Computational Fairness: Preventing Machine-Learned Discrimination*

Michael Feldman

May 8, 2015

Abstract

Machine learning algorithms called classifiers make discrete predictions about new data by training on old data. These predictions may be hiring or not hiring, good or bad credit, and so on. The training data may contain patterns such as a higher rate of good outcomes for members of certain groups (e.g. racial groups) and a lower rate of good outcomes for other groups. This is quantified by the "80% rule" of disparate impact, which is a legal measure and definition of bias. It is ethically and legally undesirable for a classifier to learn these biases from the data. We propose two methods of modifying data, called Combinatorial and Geometric repair. We test our repairs on three data sets. Experiments show that our repairs perform favorably in terms of training classifiers that are both accurate and unbiased.

1 Introduction

Disparate impact is a form of discrimination referring to practices in employment, housing, or similar institutions that prefer or reject people based on *protected* attributes such as race, gender, age, religion, and so on. The idea was formalized in the 1971 Griggs v. Duke Power Co. US Supreme Court case [3]. Duke Power required intelligence tests and high school diplomas for its higher paid jobs. It was found that this adversely affected black workers. It was also found that before this policy was implemented, the performance of promoted (white) workers did not correlate to having a diploma and required an intelligence score. The court ruled against Duke Power: despite its practices appearing neutral, in reality there was a disparate impact on race [3].

Disparate impact is not necessarily conscious or malicious as in Duke Power, and can often be unintended. A typical example is a company that wants to filter incoming resumes down to promising candidates. It may use a machine learning algorithm to

*Portions of this work were included in Certifying and Removing Disparate Impact [5].

simplify its hiring process. Machine learning algorithms can take data and produce models that can make predictions on future input based on past training data. Models that make discrete outcomes, such as hiring or not hiring an applicant, are called *classifiers*. The company may take hiring data from previous years – attributes of each applicant and whether or not they were hired – and use it to make such a classifier. However, as this data was made by humans, disparate impact may be present. A machine learning algorithm will identify these patterns and incorporate them into the model; future predictions by this model will share the biases in the data. As a more concrete example, if a company hired men at a greater rate than it hired women, the classifier will too.

This behavior is ethically and legally undesirable. Following Griggs v. Duke Power Co., the state of California and then the federal US government adopted the "80% rule" to measure disparate impact [7]. If the rate of hiring underprivileged groups is less than 80% of the rate of hiring privileged groups, then according to this guideline, there is disparate impact.

$$\frac{\text{Rate of unprivileged applicants receiving good outcome}}{\text{Rate of privileged applicants receiving good outcome}} < 0.8$$

An objective measure like this is an important step to solving the problem.

Preventing this bias is nontrivial. The *redlining effect* is a form of discrimination that relies on unprotected attributes which are predictive of protected attributes. For example, race and location are often predictive of each other; American banks historically would draw "red lines" around black neighborhoods and reduce community investment on that basis. For this reason, it isn't enough to remove racial data from individual loan applicants, because a bank could still discriminate by looking at an applicant's street address. It is also unacceptable to remove both the protected attributes and unprotected attributes that predict them, because that unprotected information may still be valuable.

Therefore, we turn to alternative solutions that modify data or learning algorithms without removing unprotected data outright.

2 Preliminaries

Here we will define several machine learning classifiers and methods of measuring their correctness. These will be referred to in a following review of relevant literature and a discussion of our methods and experiments.

2.1 Naive Bayes

Naive Bayes are probabilistic classifiers. Given a data set with attributes Y_A, ..., Y_N and a class C (its outcome), then for each $c \in C$ and each $i \in Y_I$, calculate Pr[c] (probability of any C=c) and Pr[i|c] (conditional probability of Y_I=i given C=c).

To predict the class of some collection of values $a, ..., n$, take the maximum probability

$$\frac{\text{Pr[c]} \times \text{Pr[a|c]} \times ... \times \text{Pr[n|c]}}{\text{Pr[a]} \times ... \times \text{Pr[n]}}$$

for each c. *Naive* refers to the complete independence of each attribute. Here, this means multiplying them together, such that each attribute contributes equally, instead of taking shared information into account. Despite the name, Naive Bayes performs very well in a variety of applications. Some of its advantages are that it is simple to implement, and quick to train and classify new data: it relies on easily countable information in the data. It also performs well with small data sets.

2.2 Support Vector Machines

Support Vector Machines, here referred to as *SVM*, are non-probabilistic linear binary classifiers. Given a data set of points in n-dimensions (n attributes), calculate a hyperplane (an $n-1$ subspace) that spatially separates the points by their class. The goal is to maximize the space (a *maximum margin*) on either side of the hyperplane.

To predict the class of some new point, determine on which side of the hyperplane it lies.

For example, consider a set of points in 2 dimensions. A 1-dimensional hyperplane (a line) might be drawn through the points such that all the points are separated by class. However, it is possible that no such line might be found. *Kernel methods* are functions that map points into different spaces; for example, a higher-dimensional space. Thus a set of 2-dimensional points might become 3-dimensional, at which point a 2-dimensional hyperplane (a plane) might be drawn.

Non-probabilistic refers to the fact that SVMs do not rely on probability, unlike Naive Bayes. *Linear* refers to drawing a line (in some dimension) to separate points by class. *Binary* refers to choosing one of two classes, unlike Naive Bayes, which can handle any number of classes.

2.3 Error Measurement

A *confusion matrix* is a measure of how many positive and negative guesses made by a classifier were true and false:

		Predicted	
		Yes	No
Actual	Yes	**True Positive (TP)**	**False Negative (FN)**
	No	**False Postive (FP)**	**True Negative (TN)**

A simple percentage of correct guesses is *accuracy*, or formally,

$$\frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

A more sophisticated measure of classifier performance is the *balanced error rate*, or *BER*:

$$\frac{1}{2} \times \left(\frac{\text{FN}}{\text{FN} + \text{TP}} + \frac{\text{FP}}{\text{FP} + \text{TN}} \right).$$

We measure performance as *utility* $= 1 -$ BER.

BER is *balanced*, or class-conditioned, in that it places equal weight between predictions that were actually yes and no. In data sets containing bias, protected attributes (e.g. Race) can predict the class. This means that a measure balanced this way will place equal weight between the protected attributes' values.

Consider a data set wherein 90% of the entries have values X=$\bar{\text{x}}$ (where $\bar{\text{x}}$ is a privileged group) and C=+ (where + is the good outcome), and the remaining 10% have X=x and C=−. A classifier may learn to assign every entry C=+. In this case, accuracy=0.9, but utility=0.5. A lower utility is indicative of a classifier that does not equally weigh both values of X.

3 Previous Work

Here we will review pertinent research on preventing machine learned discrimination. There are, in general, two approaches to this problem: modify the data or modify the machine learning algorithm.

3.1 Classification with No Discrimination

Kamiran and Calders propose a model they term *Classification with No Discrimination*, or *CND*[4]. As their method is the most similar to ours among those described here, we will describe CND in detail. The process of learning a CND takes a potentially biased data set D, a single binary X (a protected attribute, e.g. Race), a value x of X (an underprivileged group), and a desired outcome + as inputs. $\bar{\text{x}}$ is the other value of X. Their measure of bias will be referred to as the *Kamiran–Calders Discrimination Measure*, or *KCDM*, where:

$$KCDM = \Pr[\text{C}=+|\text{X}=\bar{\text{x}}] - \Pr[\text{C}=+|\text{X}=\text{x}].$$

The KCDM measures the disparity in two such values. For example, if KCDM = 0.40, then entries where value X=$\bar{\text{x}}$ have a 40% greater chance of receiving + than those where X = x. The goal is to create a classifier that minimizes KCDM on values of X, meaning that the probability of an entry receiving + is independent of its value of X. Using a data set with a KCDM = 0, a machine-learned classifier should also make predictions which themselves have a KCDM = 0.

Learning a CND proceeds as follows: First learn a Naive Bayes classifier on D to calculate the probability of each data entry receiving the desired outcome[1]. Then pick two groups from D with the following conditions:

[1]This part is quite flexible; any algorithm that can rank items by probability will work.

- *candidates for promotion* (*CP*) are entries with X=x and C≠+
- *candidates for demotion* (*CD*) are entries with X=$\bar{x}$ and C=+

and sort CP by decreasing probability, and CD by increasing probability. In other words, the first CP is the entry of the unprivileged group which is most likely to have gotten the desired outcome, but didn't; the first CD is the entry of the privileged group which is least likely to have gotten the desired outcome, but did.

