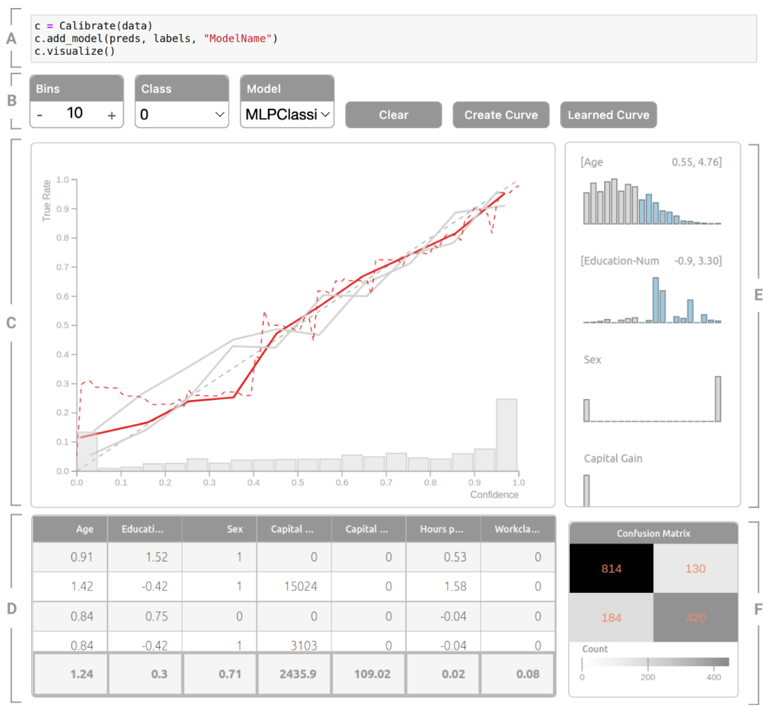
Calibrate: Interactive Analysis of Probabilistic Model Output

Peter Xenopoulos, João Rulff, Luis Gustavo Nonato, Brian Barr, Claudio Silva



```
c = Calibrate(data)
c.add_model(preds, labels, "ModelName")
c.visualize()
```

Bins: 10 · Class: 0 · Model: MLPClassi

Age	Educati...	Sex	Capital ...	Capital ...	Hours p...	Workcla...
0.91	1.52	1	0	0	0.53	0
1.42	-0.42	1	15024	0	1.58	0
0.84	0.75	0	0	0	-0.04	0
0.84	-0.42	1	3103	0	-0.04	0
1.24	**0.3**	**0.71**	**2435.9**	**109.02**	**0.02**	**0.08**

814	130
184	420

Fig. 1. Calibrate visualizing model calibration across different subgroups for income prediction on the Census Income prediction task. A) Calibrate is implemented for easy, one-line use in Jupyter Notebooks. B & C) Calibration View implements *Learned Reliability Diagrams* (red dotted line), an approach to address shortcomings in traditional reliability diagrams (blue line), as well as a histogram to show the density of predictions. D) Instance View shows instances contained by brushing prediction regions in the Calibration View. E) Feature View allows for subgroup analysis by interactive subgroup creation through brushing feature ranges. F) Performance View shows a confusion matrix for the brushed prediction region.

Abstract— Analyzing classification model performance is a crucial task for machine learning practitioners. While practitioners often use count-based metrics derived from confusion matrices, like accuracy, many applications, such as weather prediction, sports betting, or patient risk prediction, rely on a classifier's predicted probabilities rather than predicted labels. In these instances, practitioners are concerned with producing a *calibrated* model, that is, one which outputs probabilities that reflect those of the true distribution. Model calibration is often analyzed visually, through static reliability diagrams, however, the traditional calibration visualization may suffer from a variety of drawbacks due to the strong aggregations it necessitates. Furthermore, count-based approaches are unable to sufficiently analyze model calibration. We present Calibrate, an interactive reliability diagram that addresses the aforementioned issues. Calibrate constructs a reliability diagram that is resistant to drawbacks in traditional approaches, and allows for interactive subgroup analysis and instance-level inspection. We demonstrate the utility of Calibrate through use cases on both real-world and synthetic data. We further validate Calibrate by presenting the results of a think-aloud experiment with data scientists who routinely analyze model calibration.

Index Terms—calibration, performance analysis, model understanding, reliability diagram

1 Introduction

Analyzing classification model performance is an important task for machine learning stakeholders. Furthermore, understanding model performance is a complex task which necessitates an alignment between one's model performance measurements and use case goals. Many machine learning practitioners assess model performance through count-based summary performance metrics, such as accuracy or recall, which rely on predicted labels. The components used to calculate count-based metrics are most commonly viewed in confusion matrices [42]. Such a performance assessment is sensible for situations where practitioners are concerned that their model makes the correct *decisions*. However, when one is interested in a model's predicted *probabilities*, count-based model performance analysis obscures model performance with respect to the practitioner's goals. Model predicted probabilities are important for human decision making, as probabilities are more intuitive for humans than quantities like model scores or log odds [10].

- *Peter Xenopoulos, João Rulff and Claudio Silva are with New York University. E-mail: {xenopoulos, jlrulff, csilva}@nyu.edu.*
- *Luis Gustavo Nonato is with ICMC-USP, São Carlos, Brazil. E-mail: gnonato@icmc.usp.br.*
- *Brian Barr is with Capital One. E-mail: brian.barr@capitalone.com.*

Manuscript received 31 March 2022; revised 1 July 2022; accepted 8 August 2022. Date of publication 27 September 2022; date of current version 2 December 2022. This article has supplementary downloadable material available at https://doi.org/10.1109/TVCG.2022.3209489, provided by the authors.
Digital Object Identifier no. 10.1109/TVCG.2022.3209489

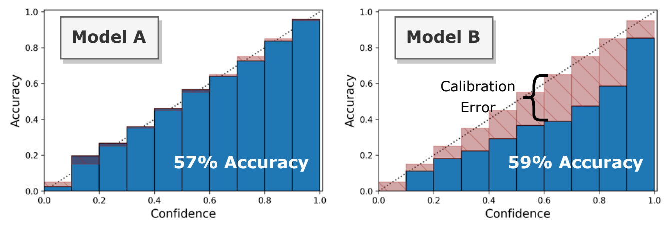


Fig. 2. Calibration plots for two deep learning models on the CIFAR100 task. Although Model B attains better accuracy, it systematically achieves a larger calibration error (i.e., the prediction confidence is far from the true prevalence).

There are many applications which depend on predicted probabilities rather than labels, such as weather forecasting, sports betting, or credit default risk prediction. In these cases, practitioners are especially concerned that their model is *calibrated*. Although there are various formal notions of model calibration, a model is generally considered "calibrated" if the model's predicted probabilities correspond to the true class occurrence. Modern machine learning models appear to achieve high performance on count-based metrics, like accuracy, yet can be systematically miscalibrated [18]. For example, consider the two models, based on ResNet-50 and built for the CIFAR-100 task, in Figure 2. While Model B has a slightly higher accuracy, it is much less calibrated than Model A, as there is a large gap between the confidence (predicted probability) and the accuracy (true class rate).

In addition to count-based metrics, it is also common to assess classifier performance through scoring rules like Brier score or log loss [6,22]. It is known that proper scoring rules are minimized if and only if classifier predictions recover the ground truth conditional distribution for all inputs. However, Kull and Flach show that proper scoring rules, like Brier score and log loss, measure more than *just* calibration [27]. Furthermore, calibration is often an important consideration for model fairness. Kleinberg et al. show that, if groups have different base rates for their labels, it is statistically impossible to ensure fairness across the balance of the positive and negative classes, as well as the calibration of the model [26]. Thus, for many applications, it is critical to deeply understand a classifier's output beyond confusion matrices, count-based metrics, and scoring rule aggregates.

To assess model calibration, practitioners typically turn to visualization, specifically to *reliability diagrams* (shown in Figure 2). Reliability diagrams are a static visualization showing the alignment between the predicted probabilities of the predicted class, referred to as the confidence, and ground-truth class prevalence for the predictions, often referred to as the accuracy. These visualizations require a data transformation that involves a binning stage, whereby predictions are binned by predicted probability, and an aggregation phase, where the average predicted probability and true class prevalence are calculated for each bin. A well-calibrated model will produce results that follow $y = x$, since the model's predictions will reflect the true class rate.

