Amazon AI Fairness and Explainability Whitepaper

This whitepaper is a technical reference for AWS customers who use features from Amazon SageMaker Clarify to help identify potential bias in machine learning models and to help explain how these models make predictions. Amazon SageMaker Clarify detects potential bias during data preparation, after model training, and in your deployed model by examining attributes you specify. For instance, you can check for bias related to age in your initial dataset or in your trained model and receive a detailed report that quantifies different types of possible bias. SageMaker Clarify also includes feature importance graphs that help you explain model predictions and produces reports which can be used to support internal presentations or to identify issues with your model that you can take steps to correct. While this whitepaper is primarily designed for data scientists and ML engineers, we also expect product managers, compliance teams, and other stakeholders to benefit from the overview of fairness and explainability provided here and from the discussions of limitations and best practices.

Introduction

Machine learning models and data-driven systems are being increasingly used to help make decisions across domains such as financial services, healthcare, education, and human resources. Machine learning applications provide benefits such as improved accuracy, increased productivity, and cost savings. This trend is the result of a confluence of factors, most notably ubiquitous connectivity, the ability to collect, aggregate, and process large amounts of fine-grained data using cloud computing, and improved access to increasingly sophisticated machine learning models that can analyze this data.

Regulatory: In many situations, it is important to understand why the ML model made a specific prediction and also whether the prediction it made was impacted by any bias, either during training or at inference. Recently, policymakers, regulators, and advocates have raised awareness about the policy challenges posed by ML and data-driven systems. In particular, they have expressed concerns about the potentially discriminatory impact of such systems, for example, due to inadvertent encoding of bias into automated decisions.

Business: The adoption of AI systems in regulated domains requires transparency. This is achieved through reliable explanations of the trained models and their predictions. Model explainability may be particularly important to certain industries with reliability, safety, and compliance requirements, such as financial services, human resources, healthcare, and automated transportation. To take a common financial example, lending applications that incorporate the use of ML models may need to provide explanations about how those models made certain predictions to internal teams of loan officers, customer service representatives, and compliance officers, as well as to end users/customers.

Data Science: Data scientists and ML engineers need tools to generate the insights required to debug and improve ML models through better feature engineering, to determine whether a model is making inferences based on noisy or irrelevant features, and to understand the limitations of their models and failure modes their models may encounter.

Developing responsible AI solutions is a process involving inputs and discussions with key stakeholders (including product, policy, legal, engineering, and AI/ML teams, as well as end users and communities) during all stages of the ML lifecycle. In this paper we focus primarily on the technical tools for bias and explainability in ML lifecycle. We do also provide a brief section on limitations and best practices for providing fairness and explainability with AI.

Fairness in Machine Learning: A Brief Overview

Algorithmic bias, discrimination, fairness, and related topics have been studied across disciplines such as law, policy, and computer science. A computer system may be considered biased if it discriminates against certain features or groups of features. The machine learning models powering these applications learn from data and this data may reflect disparities or other inherent biases. For example, the training data may not have sufficient representation of various feature groups or may contain biased labels. The machine learning models trained on datasets that exhibit these biases could end up learning them and then reproduce or even exacerbate those biases in their predictions. The field of machine learning provides an opportunity to address biases by detecting them and measuring them at each stage of the ML lifecycle.

Explainable AI: A Brief Overview

Explainability refers to the concept that the end user can understand why a prediction is made by an AI system. As a result of the increasing use of AI systems in our day-to-day experiences and recent regulatory provisions, which focus on the transparency of data-driven automated decision-making, there is a growing demand for model transparency and interpretability. Explainable AI methods can be classified based on different criteria. Interpretable models are ML models with a simple structure (such as sparse linear models or shallow decision trees) that can "explain themselves," i.e., are easy for humans to interpret. Post-hoc explanation methods attempt to analyze and explain a relatively more complex ML model after it has been trained. The explanation methods can either be model-specific (e.g., designed for neural networks or other differentiable models) or model-agnostic (i.e., applicable for any ML model, after training). Global explanation methods attempt to explain the model behavior as a whole while local explanation methods focus on explaining an individual prediction.

Bias Measurement using Amazon SageMaker

Bias Measurement Prerequisites

Measuring bias in ML models is a first step to mitigating bias. Each measure of bias corresponds to a different notion of fairness. Even considering simple notions of fairness leads to many different measures applicable in various contexts. Consider fairness with respect to gender or age, for example, and, for simplicity, that there are two relevant classes, for example, split by age at a cutoff age, say 50 years. In the case of an ML model for lending, we may want small business loans to be issued to equal numbers of both classes. Or, when processing job applicants, we may want to see equal numbers of members of each class hired. However, this approach may assume that equal numbers of both classes apply to these jobs, so we may want to condition on the number that apply. Further, we may want to consider not whether equal numbers apply, but whether we have equal numbers of qualified applicants. Or, we may consider fairness to be an equal acceptance rate of qualified applicants across classes, or, an equal rejection rate of applicants, or both. We may use datasets with different proportions of data on the attributes of interest. This imbalance can conflate the bias measure we pick. Our models may be more accurate in classifying one class than in the other. Thus, we need to choose bias measures that are appropriate for the application and the situation.

We next describe several bias metrics corresponding to different potential requirements. Bias may be measured before training and after training, as well as at inference for a deployed model. Our exposition presents the specific case of binary classification models for simplicity, but the metrics can be generalized to multi-category classification models and models where the labels are continuous. We specify the following simple notation to make presentation of the metrics easier.

Notation: The goal of binary classification is to predict an unknown binary outcome, y given an observed set of features. For example, in a loan application setting, the outcome could correspond to whether an applicant will pay back the loan or not and the observed set of features could include income, net worth, credit history, and loan amount. Suppose we have labeled training data where each example consists of the values of the observed features and the corresponding label, y (= 0 or 1). The classifier maps the observed features to a predicted label, $\hat{y}$ (= 0 or 1) that we hope agrees with the true label, y. Suppose that there is a restricted feature (which we also call the demographic group, attribute of interest, or facet) associated with each example (e.g., age), and based on the value of this feature, we designate the example as part of either a first group or class (marked/subscripted by a) or a second group (marked/subscripted by d). For example, we may split a sample by age into a first class that comprises people older than 50 years and a second class of people who have ages up to age 50. Without loss of generality, our bias metrics will measure bias in favor of the first group relative to the second.

Bias may be measured from a comparison of the true labels (*y* = 0 or 1) of the data sample with the predicted labels (ŷ = 0 or 1). Assume that ŷ = 1 corresponds to the accepted case and ŷ = 0 to the rejected case (for example, in the loan application use case). In the training dataset we may count the number of labels of values 0 and 1, grouped by the restricted feature. Denote the number of observed labels of value 0, 1 as $n^{(0)}$, $n^{(1)}$, respectively, and the number of labels of each class as n_a, n_d. These comprise labels of the first and second group, i.e., $n_a^{(0)}$, $n_a^{(1)}$ and $n_d^{(0)}$, $n_d^{(1)}$, respectively. We also have that $n_a^{(0)} + n_a^{(1)} = n_a$ and $n_d^{(0)} + n_d^{(1)} = n_d$. Corresponding to this notation for observed labels, we have a parallel notation for predicted labels ŷ, with counts $\hat{n}^{(0)}$, $\hat{n}^{(1)}$, etc.

