.06%	+0.16%	-58.19%
75th	-0.51%	+0.00%	+0.03%	-14.54%	-0.11%	-0.95%	0.0%	-21.96%
90th	+0.02%	-2.07%	-0.20%	+7.56%	+1.75%	-4.85%	-0.39%	+26.23%

Table 4: Summary statistics for the two time periods of the impact of MCGrad on 120 (first period, left) and 65 (second period, right) ML models on MLplatform#2.

time negatively affect the system performance and the user experience. Existing research claims that an *increase* of latency directly correlates with *reduced* user engagement and online business metrics [37]. MCGrad's sub-20μs inference overhead allows deployment in production, while baseline methods requiring few milliseconds per prediction can be impractical for production use.

5 Results in Production Deployment

We describe the results obtained by using MCGrad in a real-world production system. MCGrad has been deployed and used in production at Meta. Specifically, we have integrated MCGrad into two ML training platforms at Meta: the publicly released Looper [43], and an internal platform that we refer to as MLplatform#2. The evaluation consists of a total of 147 binary classification models that serve live production traffic: 27 models on Looper and 120 models on the other platform. On both ML training platforms, we run MCGrad with default hyperparameters.

Looper Results. Looper [43] is a system that automates model training, online A/B test comparisons between the new model variant and the existing production model, and launch decisions on whether or not to promote the new model variant. These launch decisions are made based on whether the new model variant outperforms the existing production model on the selected evaluation metric with statistical significance.

For 27 binary classification models that serve active production traffic on Looper, we created a variant of the model that applies MCGrad as calibration post-processing. We started online A/B tests to compare these 27 models against the active production models, all of which are GBDTs. Those production models all had previously already applied Platt scaling [50].

We found that on 24 out of 27 models, the variant with MCGrad statistically significantly outperformed the same model with Platt scaling [50], resulting in the promotion of the MCGrad-calibrated model variant to become the primary production model.

Looper also monitors area under the *precision-recall curve (PRAUC)* on online production data. PRAUC improved on 24 of the 27 models and was neutral on the rest. Due to data retention, we are able to calculate the exact PRAUC for four of the 27 Looper models for which we most recently deployed MCGrad, as well as the Brier score [6]. Table 3 summarizes the results on those models.

MLplatform#2 Results. This ML training platform is widely used at Meta. ML model training scripts on this platform define the training set, a validation set, the test set, the ML model architecture, and the set of model evaluation metrics. The ML platform allows running an *evaluation flow* that trains the model on the training set and evaluates the model on the test set using the specified set of evaluation metrics, or a *publish flow* that prepares an API endpoint that can generate predictions for live production traffic. We integrated MCGrad into the evaluation flow so that we obtain an additional set of evaluation results for MCGrad.

For every ML model evaluation on this ML platform, we collected parallel evaluation results *with* and *without* MCGrad. We conducted data collection in two separate periods. Table 4 summarizes the results. In the first data collection period, we obtained results on 120 binary classification production models. We found that applying MCGrad post-processing improved log loss for 88.7% of the models, compared to the production models without it. PRAUC improved for 76.7% of the models, AUROC for 80.2%, and Expected Calibration Error (ECE) [45] for 86%. Note that on most models where no improvement was found, there was no metric degradation either, due to the early stopping. In Table 4 we can see that the worst 10th-percentile metric effects of MCGrad still result in degradations. We manually investigated a sample of those models, and found that these were mostly explained by errors by ML practitioners in the train and test set specification. E.g., in some cases the train and test set were clearly from a different population, or sample weights were defined in the train set but not in the test set.

In a second data collection period, we obtained results on 65 binary classification models. We found that MCGrad provided improvements over the production models on the log loss metric for 80.3% of the models, on the PRAUC metric for 75.4% of the models, on the AUROC for 79.3%, and ECE for 86%.

6 Learnings from Production Deployment

Despite the recent focus of the academic community on multicalibration, it has seen little to no uptake in industry. Here we summarize various learnings from applying multicalibration in industry through production deployments of MCGrad and of the Multicalibration Error metric (MCE) [25].

Learning 1. Multicalibration has business value.
Our results demonstrate that MCGrad can improve real-world models' multicalibration as well as performance metrics that tend to correlate with business outcomes, such as PRAUC. Furthermore, baselines such as Isotonic Regression do not achieve this. This contradicts observations by Hansen et al. [28] that (1) Isotonic Regression is often competitive compared to algorithms designed specifically for multicalibration, and (2) practitioners may face a trade-off between multicalibration and predictive performance.