Reliability diagrams form the basis of many calibration-specific model performance metrics, since the binning and aggregation required to create reliability diagrams are identical for many calibration metrics. The aforementioned data transformation requires various hyperparameters, such as the number of bins or the bin creation strategy, which can drastically change the appearance of a reliability diagram [13]. Furthermore, due to the aggregations that reliability diagrams necessitate, relationships that may be apparent in certain regions of data may be washed out by aggregation. Additionally, the static nature of reliability diagrams hinders subgroup or instance-level analysis. Thus, a visual analytics approach may provide an intuitive link between performance and the underlying data [39].

In this paper, we present Calibrate (Figure 1), a visual analytics tool to analyze machine learning model calibration. We design and develop Calibrate through interviews with machine learning practitioners who routinely examine model calibration. Calibrate implements Learned Reliability Diagrams, a simple approach to construct reliability diagrams that can capture local calibration relationships. Additionally, Calibrate is integrated with Jupyter Notebooks, which enables analyses to be easily shared and deployed in a wide array of machine learning environments. We make the following contributions:

- Learned Reliability Diagrams, a new approach to construct reliability diagrams resistant to the pitfalls of conventional reliability diagrams.
- Calibrate, an interactive visual analytics tool to analyze model calibration. We design Calibrate to fulfill requirements derived from interviews with machine learning practitioners. Calibrate is designed with Jupyter Notebooks use in mind.
- An evaluation of Calibrate through use cases and expert interviews showing how Calibrate can be used to analyze calibration across subgroups and how we can use Calibrate to understand determinants of model calibration.

2 Related Work

2.1 Model Calibration

Calibration is a long-studied topic, particularly from the statistics, metereologic, and more recently, machine learning communities. However, there is limited visualization-specific work regarding calibration. In 1920, Hallenbeck introduced a tabular format to assess the probabilistic forecast of rain [20]. Reliability diagrams are a direct mapping from the tabular format introduced by Hallenbeck to a line plot, and are a standard calibration visualization [12,28,31,41]. Reliability diagrams effectively bin predictions based on an observation's predicted probability and compares the average classifier probability prediction of the bin to the average rate of the true label in that bin. Hagedorn et al. encodes the size of each bin through each marker's corresponding size in the reliability diagram [19]. Bröcker and Smith develop consistency bars to alleviate the problem of uneven distributions in bins [7]. More recently, Vaicenavicius et al. introduce a method to visualize calibration for three- and four-class problems [43].

Most prior work in calibration has revolved around metrics to assess a model's calibration or methods to calibrate a model. Brier first developed the Brier score in the 1950s to assess weather forecasts [6]. Brier score, like log loss, is a proper scoring rule, meaning that it is minimized when the probability distribution output by the classifier recovers the true conditional distribution [22, 46]. Kull and Flach show that proper scoring rules can be decomposed into a sum of various losses, one of which is calibration loss. Thus, proper scoring rules do not *only* capture calibration loss.

Guo et al. demonstrate that modern neural architectures often produce uncalibrated outputs [18]. Furthermore, they find that miscalibration can worsen even as classification error is reduced. To measure the calibration loss, they use a variety of metrics, such as expected and maximum calibration error (ECE and MCE, respectively), which measure calibration error relative to a reliability diagram [32]. Nixon et al. propose a new metric, adaptive calibration error, and show improvement over standard calibration metrics [35]. Vaicenavicius et al. propose a hypothesis test for the aforementioned calibration metrics to provide a more rigorous analysis of model calibration. Widmann et al. generalize common calibration metrics, like ECE and MCE, for multi-class problems [45]. While there has been much work done towards *quantifying* model calibration characteristics, there has been little work directed towards *visualizing* model calibration.

2.2 Assessing Model Performance

The ability to visualize common model performance metrics, like accuracy, precision, and recall, is a fundamental task for machine learning practitioners. Thus, the visualization capabilities needed to analyze model performance metrics are common to most machine learning platforms and libraries [24]. For example, the `sklearn` Python library allows one to easily visualize confusion matrices, ROC curves and reliability diagrams. Likewise, TensorFlow allows a user to visualize relevant performance metrics model comparison [1]. Hinterreiter et al. identify three levels of detail for model analysis, namely global-, class- or instance-level [23]. Global-level considers classifier performance across the whole data set, and may consider global metrics like accuracy. Class-level summarizes a model's performance over specific classes in the data. In practice, this may look like class-specific versions of global metrics or confusion matrices. Instance-level focuses on the errors of specific observations, based on predicted labels or probabilities.

Much class-level work revolves around the information contained within or derived from confusion matrices, a fundamental visualization in model performance analysis. Alsallakh et al. introduce the confusion wheel, which visualizes confusion between classes for a single classifier [2]. Gleicher et al. present Boxer, a visual system to explore the results of multiple classifiers [16]. Recently, Görtler et al. detail Neo, a system that generalizes confusion matrices [17]. Similar to class-level analysis, there is also strong demand from machine learning practitioners to analyze their models across subgroups in their data. Cabrera et al. propose FairVis, a visual analytics system which uses multiple coordinated views to investigate fairness metrics across subgroups of interest [8]. Dingen et al. introduce RegressionExplorer, a visual analytics tool which enables users to compare logistic regression models across subpopulations [14]. However, the aforementioned approaches do not allow for instance-level.

Increasingly, many class-level visualizations are turning towards designs that also allow for instance-level inspection. Amershi et al. propose ModelTracker, which arranges observations as boxes on a one-dimensional axis that conveys prediction score [3]. These boxes allow a user to select instances of interest, while grouping instances in an easy-to-understand way. Ren et al. describe Squares, an interactive system that also visualizes observations using boxes, and is built for multiclass classification problems [39]. Kahng et al. propose ActiVis, an interactive visualization system that integrates several coordinated views to explore deep learning models [25]. Although the aforementioned approaches may allow for subgroup analysis or instance-level inspection, none of them directly analyze model calibration.

3 Calibration Background

In this section, we provide a background on calibration. First, we enumerate and define various notions of calibration in Section 3.1. Then, in Section 3.2, we introduce reliability diagrams. Next, in Section 3.3, we define common calibration metrics. Finally, we outline issues with reliability diagrams in Section 3.4. We denote our classifier as $\widehat{\mathbf{p}} : X \to Y$, which outputs class probabilities for $1,...,K$ classes. Any instance $\mathbf{x} \in X$ input to $\widehat{\mathbf{p}}$ outputs a probability vector $\widehat{\mathbf{p}}(\mathbf{x})$.

3.1 Notions of Calibration

Filho et al. outline several types of model calibration [11]. The first, and most common, type of calibration, coined *confidence calibration*, requires that among all instances where the probability of the most likely class is predicted to be α, the expected accuracy is also α, where $\alpha \in [0,1]$. This is a commonly deployed notion of calibration since it is easy to implement, as one only has to consider the highest predicted probability. Thus, a classifier is confidence calibrated if

$$\mathbb{P}(Y = \arg\max(\widehat{\mathbf{p}}(X)) \mid \max(\widehat{\mathbf{p}}(X)) = \alpha) = \alpha \tag{1}$$

Another form of calibration is *classwise calibration*, proposed by Zadrozny and Elkan [48]. Classwise calibration requires that all one-vs-rest probability estimators, derived from the original model $\widehat{\mathbf{p}}$, are calibrated. That is, for class $i \in 1,...,K$ and a predicted probability s,

$$\mathbb{P}(Y = i \mid \widehat{p}_i(\mathbf{x}) = s) = s \tag{2}$$

where $\widehat{p}_i(\mathbf{x})$ represents the i-th index of $\widehat{\mathbf{p}}(\mathbf{x})$.

Finally, the strongest notion of calibration is *multiclass calibration*. Given prediction vector $\mathbf{q} = (q_1,...,q_k) \in Y$, $\widehat{\mathbf{p}}$ is multiclass calibrated if the proportion of classes among all possible instances on $\mathbf{x} \in X$ getting the same prediction $\widehat{\mathbf{p}}(X) = \mathbf{q}$ is equal to the prediction vector $\mathbf{q}$.

$$\mathbb{P}(Y = i \mid \widehat{\mathbf{p}}(\mathbf{x}) = \mathbf{q}) = q_i \text{ for } i \in 1,...,K \tag{3}$$

3.2 Reliability Diagrams

Reliability diagrams are a popular method to assess model calibration [12,34]. Reliability diagrams group predictions into discrete bins and plot the expected accuracy on the y-axis and average classifier confidence on the y-axis. We call this line calibration curve. A calibrated model is one where the difference between the expected accuracy and the average classifier confidence is small. Thus, a perfectly calibrated model would follow the line $y = x$. Reliability diagrams are specified for a particular class of interest.