Starting with the first elements in both lists, iterate through each element and change each element's class. Stop when the KCDM is 0. This will happen after a finite number of times, calculated as *num-swaps*(x, $\bar{x}$), where:

$$
\begin{aligned}
J_i &= \text{number of entries where J=i} \\
J_{i+} &= \text{number of entries where J=i and C=+} \\
J_j &= \text{number of entries where J=j} \\
J_{j+} &= \text{number of entries where J=j and C=+} \\
\text{num-swaps(i, j)} &= \frac{(J_i \times J_{j+}) - (J_j \times J_{i+})}{J_i + J_j}
\end{aligned}
$$

After num-swaps(x, $\bar{x}$) swaps, a new data set D' with KCDM=0 exists. Finally, using D', learn a classifier using any machine learning model – the result is a CND.

The authors ran experiments on a data set D of German banking data. A protected column X was Age: x is young and $\bar{x}$ is old[2]. The desired outcome + was the bank assigning *good* credit. They created four data sets: D, D without X, D', and D' without X. Naive Bayes classifiers were learned on each to produce four classifiers, the last two of which were CNDs.

Across 15 trials of random samples of input from D, the two CNDs always produced a smaller KCDM than the classifiers learned on D. The CND learned on D' had a marginally smaller KCDM than the CND learned on D' without X. The authors do not comment on or explain why including the protected attribute lowers the CND's KCDM. It is possible that by removing discrimination, including the protected attribute not only doesn't degrade the KCDM, but actually provides more data for the classifier to use, and increases its general performance.

The authors also discretized Age into old and young at different values to modify the KCDM in the original data. As KCDM increases in these data sets, so do the resultant KCDM in the classifiers produced with it; however, the two CNDs' KCDMs are always the lowest.

They measured the accuracies (percentage of correct predictions) of their classifiers. In each trial on D, all four classifiers have very similar accuracies. As KCDM increases in D', the disparity in accuracies increases, with CND having the lowest accuracy, but at most the two classifier types' accuracy varies between 70 and 73.

The two missing capabilities they note are handling non-binary protected attributes and classes, and incorporating multiple protected attributes as opposed to a single X.

[2]Ages were discretized at Old $\geq$ 25.

3.2 Three Naive Bayes models

Calders and Verwer propose and analyze three methods of creating Naive Bayes models that minimize bias [2]. Unlike the CND, their approach involves modifying the learning algorithm itself rather than the training data. They also measure bias using the KCDM.

3.2.1 Modified Naive Bayes

The first method they propose is *Modified Naive Bayes*, or *MNB*. The key difference between MNB and Naive Bayes is that MNB uses Pr[C|X] instead of Pr[X|C]. Given a classifier *M*, D, and a distribution Pr[C|X], modify M in the following way:

Test M on D and calculate the resulting KCDM and number of + labels assigned. If the number of assigned + labels is less than the number of original + labels in D, then increase Pr[C=+|X=x] (a greater chance for a privileged group member to receive +). Otherwise, increase Pr[C=+|X=$\bar{x}$] (a greater chance for an unprivileged group member to receive +). Repeat this process until M's KCDM is 0.

There will be a minimal KCDM produced from the resultant M, and the key advantage is that by design, the number of C=+ labels will be as close as possible to those in D. This is important because simply modifying the probabilities to minimize the KCDM will either create more or fewer C=+ assignments. That may be an unacceptable solution: for example, a company might only be willing to hire a specified number of people, and an unbiased classifier needs to match that number.

3.2.2 Two Naive Bayes

The second method they propose is *Two Naive Bayes*, or *2NB* [2]. Two Naive Bayes classifiers are learned on two sections of D: entries where X=x, and entries where X=$\bar{x}$. Thus there are two classifiers, each trained on data with one value of X. The overall classifier M situationally chooses which to use, based on the value of X. M's probability Pr[C|X] is modified as was described to create a Modified Naive Bayes.

The advantage of 2NB is that because each classifier is trained on data consisting solely of one value of X, X provides no useful information to each classifier. Therefore, they cannot take X into account, and it is impossible for values of an arbitrary unprotected column Y to decide the value of X, which nullifies the redlining effect (unprotected attributes predicting protected attributes).

3.2.3 Latent Variable Model

The third method they propose is *Latent Variable Model*, or *LVM* [2]. The goal is to discover a latent, i.e. unobserved, attribute *L* which has values + and − corresponding to C. L and X are independent; C depends on values of L and X. Once this Bayesian model is in place, the goal is to discover the values of L, which are the labels without taking discrimination (column X) into account.

3.2.4 Discussion

The authors tested Modified Naive Bayes, Two Naive Bayes, and Latent Variable Model on both generated and real-life data. 2NB's predictions had both the highest accuracies and the lowest KCDM, followed closely in many tests by MNB. LVM did not perform as well in general.

3.3 Prejudice Remover Regularizer

Kamishima et al. discuss several kinds of unfairness in machine learning [8]. They focus on disparate impact, which they term indirect prejudice, and provide a classifier-modifying method.

Let us first briefly define *Logistic Regression*, a classification algorithm the authors use in their experiments. A *logistic function* is

$$f(z) = \frac{1}{1 + e^{-z}},$$

where z is any number and $f(z)$ is any number between 0 or 1 inclusive (the probability of an event occuring based on z). The logistic function has a particular shape to produce this behavior: when $z > 0$, $f(z) > 0.5$; when $z < 0$, $f(z) < 0.5$. z is the sum of products consisting of a variable and a coefficient. A larger coefficient increases the probability, and a smaller coefficient decreases the probability. The greater the coefficient's absolute value, the greater impact it has on $f(z)$.

The coefficients are chosen to minimize a *cost function*, sometimes referred to as an objective function. If a variable is multiplied by a large number (its cost) in the cost function, then in order to minimize cost, its coefficient will need to be correspondingly small. We can reduce the impact that some variables have by increasing their cost and therefore decreasing their coefficients, or contribution to the result of $f(z)$. This type of constraint on variables is called a *regularizer*. One type is an L_2 regularizer, which applies higher costs to higher-order variables, as in a polynomial function. Because lower-order variables will therefore contribute more, the result is a more general function rather than a highly complex function that matches the training data too closely. Matching the training data instead of generalizing to best predict new data is *overfitting*.

The authors propose a regularizer based on a measure called *Prejudice Index*, or *PI*, defined as

$$\mathrm{PI} = \sum_{(X,C)\in D} \Pr[\mathrm{X},\, \mathrm{C}]\, \ln \frac{\Pr[\mathrm{X},\, \mathrm{C}]}{\Pr[\mathrm{X}]\Pr[\mathrm{C}]}$$

which measures how much information X and C share. This is equivalent to

$$\sum_{(X,Y)\in D} \sum_{Y\in\{0,1\}} M[\mathrm{Y}|\mathrm{X},\, \mathrm{S};\, \Theta]\, \ln \frac{\Pr[\mathrm{C}|\mathrm{X}]}{\Pr[\mathrm{C}]}$$

where M is a prediction model and Θ are its parameters. In the cost function, it is multiplied by n, a positive numeric parameter, which corresponds to how strict the regularizer is, i.e. fairness. The authors note that this is computationally heavy, and instead calculate Pr[C|X] and Pr[C] with a sample mean (an approximation). The authors use this regularizer (which minimizes PI) and an L_2 regularizer [8].

The authors tested multiple classification algorithms: Logistic Regression with the full data (LR), Logistic Regression without protected attributes (LRns), Logistic Regression with their PI regularizer (PR), Naive Bayes with the full data (NB), Naive Bayes without protected attributes (NBns), and Calders and Verwer's 2NB (CV2NB). They used the Adult Income data set which contains attributes of people and whether they earn $\geq$\$50k/year. They measured accuracy (percentage of correct predictions) and PI/MI. MI is the mutual information between the data's actual classes and the predicted classes, so a smaller PI/MI means there is a lower Prejudice Index (less bias) and a higher Mutual Information (staying close to the original classes, i.e. retaining information from the original data). Thus a high accuracy and a low PI/MI indicates a high-performing, fair classifier.

Three instances of PR were tested at n=0, 5, 10, 15, 20, 30. At PR n=5, PR's accuracy outperformed all Naive Bayes classifiers. Its PI/MI outperformed NB and NBns, but was significantly larger than CV2NB's PI/MI.

At higher values of n, the accuracy decreases, achieving the lowest PI/MI of all classifiers (except CV2NB). At n=30, PI/MI rises, meaning that there was not a significant gain fairness in exchange for decreased accuracy.

They authors also tested a synthetic data set with a class that depends on two attributes, one of which was predictive of a protected attribute. The goal was to test how PR and CV2NB perform with attributes that do not have equal prediction of protected attributes. PR was tested at values n=0 to 300. Except a brief dip at approximately n=250, PR had significantly higher accuracies than CV2NB, especially between n=0 to 100. It had higher PI/MIs as well, though the disparities were much smaller than in the previous comparisons.