Learning 2. Practitioners struggle to define protected groups. Many multicalibration algorithms [29, 31] require manual specification of the protected groups. In our experience deploying multicalibration in production, we found that many ML engineers struggle to manually define relevant protected groups, and lack guidelines and frameworks for defining them. This experience is in line with findings from a series of interviews on AI fairness adoption in industry [30], where it turns out that practitioners often lack knowledge on different types of biases and cannot identify the relevant biases to correct for. Hence, our experience is that in order to succeed in obtaining industry adoption, a method needs to relieve users of the burden of manually defining protected groups, like MCGRAD does.

Learning 3. Practitioners consider adopting multicalibration methods only when they work *out-of-the-box*, reliably, and with no risk of harming model performance.
Many multicalibration algorithms have hyperparameters. We found industry practitioners to be open to applying multicalibration to production ML models, but only if it takes limited effort. Hyperparameter tuning is often seen as complex or time-consuming. They also perceive it as a reliability risk: if a method requires hyperparameter tuning to work well, then they wonder if the hyperparameter configuration will still work well in subsequent training runs.

Therefore, for a multicalibration algorithm to find successful adoption in industry, it is important that it works *out-of-the-box*, without the need for hyperparameter tuning. Moreover, while practitioners care about multicalibration, we found that they are not willing to accept a degradation in metrics like the log loss and the PRAUC, which are often believed to relate to topline metrics.

We defined a set of hyperparameter default values for MCGRAD that achieves (1) no degradation in PRAUC and log loss, and (2) substantial reduction in MCE. By pre-computing defaults through meta-analysis across 35 datasets, we enable MCGRAD to deliver consistent improvements without requiring domain expertise or extensive experimentation. Results in Sections 4 and 5 confirm that those default values consistently produce log loss and PRAUC improvements over models calibrated with Platt scaling (on Looper) or uncalibrated models (on MLPLATFORM#2), and consistently yield log loss, PRAUC, and MCE improvements on the public datasets.

7 Related Work

Multicalibration was introduced by Hebert-Johnson et al. [29] as a learning objective requiring predictors to be calibrated across a large collection of subgroups. Subsequent work has deepened its theoretical foundations, exploring computational complexity and connections to other learning paradigms [3, 13, 14, 21].

Extensions such as *low-degree multicalibration* [22] generalize subgroup definitions using weight functions, showing that restricting to low-degree polynomials can reduce computational complexity. *Swap multicalibration* [21] strengthens the original concept and is satisfied by some existing algorithms; it is also equivalent to *swap omniprediction* for certain loss function classes. The notion of *omnipredictors* [20] further generalizes agnostic learning by requiring minimization guarantees for a class of loss functions [47]. Connections to *multi-objective learning* have also been explored, framing multicalibration as a game dynamics problem [27].

Algorithms for multicalibration. The early proposals of multicalibration algorithms, such as HKRR [29] (e.g., implemented in MCBoost [49] or by Hansen et al. [28]) and LSBoost [19], rely on discretizing the output space and calibrating the prediction for each discretization level until convergence. A discretization-free multicalibration algorithm was proposed in [31], requiring group functions as input and using a gradient boosting machine to fit an ensemble of depth-two decision trees to the residuals of an uncalibrated predictor using both the original features and the predictions of the uncalibrated predictor. For guaranteeing multicalibration within an additive error, this algorithm requires a loss saturation condition to hold, i.e., that the squared-error loss can be further reduced only by a small amount. Our algorithm uses gradient boosting similarly but does not require pre-specified groups. Instead, it leverages multi-round recursive gradient boosting and supports arbitrary user-defined decision trees in each round.

8 Conclusion & Limitations

We tackle the problem of designing a multicalibration algorithm without having access to pre-specified protected sets, that is fast and safe to deploy in production. These three aspects are relevant for practitioners as they are often (1) unable to define groups for their task, (2) forced to limit the overhead in terms of memory and speed, and (3) required to guarantee that the algorithm does not harm the base predictor. We proposed MCGRAD, a novel algorithm that achieves multicalibration by finding and fixing regions of the feature space where the miscalibration is large. Because this might create new regions with evidence of miscalibration, MCGRAD employs multiple rounds, where in each round it corrects its own output. In addition, MCGRAD's implementation is fast, lightweight, and safe to deploy. We showed that MCGRAD outperforms existing baselines on benchmark datasets and provided results obtained in a large-scale industry production deployment at Meta consisting of hundreds of production models. Finally, we included practical takeaways from our experience applying multicalibration in industry.

Limitations. Although MCGRAD shows improvements, several limitations remain. First, guardrails against harming the base predictor may not be sufficient under concept drift. Specifically, MCGRAD can potentially worsen performance on post-drift data distributions, as multicalibration is only robust to certain types of distribution shift [35, 52]. Second, MCGRAD is currently limited to binary classification settings. Although multiclass extensions exist, they often suffer from reduced sample efficiency and increased complexity, creating challenges for real-world deployment [64]. Third, while MCGRAD can be applied to other data modalities such as text and images (using embeddings), these applications have not been explored in this paper. Future research may study MCGRAD's (1) robustness to broader forms of drift, (2) extension to multiclass/regression problems, and (3) effectiveness on other data modalities.