To estimate the expected accuracy, we group data into "bins". Typically, this is done by dividing predictions into W bins of width $1/W$ along the range of $[0,1]$. We let B_w be the set of indices of observations whose prediction confidence is contained within the interval $I_w = (\frac{w-1}{W}, \frac{w}{W}]$. The accuracy of B_w, for a given class i, is defined as

$$\text{acc}(B_w) = \frac{1}{|B_w|} \sum_{j \in B_w} \mathbb{1}(\widehat{y}_j = y_j) \tag{4}$$

where $\widehat{y}_j$ and y_j are the predicted and true class labels for observation $\mathbf{x}_j$, and $\mathbb{1}$ is an indicator function. We can define the confidence of B_w as

$$\text{conf}(B_w) = \frac{1}{|B_w|} \sum_{j \in B_w} \widehat{p}_i(\mathbf{x}_j) \tag{5}$$

where $\widehat{p}_i$ is the predicted probability of class i for observation j.

In some implementations, like `sklearn`, users define the number of bins, and then a strategy which will either generate bins of equal number of samples or of equal width. These parameters can be difficult to select, as they have drastic implications on the reliability diagram itself. We show an example of a reliability diagram in Figure 3, as well as demonstrate how the parameters, namely the number of bins and binning strategy, can drastically change the visual representation of model calibration.

3.3 Calibration Metrics

Proper scoring rules are loss measures that are minimized when the predicted class distribution equals the true class posterior distribution [27]. The two most well-known proper scoring rules, which are also used as surrogate losses for optimization, are Brier score and log loss. Brier score (ϕ_{BS}) and log loss (ϕ_{LL}) are defined as

$$\begin{aligned} \phi_{BS} &= \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} (\widehat{y}_{ij} - y_{ij})^2 \\ \phi_{LL} &= -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \cdot \log(\widehat{y}_{ij}) \end{aligned} \tag{6}$$

where N is the total number of observations, K is the total number of classes, $\widehat{y}_{ij}$ is the predicted probability of class i for observation j, and y_{ij} is equal to 1 if i is the true class of observation j and 0 otherwise.

While scoring rules are minimized when the predicted class distribution from the classifier equals the true class posterior, we also know that proper scoring rules measure more than just calibration loss. This decomposition is shown in [27]. Thus, we can construct strict measures of calibration using the information calculated that we use in reliability diagrams, namely, the accuracy and confidence of each bins. From these quantities, we can calculate the expected calibration error (ECE) and maximum calibration error (MCE) [32]. We define ECE and MCE as

$$\begin{aligned} ECE &= \sum_{w=1}^{W} \frac{|B_w|}{N} |\text{acc}(B_w) - \text{conf}(B_w)| \\ MCE &= \max_{w \in \{1,...,W\}} |\text{acc}(B_w) - \text{conf}(B_w)| \end{aligned} \tag{7}$$

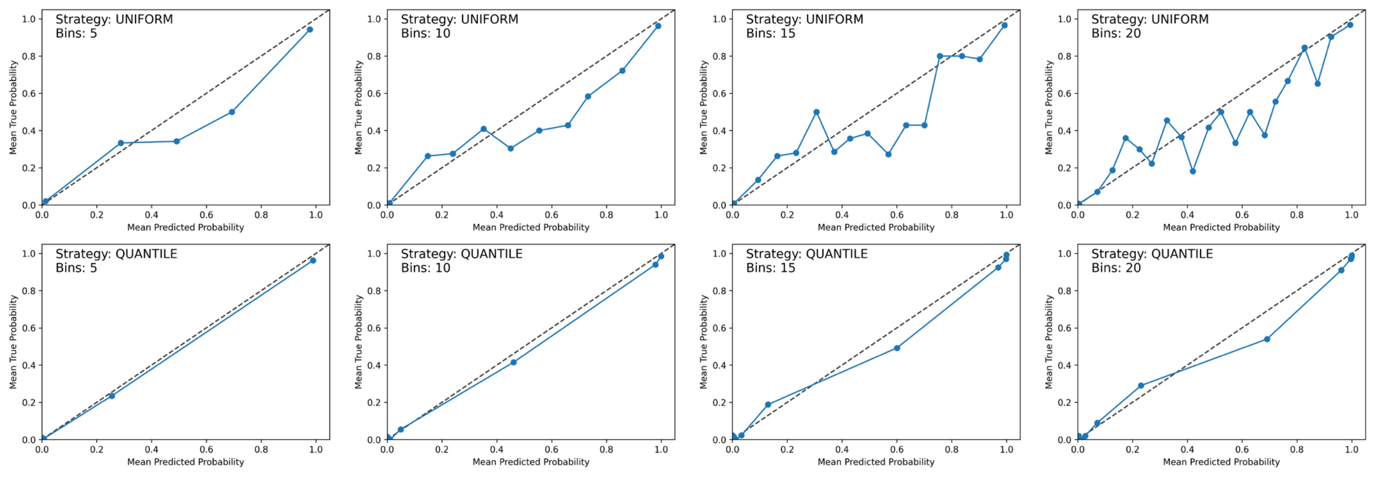


Fig. 3. Hyperparameters of reliability diagrams, namely the number of bins and binning strategy, can not only drastically change a diagram's visual representation but also the calculation of downstream metrics like Expected or Maximum Calibration Error. For example, the "quantile" strategy (bottom row), which creates bins of equal number of observations, would suggest the above model is well-calibrated. However, the "uniform" strategy (top row), which creates bins of identical width, suggests that the model is not well-calibrated between predicted probabilities of 0.5 and 0.9.

3.4 Issues with Reliability Diagrams

Reliability diagrams are simple to compute, requiring just a single pass through a model's predictions. In this pass, one assigns each prediction to a bin. Each bin contains a range of predictions, like those of $[0, 0.1)$, $[0.1, 0.2)$, and so on. The aforementioned simplicity makes sense from a historical point of view – Hallenbeck tabulated rain forecasts into a table of ten bins in 1920, long before modern computing [20]. However, we still see influences of Hallenbeck's design choices in modern day calibration analysis. For example, many works still use ten bins or fewer [18, 33, 34, 47]. The number of bins is important since it strongly influences the construction of popular calibration metrics, such as expected calibration error. Nixon et al. detail various problems with expected calibration error due to the design choices stemming from the binning and aggregation procedures required for a reliability diagram [35]. In turn, these issues may also be associated with reliability diagrams. We discuss a few salient issues below.

Issue 1: Variability from Bin Selection. Even small changes for the number of bins can have drastic effects on the produced reliability diagram. Furthermore, binning strategies, such as fixed bin width or quantile binning may induce certain shapes on the reliability diagram depending on the distribution of the predictions. Effectively, a large bin count may create many small bins that have high variance. On the other hand, a small bin count may wash out local calibration information.

Issue 2: Competing Bin Effects. One issue with reliability diagrams is that binning may wash out competing effects between overconfident and underconfident predictions in a bin. In practice, such an occurrence is common, and it hinders the ability of a practitioner to identify regions of interest in a reliability diagram. In the worst case, the effects would cancel to show no calibration error for a given bin, even though there may be substantial miscalibration within that bin. Thus, the user-specified binning parameters may have strong impacts.

Issue 3: Importance of Prediction Distribution. The distribution of predicted class probabilities is unlikely to be uniform in practice. In such a case, just a few bins will contribute the most to the expected calibration error. However, in modern reliability diagrams, each mark representing a bin is typically a single point; bin size is not usually visually encoded. Thus, it is possible for two reliability diagrams to look identical, yet have vastly different calibration errors due to the underlying distribution of predicted probabilities. The issue of conveying bin sample size is also important for uncertainty estimation – bins with low counts of observations will inherently have larger confidence intervals for their estimated average class prevalence. Possible solutions include encoding bin size through marker's size [19] or by computing a confidence interval to visualize on the reliability diagram itself [7].

4 CALIBRATE

In this section, we describe Calibrate, a visual analytics tool that enables interactive analysis of model calibration. First, we present the results of interviews with machine learning experts who routinely examine model calibration. We use these interviews to construct a list of design requirements. Then, we outline the various views in Calibrate, and how these views address the requirements identified in Section 4.1. Finally, we discuss the implementation of Calibrate, which is available for use in Jupyter notebooks. We show an example of Calibrate applied to a predictive model trained with real world data, along with a brief description of Calibrate's views, in Figure 1.

4.1 Design Requirements

We conducted interviews with four machine learning practitioners who actively work on developing, refining and deploying machine learning models, especially those which predict probabilities. We denote each practitioner as P1 through P4. Each practitioner was familiar with reliability diagrams and used them in their work. We asked each expert about their usage of reliability diagrams, particularly how they create them, what types of tasks they attempt to complete using reliability diagrams, and what issues they have encountered using reliability diagrams in practice. Where relevant, we link expert interview feedback to issues described in Section 3.4.