CV2NB outperforms PR in many measures. However, PR has several advantages. Because it is a regularizer, this approach is independent of specific implementations of learning algorithms, though it is restricted to algorithms that have cost functions. Thus their method shares some of the power that data-modifying approaches have, in that the best learning algorithm for the data set can be used. Their synthetic data experiment demonstrated that PR's ability to handle attributes independently (unlike Naive Bayes' equal treatment of attributes, for which it is called Naive) is a useful advantage, as well.

4 Repair Tool

Here we will define our repair algorithm, including pseudocode and examples.

We have implemented a data-modifying algorithm first described in [5]. The goal of this algorithm is to take a biased data set D and to create a data set D', such that

```
define unique_value_data_structures(columns):
   sorted_lists =  hashmap(key: a column name; value: a list of values)
   index_lookups = hashmap(key: a column name; value: a hashmap)
   for each column in columns:
      sorted_list = sort(unique values of column)
      sorted_lists.put(column, sorted_list)
      index_lookup = hashmap(key: a value of column;
                             value: its index in sorted_list)
      for each value in sorted_list:
         index_lookup.put(value, index)
      index_lookups.put(column, index_lookup)
   return sorted_lists, index_lookups
```

Figure 1: Create a sorted list of unique values per column, and a data structure that finds the indices of values in that list.

```
define median(list):
   return sort(list)[list.len()/2]
```

Figure 2: In the case of even-length lists, prefer the smaller item.

an entity looking at D' will not be able to use unprotected columns Y to determine stratifying columns S (e.g. Race).

First we must describe how our repair uses S columns. As an example, let's say that two columns marked S are Race = {White, Nonwhite} and Gender = {Male, Female, Nonbinary}. All stratified *groups* are made: White-Male, White-Female, ..., Nonwhite-Nonbinary. These are the values of S that a learner should be unable to determine. Store the number, or *size*, of entries in D associated with each group. From this point on, ignore all groups with size=0, meaning they are not present in the data.

The repair process for each Y column is independent of the other Y columns. To continue our example, let's use a Y column ExamScore. In order to make ExamScore values non-predictive of stratified groups, we make each group's distribution on ExamScore equal. For example, the highest ExamScore values of White-Male entries and Nonwhite-Female entries will be the same. The same is true of the entries with the lowest values. At any quantile n, all ExamScore values will be the same across all groups. Thus the ExamScore distributions are the same for each group. Again, the motivation for this is that a learner cannot look at an entry and learn anything about its Race or Gender based on its ExamScore.

Quantiles are decided as follows. We want as many quantiles as possible (meaning fewer entries per quantile) for greater precision. However, we require that each quantile contains at least one entry of each group, in order to ensure that the distributions are equal. The largest number of quantiles we can choose while guaranteeing that there will be at least one entry per quantile is equal to the size of the smallest group.

The unique values in each Y column are found and stored in sorted lists per

column. Recall that all values in Y columns must be orderable. ExamScore's unique values = {3, 5, 6, 8, 9, 10, 11, 13, 15, 18}, and the values at the 1st quantile of Nonwhite-Males = {9, 13, 15}. We choose the median[3] value, 13. Say that median values for other groups were chosen the same way, and are {8, 9, 10, 13, 13, 15}. We choose the median value among these values, preferring the smaller item in the case of even-length lists, which is 10. We will call 10 the *target* for all ExamScore values in the 1st quantile of each group.

Once the target is chosen, the two repair methods diverge into *Combinatorial* or *Geometric* repairs. They each allow the notion of *partial repair* which relies on a repair amount λ, an input between 0 and 1 inclusive. At λ=0, D=D'. At λ=1, distributions of Y columns in D' are as equal as possible among groups, i.e. every value in a given quantile is equal to its quantile's target value. When $\lambda < 1$, there is no guarantee that the target values will be the same per quantile, and thus no guarantee that the distributions will be equal.

A partial repair might be motivated by simultaneous goals of maximizing utility and minimizing disparate impact. While a full repair may minimize disparate impact, it may come at the cost of utility, which is undesirable. There may exist a λ at which disparate impact is reduced to some acceptable amount (recall the legally defined 80% rule), thus preventing the need to use a data set repaired at λ=1, which would have further decreased utility.

Combinatorial repair

The first of two repair methods at this point is Combinatorial repair. We use a sorted list *SL* of all unique ExamScore values[4], the index[5] g of the original value in SL, and the index t of the target value in SL. We choose

$$\text{repair index } ri = \text{round}(\text{g} + (\lambda \cdot (\text{t} - \text{g})))$$

rounding ri to the nearest integer. The original value is updated to SL[ri]. Once a target value is chosen for a quantile, this process must be repeated individually for every entry at that quantile.

As an example at λ=0.5, recall that the algorithm chose 10 as the target value for the 1st quantile. One of the original values to be repaired in this example is 13. Thus in the above SL of ExamScore values, SL[g]=13, SL[t]=10, g=7, t=5. We find that ri is 6:

$$6 = 7 + (0.5 \cdot (5 - 7))$$

[3]We choose medians rather than means because our algorithm has no knowledge of the contexts of these values. Choosing only from the pool of existing values ensures that we don't make nonsensical repairs. This is only guaranteed for Combinatorial repair.

[4]The previous list of ExamScore's values is sorted, and thus SL for this example.

[5]Zero-indexed in this example.

```
define repair(requested_repair_type, Y_columns, all_stratified_combinations,
              number_of_quantiles, sorted_lists, index_lookups, lambda):
   quantile_unit = 1.0/number_of_quantiles
   for each column in Y_columns:
      group_offsets = hashmap(key: stratified group; value: offset)
      for each quantile in [0, ..., number_of_quantiles]:
         median_values_at_quantile = []
         entries_at_quantile = []
         for each group in all_stratified_groups:
            offset = round_to_nearest_int(group_offsets.get(group)*group.size())
            number_of_entries_to_get = round_to_nearest_int(
               (group_offsets.get(group) + quantile_unit)*group.size()) - offset

            select entry ID and value from column, in D, where S = group,
               sorted by column, from offset to number_of_entries_to_get
            entries_at_quantile.append(entry IDs)
            median_values_at_quantile.append(median(values))

         target_value = median(median_values_at_quantile)
         position_of_target = index_lookups.get(column).get(target_value)

         for each entry ID in entries_at_quantile:
            select value from column, in D, where ID = entry ID

            if requested_repair_type == combinatorial:
               position_of_original_value = index_lookups.get(column).get(value)
               distance = target_value - position_of_original_value
               distance_to_repair = round_to_nearest_int(distance * lambda)
               index_of_repair_value = position_of_original_value + distance_to_repair
               repair_value = sorted_lists.get(col_name)[index_of_repair_value]
            else:
               repair_value = (1 - lambda)*value + lambda*target_value

            update D', set column = repair_value, where ID = entry ID
```

Figure 3: The repair algorithm.

```
sorted_lists, index_lookups = unique_value_data_structures(X_columns and Y_columns)

all_stratified_combinations = cartesian_product(each column in S)

for each combination in all_stratified_combinations:
   if combination.size() == 0:
      all_stratified_combinations.remove(combination)

number_of_quantiles = min(combination.size() for each combination in
                          all_stratified_combinations)

D' = clone(D)

repair(requested_repair_type, Y_columns, all_stratified_combinations,
       number_of_quantiles, sorted_lists, index_lookups, lambda)
```

Figure 4: The setup and call to repair with parameters.

The entry with ExamScore=13 is now repaired to SL[6], which is 11.

Geometric repair

The second of two repair methods at this point is Geometric repair. Given an original value and a target value, choose:

$$\text{repair value } rv = ((1 - \lambda) \cdot \text{original}) + (\lambda \cdot \text{target})$$

The original value is updated to rv. Note that in Geometric repair, it is possible that D' will contain values not present in D.

•

After the Combinatorial or Geometric repair is finished, remove all protected X columns. The resulting data set is D'.

5 Experimental Setup

Here we describe our experimental setup and the three data sets we use to test. In order to measure the effects of the repairs, three data sets were used. Preprocessing was applied to each data set as described in the following sections. Recall that Combinatorial repair refers to choosing a repair index of possible values between the original and target values, and Geometric repair refers to choosing a repair value on a continuous number line between the original and target values.

For each experiment, 22 data sets were generated after preprocessing: 11 versions modified by Combinatorial repair at λ in the interval [0.0, 0.1, 0.2, ..., 1.0], and 11 versions modified by Geometric repair at the same λ.[6]

[6]At λ=0.0, both repair algorithms output the same file (identical to the original).