Acknowledgments

We gratefully acknowledge Nicolas Stier, Udi Weinsberg, Ido Guy, Thomas Leeper, and Mark Tygert for their valuable discussions and support, which contributed to the success of this work.

References

[1] Imanol Arrieta-Ibarra, Paman Gujral, Jonathan Tannen, Mark Tygert, and Cherie Xu. 2022. Metrics of calibration for probabilistic predictions. *Journal of Machine Learning Research* 23, 351 (2022), 1–54.

[2] Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. 2010. Calibration of machine learning models. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global Scientific Publishing, 128–146.

[3] Jarosław Błasiok, Parikshit Gopalan, Lunjia Hu, Adam Tauman Kalai, and Preetum Nakkiran. 2024. Loss Minimization Yields Multicalibration for Large Neural Networks. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 287)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 17:1–17:21.

[4] Jarosław Błasiok and Preetum Nakkiran. 2023. Smooth ECE: Principled reliability diagrams via kernel smoothing. *arXiv preprint arXiv:2309.12236* (2023).

[5] Gail Blattenberger and Frank Lad. 1985. Separating the Brier score into calibration and refinement components: A graphical exposition. *The American Statistician* 39, 1 (1985), 26–32.

[6] Glen W Brier. 1950. Verification of forecasts expressed in terms of probability. *Monthly Weather Review* 78, 1 (1950), 1–3.

[7] Sougata Chaudhuri, Abraham Bagherjeiran, and James Liu. 2017. Ranking and calibrating click-attributed purchases in performance display advertising. In *Proceedings of the ADKDD'17*. 1–6.

[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794.

[9] Zhenbang Chen and Peng Qi. 2022. *Calibrating CTR Prediction with Transfer Learning in Instacart Ads*. Instacart Tech Blog. https://tech.instacart.com/calibrating-ctr-prediction-with-transfer-learning-in-instacart-ads-3ec88fa97525

[10] A Feder Cooper, Solon Barocas, Christopher De Sa, and Siddhartha Sen. 2023. Variance, self-consistency, and arbitrariness in fair classification. *arXiv preprint arXiv:2301.11562* (2023), 1–84.

[11] Diego Corrêa da Silva and Dietmar Jannach. 2025. Calibrated Recommendations: Survey and Future Directions. *arXiv preprint arXiv:2507.02643* (2025).

[12] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international Conference on Machine Learning*.

[13] Zhun Deng, Cynthia Dwork, and Linjun Zhang. 2023. HappyMap : A Generalized Multicalibration Method. In *14th Innovations in Theoretical Computer Science Conference (ITCS 2023) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 251)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 41:1–41:23.

[14] Gianluca Detommaso, Martin Bertran Lopez, Riccardo Fogliato, and Aaron Roth. 2024. Multicalibration for Confidence Scoring in LLMs. In *Forty-first International Conference on Machine Learning*.

[15] 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), 6478–6490.

[16] Cynthia Dwork, Michael P Kim, Omer Reingold, Guy N Rothblum, and Gal Yona. 2022. Beyond bernoulli: Generating random outcomes that cannot be distinguished from nature. In *International Conference on Algorithmic Learning Theory*. PMLR, 342–380.

[17] Yewen Fan, Nian Si, and Kun Zhang. 2023. Calibration Matters: Tackling Maximization Bias in Large-scale Advertising Recommendation Systems. In *The Eleventh International Conference on Learning Representations*.

[18] Sarah Flood, Miriam King, Renae Rodgers, Steven Ruggles, J. Robert Warren, and Michael Westberry. 2021. Integrated Public Use Microdata Series, Current Population Survey: Version 9.0. IPUMS. https://cps.ipums.org/cps/

[19] Ira Globus-Harris, Declan Harrison, Michael Kearns, Aaron Roth, and Jessica Sorrell. 2023. Multicalibration as Boosting for Regression. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, 11459–11492.

[20] Parikshit Gopalan, Adam Tauman Kalai, Omer Reingold, Vatsal Sharan, and Udi Wieder. 2022. Omnipredictors. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 215)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 79:1–79:21.

[21] Parikshit Gopalan, Michael Kim, and Omer Reingold. 2023. Swap Agnostic Learning, or Characterizing Omniprediction via Multicalibration. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 39936–39956.

[22] Parikshit Gopalan, Michael P. Kim, M. Singhal, and Shengjia Zhao. 2022. Low-Degree Multicalibration. In *Annual Conference Computational Learning Theory*.