For each practitioner, reliability diagrams were a common tool used in their analyses. P2 remarked that, especially recently, their organization is using more reliability diagrams to evaluate their models. A common characteristic among all practitioners was that reliability diagrams were most used in content created for other technical staff, such as analysts, data scientists, or data engineers, rather than for less technical stakeholders. However, P4 also mentioned that reliability diagrams may be more intuitive when they present a model's performance to non-technical stakeholders, noting that reliability diagrams are "generally easy to explain" and they "answer a natural question." Most practitioners engaged in binary classification tasks. For multiclass problems, they would typically use a one-versus-rest approach when constructing reliability diagrams.

All practitioners identified subset analysis as a routine task when analyzing model calibration with reliability diagrams. This task involved manually filtering data to create subgroup of interest and then creating reliability diagrams for these subgroups. Each practitioner mentioned that subgroup analysis was primarily a manual process and involved much domain knowledge to construct subgroups. P1 mentioned that typically they went into this subset analysis knowing specific subsets to analyze, based on domain knowledge. On the other hand, P3 remarked that they would "sanity check" their model across broad subgroups, particularly across subgroups that they may expect to be harder to predict. While these subgroups would be defined by filtering the data by its features, P4 noted that sometimes, although rarely, they would construct subgroups by manually grouping instances.

When practitioners found a subgroup which was miscalibrated, many of them turned to feature engineering rather than post-hoc corrections like Platt scaling or isotonic regression. For example, when they found a miscalibrated subgroup, P4 mentioned they would typically go back and attempt to create new features to target that specific subgroup. Similarly, P1 remarked that when they found an important subgroup which was miscalibrated, they would either perform more feature engineering, or they would create a separate model for their subgroup. In some cases, practitioners would change their model choice, as there was some understanding among the practitioners that model architecture had an impact on calibration. There were also shared concerns on class imbalance and their effects on calibration by P1, P2 and P4. We investigate this topic further in Section 5.2.

Displaying the density of the predictions was a common complement to reliability diagrams. In particular, this visualization was deemed by most practitioners to be important for understanding where their model's calibration estimate was more unreliable (Issue 3). Intuitively, where one has lower density of predictions, one would expect the "error" to be higher, as mentioned by P2. P1 mentioned that they sometimes used error bars on their reliability diagrams, but oftentimes opted to plot the distribution of the predictions underneath the reliability diagram, as most other stakeholders could infer which regions had highest variability. Furthermore, P1 described uncertainty estimation in common calibration libraries as having little support.

A common issue with conventional reliability diagrams was selecting the number of bins. Most practitioners used the default parameters or between 10 and 20 bins. P1 mentioned that they had "only ever seen bins between 10 to 20", and that they found a many bins as unhelpful in analyzing model calibration. P4 noted that they were not familiar with "industry practice" and that they had concerns over reliability diagrams potentially returning different conclusions due to different numbers of bins (Issue 1). Although the practitioners were seemingly less worried about competing effects within bins, P1 mentioned that "the binning obscures a lot of fine detail" (Issue 2), and that the static nature of reliability diagrams does not encourage exploration.

The practitioners unanimously stated that they implement reliability diagrams using popular packages in Python and R, and in particular, in Jupyter Notebooks. For example, P1 mentioned that they "perform virtually all model analysis and produce all reliability diagrams in Jupyter". One common aspect mentioned by the practitioners was that Jupyter Notebooks are easily reproducible, and therefore they were a desired for disseminating model performance analysis to other stakeholders. Additionally, P2 mentioned that much of their organization's model performance analysis is done in notebooks where the model building also takes place. P3 noted that although model training occurred outside of Jupyter, they would frequently read model predictions into a Jupyter notebook where they would analyze the model's calibration.

From the above interviews, we compiled the following requirements:

R1 Allow for identification of interesting calibration regions. Hyperparameter choice is a difficult task for our practitioners and indeed has effects on the visual representation of model calibration. In many cases, practitioners simply use the default parameters given by whichever library they use. However, as we describe in Section 3, traditional diagrams suffer from a variety of issues. Our system should address the issues with traditional reliability diagrams, while also allowing a user to easily change the parameters for conventional reliability diagrams.

R2 Connect performance to data, especially in bins. Due to the static nature of reliability diagrams, it is difficult for practitioners to inspect specific bins and analyze the instances that comprise the selected bin. Users should be able to select predictions regions, aside from the bins returned by conventional reliability diagrams, and analyze the instances contained within the region.

R3 Allow for subgroup analysis for reliability diagrams. A fundamental task is to analyze calibration on subgroups of predictions. Finding these subgroups is generally a manual process, whereby the practitioners rely on prior knowledge to manually define and filter subgroups. Thus, we should allow for users to interactively investigate and compare calibration among subgroups of predictions.

R4 Integrate with Jupyter Notebooks. The practitioners that we interviewed made their reliability diagrams through calibration-specific packages in Python and to a lesser degree, R. Overwhelmingly, they also deployed the code to produce these diagrams in Jupyter Notebooks, which they found easier to disseminate to other stakeholders in their analytics processes. Furthermore, because many practitioners built their models in Jupyter, they prefer to perform their model analysis near to the model training workload. Therefore, it is important for the tool's implementation to be compatible with Jupyter Notebooks.

4.2 Learned Reliability Diagrams

Reliability diagrams are effectively an attempt to estimate the relationship between a model's predicted probabilities and their associated true outcomes. However, since we only know an observation's label outcomerather than its true probability, we use binning to estimate the "true probability" for a collection of samples. In practice, traditional approaches to reliability diagrams use 10 to 20 bins [18]. Typically, these bins do not overlap, and they encompass distinct, uniform ranges of the model predictions. One variation is to use a "quantile" binning strategy, whereby bins are created according to the distribution of the data. However, as we show in Figure 3, the quantile binning strategy, for some model outputs, can cause most bins to be created near 0 or 1, which complicates analysis of model calibration between the extreme predictions. Another approach to solve the aforementioned issues with reliability diagrams is to use overlapping bins. To the best of our knowledge, the only prior work using overlapping bins is that of Caruana and Niculescu-Mizil, which sorts predictions and creates rolling bins of 100 samples each to calculate confidence and accuracy [9]. One issue with overlapping bins is that the distribution of the data may result in bins that encompass large ranges of the predicted probability distribution. Thus, local calibration changes may be muted.

In general, setting the appropriate binning parameters, such as the number of bins or binning strategy, is a difficult activity for many practitioners, and many rely on comfortable past choices. The choice of bins can lead to drastically different conclusions. In Figure 4, we show various bin sizes for predictions from a random forest classifier trained on the Wisconsin Breast Cancer dataset. By simply considering ten instead of eight bins, we see a large difference in the conclusion one may draw about the model's calibration. Oftentimes, a small number of bins may suggest the model is calibrated, as the bins encompass larger regions, and a large number of bins suggests the opposite, as the bins have few samples in them.

To address the issues arising from binning parameter selection for conventional reliability diagrams (**R1**), we propose to *learn* reliability diagrams. Taking a set of predicted probabilities X, along with the true labels for these predictions Y, we learn a new function $f : X \to Y$. This univariate function can be learned through any classifier that produces probabilistic predictions, such as a boosted tree or a generalized additive model. If our underlying predictions are calibrated, then f will closely follow the 45-degree line for all inputs. Our approach is similar to post-hoc calibration. In post-hoc calibration, one attempts to fix a model's predicted probabilities after the model has generated its predictions. Typically, one imposes a monotonic functional form on f, such as through a sigmoid (known as Platt scaling) [38] or isotonic regression [48]. However, in our case, the goal is not to fix the original predicted probabilities, but rather *summarize* their relationship with the true probabilities in a manner that is less sensitive to binning parameter choices. Thus, while in practice f can be any classifier, we generally want to choose f in a way that is agnostic to the f's shape (e.g., using logistic regression for f would impose a sigmoid shape) and in a manner that is stable under parameter changes, since many of choices of f

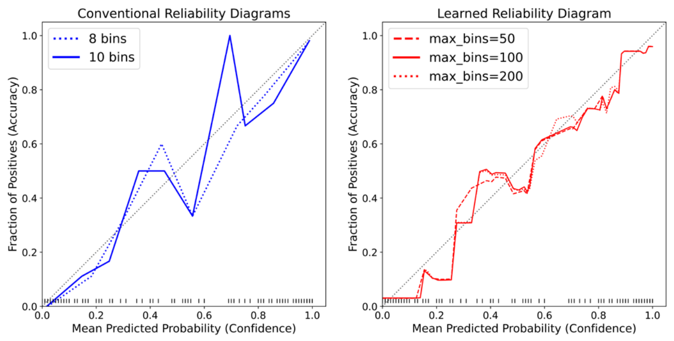


Fig. 4. Example of conventional 8 and 10 bin diagrams (left) along with learned reliability diagrams for a random forest classifier trained on the Wisconsin Breast Cancer dataset. The density of the predictions is shown below each graph using a rug plot. Learned reliability diagrams are resistant to parameter change, while simply going from 8 to 10 bins induces large changes.

would require specific parameters. After learning f, we can plot $f(x)$ for all $x \in [0,1]$ to visualize the *learned reliability diagram* (Figure 4). A learned reliability diagram is understood the same way as a conventional reliability diagram. We can still calculate ECE for our learned reliability diagram by determining the total area between the curve and the 45-degree line. Lastly, since our prediction task only takes a single variable as input (the original model predictions), learned reliability diagrams are fast to generate.