Pre-Processing Bias Metrics for Training Data

Training on biased data is liable to exacerbate any pre-existing bias in that data. To identify bias in the data before expending time/money on training, we want to develop metrics that can be computed on the raw dataset before training. One option is to conduct a survey to determine the "golden truth" and compare it to the dataset to make sure the data is not too contaminated with bias to be useful. The golden truth is the joint statistical distribution of model inputs we would ideally have to train any model fairly. These distributions are not always available, so pre-training bias metrics provide measures for comparison to a golden truth, were it to be available. If not, modelers will at least be able to assess whether the pre-training bias metrics are in violation of a judgment threshold level. The following pre-training metrics are of course model-independent.

[1] Class imbalance (CI): Bias is often generated from an under-representation of one of the classes in the dataset, especially if the desired "golden truth" is equal representation across groups. As an example, if a machine learning model is trained primarily on data from middle-aged individuals, it may be less accurate when making predictions involving younger and older people. If n_a is the number of members of the first class and n_d the number for the second class, we can represent the class imbalance measure as

$$\frac{n_a - n_d}{n_a + n_d}$$

We note that this metric must lie in the range (-1,+1).

[2] Difference in positive proportions in labels (DPL): This metric compares the proportion of positive labels for the second class versus that of the first class in the dataset. For example, in the case of college admissions, positive labels could be associated with applicants granted admission and negative labels with applicants not granted admission. If the applicants in the dataset could be divided into two groups, group 1 (class *a)* and group 2 (class d), was each group given college admission in proportion to the number of applicants from that group? Let the proportion of group 1 who got admission be equal to $q_a = n_a^{(1)}/n_a$, where $n_a^{(1)}$ is the number of people in group 1 who got admission (as opposed to those who did not get admission, $n_a^{(0)}$). Likewise, the proportion of people in group 2 who were granted admission is $q_d = n_d^{(1)}/n_d$. Our bias metric is DPL = $q_a - q_d$. If DPL is close to 0, then we say that "*demographic parity*" has been achieved ex-post (i.e., already in the historical record).

$$q_a - q_d = \frac{n_a^{(1)}}{n_a} - \frac{n_d^{(1)}}{n_d}$$

Whereas Class Imbalance examines lopsidedness in the numbers of class a versus d, demographic parity is concerned with the imbalance in label distributions between classes. This label imbalance metric must lie in the range (-1,+1).

[3] KL Divergence (KL): We compare the probability distribution of labels of the first class (P_a) with that of the second class (P_d), using KL divergence. KL measures how much the label distribution of each class differs. This measure generalizes easily to multiple label types, not just binary ones. For example, take college admissions, where an applicant may be assigned by a model to three categories: x = {rejected, wait-listed, or accepted}. We compute how different the distribution P_a is from P_d. KL is essentially a measure of entropy. For all three categories, denoted by x, we compute the ratio $\log[P_a(x)/P_d(x)]$, which is a measure of distance between the probability distributions for a label x. We then take the probability weighted sum of this measure, weighted by the probability of $P_a(x)$, which gives the KL measure of divergence of class a from class d. The measure is also a label imbalance metric and is denoted as

$$KL(P_a||P_d) = \sum_x P_a(x) \log\left(\frac{P_a(x)}{P_d(x)}\right)$$

This metric is non-negative, KL >= 0.
$P_d(x) > 0$ for all x, else the metric is not defined.
We say that it gives the distance of P_a from P_d.
This measure is not symmetric (reversing the distributions gives a different result), but is still a meaningful measure of difference in label distributions.

[4] Jensen-Shannon divergence (JS): Denoting the average of the label distributions of the two classes as P, we can compute the JS divergence as the average of the KL divergence of the probability distribution of the first class vs. P and the KL divergence of the probability distribution of the second class vs. P. This is an extension of the KL divergence measure for label imbalance. In contrast to the KL divergence, this metric is symmetric and bounded above. The measure is computed as

$$JS(P_a, P_d, P) = \frac{1}{2}[KL(P_a, P) + KL(P_d, P)]$$

This metric is non-negative, and bounded above by ln(2), that is, 0 <= JS <= ln(2), assuming that natural logarithm is used in the KL divergence computation.
It provides a symmetric difference between label distributions P_a and P_d.

[5] L_p-Norm (LP): Another measure of distance in label distributions is the normed direct distance between the distributions. For every label category, e.g., x = {rejected, wait-listed, accepted} in college admissions, we take the difference and take the p-polynomial mean, as follows

$$L_p(P_a, P_d) = \left[\sum_x |P_a(x) - P_d(x)|^p\right]^{1/p}$$

This metric is non-negative, LP >= 0.
In this example, there will be three items to be summed up.

[6] Total variation distance (TVD): this is half the L_1-norm of the difference between the probability distribution of labels of the first class and the probability distribution of labels of the second class.

$$TVD = \frac{1}{2}L_1(P_a, P_d) \geq 0$$

This metric is non-negative, TVD >= 0.
This is a special case of the LP metric where p=1.

[7] Kolmogorov-Smirnov (KS), two-sample approximated version: This metric evaluates the KS statistical test between the probability distribution of labels of the first class and the probability distribution of labels of the second class. This metric indicates whether there is a big divergence in one of the labels across classes. It complements the other measures by zoning in on the most imbalanced label.

$$KS = \max(|P_a(x) - P_d(x)|)$$

This statistic is in the the range (0,1).

[8] Conditional Demographic Disparity in Labels (CDDL): This metric examines disparity of outcomes (labels) between two classes, 1 and 2, but it also examines this disparity in subgroups, by stratifying the data using a “group” variable. The metric examines whether the second class has a bigger proportion of the rejected outcomes than the proportion of accepted outcomes for the same class. For example, in the case of college admissions, if class 2 applicants comprised 60% of the rejected applicants and comprised only 50% of the accepted applicants, then we say that there is demographic disparity (DD) because the rate at which class 2 applicants are rejected is greater than the rate at which they are accepted. Formally, we define demographic disparity (DD) as the difference between the proportion of rejected outcomes and the proportion of accepted outcomes for a given class. For simplicity, assume that all the applicants can be grouped into two groups. Suppose the first group comprised 40% of the rejected applicants and 40% of the accepted applicants, and hence, the second group comprised 60% of the rejected applicants and 60% of the accepted applicants. In this case, there is no demographic disparity. We note here that demographic parity is different from demographic disparity, though the nomenclature is confusingly similar. Conditional Demographic Disparity (CDDL) is defined as the weighted average of demographic disparity (DD) over different classes, with each class weighted by its size.

$$CDD = \frac{1}{n}\sum_i n_i \cdot DD_i, \quad \sum_i n_i = n$$

Both DD and CDDL lie in the range (-1,+1).

Suppose that there were 120 applicants from the first group and 80 applicants from the second group in the above example, of whom 48 from the first group and 32 from the second group are accepted. Total acceptances are 80, of whom 60% belong to the first group. Total rejections are 120 of whom also 60% belong to the first group. 40% of the acceptances and rejections belong to the second group. Demographic disparity = 0.

At UC Berkeley, men were admitted at a higher rate than women, but when examined at the school level, women in fact were admitted at a higher rate in almost each and every school. This observation can be explained by the well-known Simpson's paradox, and in the case of Berkeley admissions it arose because women applied to schools that were more competitive so fewer women were admitted overall compared to men, even though school by school they in fact were admitted at a higher rate.