Each generated data set was used to train and test an SVM classifier from the scikit-learn Python package (except the *Ricci* experiment, which used a simple classifier based on one column). For each trained classifier, utility (1 − BER), accuracy (percentage of correct predictions), Zemel Fairness (rate of privileged entries receiving good outcomes minus unprivileged entries receiving good outcomes), and Disparate Impact scores (rate of unprivileged entries receiving good outcomes divided by privileged entries receiving good outcomes) were calculated.

5.1 Ricci v. DeStefano data

The Ricci v. DeStefano data set contains the results of a fire department's exam used for promotions. There are five columns in the set, but only Race and Combine were used in this experiment. Race is categorical: a given person is either Black, Hispanic, or White. For all experiments in this paper, Race columns were transformed into binary White and Nonwhite. Combine is continuous numeric between 0 and 100 inclusive. A Combine ≥ 70 is a pass on the exam.

Recall that protected columns are marked X (e.g. Race); unprotected columns are marked Y (e.g. interview score); class columns are marked C (e.g. hiring decision); stratifying columns are marked S (the full repair alters Y such that Y cannot predict S), which may apply to both X and Y columns. After preprocessing Race into binary values, the repaired data sets were generated. For this experiment, Race was X and S, and Combine was Y rather than C. This was done for two reasons. The first is that Combine is merely a 60/40 weight of two other columns (which are scores on the written and oral portions of the exam) and therefore it would be trivial to predict if those columns were included in the data. The second is that the goal of this experiment was to study the effects of the repair on only one column.

To that effect, a simple classifier was used in place of an SVM classifier: as in the actual promotion exam, a Combine of 70 or greater is a pass. We applied the same rule to our repaired data's Combine values. We calculated utility, accuracy, Zemel Fairness, and Disparate Impact scores comparing the repaired Combine to the original Combine.

5.2 German Credit

The German Credit data set[7] contains attributes of people and whether they were classified as having good or bad credit. We discarded columns that were categorical and not clearly orderable (i.e. not having a clear best and worst value), because the repair process requires this. We kept the following columns: Status of existing checking account, Duration in month, Credit history, Credit amount, Savings account/bonds, Present employment since, Installment rate in percentage of disposable income, Personal status and sex, Present residence since, Property, Age in years, Number of existing credits at this bank, Job, Number of people being liable to provide

[7]From UC Irvine's machine learning repository: *http://archive.ics.uci.edu/ml*.

maintenance for, Class. These columns consist of numerical and orderable categorical data. We were interested in stratifying by Age, which was transformed into binary Old and Young values at Old $\geq$ 25 in accordance with other papers that use this data[4].

After this preprocessing, the repaired data sets were generated. Sex was a protected column and Age was a protected stratify column. The scikit-learn SVM was applied to the repaired data sets. The SVM in all our experiments used regularizers for L_2 (to avoid overfitting too closely to the training data) and hinge loss (to maximize maximum-margin).This experiment on the German Credit data was repeated ten times using ten pseudorandom distributions of $\frac{2}{3}$ train and $\frac{1}{3}$ test splits on all 22 repaired data sets. The resulting metrics were averaged.

5.3 Adult Income

The Adult Income data set[8] contains attributes of people and whether they earn greater than or equal to \$50k/year. As with the German Credit experiment, we discarded non-orderable columns. We kept the following columns: Age, fnlwgt (the sampling weight), education-num, gender, race, capital-gain, capital-loss, hours-per-week, class. Some data cells in the data set contain unknown values. However, there are no unknowns in the columns we kept, so this was not a problem in our experiments.

Race was transformed into binary White and Nonwhite values. Gender and Race were protected columns, and we generated three versions of repairs by stratifying based on Gender, Race, and both Gender and Race.

The SVM was applied to the repaired data sets. The Adult Income data was provided in $\frac{2}{3}$ train and $\frac{1}{3}$ test splits (both sets were repaired), so no random distributions and averaging was necessary.

6 Experimental Results

Here we discuss the results of the experiments on the Ricci v. DeStefano, German Credit, and Adult Income data sets.

We rely on four measurements: utility (1 − BER), accuracy (percentage of correct predictions), and two previously unmentioned measures. Zemel Fairness is defined [9] as

$$\textit{Zemel Fairness} = \Pr[\mathrm{C}{=}+|\mathrm{X}{=}\bar{\mathrm{x}}] - \Pr[\mathrm{C}{=}+|\mathrm{X}{=}\mathrm{x}].$$

Note that this is equivalent to Kamiran and Calders' KCDM [4]. We also define a formalized definition of the Disparate Impact Score as

[8]From UC Irvine's machine learning repository: *http://archive.ics.uci.edu/ml.*

$$DI = \frac{\Pr[\mathrm{C}=+|\mathrm{X}=\mathrm{x}]}{\Pr[\mathrm{C}=+|\mathrm{X}=\bar{\mathrm{x}}]},$$

where + is a good outcome of class C and $\bar{\mathrm{x}}$ is a privileged group of protected attribute X. Note that the conditional probabilities in these definitions can be understood as the rate of privileged entries receiving the good outcome compared to the rate of unprivileged entries receiving the good outcome.

We will make two kinds of graphs to understand the results of our experiments:

A *performance graph* compares utility and accuracy to λ, the variable which determines partial repair (λ=1 is a full repair). It measures how classifier performance degrades as D' (the repaired data) becomes more different than D. Performance graphs demonstrate the difference in comparing performance with accuracy and utility. Note that "better" in this context means data points at high utility or accuracy and high λ; this signifies data that was more thoroughly repaired and trained a high-performing classifier.

A *fairness graph* compares utility to Disparate Impact and Zemel Fairness scores. While λ is an abstract concept more relevant to the repair algorithm, DI and Zemel Fairness scores are more relevant to people using these data sets for legal and ethical reasons. They would also be concerned with the performance of their classifiers. Thus we compare these types of measurements directly, focusing on utility. Recall that we chose $1 - \mathrm{BER}$ as utility because BER tends to be a more stable measurement than accuracy (as is apparent in later data). A second reason for preferring utility is that it is class-conditioned, i.e. equally weighs the predictions that were good outcomes in D and those that were bad. In data sets that contain bias, this loosely corresponds to placing equal weight on predictions pertaining to the privileged and unprivileged groups, respectively. "Better" has a similar meaning in fairness graphs as in performance graphs: it means data points at high utility and high Disparate Impact or Zemel Fairness scores.

6.1 Ricci v. DeStefano data

We first discuss the results of the Ricci v. DeStefano experiments using the simple classifier.

Recall that experiments using the Ricci data used a mock classifier rather than an SVM. The performance graph in Figure 5 shows that utility and accuracy degrade steadily as λ increases. This trend will be the case in most following experiments. At λ=0.4, the measures diverge, and it's clear that Geometric repair slightly performed better than Combinatorial at higher values of λ.

The results in the fairness graph in Figure 6 are remarkably stable; neither type of repair has any significant difference by these measures. In other words, there are two clear trends in the graph, which correspond to the two types of fairness measurements. Neither measurement shows a difference between Combinatorial and Geometric repairs.

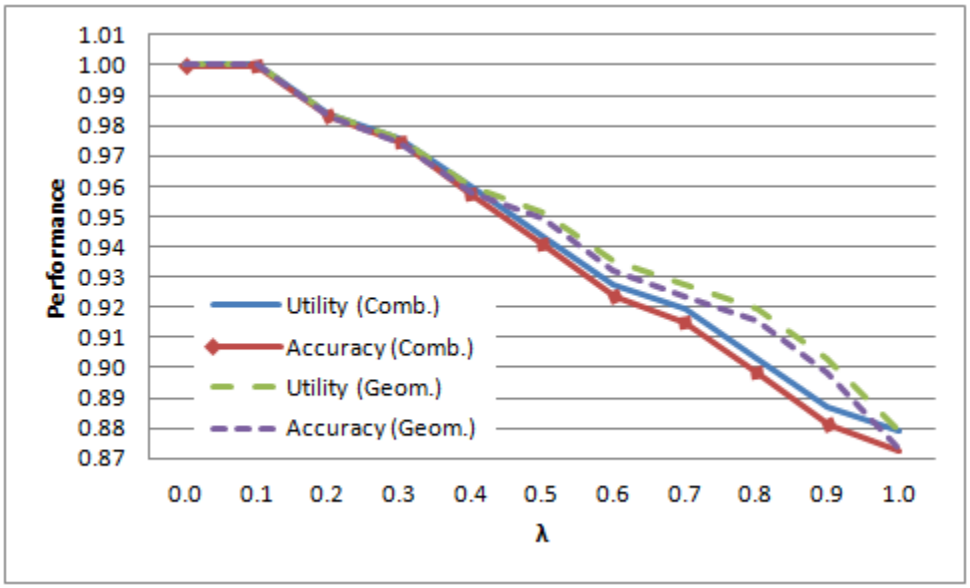

Figure 5: Ricci v. DeStefano Performance Graph. Utility and accuracy versus λ. Stratified by Race.