By estimating the relationship between the predicted and true probabilities through a continuous function rather than through discrete intervals, we gain a few advantages. First, the issues of competing effects within a bin or variability imposed by binning parameter selection are largely avoided since learned reliability diagrams are beholden to the distribution of the predictions rather than arbitrary, user-selected binning parameters. Furthermore, many choices of f grant us the ability to easily extract confidence interval estimates. For our work, we use Explainable Boosting Machines (EBMs) [29, 30, 36], which are available in the `interpret` Python package. While some choices of f may still necessitate parameter tuning, we find that EBMs provide stable results across a wide parameter set. For example, in Figure 4, we vary the “max bins” parameter in EBMs and find that the resulting learned reliability diagrams are quite resistant to parameter changes.

4.3 Calibration View

Building upon users’ familiarity with reliability diagrams, the Calibration View (Figure 1-C) visualizes both traditional and learned reliability diagrams (**R1**). Users may change the number of bins through a selection box at the top of the view (Figure 1-B). To visualize calibration for different classes, we allow a user to select the class of interest, which presents the confidence calibration for the selected class. Additionally, users may interact to display learned reliability diagrams or to clear the current view. The calibration view is tied to the other views through select and brushing operations. Since a user may create many calibration curves, we allow the user to hover on a calibration curve, which displays a tooltip with salient information about the selected set of predictions, like expected calibration error. Furthermore, the user may click to “select” the calibration curve, which identifies which model to consider for the brushing operation which populates the Instance and Performance views. With a curve selected, users may also brush to select a prediction region of interest. Using the x-axis values of the brushed region, we then populate the Instance and Performance views with predictions that are within the brushed region.

Each interviewed expert mentioned that they often coupled a plot of the prediction distribution with the reliability diagram, as it helped to identify areas with high uncertainty due to low sample size. At the bottom of the selected curve, Calibrate provides a histogram of predictions for the observations used to create the selected curve. Understanding the distribution of predictions is important since, especially for difficult multiclass prediction problems, models may not even produce predictions with high confidence. Furthermore, with a density plot, users will be able to gauge the uncertainty of a prediction region, as regions with small amounts of samples will generally have more uncertain estimates.

4.4 Feature View

Subgroup analysis is an important yet arduous task for most users. To analyze subgroup calibration, users first define subgroups of interest, often on a single feature, but potentially on groups of features. Typically, these actions are performed manually in Jupyter Notebooks cells. Thus, with our Feature View (Figure 1-E), we seek to make the subgroup creation process interactive to enable quick and iterative analysis. P1 mentioned that they typically view subgroups in terms of “distributions”. Thus, we provide a view for users to create subgroups through brushing feature distributions (**R3**). Being able to visualize the distributions of features may also help practitioners understand the limitations of their own data. For example, P1 mentioned that through exploratory data analysis on their features, they sometimes find important subgroups for which they may lack data, and thus they investigate the calibration characteristics a subgroup defined in this low density region.

In the Feature View, a user must first select what features to visualize. Users may brush each histogram by dragging across a region of the histogram. Once a user has brushed a region, the user then can create a new reliability diagram. The samples that fit the brushed selection will populate a reliability diagram in the calibration view. One may one-hot encode categorical variables, therefore making subgroup creation a matter of brushing either the 0 or 1 part of the feature’s domain. Within the Feature View, a user can scroll to see the full list of features.

4.5 Instance View

Exploring errors on the instance-level is an important aspect of model performance analysis. However, due to the binning required, along with the static nature of reliability diagrams, instance-level inspection is usually not performed in calibration analysis. Furthermore, it is unclear how to assess calibration error on the instance level – two instance may differ in true and predicted labels, but assessing the difference between predicted probability and outcome is akin to a proper scoring rule, which is why binning is employed to measure calibration. Thus, it is difficult to connect performance (confidence regions) to particular instances. Nevertheless, inspecting individual instances manually can still yield interesting outcomes, and may guide model improvement. To satisfy **R2**, we implement the Instance View (Figure 1-D). As users brush on the calibration view, the instance view updates to reflect the instances in the selected prediction rage. Each row contains information on the features of each instance and the mean of each feature is shown at the bottom of the instance view.

4.6 Performance View

Ultimately, confusion matrix-derived metrics are still important to visualize for model performance analysis, and was a standard procedure among the experts we interviewed. Thus, it is important to present these classic model performance measures alongside calibration performance measures. Therefore, we implement the Performance View (Figure 1-F), which visualizes a confusion matrix, which forms the basis of many count-based performance metrics. The performance view is updated as users brush on the calibration view to select confidence ranges of interest or when a user selects another calibration curve.

4.7 Implementation

Given many data scientists’ background in Jupyter, we provide a solution compatible with Jupyter notebooks (**R4**). The main justification in providing a Jupyter-first tool was that, from our expert interviews, practitioners much preferred to keep their analysis within Jupyter notebooks. In fact, none of our interviewed experts relied on outside programs to analyze their models. The two main avenues of model analysis were either (1) combine model training and performance analysis into a single notebook or (2) model training, particularly for large models, was offloading to another resource, and a notebook read in the predictions for model analysis.

Calibrate is available as a Python library and is designed with Jupyter Notebooks use in mind. The front-end is implemented via JavaScript with React and D3 [5]. The back-end, which creates reliability diagrams, is implemented in Python using Numpy [21], Pandas [44] and Interpret [37]. We use Interpret for its EBM implementation with its default parameters. To use Calibrate, a user first creates the `Calibrate(data)` class, where `data` is a dataframe with rows as observations and columns as features. To add models to the Calibrate, users use the `.add_model(preds, labels, model_name)` method, where `preds` is an $N \times K$ matrix of predicted probabilities, `labels` is a $N \times K$ one-hot encoded matrix representing the labels of the observations in `data`, and `model_name` is a string indicating the model name. K is equal to the total number of classes. Finally, to visualize the Calibrate widget, a user may call the `.visualize()` method.

5 Evaluation

5.1 Case Study 1: Identifying Miscalibration in Subgroups

Analyzing model calibration among subgroups is an important task. Oftentimes, discovering that a model is miscalibrated for specific subgroups leads to important business or technical decisions. For example, if a sports bettor discovers their game outcome model is miscalibrated for games that occur at night, they may decide not to bet on such games. Like our experts indicated, if a machine learning practitioner uncovers miscalibration among subgroups in their model, they may decide to engineer more features or to change their model architecture. In this use case, we show how Calibrate can be used to analyze model miscalibration among subgroups in a real world data set.

The COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) data was released by ProPublica in 2016 and is based on Broward County data from January 2013 to December 2014 [4]. The data consists of information about defendants, such as their sex, race, or previous arrests, which are used to create models to predict a defendant's risk of recidivism. These models have increasingly come under scrutiny, due to their ability to cause significant harm to individuals. While the issue of whether or not these models should be used in practice is best left to other domains, understanding recidivism model calibration is nonetheless important and provides a unique example of how miscalibration may harm individuals.

We split our data with 50% of observations going to training and 50% going to testing. To start, we consider a defendant's sex, charge degree (misdemeanor or felony), juvenile misdemeanor and felony counts, as well as total prior charge count. We train a random forest classifier, which was noted by our interviewed practitioners as a common model choice they use in practice. Additionally, we consider a multilayer perceptron (MLP) and a logistic regression as other candidate models. We use the default parameters and implementation in sklearn for each model. Using Calibrate, we show the model's associated reliability diagram, along with its learned reliability diagram, in Figure 5.