Comments:

1. Of the eight pre-training metrics discussed here, the first one examines only class imbalance and does not examine labels. The remaining seven metrics all have to do with class imbalance. There is redundancy here but you are given a lot of choice. Our recommended approach is to use CI, DPL, KL, and CDDL at the first pass. If the other remaining four measures (JS, LP, TVD, KS) are desired, you may include those as well.
2. We note that class imbalance measures are usually applied to binary classes. To generalize to the case where there are classes of more than two levels, you can assign each class, one at a time, to be the protected class and work out the respective pre-training metrics for each class.
3. For label imbalance we can have multicategory labels, i.e., more than two categories. In this case, two approaches that are possible: (a) Collapse the multiple categories of labels into binary and then compute the various label imbalance measures. To do this you specify which labels will be grouped into one binary category (1), the others are than all grouped into the other category (0), and then the label imbalance is computed. Or (b) we compute label imbalances across all the multiple categories. Note that (a) is a special case of (b). For now, we require you to group labels into binary categories, scheme (3a) above. We believe that this helps you think about the labels and examine the imbalance across binary sets to discover which ones are the most significant. Therefore, we require you to specify the values of the labels that go into binary categories (which labels are 1 and which are 0). This helps you find the most important imbalances in the labels and becomes necessary when the labels are ordinal, i.e., they have an order. For example, take college admissions, considered as a use case above. There may be three outcome labels: {rejected, wait-listed, accepted}. These are clearly ordinal, and the you may want to group them into binary: {rejected}, {wait-listed, accepted}. Here, {wait-listed, accepted}=1, {rejected}=0. However, another grouping might be: {rejected, wait-listed}=0, {accepted}=1. By requiring you to choose, we are eliciting the more important label imbalance, but also requiring a level of data examination and feature analysis that is important to carry out before using the data to train machine leaning model.

Post-Processing Bias Metrics for the Trained Model

In the preceding section, we presented eight pre-training metrics. All these metrics needed information available during pre-training such as the distribution of class membership (a,d) and/or the distribution of labels (y). After training the ML model, we gain additional information from the model itself, in particular the predicted probabilities from the model (p') and the predicted labels (ŷ). These allow an additional set of bias metrics to be calculated. Any bias that arises post-training may emanate from biases in the data and/or biases in the model's classification and prediction.

[1] Difference in positive proportions in predicted labels (DPPL): This is defined as the difference in the proportion of positive predictions (ŷ = 1) for the first class versus the proportion of positive predictions (ŷ = 1) for the second class. For example, if the model grants loans to 50% of class 2 and to 60% of class 1, then it may be biased against class 2. We would have to decide whether a 10% difference is material. A similar bias would be detected in hiring situations, college admissions, etc. The measure would be

$$\hat{q}_a = \frac{\hat{n}_a^{(1)}}{n_a} \quad \hat{q}_d = \frac{\hat{n}_d^{(1)}}{n_d} \quad \hat{q}_a - \hat{q}_d$$

The metric lies in the range (-1,+1).
The carat above the variable denotes model "predicted" or estimated label numbers.

[2] Disparate (Adverse) Impact (DI)**:** The same metric may be assessed in the form of a ratio.

$$DI = \frac{\hat{q}_d}{\hat{q}_a}$$

This metric is known as a measure of "disparate impact" (DI).
DI >= 0
This measure may sometimes be considered fair if it resides in the (4/5, 5/4) range. However, this is purely indicative and it is up to you to decide appropriate thresholds.

[3] Difference in conditional outcome (DCO): This metric compares the observed labels in the data to the predicted labels from the model and assesses whether the actual vs. observed label balance is the same across classes. Consider for example a dataset of 100 people from class 1 and 50 people from class 2 who applied for loans where the model recommended that 60 class 1 and 30 class 2 people be given loans. So the predicted proportions are DPPL-fair (the model grants loans to 60% of candidates from each class), but the observed labels show that 70 people from class 1 and 20 from class 2 were granted loans. In other words, the model granted loans to 17% more people from class 1 than the observed labels in the training data suggested (70/60 = 1.17) and granted loans to 33% fewer people from class 2 than the observed labels suggested (20/30 = 0.67). So the increase in group-level acceptances from observed to predicted labels between people from class 1 and class 2 is imbalanced. The DCO metric captures such imbalances.

DCO may be broken down into two types, **Difference in Conditional Acceptance** (DCA) and **Difference in Conditional Rejection** (DCR). First, we present DCA:

$$c_a = \frac{n_a^{(1)}}{\hat{n}_a^{(1)}} \quad c_d = \frac{n_d^{(1)}}{\hat{n}_d^{(1)}} \quad DCA = c_a - c_d$$

DCA is unbounded. It is possible that the denominator in these expressions is zero, in which case the allocations to each class are too small and a warning should be issued.

The same idea is also applied to difference in rejection rates (or DCR):

$$r_a = \frac{n_a^{(0)}}{\hat{n}_a^{(0)}} \quad r_d = \frac{n_d^{(0)}}{\hat{n}_d^{(0)}} \quad DCR = r_d - r_a$$

DCR is unbounded. It is possible that the denominator in these expressions is zero, in which case the allocations to each class are too small and a warning should be issued.

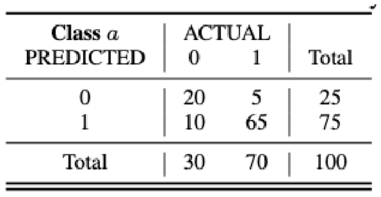
When both DCA and DCR are very close to 0, we can conclude that the proportion of qualified (as suggested by observed labels) applicants accepted by the model and the proportion of unqualified applicants rejected are nearly equal across both classes.

[4] Recall Difference (RD): We may be interested in knowing whether there is a difference in recall of the model across the attributes of interest. Recall is how often the model correctly captures the cases that should receive a positive outcome. For example, of all the people who should be given loans, how many are detected correctly by the model? Recall is perfect for a class if all y = 1 cases are correctly called as ŷ = 1 for that class. If recall is high for lending to the first group but low for lending to the second group, then the difference is a measure of bias against the second group (the second group can be defined in many ways, such as by gender or age). Here, higher recall for the first class suggests that the ML model predicts fewer false negatives for the first class than for the second class, i.e., it finds more of the actual true positives for the first class than the second class, which is a form of bias. We define RD as the difference in recall for the first vs. second group.

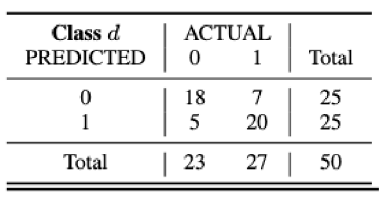
$$RD = \frac{TP_a}{TP_a + FN_a} - \frac{TP_d}{TP_d + FN_d}$$

We note that recall is greater when the model minimizes false negatives, i.e., Type II error.
RD is in the range (-1,+1).

As an example, we can look at the following pair of confusion matrices:



Class a	ACTUAL		
PREDICTED	0	1	Total
0	20	5	25
1	10	65	75
Total	30	70	100



Class d	ACTUAL		
PREDICTED	0	1	Total
0	18	7	25
1	5	20	25
Total	23	27	50

Class-wise confusion matrices

We obtain the class-wise confusion matrices for each class as shown above. We then compute recall for each class separately and then take the difference.
Recall for class ‘a’ = 65/70 = 0.9285
Recall for class ‘d’ = 20/27 = 0.7407
RD = 0.1878

[5] Difference in label rates (DLR): labels may be positive or negative outcomes. The difference in the rates of these positive and negative predicted outcomes across the first and second classes is a measure of bias. We have two types of bias metrics here. (a) **Difference in acceptance rates** (DAR): This metric measures whether qualified applicants from the first and second classes are accepted at the same rates. It is the difference in the ratio of true positives to predicted positives for each class. (b) **Difference in rejection rates** (DRR): This metric measures whether unqualified applicants from the first and second class are rejected at the same rates. It is the difference in the ratio of true negatives to predicted negatives for each class.

$$DAR = \frac{TP_a}{\hat{n}_a^{(1)}} - \frac{TP_d}{\hat{n}_d^{(1)}} \qquad DRR = \frac{TN_d}{\hat{n}_d^{(0)}} - \frac{TN_a}{\hat{n}_a^{(0)}}$$

DAR, DRR lie in the range (-1,+1).
DAR is also the same as precision difference between the first and second classes.