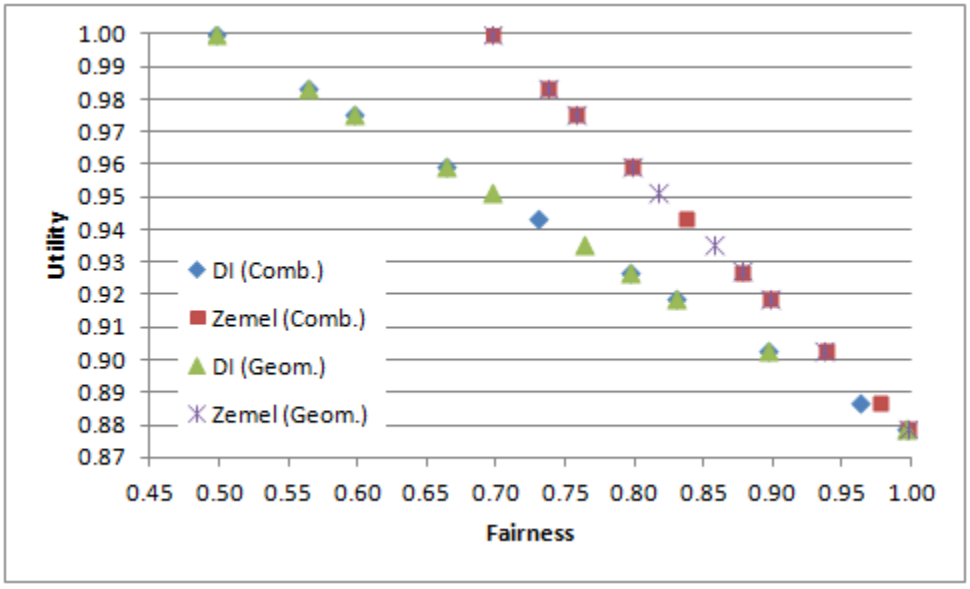

Figure 6: Ricci v. DeStefano Fairness Graph. Utility versus Disparate Impact and Zemel scores. Stratified by Race.

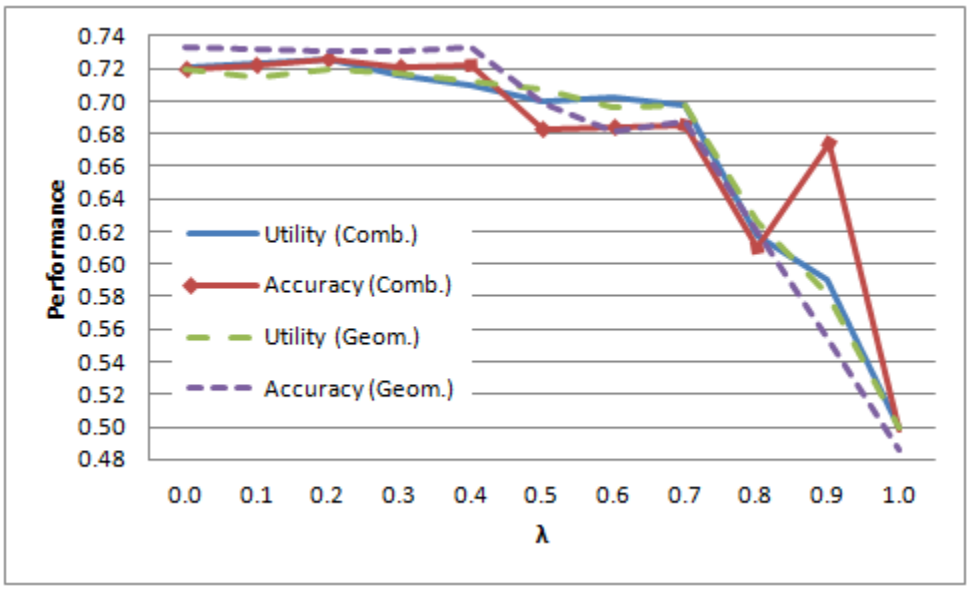

Figure 7: German Credit Performance Graph. Utility and accuracy versus λ. Stratified by Age.

6.2 German Credit

We next analyzed the German Credit data repairs.

The performance graph in Figure 7 shows close similarities in performance of both repairs. Though they fluctuate slightly, they share nearly identical results in each respective measure at each value of λ. However, at λ=0.9, there is an unusual spike in the Combinatorially repaired data's accuracy (though not its utility). Data indicates that high accuracy and low utility is caused by a classifier predicting C=+ at higher rates for a majority group and C=− at higher rates for a minority group. This may be a quirk of the repair process; one advantage of our algorithm is that it is easy to generate a variety of repairs at different values of λ and choose the best performing D'.

This particular value aside, it is possible that the overall consistency in data is due to a significantly larger data set (1,000 entries) compared to Ricci v. DeStefano (118 entries), pointing to a trend more inherent to the data itself than the particular repair method. Another possibility is that there are no significant differences between the two methods. Until further experiments on similarly large data sets show a difference between Combinatorial and Geometric repairs, we cannot assume either case.

All performance measures sharply converge on 0.5 starting around λ=0.8, and are less than or equal to 0.5 at λ=1.0. These measures being 0.5 is no coincidence: on this data set, at λ=1.0, each repaired column in D' contains the same value. The SVM chooses at random between predicting all good outcomes once trained and all bad. Of course, the resulting SVM is useless when trained on this data. One possible explanation for this is that the data set was thoroughly biased, in that every piece of data was predictive of Age. Thus, removing all bias removed all useful data.

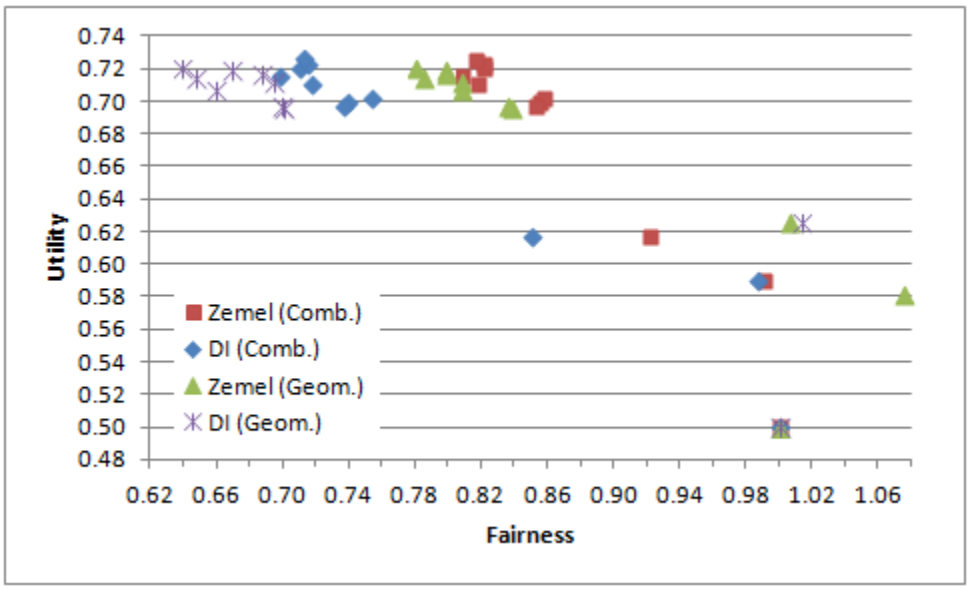

Figure 8: German Credit Fairness Graph. Utility versus Disparate Impact and Zemel scores. Stratified by Age.

The fairness graph in Figure 8 shows where the repairs diverge. Unsurprisingly, points scattered without pattern in the bottom right correspond to λ=0.8, 0.9, and 1.0, discussed in the previous paragraph. The points before that range, however, are tightly clustered in a range between utility=0.69 and 0.73. The Combinatorial repair slightly outperformed the Geometric repair as measured by greater fairness at comparable utility.

6.3 Adult Income

We next analyzed the Adult Income data repairs.