Figure 5 shows that the models are well-calibrated, with the exception of our random forest model in the very low and very high predictions, where we see under and overconfidence, respectively. Overconfidence is indicated by values of the reliability diagram below the $y = x$ perfect calibration line. These are samples which are generally predicted as having a higher probability than the true occurrence rate. Conversely, underconfidence implies that the model predicts a lower probability than the true occurrence rate. For our domain, we may be especially concerned with model overconfidence, as such a characteristic results in higher predicted probabilities than the true rate, meaning a defendant will be rated as higher risk than they truly are, on average.

While investigating global calibration is typically the beginning of any calibration analysis, it is also important to check a model's calibration characteristics across subgroups of interest. In practice, one would perform subgroup analysis by using domain knowledge to manually define subgroups, through packages like Pandas [44]. Analyzing regions of the global reliability diagram can also be a useful direction to identify subgroups. Using the brushing interaction, we analyze both the underconfident and overconfident regions in Figure 5. Interestingly, through the instance view, we observe many young defendants in the overconfident region, which has an average age of 33. In this region,

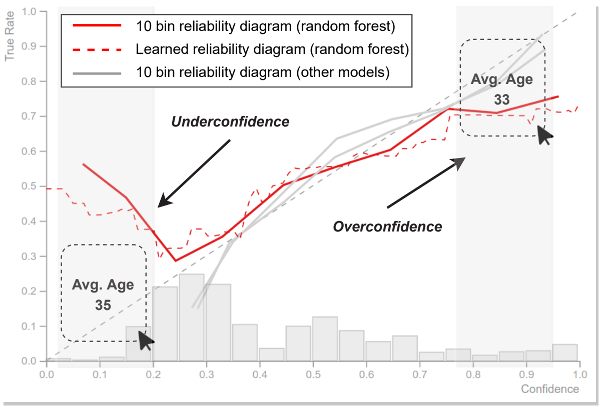


Fig. 5. Reliability diagrams for each model, which consider a defendant's sex, charge type, and prior offense counts, on the full COMPAS test set. We can see significant regions of under and overconfidence in the random forest model (red, selected). The random forest's learned reliability diagram is shown as the red dotted line, and its histogram of the predicted probabilities is shown along the x-axis. Brushing indicates regions of predictions with above and below average age through the instance view.

the average predicted probability is about 0.9, yet the average true rate is only about 0.75. Conversely, we see that the underconfident region is slightly older, with an average age of 35. At the same time, at the extremely low end of the underconfident region (i.e., prediction ≤ 0.1), the average age is just 31.

Thus, it is reasonable to posit that the miscalibration, particularly in the low and high predictions, may be correlated to some degree with defendant age. Therefore, we define a subset of data encompassing older defendants, which we define as those of 45+ years of age. Using Calibrate's feature view, we create the associated reliability diagram for the "old" subset in Figure 6. Interestingly, we see that this group faces systematic overconfidence in its predictions, regardless of the selected model choice. Many of our interviewed experts noted that when faced with model miscalibration, it was common for them to hypothesize what features they may be missing. In this instance, it is reasonable to assume that, since we do not include age as a feature, that we may be observe miscalibration associated with age. Thus, we may also include a defendant's age in our model and observe the resulting reliability diagram in Figure 6. We see that the models become more calibrated on the old age subgroup, yet suffer no large deviation in accuracy. Interestingly, we also see that the characteristics of the lower 20% of predictions changes. For example, without "age" as a feature, these predictions are about half male and half female with an average of 17 prior arrests. However, when considering age, this subset of predictions drops to an average of about 1.5 prior arrests. Thus, the instance view can be useful for learning about the characteristics of prediction regions.

P1 noted that they sometimes noticed miscalibration with certain model choices. For example, P1 stated "we often use logistic regression as we noticed its outputs were well-calibrated for many prediction tasks". Aside from feature engineering, another avenue one may consider exploring is model architecture. While this may seemingly address global model calibration issues, as demonstrated in Figure 5, subset analysis is still important. For example, consider the right panel of Figure 6, where two models can exhibit similar performance on labels, yet have markedly different calibration characteristics.

5.2 Case Study 2: Analyzing Determinants of Calibration

Understanding the factors that may induce miscalibration in a model is important. The experts we interviewed typically hypothesized miscalibration as a consequence of one of two factors: (1) lack of useful features or (2) class imbalance. For example, P4 mentioned that if a model was miscalibrated, they would try to answer questions like "what

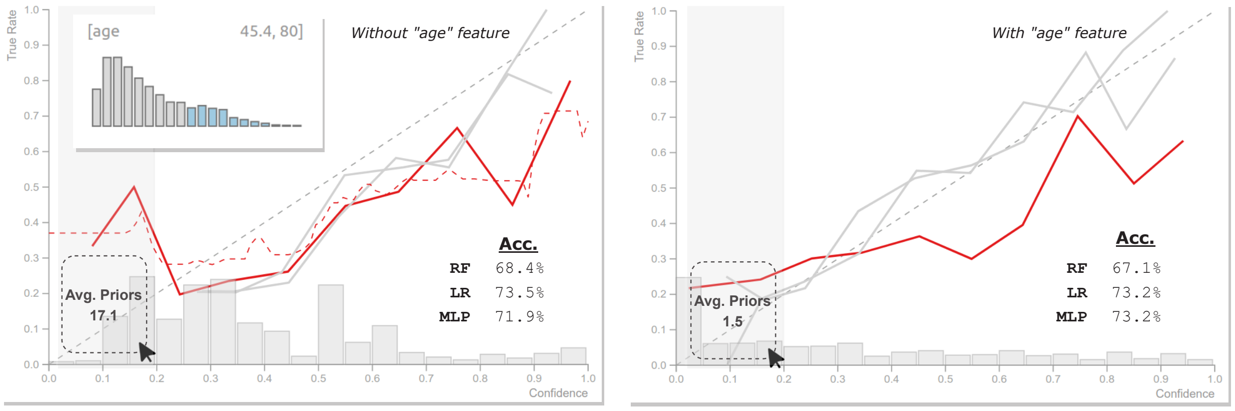


Fig. 6. Reliability diagram for "old age" subgroup. Red indicates random forest model, gray indicates other models. Histograms of random forest predictions are shown along the x-axis. Left panel shows models *without* age as a feature, and the right panel shows models *with* age as a feature. We see systemic overconfidence in the predictions for all models that do not consider age as a feature.

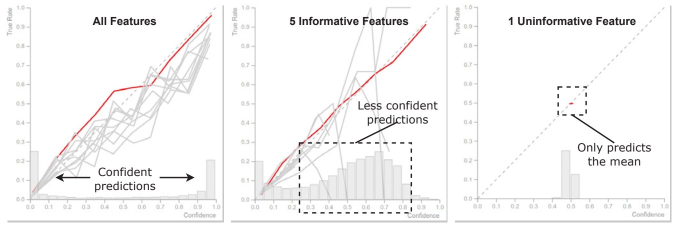


Fig. 7. Dominant class reliability diagram by feature set (red). Each minority class is in gray. Histograms of dominant class predictions are shown along the x-axis. Models trained on all features or 5 informative features are well-calibrated, although the latter has less confident predictions (further from 0 or 1). Using a single uninformative feature produces a model which purely estimates base class – this model can deceptively yield zero calibration error (calibration curve shown in dotted box) yet be an uninformed model.

features am I missing?" and "what is the balance of the classes in the data sets?". In this use case, we investigate how the aforementioned factors influence model calibration through experiments with synthetic data. We use sklearn's "make_classification" implementation to design a classification task with 10 classes. We generate 20 features for each observation, with 10 of the features as informative and the remaining 10 as noise. We randomly select 50% of the generated samples for training and the remaining 50% for testing. We train a multilayer perceptron using the default parameters provided by sklearn.

Feature Set

Many experts mentioned that they would turn to feature engineering or to collecting different features when faced with model miscalibration. For example, P1 pointed to a real-life use case where they identified a subgroup that was poorly calibrated, which they fixed by creating bespoke features for the subgroup. Thus, it is important to understand the effect of the feature set on model calibration. We also visually demonstrate how, somewhat paradoxically, a model with only uninformative features is often calibrated. We consider a classification task on 10,000 total observations where one class comprises 50% of all observations, and the remaining 50% of observations are spread equally among the other 9 classes.