[23] Mert Gulsoy, Emre Yalcin, Yucel Tacli, and Alper Bilge. 2025. DUoR: Dynamic User-oriented re-Ranking calibration strategy for popularity bias treatment of recommendation algorithms. *International Journal of Human-Computer Studies* (2025), 103578.

[24] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International conference on machine learning*. PMLR, 1321–1330.

[25] Ido Guy, Daniel Haimovich, Fridolin Linder, Nastaran Okati, Lorenzo Perini, Niek Tax, and Mark Tygert. 2025. Measuring multi-calibration. arXiv:2506.11251 [stat.ME] https://arxiv.org/abs/2506.11251

[26] Nika Haghtalab, Michael Jordan, and Eric Zhao. 2023. A Unifying Perspective on Multi-Calibration: Game Dynamics for Multi-Objective Learning. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 72464–72506. https://proceedings.neurips.cc/paper_files/paper/2023/file/e55edcdb01ac45c839a602f96e09fbcb-Paper-Conference.pdf

[27] Nika Haghtalab, Michael Jordan, and Eric Zhao. 2023. A Unifying Perspective on Multi-Calibration: Game Dynamics for Multi-Objective Learning. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 72464–72506.

[28] Dutch Hansen, Siddartha Devic, Preetum Nakkiran, and Vatsal Sharan. 2024. When is multicalibration post-processing necessary? *Advances in Neural Information Processing Systems* 37 (2024), 38383–38455.

[29] Ursula Hebert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. 2018. Multicalibration: Calibration for the (Computationally-Identifiable) Masses. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 1939–1948.

[30] Kenneth Holstein, Jennifer Wortman Vaughan, Hal Daumé III, Miro Dudik, and Hanna Wallach. 2019. Improving fairness in machine learning systems: What do industry practitioners need?. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–16.

[31] Hongyi Henry Jin, Zijun Ding, Dung Daniel Ngo, and Zhiwei Steven Wu. 2025. Discretization-free Multicalibration through Loss Minimization over Tree Ensembles. arXiv:2505.17435 [cs.LG] https://arxiv.org/abs/2505.17435

[32] Christopher Jung, Changhwa Lee, Mallesh Pai, Aaron Roth, and Rakesh Vohra. 2021. Moment multicalibration for uncertainty estimation. In *Conference on Learning Theory*. PMLR, 2634–2678.

[33] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 3149–3157.

[34] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems* 30 (2017).

[35] Michael P Kim, Christoph Kern, Shafi Goldwasser, Frauke Kreuter, and Omer Reingold. 2022. Universal adaptability: Target-independent inference that competes with propensity scoring. *Proceedings of the National Academy of Sciences* 119, 4 (2022), e2108097119.

[36] Ian D. Kivlichan, Zi Lin, Jeremiah Liu, and Lucy Vasserman. 2021. Measuring and Improving Model-Moderator Collaboration using Uncertainty Estimation. In *WOAH 2021 - 5th Workshop on Online Abuse and Harms, Proceedings of the Workshop*.

[37] Ron Kohavi, Alex Deng, Roger Longbotham, and Ya Xu. 2014. Seven rules of thumb for web site experimenters. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1857–1866.

[38] Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. 2019. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in Neural Information Processing Systems* 32 (2019).

[39] Meelis Kull, Telmo Silva Filho, and Peter Flach. 2017. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Artificial intelligence and statistics*. PMLR, 623–631.

[40] Wonbin Kweon, SeongKu Kang, and Hwanjo Yu. 2022. Obtaining calibrated probabilities with personalized ranking models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 4083–4091.

[41] Hongfu Liu, Hengguan Huang, Xiangming Gu, Hao Wang, and Ye Wang. 2024. On calibration of LLM-based guard models for reliable content moderation. *arXiv preprint arXiv:2410.10414* (2024).

[42] Haihao Lu and Rahul Mazumder. 2020. Randomized Gradient Boosting Machine. *SIAM Journal on Optimization* 30, 4 (2020), 2780–2808.

[43] Igor L Markov, Hanson Wang, Nitya S Kasturi, Shaun Singh, Mia R Garrard, Yin Huang, Sze Wai Celeste Yuen, Sarah Tran, Zehui Wang, Igor Glotov, et al. 2022. Looper: An end-to-end ml platform for product decisions. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3513–3523.

[44] S. Moro, P. Rita, and P. Cortez. 2014. Bank Marketing. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5K306.

[45] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 29.