Across all three Stratification choices (stratifying by Gender, by Race, and by both Gender and Race), the performance graphs in Figures 9–11 are similar, much to the degree that the German Credit performance graphs were. Whereas utility has a consistent downward trend in each graph, several accuracy data points are significantly more or less accurate than surrounding points. There are several values of λ for which the Combinatorial or Geometric data performs better, but most are the same value in a given measure, or very close. Still, the differences between these numbers, even at the values with a spike in performance, are quite small.

As before, it is possible that differences between the two repair methods are smoothed out by the sheer size of the Adult data set, or there are no real differences at all. The fairness graphs in Figures 12–14 are also fairly consistent, and do not favor one type of repair over another.

We compared the results of all stratification options (Gender, Race, and Gender+Race) side by side. Due to the overall similarities of both Combinatorial and

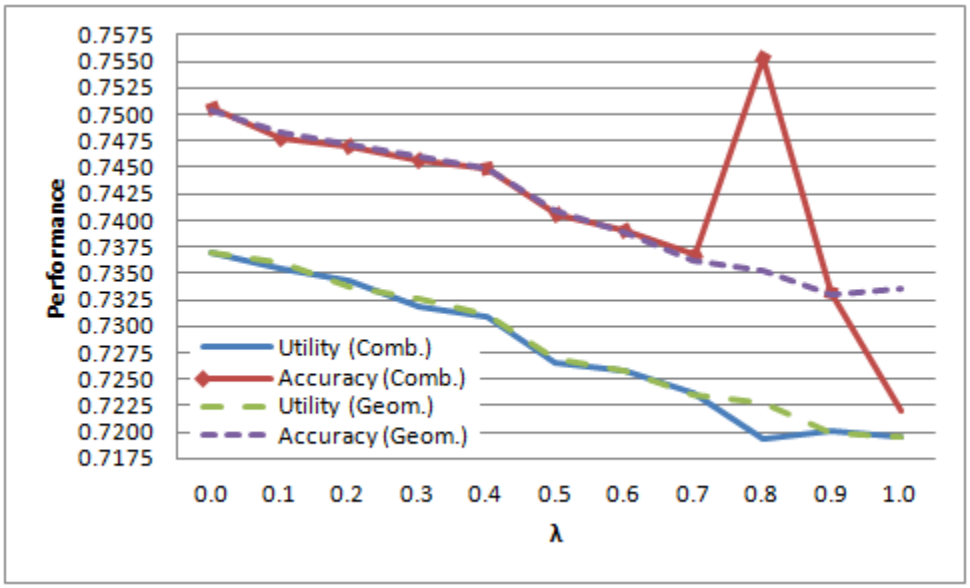

Figure 9: Adult Income Performance Graph. Utility and accuracy versus λ. Stratified by Gender.

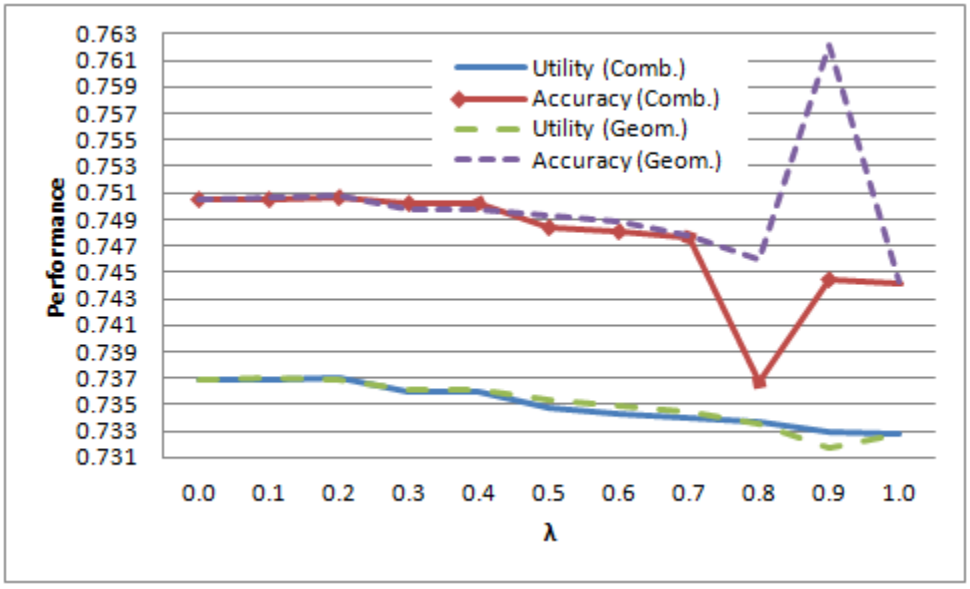

Figure 10: Adult Income Performance Graph. Utility and accuracy versus λ. Stratified by Race.

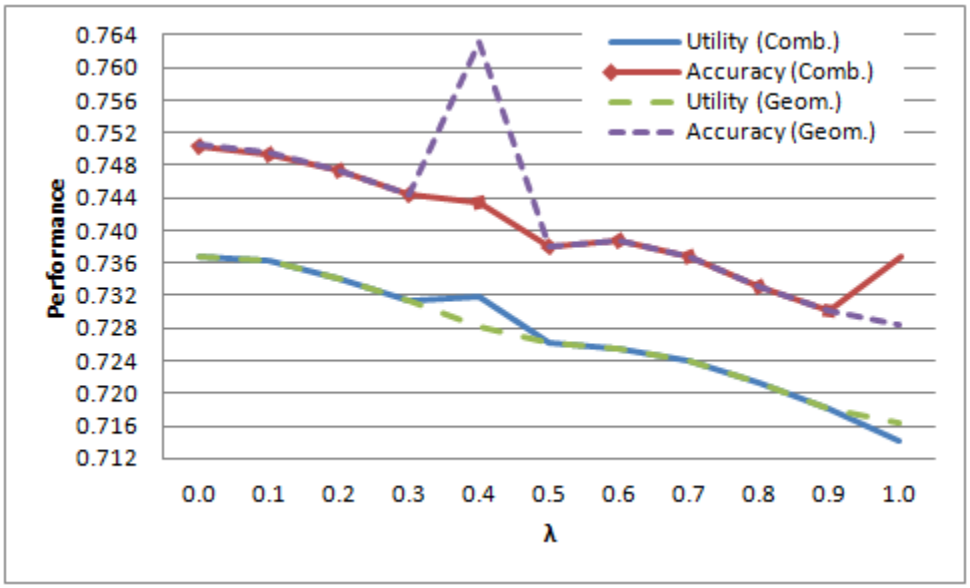

Figure 11: Adult Income Performance Graph. Utility and accuracy versus λ. Stratified by Gender and Race.

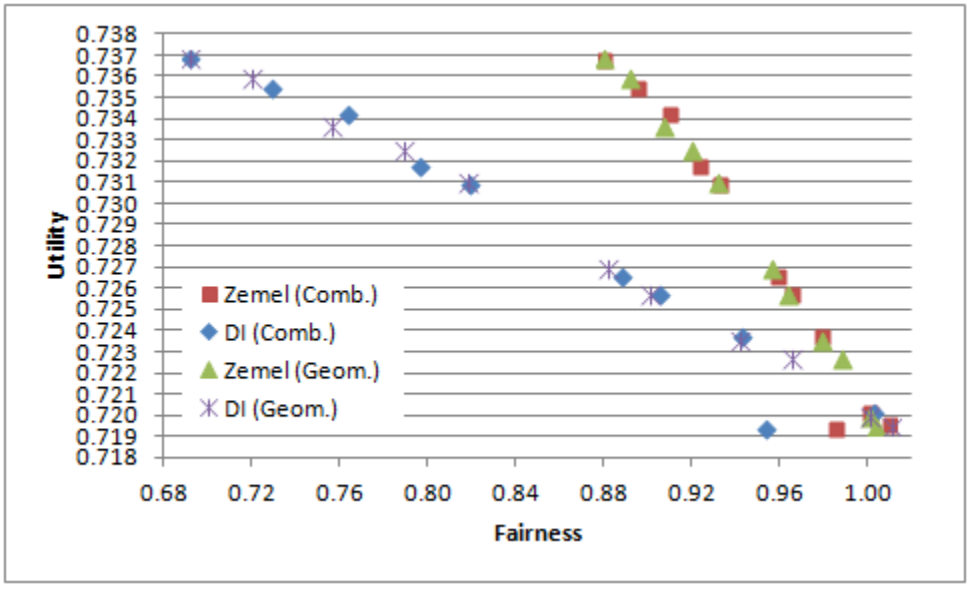

Figure 12: Adult Income Fairness Graph. Utility versus Disparate Impact and Zemel scores. Stratified by Gender.