We consider three feature set scenarios: (1) all features, (2) only 5 informative features, (3) 1 uninformative feature. In Figure 7, we visualize the calibration characteristics of the most prevalent class in red, along with the minority classes each as a gray line. We see that the model is well-calibrated when all features are used, as well as when only 5 informative features are used. Although our model is well-calibrated in (1) and (2), the model attains a much lower accuracy in scenario (2). Thus, although two models may both be well-calibrated, shown in the calibration view, such a finding does not guarantee similar accuracy, found in the performance view. Furthermore, using the histogram, we notice that as the predictions become less confident (farther away from 1 or 0) as the number of informative feature decreases.

For scenario (3), we see a single point around 0.5 confidence and accuracy. Effectively, our model is learning no relationships in this case. Thus, for our dominant class, the model always predicts its base rate – 50%. In a sense, this model is very well-calibrated since its expected calibration error will be 0. However, intuitively, this model has low accuracy. Thus, we show a canonical example with zero calibration loss, but high error rate (and thus high log loss). Furthermore, this demonstrates a strong case for the use of reliability diagrams in investigating calibration – simply calculating expected calibration error would not have led to the same conclusion.

Class Imbalance

Class imbalance, where a particular class is more prevalent than others (known as the dominant or majority class), is common in real-world data. Consider a simple data set with two classes, A and B, where the ratio of A instances to B is 99 : 1. A naive model which always predicts class A as 0.99 and class B as 0.01 will achieve 99% accuracy and *technically* be perfectly calibrated, as we saw above. However, this model would unlikely be used in practice, since it provides no new information to practitioners. Furthermore, imbalanced class scenarios often carry imbalanced costs between errors for different classes, which make identifying minority class instances important. Therefore, it is important to understand class imbalance's relationship with calibration.

We consider three scenarios: (1) equal class balance, (2) dominant class is 50% of samples, (3) dominant class is 90% of samples. In total we use 10 classes, and the non-dominant classes consist of equal portions of the remaining samples. In Figure 8, we show the reliability diagrams for the dominant class (red) and the non-dominant classes (gray), which we created by toggling the selected class in the calibration view. Intuitively, the majority class instances are well-calibrated across the three imbalance levels. However, we see that as class imbalance increases, minority instances become systemically miscalibrated, as indicated by the many gray lines below the 45-degree dotted line. The direction of the miscalibration is also towards overconfidence. Overconfidence implies that the model predicts a class with high probability, yet the true probability, as indicated by the reliability diagram, is low.

One common approach to address class imbalance is to resample the training data in a way that evens out the class distribution. Undersampling evens the class distribution by sampling only a portion of the majority class so that the sampled number of points equals the total minority sample count. Oversampling resamples the minority classes so that the sampled number of points equals the total majority sample count. We run an experiment using the data from the 90%

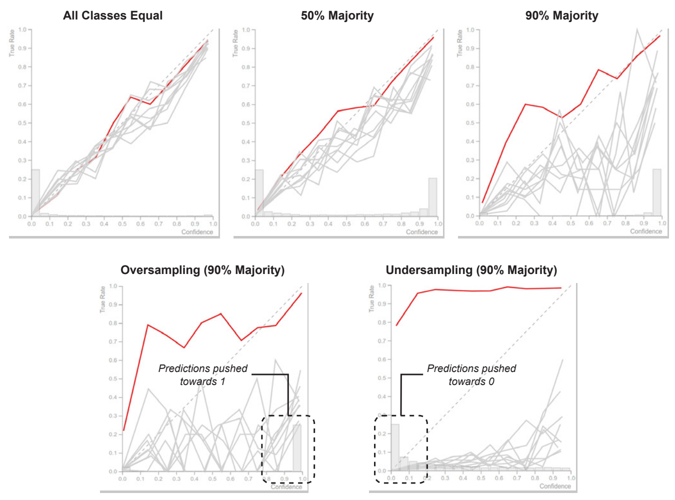


Fig. 8. Reliability diagram by class imbalance showing the majority class (red) and minority clsses (gray). The majority class histogram is shown along the x-axis. Each model is well-calibrated for the majority class. As class imbalance increases, minority class calibration worsens, and trends towards overconfidence. Class balancing techniques appear to make model predictions even more miscalibrated.

majority dataset. We show the results of applying basic over and undersampling in Figure 8. In our experiment, class-based resampling has significant effects on the resulting diagram, which appeared to make our predictions miscalibrated. Specifically, the majority class predictions became severely underconfident, while minority class predictions became severely overconfident. Furthermore, the prediction histogram shows that oversampling pushed majority class predictions towards 1, while undersampling pushed majority class predictions towards 0.

5.3 Expert Feedback

We conducted "think aloud" interviews with the same experts who had previously helped us identify the system requirements. In this session, the experts had access to Calibrate via a Jupyter notebook. The notebook was preloaded with a trained model on the Census Income data set [15]. Before participants interacted with Calibrate, they were given a brief demo of Calibrate's functionality. Participants were free to explore the results of Calibrate in whatever way they wanted.

User feedback was very positive. The experts mentioned that they found the system useful and were interested in using Calibrate for their own work. In particular, the experts appreciated the ability to quickly create subgroups through interaction. P1 mentioned that in the future, it may be useful to have an ordering to the features, such as by their variance. The experts also enjoyed the interaction of selecting a calibration curve, and seeing both the prediction distribution and instances. P2 remarked that the information displayed in Calibrate was clear and interpretable, and that they may consider using the tool for non-technical stakeholders as well. P3 found Calibrate easy to use, especially due to its Jupyter implementation.

P1 mentioned that although the learned reliability diagram was intuitive as a concept, that the shape of the learned diagram was unintuitive. For the given demo, we utilized an explainable boosting machine [37] for the learned reliability diagram, which resulted in a "step-like" curve, according to P1. Because of this, P1 mentioned that they confused the steps in the curve as areas of significant calibration changes. When shown a learned reliability diagram using splines, P1 remarked that the smooth function created by the splines was much more intuitive.

6 Discussion

One limitation of our work is that we do not directly tackle the issue of multiclass calibration visualization, and we opt for a one-versus-rest approach instead. In part, this design choice is due to the ubiquity of traditional reliability diagrams, which appear to be easily understood by both technical and non-technical audiences. Furthermore, it was unclear from our domain requirements interviews that multiclass calibration was commonly analyzed, and if so, a one-versus-rest approach was often deployed. Works like Vaicenavicius et al. propose a method to visualize calibration across prediction tasks with three and four classes [43]. One solution to multiclass calibration visualization may be to adopt a Squares-like [39] approach, whereby each class is represented on a single horizontal axis by a vertical reliability diagram. Parallel coordinates could also be used to link bins and instances across classes.

One limitation of learned reliability diagrams is that the selected model architecture may induce particular reliability diagram shapes. For example, if one uses logistic regression as the basis of their learned reliability diagram, then one would see a sigmoid relationship, which, in some instances, may imply overconfidence in the high probabilities and underconfidence in low probabilities. For the case of EBMs, which we use to produce our learned reliability diagrams, we found that, for low max bin counts, the learned reliability diagram sometimes appeared underconfident at the low end of the predictions. However, as shown in Figure 4, setting the max bins parameter to a high number has little effect on the shape of the learned reliability diagram. Thus, by using the default parameters of EBMs, one may avoid this problem.

While calibration has traditionally been a topic covered mostly in data mining or machine learning, there are exciting avenues for calibration work in visualization. For example, developing and evaluating new visual encodings for model calibration is particularly interesting. Reliability diagrams rely on many visual design decisions, such as whether to use points or bars, how to visualize uncertainty, or how to set the number of bins. Furthermore, as we saw in our expert study, some users may find certain learned diagram shapes confusing. Many of the questions regarding the aforementioned design decisions may ultimately best be examined through user studies. Indeed, the evaluation of Calibrate is another limitation of our work. While we primarily demonstrate Calibrate's effectiveness through case studies and expert interviews, a quantitative user study, like that performed by Sahann et al. [40], would be helpful to further demonstrate the effectiveness of Calibrate's features in aiding machine learning model calibration analysis. Furthermore, we note that our presented use cases represent starting points to generate hypotheses, and should be followed by deeper dives.

7 Conclusion

Machine learning model performance analysis is an important task that influences real-life model deployment. Much of current model performance analysis is centered around analyzing a model's predicted labels. However, many applications rely on predicting correct *probabilities* rather than labels. When a model's predicted probabilities align with reality, the model is said to be calibrated. Typically, calibration is explored through static visualizations called reliability diagrams. Since models can achieve high accuracy yet be poorly calibrated, it is important to analyze model calibration. In this work, we present Calibrate, an interactive tool to analyze model calibration. Additionally, we introduce Learned Reliability Diagrams, a simple approach to address the issues with standard binning procedures for traditional reliability diagrams. Calibrate allows users to perform a wide array of analyses, such as subgroup and instance-level analysis. We implement Calibrate for easy use within Jupyter Notebooks, so that calibration analyses can be easily created, reproduced and disseminated. To evaluate Calibrate, we present use cases which perform subgroup analysis and explore model calibration factors, along with expert interviews.