We take the example of college admissions. The total number of applicants in class 1 is 180, and in class 2 is 150. If the model predicts that 100 class 2 people are qualified to be admitted of whom the college admits 40, and 70 of class 1 are qualified to be admitted of whom the college admits 35, then DAR = 35/70 - 40/100 = 0.10. If the model designates 50 people in class 2 as unqualified and rejects 40 of them and designates 100 in class 1 as unqualified and rejects 80 of them, then DRR = 40/50 - 80/100 = 0 (no bias). Both, DAR and DRR are related to the concepts of equality of opportunity and equalized odds. For example, we note that admission quotas can also cause differences in label rates, which would show up in this metric.

We also define **Precision difference** (PD): If the precision for the first class is greater than the precision for the second class, it implies that more of the predicted true positives are valid compared to false positives for the first class. This suggests that predictions favor the first class. We define PD as the difference in precision for the first vs. second classes. Precision difference is the same as DAR. More importantly, it is a measure of Type I error and the higher the precision, the lower the number of false positives. Precision difference is in the range (-1,+1).

[6] Accuracy Difference (AD): Classification by the ML model may be more accurate for one class than the other. This is a useful metric because it indicates that one class incurs a greater proportion of Type I and Type II errors. For example, if loan approvals are made with much higher accuracy for applicants in class 1 than for applicants in class 2, it means that a greater proportion of qualified applicants in class 2 are denied a loan and/or a greater proportion of unqualified applicants in class 2 get a loan. This leads to within group unfairness for class 2 even if the proportion of loans granted is nearly the same for both classes (DPPL close to 0).

$$AD = \frac{TP_a + TN_a}{TP_a + TN_a + FP_a + FN_a} - \frac{TP_d + TN_d}{TP_d + TN_d + FP_d + FN_d}$$

AD lies in the range (-1,+1).

[7] Treatment Equality (TE): This is defined as the difference in the ratio of false positives to false negatives for the first vs. second class. Even if the accuracy across classes is the same, is it the case that errors are more harmful to one class than another? TE measures whether errors are compensating in the same way across classes. Example: 100 class 1 applicants and 50 class 2 applicants apply for a loan. 8 people in class 2 were wrongly denied a loan and another 6 were wrongly approved. For class 2, 5 were wrongly denied and 2 were wrongly approved. The ratio of false positives to false negatives equals 0.75 for class 1 and 0.40 for class 2, and hence TE = 0.75 - 0.40 = 0.35, even though both classes have the same accuracy of 0.86.

$$\frac{FP_a}{FN_a} - \frac{FP_d}{FN_d}$$

Care needed when FN = 0.
The metric can be unbounded.

[8] Conditional Demographic Disparity in Predicted Labels (CDDPL): A comparison of difference in predicted rejection proportion and predicted acceptance proportion across classes. This metric is exactly the same as the pre-training metric except that it is computed off the predicted labels instead of the actual ones. This metric lies in the range (-1,+1).

[9] Counterfactual Fliptest (FT): The fliptest is an approach that looks at each member of the second class and assesses whether similar members of the first class have different model predictions. The members of the first class are chosen to be k-nearest neighbors of the observation from the second class. We assess how many nearest neighbors of the opposite class receive a different prediction, where the flipped prediction can go from positive to negative and vice versa. The formula for the counterfactual fliptest is the difference in the cardinality of two sets divided by the number of members of the second class:

$FT = (F^+ - F^-)/n_d$

where:

- F^+ is the number of second group members with an unfavorable outcome whose nearest neighbors in the first group received a favorable outcome.
- F^- is the number of second group members with a favorable outcome whose nearest neighbors in the first group received an unfavorable outcome.
- n_d is the size of the second group

The range of values for the counterfactual fliptest for binary and multi-category facet labels is [-1, +1]. The FT metric is not defined for continuous labels.

- Positive values occur when the number of unfavorable counterfactual fliptest predictions for the second group exceeds the favorable ones.
- Values near zero occur when the number of unfavorable and favorable counterfactual fliptest predictions balance out.
- Negative values occur when the number of unfavorable counterfactual fliptest predictions for the second group is less than the favorable ones.

Monitoring Bias Metrics for the Deployed Model

Measuring bias *only* during the train-and-deploy phase is not sufficient. It is quite possible that after the model has been deployed, the distribution of the data that the deployed model sees, that is, the *live data,* is different from that of the training dataset. This change, also known as concept drift, may cause the model to exhibit more bias than it did on the training data. The change in the live data distribution might be temporary (e.g., due to some short-lived behavior like the holiday season) or permanent. In either case, it might be important to detect these changes.

To detect these changes, we provide functionality to monitor the bias metrics of a deployed model continuously and raise automated alerts if the metrics exceed a threshold. To be precise, let us take the DPPL bias metric as an example. Let us say that you specify an allowed range of values $A = (a_{\min}, a_{\max})$, for instance (-0.1, 0.1), that DPPL should lie in during deployment – any deviation from this range should raise a “bias detected” alert. Our offering allows you to perform these checks at regular intervals.

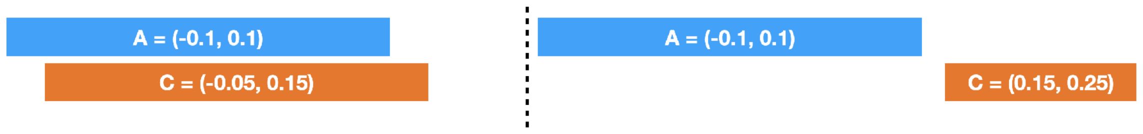
For instance, you can set the frequency of the checks to be every 2 days. This means that every 2 days, we will compute the DPPL on the data $\mathrm{D}_{\mathrm{win}}$ that the model processed during last 2 days time window. We would like an alert to be issued if the DPPL value b_{win} computed on $\mathrm{D}_{\mathrm{win}}$ falls outside of the allowed range A.

The above approach of checking if b_{win} is outside of A can be somewhat noise-prone: $\mathrm{D}_{\mathrm{win}}$ might consist of very few samples and might not be representative of the live data distribution. The small sample size means that the value of bias b_{win} computed over $\mathrm{D}_{\mathrm{win}}$ may not be a very robust estimate. In fact, very high (or low) values of b_{win} may be observed purely due to chance. To ensure that the conclusions drawn from the observed data $\mathrm{D}_{\mathrm{win}}$ are statistically significant, we make use of confidence intervals. Specifically, we use the Normal Bootstrap Interval method to construct an interval $C = (c_{\min}, c_{\max})$ such that we are confident that the true bias value computed over the full live data is contained in C with high probability. For more details on interpretation of confidence intervals and their computation, see *All of Statistics* by Wasserman and *An Introduction to the Bootstrap* by Efron and Tibshirani.

Now, if our confidence interval C overlaps with the allowed range A, we interpret it as “it is likely that the bias metric value of the live data distribution falls within our allowed range”. If C and A are disjoint, we can be confident that the bias metric does not lie in A and raise an alert.