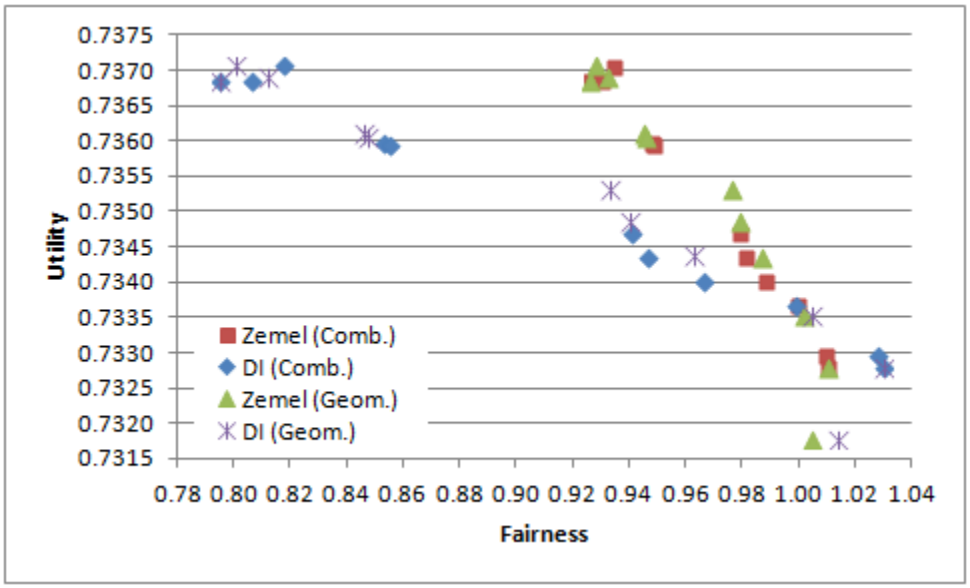

Figure 13: Adult Income Fairness Graph. Utility versus Disparate Impact and Zemel scores. Stratified by Race.

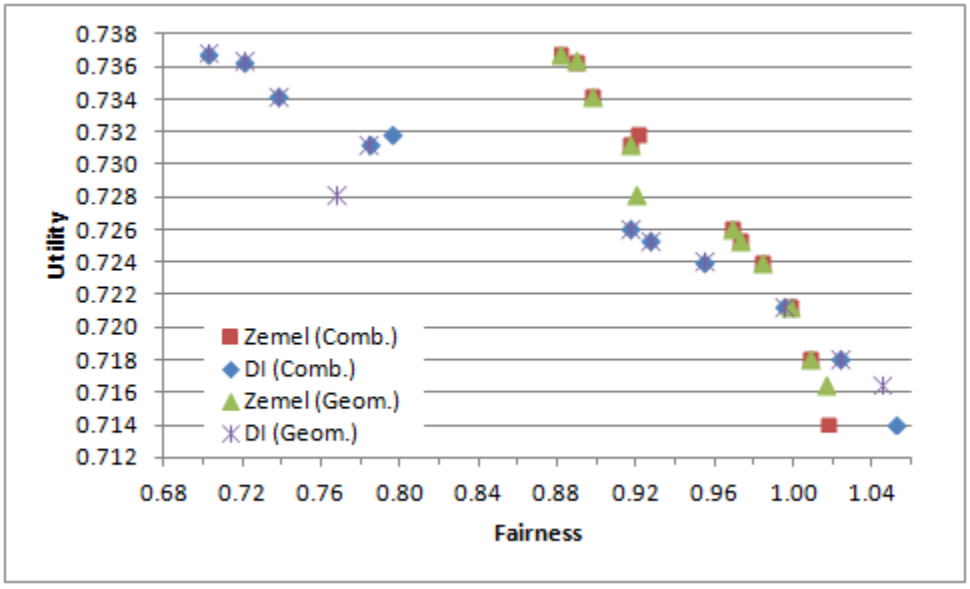

Figure 14: Adult Income Fairness Graph. Utility versus Disparate Impact and Zemel scores. Stratified by Gender and Race.

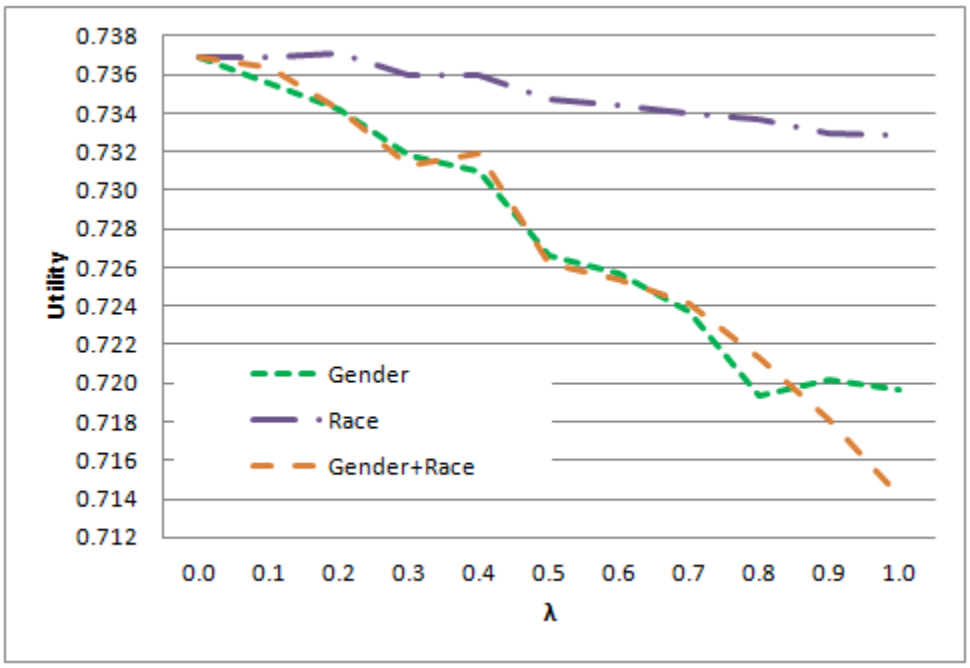

Figure 15: Adult Income Performance Graph. Utility versus λ. Only Combinatorial repair data shown. All stratifications (Gender, Race, Gender and Race) shown.

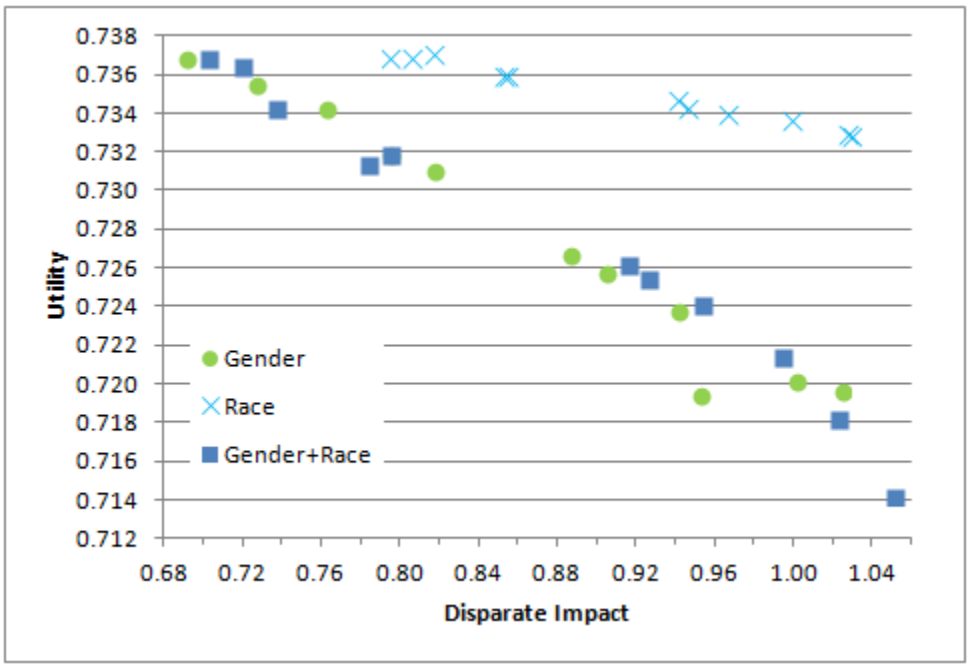

Figure 16: Adult Income Fairness Graph. Utility versus Disparate Impact. Only Combinatorial repair data shown. All stratifications (Gender, Race, Gender and Race) shown.

Geometric repairs on this data set, we looked only at the Combinatorial results. Figures 15 and 16 show the performance and fairness graphs respectively. We can see that the fairness measures of the data repaired by Race are always higher than fairness measures against other stratifications. Stratifying by Race yielded the best performance, i.e. the smallest degradation of utility and accuracy as λ, the DI score, and Zemel Fairness each approached and reached 1.0. It is reasonable to conclude that high performance was preserved because Race is not as predictive of income in the data set as Gender is, and thus data was not lost in repair.