Acknowledgments

This collaboration has been supported by a grant from Capital One. Silva's research has also been supported by NASA; NSF awards CNS-1229185, CCF1533564, CNS-1544753, CNS-1730396, CNS-1828576, CNS-1626098; and DARPA PTG and D3M. Nonato's research has been supported by São Paulo Research Foundation (FAPESP)-Brazil (grant 2013/ 07375-0) and CNPq-Brazil (grant 307184/2021-8). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, NSF, NASA, FAPESP, CNPq, or Capital One.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[2] B. Alsallakh, A. Hanbury, H. Hauser, S. Miksch, and A. Rauber. Visual methods for analyzing probabilistic classification data. *IEEE Trans. Vis. Comput. Graph.*, 20(12):1703–1712, 2014.

[3] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Y. Simard, and J. Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In B. Begole, J. Kim, K. Inkpen, and W. Woo, editors, *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*, pages 337–346. ACM, 2015.

[4] J. Angwin, J. Larson, S. Mattu, and L. Kirchner. Machine bias. In *Ethics of Data and Analytics*, pages 254–264. Auerbach Publications, 2016.

[5] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2301–2309, 2011.

[6] G. W. Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[7] J. Bröcker and L. A. Smith. Increasing the reliability of reliability diagrams. *Weather and forecasting*, 22(3):651–661, 2007.

[8] Á. A. Cabrera, W. Epperson, F. Hohman, M. Kahng, J. Morgenstern, and D. H. Chau. FAIRVIS: visual analytics for discovering intersectional bias in machine learning. In R. Chang, D. A. Keim, and R. Maciejewski, editors, *14th IEEE Conference on Visual Analytics Science and Technology, IEEE VAST 2019, Vancouver, BC, Canada, October 20-25, 2019*, pages 46–56. IEEE, 2019.

[9] R. Caruana and A. Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 69–78. ACM, 2004.

[10] L. Cosmides and J. Tooby. Are humans good intuitive statisticians after all? rethinking some conclusions from the literature on judgment under uncertainty. *Cognition*, 58(1):1–73, 1996.

[11] T. de Menezes e Silva Filho, H. Song, M. Perelló-Nieto, R. Santos-Rodríguez, M. Kull, and P. A. Flach. Classifier calibration: How to assess and improve predicted class probabilities: a survey. *CoRR*, abs/2112.10327, 2021.

[12] M. H. DeGroot and S. E. Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(1-2):12–22, 1983.

[13] T. Dimitriadis, T. Gneiting, and A. I. Jordan. Stable reliability diagrams for probabilistic classifiers. *Proceedings of the National Academy of Sciences*, 118(8), 2021.

[14] D. Dingen, M. van 't Veer, P. Houthuizen, E. H. J. Mestrom, H. H. M. Korsten, A. R. A. Bouwman, and J. J. van Wijk. Regressionexplorer: Interactive exploration of logistic regression models with subgroup analysis. *IEEE Trans. Vis. Comput. Graph.*, 25(1):246–255, 2019.

[15] D. Dua and C. Graff. UCI machine learning repository, 2017.

[16] M. Gleicher, A. Barve, X. Yu, and F. Heimerl. Boxer: Interactive comparison of classifier results. *Comput. Graph. Forum*, 39(3):181–193, 2020.

[17] J. Görtler, F. Hohman, D. Moritz, K. Wongsuphasawat, D. Ren, R. Nair, M. Kirchner, and K. Patel. Neo: Generalizing confusion matrix visualization to hierarchical and multi-output labels. *arXiv preprint arXiv:2110.12536*, 2021.

[18] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017.

[19] R. Hagedorn, F. J. Doblas-Reyes, and T. N. Palmer. The rationale behind the success of multi-model ensembles in seasonal forecasting—i. basic concept. *Tellus A: Dynamic Meteorology and Oceanography*, 57(3):219–233, 2005.

[20] C. Hallenbeck. Forecasting precipitation in percentages of probability. *Monthly Weather Review*, 48(11):645–647, 1920.

[21] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[22] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009.

[23] A. P. Hinterreiter, P. Ruch, H. Stitz, M. Ennemoser, J. Bernard, H. Strobelt, and M. Streit. Confusionflow: A model-agnostic visualization for temporal analysis of classifier confusion. *IEEE Trans. Vis. Comput. Graph.*, 28(2):1222–1236, 2022.

[24] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Trans. Vis. Comput. Graph.*, 25(8):2674–2693, 2019.

[25] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. P. Chau. Activis: Visual exploration of industry-scale deep neural network models. *IEEE Trans. Vis. Comput. Graph.*, 24(1):88–97, 2018.

[26] J. M. Kleinberg, S. Mullainathan, and M. Raghavan. Inherent trade-offs in the fair determination of risk scores. In C. H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 43:1–43:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[27] M. Kull and P. A. Flach. Novel decompositions of proper scoring rules for classification: Score adjustment as precursor to calibration. In A. Appice, P. P. Rodrigues, V. S. Costa, C. Soares, J. Gama, and A. Jorge, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I*, volume 9284 of *Lecture Notes in Computer Science*, pages 68–85. Springer, 2015.

[28] S. Lichtenstein, B. Fischhoff, and L. D. Phillips. Calibration of probabilities: The state of the art. *Decision making and change in human affairs*, pages 275–324, 1977.

[29] Y. Lou, R. Caruana, and J. Gehrke. Intelligible models for classification and regression. In Q. Yang, D. Agarwal, and J. Pei, editors, *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 150–158. ACM, 2012.

[30] Y. Lou, R. Caruana, J. Gehrke, and G. Hooker. Accurate intelligible models with pairwise interactions. In I. S. Dhillon, Y. Koren, R. Ghani, T. E. Senator, P. Bradley, R. Parekh, J. He, R. L. Grossman, and R. Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 623–631. ACM, 2013.

[31] A. H. Murphy and R. L. Winkler. Reliability of subjective probability forecasts of precipitation and temperature. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 26(1):41–47, 1977.

[32] M. P. Naeini, G. F. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In B. Bonet and S. Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2901–2907. AAAI Press, 2015.

[33] A. Niculescu-Mizil and R. Caruana. Obtaining calibrated probabilities from boosting. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, page 413. AUAI Press, 2005.

[34] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In L. D. Raedt and S. Wrobel, editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pages 625–632. ACM, 2005.

[35] J. Nixon, M. W. Dusenberry, L. Zhang, G. Jerfel, and D. Tran. Measuring calibration in deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 38–41. Computer Vision Foundation / IEEE, 2019.

[36] H. Nori, S. Jenkins, P. Koch, and R. Caruana. Interpretml: A unified framework for machine learning interpretability. *CoRR*, abs/1909.09223, 2019.

[37] H. Nori, S. Jenkins, P. Koch, and R. Caruana. Interpretml: A unified

framework for machine learning interpretability. *CoRR*, abs/1909.09223, 2019.

[38] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[39] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Trans. Vis. Comput. Graph.*, 23(1):61–70, 2017.

[40] R. Sahann, T. Möller, and J. Schmidt. Histogram binning revisited with a focus on human perception. In *2021 IEEE Visualization Conference, IEEE VIS 2021 - Short Papers, New Orleans, LA, USA, October 24-29, 2021*, pages 66–70. IEEE, 2021.

[41] F. Sanders. On subjective probability forecasting. *Journal of Applied Meteorology and Climatology*, 2(2):191–201, 1963.

[42] J. T. Townsend. Theoretical analysis of an alphabetic confusion matrix. *Perception & Psychophysics*, 9(1):40–50, 1971.

[43] J. Vaicenavicius, D. Widmann, C. R. Andersson, F. Lindsten, J. Roll, and T. B. Schön. Evaluating model calibration in classification. In K. Chaudhuri and M. Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 3459–3467. PMLR, 2019.

[44] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[45] D. Widmann, F. Lindsten, and D. Zachariah. Calibration tests in multi-class classification: A unifying framework. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12236–12246, 2019.

[46] R. L. Winkler. Scoring rules and the evaluation of probability assessors. *Journal of the American Statistical Association*, 64(327):1073–1078, 1969.

[47] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In C. E. Brodley and A. P. Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 609–616. Morgan Kaufmann, 2001.

[48] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 694–699. ACM, 2002.