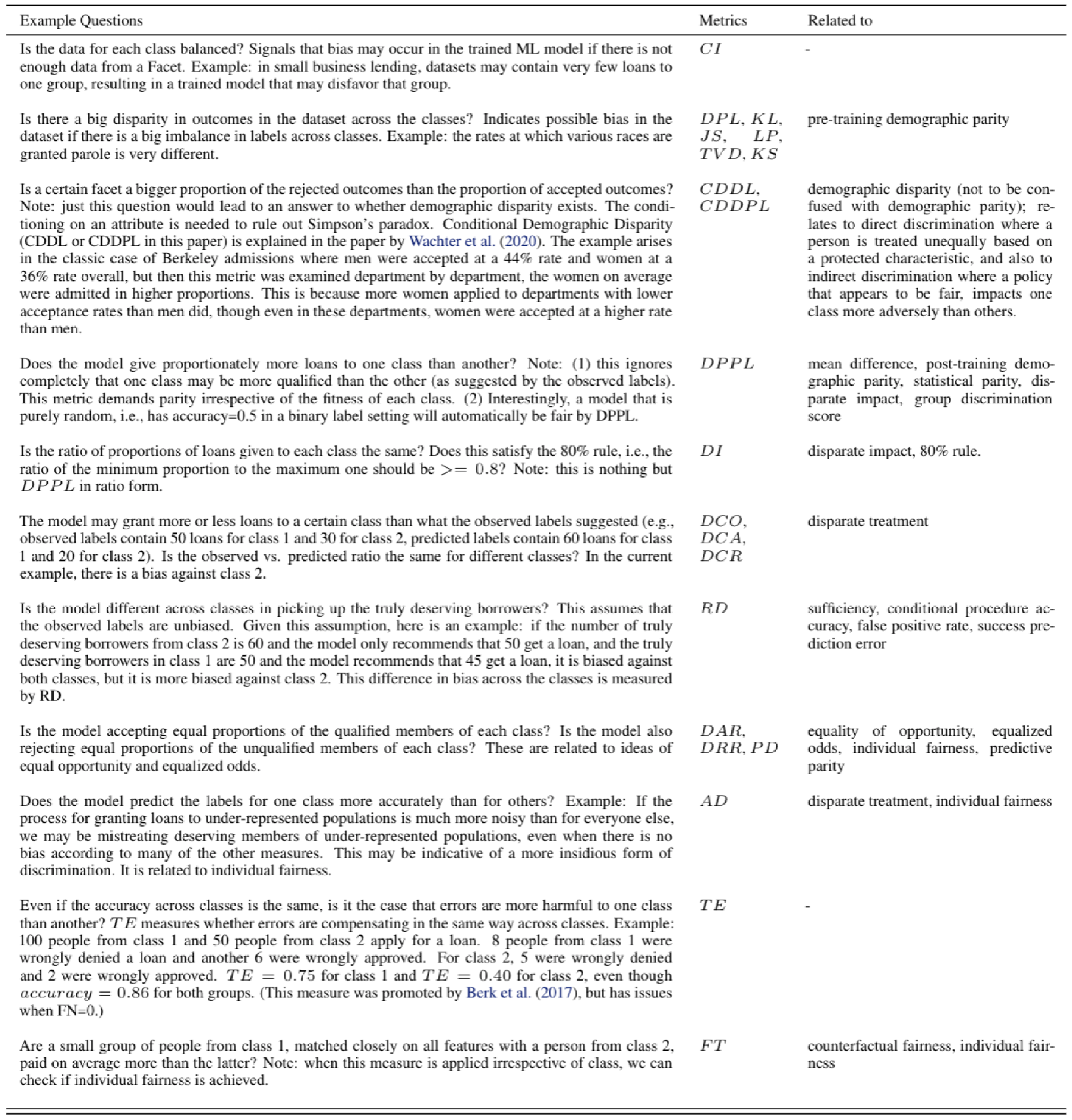
No bias alert
Confidence interval C overlaps with the allowed range A.

Issue bias alert
Confidence interval C and the allowed range A are disjoint.



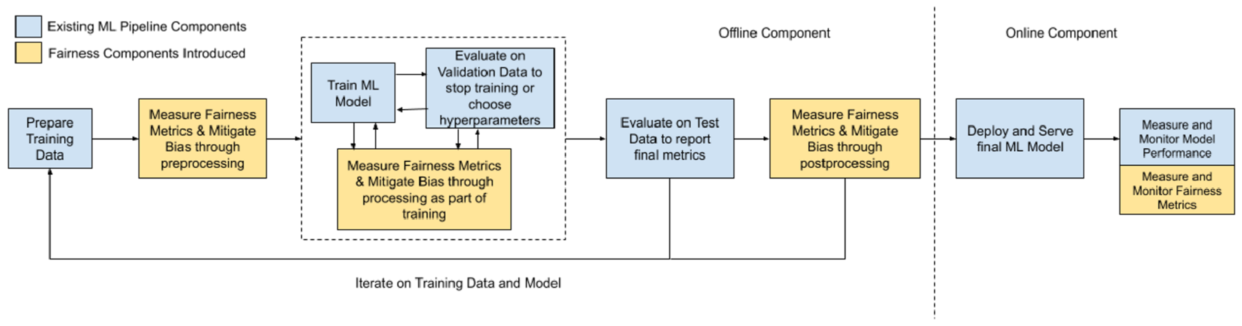
Bias Metrics Explained

As various bias metrics examine different nuances and ways in which bias may arise, and there is not a single bias metric applicable across all scenarios, it is not always easy to know which ones apply in a particular situation or domain. Hence, we present a checklist of questions that may be helpful to a customer to determine the most relevant bias metric(s) for their application context. In the table below, we state such example questions (in the setting of a financial ML model for concreteness), along with the bias metrics most relevant for each question. Please see our paper, Fairness Measures for Machine Learning in Finance for more information.



Example Questions	Metrics	Related to
Is the data for each class balanced? Signals that bias may occur in the trained ML model if there is not enough data from a Facet. Example: in small business lending, datasets may contain very few loans to one group, resulting in a trained model that may disfavor that group.	CI	-
Is there a big disparity in outcomes in the dataset across the classes? Indicates possible bias in the dataset if there is a big imbalance in labels across classes. Example: the rates at which various races are granted parole is very different.	DPL, KL, JS, LP, TVD, KS	pre-training demographic parity
Is a certain facet a bigger proportion of the rejected outcomes than the proportion of accepted outcomes? Note: just this question would lead to an answer to whether demographic disparity exists. The conditioning on an attribute is needed to rule out Simpson's paradox. Conditional Demographic Disparity (CDDL or CDDPL in this paper) is explained in the paper by Wachter et al. (2020). The example arises in the classic case of Berkeley admissions where men were accepted at a 44% rate and women at a 36% rate overall, but then this metric was examined department by department, the women on average were admitted in higher proportions. This is because more women applied to departments with lower acceptance rates than men did, though even in these departments, women were accepted at a higher rate than men.	$CDDL$, $CDDPL$	demographic disparity (not to be confused with demographic parity); relates to direct discrimination where a person is treated unequally based on a protected characteristic, and also to indirect discrimination where a policy that appears to be fair, impacts one class more adversely than others.
Does the model give proportionately more loans to one class than another? Note: (1) this ignores completely that one class may be more qualified than the other (as suggested by the observed labels). This metric demands parity irrespective of the fitness of each class. (2) Interestingly, a model that is purely random, i.e., has accuracy=0.5 in a binary label setting will automatically be fair by DPPL.	$DPPL$	mean difference, post-training demographic parity, statistical parity, disparate impact, group discrimination score
Is the ratio of proportions of loans given to each class the same? Does this satisfy the 80% rule, i.e., the ratio of the minimum proportion to the maximum one should be $>= 0.8$? Note: this is nothing but $DPPL$ in ratio form.	DI	disparate impact, 80% rule.
The model may grant more or less loans to a certain class than what the observed labels suggested (e.g., observed labels contain 50 loans for class 1 and 30 for class 2, predicted labels contain 60 loans for class 1 and 20 for class 2). Is the observed vs. predicted ratio the same for different classes? In the current example, there is a bias against class 2.	DCO, DCA, DCR	disparate treatment
Is the model different across classes in picking up the truly deserving borrowers? This assumes that the observed labels are unbiased. Given this assumption, here is an example: if the number of truly deserving borrowers from class 2 is 60 and the model only recommends that 50 get a loan, and the truly deserving borrowers in class 1 are 50 and the model recommends that 45 get a loan, it is biased against both classes, but it is more biased against class 2. This difference in bias across the classes is measured by RD.	RD	sufficiency, conditional procedure accuracy, false positive rate, success prediction error
Is the model accepting equal proportions of the qualified members of each class? Is the model also rejecting equal proportions of the unqualified members of each class? These are related to ideas of equal opportunity and equalized odds.	DAR, DRR, PD	equality of opportunity, equalized odds, individual fairness, predictive parity
Does the model predict the labels for one class more accurately than for others? Example: If the process for granting loans to under-represented populations is much more noisy than for everyone else, we may be mistreating deserving members of under-represented populations, even when there is no bias according to many of the other measures. This may be indicative of a more insidious form of discrimination. It is related to individual fairness.	AD	disparate treatment, individual fairness
Even if the accuracy across classes is the same, is it the case that errors are more harmful to one class than another? TE measures whether errors are compensating in the same way across classes. Example: 100 people from class 1 and 50 people from class 2 apply for a loan. 8 people from class 1 were wrongly denied a loan and another 6 were wrongly approved. For class 2, 5 were wrongly denied and 2 were wrongly approved. $TE = 0.75$ for class 1 and $TE = 0.40$ for class 2, even though $accuracy = 0.86$ for both groups. (This measure was promoted by Berk et al. (2017), but has issues when FN=0.)	TE	-
Are a small group of people from class 1, matched closely on all features with a person from class 2, paid on average more than the latter? Note: when this measure is applied irrespective of class, we can check if individual fairness is achieved.	FT	counterfactual fairness, individual fairness