[46] Georgy Noarov and Aaron Roth. 2023. The statistical scope of multicalibration. In *International Conference on Machine Learning*. PMLR, 26283–26310.
[47] Princewill Okoroafor, Robert Kleinberg, and Michael P. Kim. 2025. Near-Optimal Algorithms for Omniprediction. arXiv:2501.17205 [stat.ML] https://arxiv.org/abs/2501.17205
[48] Gustavo Penha and Claudia Hauff. 2021. On the calibration and uncertainty of neural learning to rank models. *arXiv preprint arXiv:2101.04356* (2021).
[49] Florian Pfisterer, Christoph Kern, Susanne Dandl, Matthew Sun, Michael P. Kim, and Bernd Bischl. 2021. mcboost: Multi-Calibration Boosting for R. *Journal of Open Source Software* 6, 64 (aug 2021), 3453.
[50] John Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers* (1999).
[51] Rebecca Roelofs, Nicholas Cain, Jonathon Shlens, and Michael C Mozer. 2022. Mitigating bias in calibration error estimation. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 4036–4054.
[52] Aaron Roth. 2022. Uncertain: Modern topics in uncertainty quantification. https://www.cis.upenn.edu/~aaroth/uncertainty-notes.pdf
[53] Eliran Shabat, Lee Cohen, and Yishay Mansour. 2020. Sample complexity of uniform convergence for multicalibration. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1118, 10 pages.
[54] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.
[55] Shubham Sharma, Alan H Gee, David Paydarfar, and Joydeep Ghosh. 2021. Fair-n: Fair and robust neural networks for structured data. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 946–955.
[56] Telmo Silva Filho, Hao Song, Miquel Perello-Nieto, Raul Santos-Rodriguez, Meelis Kull, and Peter Flach. 2023. Classifier calibration: a survey on how to assess and improve predicted class probabilities. *Machine Learning* 112, 9 (2023), 3211–3260.
[57] Harald Steck. 2018. Calibrated recommendations. In *Proceedings of the 12th ACM conference on recommender systems*. 154–162.
[58] Mark Tygert. 2023. Calibration of P-values for calibration and for deviation of a subpopulation from the full population. *Advances in Computational Mathematics* 49, 5 (2023), 70.
[59] Bertie Vidgen, Scott A Hale, Sam Staton, Tom Melham, Helen Margetts, Ohad Kammar, and Marcin Szymczak. 2020. Recalibrating classifiers for interpretable abusive content detection. In *Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science*. 132–138.
[60] Jiayun Wu, Jiashuo Liu, Peng Cui, and Steven Z Wu. 2024. Bridging multicalibration and out-of-distribution generalization beyond covariate shift. *Advances in Neural Information Processing Systems* 37 (2024), 73036–73078.
[61] Shuai Yang, Hao Yang, Zhuang Zou, Linhe Xu, Shuo Yuan, and Yifan Zeng. 2024. Deep Ensemble Shape Calibration: Multi-Field Post-hoc Calibration in Online Advertising. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6117–6126.
[62] I-Cheng Yeh. 2009. Default of Credit Card Clients. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C55S3H.
[63] Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 694–699.
[64] Shengjia Zhao, Michael Kim, Roshni Sahoo, Tengyu Ma, and Stefano Ermon. 2021. Calibrating predictions to decisions: A novel approach to multi-class calibration. *Advances in Neural Information Processing Systems* 34 (2021), 22313–22324.

A Bridging Multicalibration Error and α−MC

Several definitions of approximate multicalibration, known as α-multicalibration, exist in the literature. A common approach uses a uniform bound on absolute deviation $\Delta_{h,g}(f) \le \alpha$ where the precise choice of h and g varies [3, 13, 22, 26]. Another common choice is a uniform bound on the conditional deviation [29, 53], i.e. $\delta_{h,g}(f) := |\mathbb{E}[h(X)g(f(X))(Y - f(X)) \mid h(X)g(f(X)) = 1]| \le \alpha$. Recent work [26] improved these by showing a tighter bound $\Delta_{h,g}(f) \le \alpha\sqrt{\mathbb{E}[h(X)]}$ can be achieved with no extra requirements.

We next show that Definition 2.1, for the expected deviation corresponding to the sample mean deviation over dataset D, i.e.,

$$\mathbb{E}_D[h(X)g(f(X))(Y - f(X))] = \frac{1}{n}\sum_{k=1}^{n} h(x_k)g(f(x_k))(y_k - f(x_k)),$$

with scale parameter chosen $\tau_h(f) = \sqrt{\mathbb{E}_D[h(X)f(X)(1 - f(X))]}$, matches with the MCE metric defined in Eq. (2).