An important observation to make of the data is that stratifying by both Gender+Race did not significantly worsen the performance compared to stratifying by only one column. In fact, utility (the more stable measurement) was nearly identical in both Gender and Gender+Race. To fully repair values to their target values at λ=1.0, the Gender+Race utility dips slightly below the utility of Gender, but this is a difference of less than one percent utility. Further experiments should explore whether this is maintained when more than two columns are stratified. However, it is a promising demonstration that our repair can handle multiple stratifications at no significant extra cost to performance.

6.4 Comparisons to Previous Work

We next compared our results to other contributors' work, using information collected by Zemel et al [9].

Figure 17 shows these comparisons.

Recall that "better" refers to data points having both high accuracy and fairness. Several data points in both graphs are better than many other implemented classifiers, or approximately as good as them. This comparison reveals two important advantages of our method. One advantage is that while our data was collected by training an SVM, we could train any type classifier on a repaired data set D' in order to choose the best result. For instance, the Kamiran and Calders' CND did better than several of our points on the German Credit and Adult Income data. Their experiments used a Naive Bayes classifier. In fact, when our algorithm was used to create a D' that trained a Naive Bayes classifier, it actually outperformed CND [5]. The other advantage is that partial repair allows many variations of D', which also allows the user to choose the most favorable tradeoff of performance and fairness.

7 Conclusion

We have proposed two methods, Combinatorial and Geometric repair. We demonstrated that they effectively remove bias from data sets which train high-performing classifiers using three data sets. Further, we have introduced an American legal measure of bias to the field, whereas prior contributors often create their own measures.

Our methods have several advantages. One is that any classification algorithm can be used after the repair; presumably, the highest-performing classifier on a given

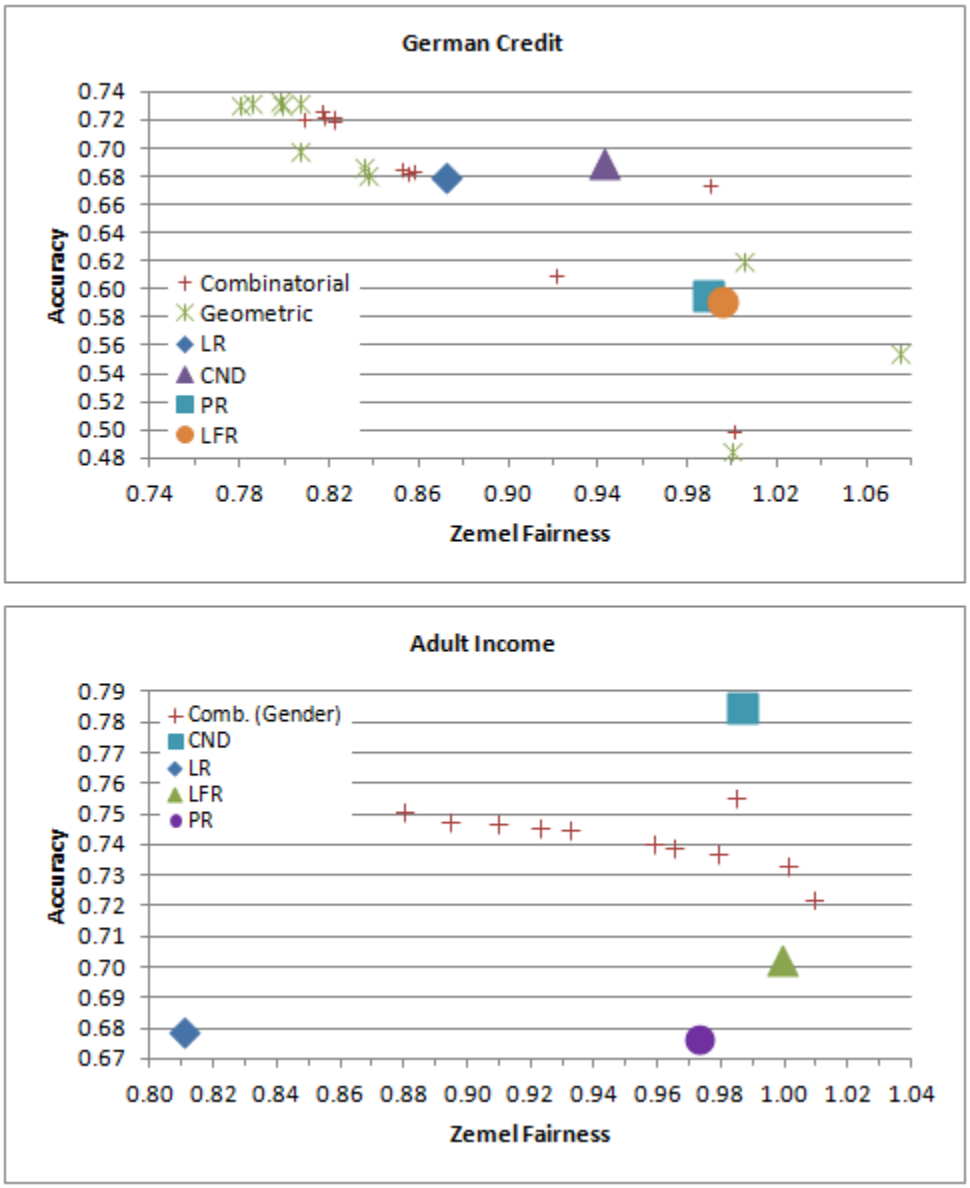

Figure 17: Comparison to Other Results on German Credit and Adult Income. accuracy versus Zemel Fairness. The Adult data is only repaired for Gender. *LR* is Logistic Regression applied to the original data set. *CND* is the Kamiran and Calders Classifier with No Discrimination. *PR* is the Kamishima et al. Prejudice Regularizer applied to logistic regression. *LFR* is the Zemel et al. Learned Fair Representation (not discussed in this paper). This data comes from the Zemel et al. Learning Fair Representations: Supplementary Materials [9].

repaired data set D' will be chosen. Similarly, a second advantage is that partially repairing data is built into the algorithm, thus allowing the user to choose the data set D' with the most satisfactory tradeoff of performance and fairness. Of course, there is the additional option between the two types of repairs. A third advantage is that our repair easily handles multiple stratified attributes. A fourth advantage is that our repairs trained classifiers that performed very well against other approaches to this problem.

Future work should expand on our contributions. At most, we stratified with two protected groups at once (Gender and Race in the Adult Income data). Our results indicated that this did not significantly worsen performance, but we would like further testing to confirm or reject this hypothesis. An important limitation of our method is that it must discard all information that is not orderable (to create and repair distributions). A new approach that expands our idea to include these types of columns could perform even better. Perhaps our Combinatorial or Geometric repair could be applied to orderable columns in a data set D, and a different repair could be applied to nonorderable columns.

Additionally, a small improvement can be made on the algorithm. When the Combinatorial repair chooses the target, it takes a median, in order to choose only values that exist in the original data. Currently our medians break ties by choosing the smaller of two values. However, it could take advantage of the Sorted Lists to choose a value (which exists in the original data) in between two values.

References

[1] Solon Barocas and Andrew D. Selbst. Big Data's Disparate Impact. *California Law Review*, 104, 2015.

[2] Toon Calders and Sicco Verwer. Three Naive Bayes Approaches for Discrimination–Free Classification. *Data Mining and Knowledge Discovery*, 21:277–292, 2010.

[3] David J. Garrow. Toward a Definitive History of Griggs v. Duke Power Co. *Vanderbilt Law Review*, 67(1), 2014.

[4] Faisal Kamiran and Toon Calders. Classifying Without Discriminating. In *Classifying Without Discriminating*. Computer, Control and Communication, 2010.

[5] Michael Feldman, Sorelle Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and Removing Disparate Impact. 2015. http://arxiv.org/abs/1412.3756.

[6] Richard Primus. The Future of Disparate Impact. *Michigan Law Review*, 108, 2010.

[7] Technical Advisory Committee on Testing. *Uniform Guidelines on Employee Selection Procedures(1978).*, 1978. Section 4D.

[8] Toshihiro Kamishima et al. Fairness–Aware Classifier with Prejudice Remover Regularizer. *Machine Learning and Knowledge Discovery in Databases*, 2012.

[9] Rich Zemel, Toni Pitassi Yu Wu, Kevin Swersky, and Cynthia Dwork. Learning Fair Representations. *The International Machine Learning Society*, 2013.