Broader Perspective


Existing ML Pipeline Components
Fairness Components Introduced
Offline Component
Online Component
Prepare Training Data
Measure Fairness Metrics & Mitigate Bias through preprocessing
Train ML Model
Evaluate on Validation Data to stop training or choose hyperparameters
Measure Fairness Metrics & Mitigate Bias through processing as part of training
Evaluate on Test Data to report final metrics
Measure Fairness Metrics & Mitigate Bias through postprocessing
Deploy and Serve final ML Model
Measure and Monitor Model Performance
Measure and Monitor Fairness Metrics
Iterate on Training Data and Model

Although our current offering focuses on bias measurement for training data and trained model, we provide a broader perspective in terms of bias measurement and mitigation during all stages of the ML lifecycle (schematic diagram above).

Training Data Collection: Bias measurement during this stage consists of measuring metrics for representativeness and label distribution across different subgroups. By representativeness, we refer to measures for determining whether the training data is representative of different subgroups (e.g., Class Imbalance (CI) discussed above). By label distribution, we refer to measures for comparing the label distributions across different subgroups (with each other), towards the goal of uncovering potential labeling biases within groups (e.g., DPL, KL, JS, TVD discussed above).

Bias mitigation during this stage (pre-processing) includes the efforts prior to model training such as representative training data collection and modifying the features or the labels in the training data. This involves various techniques such as sampling/re-weighting, adjusting labels, and learning unbiased representations to remove (or reduce) the bias in the training data.

Model Training & Tuning: Bias measurement post model training involves comparing predicted score distributions across different subgroups (similar to measurements on training data) and directly comparing performance metrics (such as accuracy, precision, and recall) across different subgroups. In non-trivial cases of unequal prevalence rates, differences in model performance across subgroups can be expected due to the impossibility results in the fairness literature. In other words, we would not be able to achieve parity with respect to multiple bias metrics. Hence, we encourage you to choose the fairness notions appropriate for your ML application, and focus on the corresponding bias metrics.

Bias mitigation during this stage includes incorporating fairness constraints as part of the objective function during model training and applying post-processing methods such as adjusting cutoffs in the trained model. We recently demonstrated that fair and accurate models can be obtained by tuning a machine learning model's *hyperparameters*. We also demonstrated how to minimize the largest group error rate (that is, make sure that the worst-off group is as well-off as possible) using the minimax group fairness framework, since this notion may be preferable in ML applications in which most or even all of the targeted population is disadvantaged.

A/B Testing, Deployment and Monitoring of ML Models: The final component of the ML lifecycle involves measuring and monitoring fairness metrics on an ongoing basis. This step can not only be used for measuring these metrics during A/B tests, but also help with detecting issues like model drift with respect to fairness.

Explainable AI using Amazon SageMaker

Explainable AI Prerequisites

Our goal is to offer tools to provide global explanations of models and to explain the predictions of a deployed model producing inferences. Such model explanation tools can help ML modelers and developers and other internal stakeholders understand model characteristics as a whole prior to deployment and to debug predictions provided by the model once deployed. We focus on feature attribution approaches, which can be used both for global understanding of a model after training and for providing per-instance explanation during inference. Our current offering includes a scalable and efficient implementation of SHAP (SHapley Additive exPlanations), based on the concept of the Shapley value from the field of cooperative game theory that assigns each feature an importance value for a particular prediction.

What is the function of an explanation in the machine learning context? An explanation can be thought of as the answer to a why-question, thereby helping a human understand the cause of a prediction. In the context of a machine learning model, we may be interested in answering questions such as “Why did the model predict a negative outcome (e.g., loan rejection) for a given user?”, “How does the model make predictions?”, “Why did the model make an incorrect prediction?”, and “Which features have the largest influence on the behavior of the model?” Thus, explanations can be useful for auditing and meeting regulatory requirements, building trust in the model and supporting human decision making, and debugging and improving model performance.

Some customers may care about *contrastive explanations*, or explanations of why an event X happened instead of some other event Y that did not occur. Here, X is the event that happened (an unexpected or surprising outcome as discussed above), and Y corresponds to an expectation based on their existing mental model. Note that for the same event X, different people may seek different explanations depending on their point of view or mental model Y. In the context of explainable AI, we can think of X as the example being explained and Y as a “baseline” that is typically chosen to represent an uninformative or average example in the dataset. Sometimes, the baseline may be implicit, e.g., an image with all pixels of the same color in the case of ML models for images.

Feature Attributions using Shapley Values

Our offering provides feature attributions based on the concept of Shapley values, from which you can determine the contribution that each feature made to model predictions. These attributions can be provided at an instance level for specific predictions and at a global level for the model as a whole. For example, for an ML model used for college admissions, the explanations can help determine whether the GPA and the SAT score were the features most responsible for the model’s predictions and we can determine how responsible they were for an admission prediction about a particular student.

We have adopted the concept of Shapley values from game theory and deployed it in a machine learning context. The Shapley values quantify the contribution of each player to a game, and hence provide the means to distribute the total payoff generated by a game to its players based on their contributions. Before describing how Shapley values can be used to assign feature attributions in the context of ML models, we first provide an illustrative example to give insight into their computation and their axiomatic properties which make them an attractive method for ML explainability.

Assume a coalition of three players $\{P_1, P_2, P_3\}$ cooperates in playing a game and gets a combined payoff of $1000. Our goal is to decide how to distribute the total payoff into three players, which naturally, should be commensurate with the contribution of each player to the game. So the question boils down to “How can we compute the contribution of each player?”. Let us also assume that we are given the information about the payoff for all possible coalitions of players.