Proposition A.1. *Given a set $\mathcal{H}$ of group membership functions, a probabilistic predictor f is α-multicalibrated with respect to $\mathcal{H}$, with the scale parameter $\tau_h(f)$, if and only if $MCE(f) \le \alpha\sqrt{n}$.*

Proof. Note that $\Delta_{h,g}(f) = \frac{1}{n}\left|\sum_{k=1}^{n} h(x_k)g(f(x_k))(y_k - f(x_k))\right|$. Combining with the definition of $\mathrm{ECCE}_h(f)$, we get that

$$\max_{g\in\mathcal{G}} \Delta_{h,g}(f) = \frac{n_h}{n}\mathrm{ECCE}_h(f).$$

Plugging in the definition of $\mathrm{MCE}(f)$, as in Eq. (2), we have

$$\frac{1}{\sqrt{n}}\mathrm{MCE}(f) = \frac{1}{\sqrt{n}}\max_{h\in\mathcal{H}}\frac{\mathrm{ECCE}_h(f)}{\sigma_h(f)} = \max_{h\in\mathcal{H},g\in\mathcal{G}}\frac{\sqrt{n}\Delta_{h,g}(f)}{n_h\sigma_h(f)} = \max_{h\in\mathcal{H},g\in\mathcal{G}}\frac{\Delta_{h,g}(f)}{\tau_h(f)}.$$

Now, from the last expression, α-MC with respect to $\mathcal{H}$ holds if and only if $\mathrm{MCE}(f) \le \alpha\sqrt{n}$. □

B Convergence Analysis

We analyze the convergence of Algorithm 1 under specific conditions for the GBM used in each round. We show that (1) the loss does not increase with more rounds and decreases unless the predictor has converged, and (2) the α-multicalibration error after T rounds is bounded by the difference between predictors F_{T+1} and F_T and that 0-multicalibration is achieved at convergence. Unlike Section 3.1, which assumes adding an optimal linear combination of weak learners each round, here the GBM incrementally adds regression trees to F_t to form F_{t+1}.

We consider a GBM that, in round t, iteratively combines M_t regression trees as $F_t^{m+1}(x) = F_t^m(x) + \rho_m h_{j_m}(x, f_t(x))$ where j_m is the index of the regression tree added in the m-th iteration, ρ_m is the step size, and $m = 0, 1, \ldots, M_t - 1$. Let

$$\mathcal{L}_t(w) = \mathbb{E}_D\left[\mathcal{L}\left(F_t(X) + \sum_{k=1}^{K} w_k h_k(X, f_t(X)), Y\right)\right].$$

The gradient of the loss function with respect to w_k is given by

$$\nabla_k \mathcal{L}_t(w) = \mathbb{E}_D[h_k(X, f_t(X))(Y - f_{t+1}(X))]$$

where $f_{t+1}(x) = \mu(F_t(x) + \sum_{k=1}^{K} w_k h_k(x, f_t(x)))$.

A GBM can be shown to correspond to a coordinate gradient descent in the coefficient space [42], with updates of the form $w^{m+1} = w^m - \rho\nabla_{j_m}\mathcal{L}_t(w^m)e_{j_m}$, where e_j is the j-th standard basis vector in $\mathbb{R}^K$ and ρ is the step size.

Decreasing Loss Property.

Proposition B.1. *Assume that the loss function $\mathcal{L}$ is a convex and L-smooth (for the log loss, $L = 1/4$) and that the step size ρ is set as $\rho = 1/L$. Then, for every $t \ge 0$, the following inequality holds*

$$\mathbb{E}_D[\mathcal{L}(F_{t+1}(X), Y)] \le \mathbb{E}_D[\mathcal{L}(F_t(X), Y)]$$

and is strict if $\nabla_{j_m}\mathcal{L}_t(w^m) \ne 0$ for some $m \in \{0, \ldots, M_t - 1\}$.

Proof. Note the following relations, for any ρ_m:

$$\mathcal{L}_t(w^{m+1}) \le \mathcal{L}_t(w^m + \rho_m e_{j_m}) \le \mathcal{L}_t(w^m) + \rho_m \nabla\mathcal{L}_t(w^m)^\top e_{j_m} + \frac{L}{2}\rho_m^2.$$

By taking $\rho_m = -(1/L)\nabla_{j_m}\mathcal{L}_t(w^m)$, we have

$$\mathcal{L}_t(w^{m+1}) \le \mathcal{L}_t(w^m) - \frac{1}{2L}(\nabla_{j_m}\mathcal{L}_t(w^m))^2.$$

It follows that $\mathcal{L}_t(w^{M_t}) - \mathcal{L}_t(w^0) \le -\frac{1}{2L}\sum_{m=0}^{M_t-1}(\nabla_{j_m}\mathcal{L}_t(w^m))^2$, which allows us to conclude that $\mathcal{L}_t(w^{M_t}) - \mathcal{L}_t(w^0) \le 0$ where the inequality is strict if $\nabla_{j_m}\mathcal{L}_t(w^m) \ne 0$ for some $0 \le m < M_t$. □