Coalition #	**Players**	**Payoff ($)**
1	$\{\}$	0
2	$\{P_1\}$	25
3	$\{P_2\}$	100
4	$\{P_3\}$	150

5	$\{P_1, P_2\}$	550
6	$\{P_1, P_3\}$	650
7	$\{P_2, P_3\}$	50
8	$\{P_1, P_2, P_3\}$	1000

One intuitive way to assign the contribution to P_i would be to see how much payoff the game will generate if P_i played alone. For instance, the payoff of the game with P_1 as the sole player is \$25 whereas with P_2 the payoff is \$100. So we could assign the contribution of P_2 to be ($\frac{100}{25} =$) 4 times more than that of P_1.

However, this assignment might be problematic: even though a game with P_1 as the sole player only produces a \$25 payoff, the addition of P_1 into a game with P_2 massively increases the payoff from \$100 (coalition #3) to \$550 (coalition #5). A similar situation occurs when we go from coalition #3 to coalition #6. Similarly, while the coalition $\{P_2, P_3\}$ generates a payoff of \$50 only, adding P_1 to it increases the payoff to \$1000. Note how the payoff of $\{P_2, P_3\}$ is smaller than the payoff with either of them as a sole player – perhaps they do not play together very well. So it seems fair to assign a contribution to P_1 that not only considers the payoff of the game with P_1 as the sole player, but also considers how much the payoff improves as P_1 is added to a game consisting of other players.

That is exactly what Shapley values do! The Shapley value of player P_i is the average marginal payoff over all possible coalitions when P_i is added to them. According to the original 1951 paper titled *The Value of an n-Person Game* by Lloyd Shapley, the contribution of player P_i can be computed as:

$$\sum_{S \subseteq P \setminus \{P_i\}} \frac{|S|!(|P|-|S|-1)!}{|P|!} \times (v(S \cup \{P_i\}) - v(S))$$

where P, which in our case is $\{P_1, P_2, P_3\}$, is the set of all players. The coalition S is a subset of all players P not containing P_i and the operator $|.|$ represents the size of a set. As we discussed before, the sum is computed over all such coalitions. Finally, the function $v(T)$ returns the payoff for a given coalition T. The second term inside the summation corresponds to the marginal payoff – the increase/decrease in payoff – when adding our player P_i into the coalition S. The first term is included for the sake of scaling and accounts for the fact that the coalition S of players can be constructed in many different orders. With this computation, the Shapley values of P_1, P_2, and P_3 come out at \$483, \$221, and \$296, respectively.

It turns out that Shapley value is the *only* solution that satisfies the following axiomatic properties:

Efficiency: The sum of Shapley values adds up to the total payoff. We can confirm this from our example above where 483 + 221 + 296 = \$1000.
Symmetry: With two players P_i and P_j, if the marginal contribution of both is the same for all coalitions not containing either of the players, then their Shapley values are also the same.
Linearity: For two different games, the sum of the Shapley values of a player P_i over the two games is the same as the Shapley value of P_i over the sum of the two games.
Dummy: If P_i is a null player, that is, if P_i has zero marginal gain over all coalitions, then the Shapley value for this player is 0.

Now moving back to the ML context, we can compute the feature attributions to be the Shapley values in a game where the total payoff is the model prediction with all the features included, and the players are the individual features. Taking the example of our college admission scenario, consider a model with features {SAT Score, GPA, Class Rank}. Let us assume that we want to explain the model prediction for a candidate. The model prediction ranges from 0 to 1. The prediction for our candidate is 0.95. Then in our game, the total payoff would be 0.95 and the players would be the three individual features. If for our candidate, the Shapley values are {0.65, 0.7, -0.4}, then we learn that the the GPA affects the prediction the most, followed by the SAT score. We also learn that while GPA and SAT score affect the prediction positively, the class rank affects it negatively (note that a lower rank is better).

You can already see how satisfying the above axioms is a very attractive property for a ML explainability method. For instance, taking the Efficiency axiom, this would mean that the sum of all feature attributions sums up to the total model prediction! Or, if adding a feature never changes the model prediction, then the feature should get an attribution score of zero.

As we already saw, computing the Shapley values involves considering all possible coalitions of features. This means that given d features, there are 2^d such possible feature coalitions, each corresponding to a potential model that needs to be trained and evaluated. Even for reasonable values of d, say 50 features, it is computationally prohibitive and impractical to train 2^d possible models, and hence we need to make use of various approximation techniques. For this purpose, we have adopted SHAP, which incorporates such approximations, for instance, sampling coalitions in a specific manner and using a different scaling term than in the original Shapley value computation. We further devised a scalable and efficient implementation of the Kernel SHAP algorithm through additional optimizations.

Baselines
As noted earlier, explanations are typically contrastive, that is, they account for deviations from a baseline. As a result, for the same model prediction, we can expect to get different explanations with respect to different baselines. Hence the choice of a baseline is crucial. The baseline in a machine learning context corresponds to a hypothetical instance that can be either uninformative or informative. During the computation of Shapley values, we generate several new instances between the baseline and the given instance, in which the absence of a feature is modeled by setting the feature value to that of the baseline and the presence of a feature is modeled by setting the feature value to that of the given instance. Thus, the absence of all features corresponds to the baseline and the presence of all features corresponds to the given instance.

How can we choose good baselines? Often it is desirable to select a baseline with very low information content. For example, we can construct an average instance from the training dataset by taking either the median or average for numerical features and the mode for categorical features. For the college admissions example, we may be interested in explaining why a particular applicant was accepted as compared to a baseline consisting of an average applicant.

Alternatively, we can choose to generate explanations with respect to informative baselines. For the college admissions scenario, we might like to explain why a particular applicant was rejected when compared with other applicants from similar demographic backgrounds. In this case, we can choose a baseline that represents the applicants of interest, namely those from a similar demographic background. Thus, informative baselines can be used to concentrate our analysis on the specific aspects of a particular model prediction. They can isolate the actionable features for assessment by setting demographic attributes and other non-actionable features to the same value as in the given instance.

SageMaker Clarify allows you to specify the rows of instances, which can be one or many, to use as the baseline dataset for the SHAP algorithm.

Global Explanations of Models via Aggregation
In addition to providing explanations for per-instance inferences, we also support global explanation for ML models that helps us understand the behavior of a model as a whole in terms of its features. We generate a global explanation of an ML model by aggregating the Shapley values over multiple instances. SageMaker Clarify supports three different ways of aggregation:

1. "`mean_abs`": mean of absolute SHAP values for all instances
2. "`median`": median of SHAP values for all instances
3. "`mean_sq`": mean of squared SHAP values for all instances

Monitoring Feature Attributions for the Deployed Model

Much like what we saw in the section on monitoring bias metrics, a drift in the live data distribution can result in a corresponding drift in the feature attribution values. Taking the example of our hypothetical college admission scenario, let us say that we observe the following (aggregated) feature attribution values in the training data and the live data:

Feature	**Attribution in Training Data**	**Attribution in Live Data**
SAT Score	0.70	0.10
GPA	0.50	0.20
Class Rank	0.05	0.70

The change from training data to live data seems pretty big; the feature ranking has completely reversed. Similar to the bias drift, the feature attribution drifts may be caused by a change in the live data distribution and warrant a closer look into the model behavior on the live data. Again, the first step in these scenarios is to raise an alarm that a drift has happened.

Naturally, we can detect the drift by comparing how the ranking of the individual features changed from training data to live data. In addition to being sensitive to changes in *ranking order* only, we also want to be sensitive to the *raw attribution score* of the features. For instance, given two features that fall in the ranking by same number of positions going from training to live data, we want to be more sensitive to the feature that had a higher attribution score in the training data. With these properties in mind, we use the Normalized Discounted Cumulative Gain (NDCG) score for comparing the feature attributions rankings of training and live data.