For a GBM that, in each iteration, greedily selects the weak learner with the largest absolute value of the loss gradient, i.e., $j_m \in \arg\max_j |\nabla_j \mathcal{L}_t(w^m)|$, we can establish a convergence rate to the loss of the optimum linear combination of weak learners using Theorem 4.2 [42]. Let $w^*(t) = \arg\min_w \mathcal{L}_t(w)$. Then,

$$\mathbb{E}_D\left[\mathcal{L}(F_{t+1}(X), Y)\right] - \mathcal{L}_t(w^*(t)) \le 0.5 \cdot \mathrm{Dist}_t^2 \frac{1}{M_t} \tag{4}$$

where Dist_t denotes the distance between the level set $\{w : \mathcal{L}_t(w) \le \mathcal{L}_t(w^0)\}$ and $w^*(t)$, as defined in [42].

This result justifies modeling each round as updating the predictor with an optimal ensemble of weak learners that minimizes loss. Proposition B.1 shows the training loss never increases with more rounds and converges to a limit as rounds go to infinity.

A Bound on the Multicalibration Error. The bound in Proposition B.2 provides a theoretical upper-bound guarantee, showing that the multicalibration error converges to zero when C_T is bounded by a constant, the gap between prediction values F_{T+1} and F_T vanishes as the number of rounds T increases, and the GBM achieves optimal loss in round T. Before proving this, we need a preliminary result:

Lemma B.1. *If f_T is such that $|\mathbb{E}[h(X)g(f_T(X))(y - f_T(X))]| \ge \alpha\sqrt{\mathbb{E}[h(X)f_T(X)(1-f_T(X))]}$, for some $h \in \mathcal{H}$ and $g \in \mathcal{G}$, then there exists $\tilde{w} \in \mathbb{R}^K$ such that for $\tilde{h}(x) = \sum_{k=1}^K \tilde{w}_k h_k(x, f_T(x))$:*

$$\alpha \le \frac{2}{\sqrt{3}} C_T \sqrt{\mathbb{E}\left[\mathcal{L}(F_T(X), Y)\right] - \mathbb{E}\left[\mathcal{L}(F_T(X) + \tilde{h}(X), Y)\right]}$$

$$\text{where } C_T^2 := \max_{h\in\mathcal{H}, g\in\mathcal{G}} \frac{\mathbb{E}\left[g(f_T(X)) \mid h(X) = 1\right]}{\mathbb{E}\left[f_T(X)(1-f_T(X)) \mid h(X) = 1\right]}.$$

Proof. For any F, h and any convex, L-smooth loss function $\mathcal{L}$,

$$\begin{aligned}\mathcal{L}(F(x), y) - \mathcal{L}(F(x) + h(x), y) &\ge \frac{d}{dF}\mathcal{L}(F(x) + h(x), y)(-h(x)) \\ &\ge \frac{d}{dF}\mathcal{L}(F(x), y)(-h(x)) - Lh(x)^2\end{aligned}$$

where the first inequality holds as $\mathcal{L}$ is convex and the second inequality holds because $\mathcal{L}$ is L-smooth.

For the log loss $(d/dF)\mathcal{L}(F, y) = -(y - \mu(F))$, hence, we have

$$\mathbb{E}\left[\mathcal{L}(F(X), Y) - \mathcal{L}(F(X) + h(X), Y)\right] \ge \mathbb{E}\left[(Y - f(X))h(X)\right] - L\mathbb{E}\left[h(X)^2\right]$$

Let

$$(h^*, g^*) \in \arg\max_{h\in\mathcal{G}, g\in\mathcal{G}} \frac{|\mathbb{E}[h(X)g(f_T(X))(Y - f_T(X))]|}{\sqrt{\mathbb{E}[h(X)f_T(X)(1-f_T(X))]}}$$

and, let $\tilde{w}$ be defined as:

$$\tilde{w}_{h,g} = \frac{\mathbf{E}[h(X)g(f_T(X))(Y - f_T(X))]}{\mathbb{E}\left[(h(X)g(f_T(X)))^2\right]}$$

if $(h, g) = (h^*, g^*)$, otherwise $\tilde{w}_{h,g} = 0$. For $w = \tilde{w}$, we have both

$$\mathbb{E}\left[(Y - f_T(X))\sum_{k=1}^K w_k h_k(x, f_T(X))\right] = \frac{\mathbb{E}\left[h^*(X)g^*(f_T(X))(Y - f_T(X))\right]^2}{\mathbb{E}\left[(h^*(X)g^*(f_T(X)))^2\right]}$$

$$\mathbb{E}\left[\left(\sum_{k=1}^K w_k h_k(X, f_T(X))\right)^2\right] = \frac{\mathbb{E}\left[h^*(X)g^*(f_T(X))(Y - f_T(X))\right]^2}{\mathbb{E}\left[(h^*(X)g^*(f_T(X)))^2\right]}.$$