Specifically, let us assume we have the following:

- $\mathrm{F} = [f_1, \ldots, f_m]$ is the list of features *sorted w.r.t. their attribution scores in the training data* where m is the total number of features. For instance, in our case, $\mathrm{F} =$[SAT Score, GPA, Class Rank].
- $a(f)$ be a function that returns the *feature attribution score on the training data* given a feature f. For instance, in our case, a(SAT Score) = 0.70.
- $\mathrm{F}' = [f_1', \ldots, f_m']$ is the list of features *sorted w.r.t. their attribution scores in the live data*. For instance, in our case, F' = [Class Rank, GPA, SAT Score].

Then, we can compute the NDCG as:

$$\mathrm{NDCG} = \frac{\mathrm{DCG}}{\mathrm{iDCG}} \text{ with } \mathrm{DCG} = \sum_{i=1}^{m} \frac{a(f_i')}{\log_2(i+1)} \text{ and } \mathrm{iDCG} = \sum_{i=1}^{m} \frac{a(f_i)}{\log_2(i+1)}$$

The quantity DCG measures if features with high attribution in the training data are also ranked higher in the feature attribution computed on the live data. The quantity iDCG measures the "ideal score" and is just a normalizing factor to ensure that the final quantity resides in the range [0, 1], with 1 being the best possible value. A NDCG value of 1 means that the feature attribution ranking in the live data is the same as the one in the training data. In our particular example above, since the ranking changed by quite a bit, the NDCG value is 0.69.

In SageMaker Clarify, if the NDCG value is below 0.90, we automatically raise an alert.

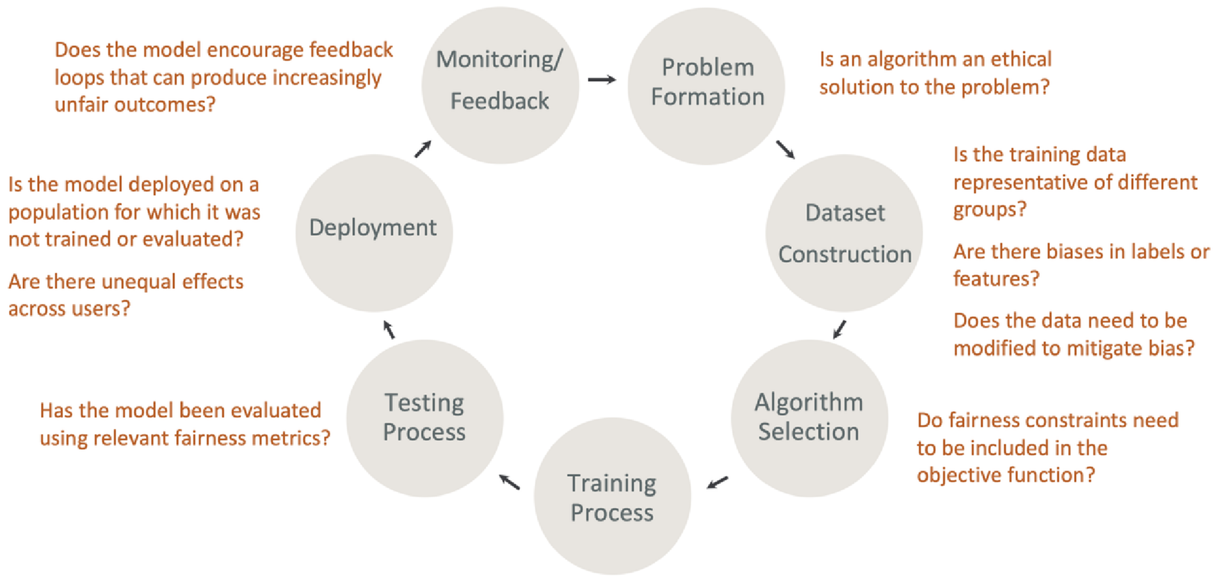
Broader Perspective

Although our current offering focuses on Shapley value based feature attribution methods for global understanding of a trained model and for providing per-instance explanation during inference, we also aim to provide a broader perspective on explainable AI. In addition to scenarios involving tabular data, explainability is becoming increasingly important for models based on NLP, computer vision, and multimodal domains. The field of explainable AI is rapidly evolving, and several new explainability techniques are being developed and improved upon. These techniques range from feature attribution methods to counterfactual explanation methods to causality based approaches to distillation methods. In addition, there is growing awareness on the need for robust explanations and protection against model reconstruction or misleading explanations.

Best Practices and Limitations

Fairness as a Process: We recognize that the notions of bias and fairness are highly application dependent and that the choice of the attribute(s) for which bias is to be measured, as well as the choice of the bias metrics, may need to be guided by social, legal, and other non-technical considerations. Building consensus and achieving collaboration across key stakeholders (such as product, policy, legal, engineering, and AI/ML teams, as well as end users and communities) is a prerequisite for the successful adoption of fairness-aware ML approaches in practice.

Fairness and Explainability by Design in the ML Lifecycle: Fairness and explainability should be taken into account during each stage of the ML lifecycle, for example, Problem Formation, Dataset Construction, Algorithm Selection, Model Training Process, Testing Process, Deployment, and Monitoring/Feedback. It is important to have the right tools to do this analysis. To encourage engaging with these considerations, here are a few example questions worth asking during each of these stages.


Does the model encourage feedback loops that can produce increasingly unfair outcomes?
Monitoring/ Feedback
Problem Formation
Is an algorithm an ethical solution to the problem?
Is the training data representative of different groups?
Are there biases in labels or features?
Does the data need to be modified to mitigate bias?
Dataset Construction
Is the model deployed on a population for which it was not trained or evaluated?
Are there unequal effects across users?
Deployment
Algorithm Selection
Do fairness constraints need to be included in the objective function?
Has the model been evaluated using relevant fairness metrics?
Testing Process
Training Process

Model Explanations and Privacy Risks: Explanations help build trust by providing insights into model predictions. However, there is the possibility that these insights may be negatively used to compromise privacy and leak information about the model internals. The research community is beginning to identify and analyze such risks.

Technical Limitations: SageMaker Clarify helps detect potential bias during data preparation, after training, and in your deployed model by examining attributes you specify, but does not support bias mitigation methods. Moreover, algorithmic fairness can also be defined at the individual level instead of facets. However, such definitions require specifying a similarity metric which may vary widely from one ML task to another. As a result, SageMaker Clarify does not include individual-level fairness metrics. For the explainability analysis, SageMaker Clarify uses an optimized implementation of the open source SHAP library to support tabular datasets (in CSV or JSONLines format), but does not currently support other formats such as text and images or other explanation methods. Moreover, SHAP implements an approximate computation of the Shapley values in order to overcome the exponential runtime, and hence the estimated Shapley values may differ from the true Shapley values. The accuracy of the estimate can, however, be increased by tuning the `n_samples` hyperparameter.

Conclusion

Amazon SageMaker Clarify is a new functionality in Amazon SageMaker that helps detect bias in data and models and helps explain predictions made by ML models. It is important to realize that this is an ongoing effort that is part of a larger process. We have focused on providing tools for bias and explainability in the context of an ML lifecycle. It cannot be overemphasized that developing AI solutions needs to be thought of more broadly as a process involving iterated inputs from and discussions with key stakeholders such as product, policy, legal, engineering, and AI/ML teams as well as end users and communities, and asking relevant questions during all stages of the ML lifecycle.