Hence, we have

$$\begin{aligned}&\mathbb{E}\left[\mathcal{L}(F_T(X), Y) - \mathcal{L}\left(F_T(X) + \sum_{k=1}^K \tilde{w}_k h_k(X, f_T(X)), Y\right)\right] \\ \ge\ & (1 - L)\frac{\mathbb{E}\left[h^*(X)g^*(f_T(X))(Y - f_T(X))\right]^2}{\mathbb{E}\left[(h^*(X)g^*(f_T(X)))^2\right]} \\ \ge\ & (1 - L)\frac{\mathbb{E}\left[f_T(X)(1 - f_T(X)) \mid h^*(X) = 1\right]}{\mathbb{E}\left[g^*(f_T(X)) \mid h^*(X) = 1\right]}\alpha^2.\end{aligned}$$

Because $L = 1/4$ for the log loss, we conclude the proof

$$\alpha^2 \le \frac{4}{3}C_T^2\left(\mathbb{E}\left[\mathcal{L}(F_T(X), Y)\right] - \mathbb{E}\left[\mathcal{L}\left(F_T(X) + \tilde{h}(X), Y\right)\right]\right)$$

□

Now, we can derive the upper bound of the MC error:

Proposition B.2. *f_T is α-MC with $\alpha \le \frac{2}{\sqrt{3}}C_T\sqrt{\phi_T + \epsilon_T}$ where*

$$\begin{aligned}\phi_T &:= 2\,\mathbb{E}\left[|F_{T+1}(X) - F_T(X)|\right] + \frac{1}{8}\mathbb{E}\left[(F_{T+1}(X) - F_T(X))^2\right] \\ \epsilon_T &:= \mathbb{E}\left[\mathcal{L}(F_{T+1}(X), Y)\right] - \mathcal{L}_T(w^*(T)),\end{aligned}$$

and C_T is defined as in Lemma B.1.

Proof. For any $w \in \mathbb{R}^K$,

$$\begin{aligned}&\mathbb{E}\left[\mathcal{L}(F_t(X), Y)\right] - \mathbb{E}\left[\mathcal{L}\left(F_t(X) + \sum_{k=1}^K w_k h_k(X, f_t(X)), Y\right)\right] \\ \le\ & \underbrace{\mathbb{E}\left[\mathcal{L}(F_t(X), Y) - \mathcal{L}(F_{t+1}(X), Y)\right]}_{(*)} + \underbrace{\mathbb{E}\left[\mathcal{L}(F_{t+1}(X), Y)\right] - \mathcal{L}_t(w^*(t))}_{(**)}\end{aligned}$$

The first term $(*)$ captures the gap of the loss function values at two successive rounds of the algorithm and can be bounded by $2\mathbb{E}\left[|F_{t+1}(X) - F_t(X|\right] + \frac{L}{2}\mathbb{E}\left[(F_{t+1}(X) - F_t(X))^2\right]$. The second term $(**)$ measures the gap between the loss at the end of round t and the loss from an optimal linear combination of weak classifiers. This gap can be reduced by increasing the number of GBM iterations and is bounded as in Eq (4). The lemma follows from Lemma B.1. □

The bound on the multicalibration error depends on three terms: (a) C_T, which depends inversely on the average prediction variance within subgroups, (b) ϕ_T, which depends on the gap between the prediction values F_{T+1} and F_T, and (c) ϵ_T, which measures the gap between the loss achieved by the predictor F_{T+1} and the optimum loss in round T. If C_T is bounded by a constant, the MC error can be made arbitrarily small by reducing ϕ_T and ϵ_T. As T increases, ϕ_T approaches zero, and ϵ_T can be minimized by increasing the number of GBM iterations M_T, as shown in Eq. (4). Also, note that

$$C_T^2 \le \frac{1}{\min_{h\in\mathcal{H}}\mathbb{E}\left[f_T(X)(1 - f_T(X)) \mid h(X) = 1\right]}.$$

This bound is finite if and only if $\mathbb{E}\left[f_T(X)(1 - f_T(X)) \mid h(X) = 1\right] > 0$ for every group $h \in \mathcal{H}$. Equivalently, for every group h, there exists at least one member x such that $0 < f_T(x) < 1$. This condition clearly holds if $0 < f_T(x) < 1$ for all $x \in \mathcal{X}$ which is satisfied whenever $F_T(x) \in [-R, R]$ for every $x \in \mathcal{X}$, for some constant $R > 0$. Moreover, if the predictor f_T takes values in $[\epsilon, 1 - \epsilon]$, for some $0 < \epsilon \le 1/2$, then $C_T^2 \le \frac{2}{\epsilon}$.