A Comparison of Fairness-Aware Machine Learning Algorithms

Derek Roth

January 10, 2018

Abstract

Fairness, as it applies to algorithms, implies that the decisions made by an algorithm are being made such that there is no discrimination against individuals or groups in labeled data sets. In this paper, I give a summary of the relevant findings by computer scientists regarding fair algorithms, discuss three techniques used to reduce/remove bias in algorithms, and examine three case studies that clearly demonstrate the importance of this field of study. The goal of this thesis is to determine the best practices for case studies on this topic and to discover ways of developing algorithms that are unbiased. I apply existing algorithms to four data sets and compare their results in order to determine which are the most useful in a specific situation.

1 Introduction

As we begin to rely more and more on machine learning algorithms to make important decisions for us, the issue of the fairness of those algorithms becomes paramount. Nowadays, everything from loan approval/disapproval to college admissions is being handled, at least in part, by an algorithm. On the surface, this seems like the best and easiest way to ensure complete objectivity in decision making. Unfortunately, there are ways in which the algorithms can learn our own human biases and thus be compromised in their decision-making.

To put things into perspective, let's say there is a bank that needs to determine whether or not to give out loans. This bank uses an algorithm to make this decision to save time and to reduce discrimination. The bank's classification algorithm takes a data set made up of a number of individuals with parameters such as age, sex, whether or not previous loans have been repaid, and whether or not the individual received a loan. Using this *training*

data, the algorithm finds patterns that can help it to make predictions as to whether or not an individual will receive a loan given new data without a classification attached. However, let's say this particular bank is located in a predominantly white area where most of the non-white population struggles financially. Based on the training data, the algorithm will most likely make decisions that benefit those who have repaid previous loans more than those who have not repaid previous loans. Then, the area's white population will be given loans much more often than the area's non-white population. This is a form of discrimination called *indirect discrimination*. While race was not given explicitly as a parameter, one racial group was negatively impacted by the decisions of the algorithm.

So, how do we recognize an algorithm that is discriminatory in its decision-making? What do we do to improve it? In recent years, computer scientists have come up with a number of different techniques and tested them across several different data sets. The issue with this, of course, is that it is hard to compare the techniques when they haven't all been tested on the same data. It is for this reason that case studies are important. When we focus on one data set that we have some information about already, we can test the different techniques and decide which ones work best for predicting bias in algorithms and reducing bias in algorithms.

In this paper, I will first discuss the definition of fairness and the ways in which we measure discrimination and fairness. In Section 3, I will examine three models used to reduce algorithmic bias and some of their strengths and weaknesses. Then, in Section 4, I will focus on individual case studies, going through all of their procedures and explaining how they might have done things differently. In Section 5, I will discuss the set-up and conditions for the experiments I ran myself, the goal of which was to determine which of the group fairness algorithms is the most useful for different types of data. Finally, in Section 6, I will discuss the results of my experiments, highlighting the algorithms that are the most stable and the most accurate. I will also look into each of the measured metrics and discuss the tradeoffs involved in maximizing one over another.

2 Background Information

In this section, I will define two different types of fairness: individual fairness and group fairness. Both types are important in understanding how bias affects individuals and groups in relation to each other.

2.1 Types of Fairness

According to Dwork et al. [4], there are two types of fairness. The first of these, individual fairness, has to do with whether similar individuals

are getting similar outcomes. It was defined "by the principle that any two individuals who are similar *with respect to a particular task* should be classified similarly." Let's say there are two equally qualified candidates for a position in a company, one of them white and the other a person of color. Individual fairness says that both candidates should have an equal chance of getting the position as they are both up to the task.

Group fairness, as opposed to individual fairness, is more concerned with the number of positive outcomes being evenly distributed across groups. Instead of comparing individuals, group fairness looks at the bigger picture. For example, someone studying group fairness might look at the percentage of people hired from each race, expecting them to be the same or similar.

3 Techniques To Reduce Algorithmic Bias

In the last decade, several techniques to reduce algorithmic bias have been developed. The models discussed below are the Classification with No Discrimination model, a group of similar Naive Bayesian models, and a regularization model. All of these models define fairness as group fairness as opposed to individual fairness. Then, the classifier is "fair" when positive outcomes are evenly distributed across groups.

The Classification with No Discrimination Model introduced the idea of splitting individuals into groups depending on their classifications and the value of their sensitive attribute. It is important to look at this model because training two classifiers, one for the sensitive attribute and one for the favored attribute, essentially takes the value of the sensitive attribute out of the decision-making process.

Following that, the Naive Bayesian models looked into increasing the probability of discriminated sensitive values given the positive class and decreasing the probability of the favored sensitive values given the positive class. The second model actually learns two models, one for the set of individuals with positive sensitive values and one for the set of individuals with negative sensitive values. The final Bayesian model looks into the use of latent variables to hide actual class labels, ensuring that each individual has an equal probability of being discriminated against. These models expand on the ideas used to develop the CND model.

Finally, regularizers were introduced to a classifier's loss function to prevent over-fitting and to improve the fairness of the results. This was a new way of approaching the problem and helps to examine one of the many parts of fairness algorithms that is often thought of as static, the loss function.

3.1 Classi cation with No Discrimination Model

Calders and Kamiran [7] explain that merely removing the sensitive attributes from the training data is not enough to solve the problem of bias in classifiers. Certain attributes can be linked to sensitive attributes, making it easy for the classifier to identify the discriminated community.

To combat this problem, they suggest the *Classification with No Discrimination* model (CND). This model massages the data, first separating individuals into two groups: Candidates for Promotion (CP) and Candidates for Demotion (CD). The first group is made up of those individuals who have the sensitive attribute, but did not receive the positive classification, while the second group is made up of those individuals who do not have the sensitive attribute and did receieve the positive classification. The algorithm then ranks the members of each group based on probability for the positive classifier (decreasing order for CP and increasing order for CD) and promotes the best candidate in CP while demoting the worst candidate in CD. In this way, the balance between the two classes is maintained and the training data is edited until the discrimination score is zero. This modified training data will be used to train the classifier.

3.2 Naive Bayes Models for Discrimination-free Classi cation

Calders and Verwer [3] discussed three Naive Bayes approaches to classification without discrimination, all of which give similar results as far as accuracy and discrimination. In this study, the *discrimination score* is calculated as the difference in probability of those with the favored attribute and those with the sensitive attribute receiving the positive classification.

The first of these approaches, Algorithm 1, involves an algorithm that modifies the probability distribution of the sensitive attribute values S, given the class values C. The general idea behind this first approach is to increase the probability of discriminated sensitive values S given the positive class C_+ while decreasing the probability of favored sensitive values S_+ given the positive class. In their paper, Calders and Verwer bring up the problem with this straightforward approach as it tends to either increase or decrease the number of positive labels depending on how frequently the favored sensitive values appear in the data set. The best thing to do, when taking this approach to discrimination-free classification, is to keep the number of positive labels assigned by the classifier the same as the number of positive labels in the original data set. They do this by making a small change to the Naive Bayes model by changing $P(SjC)$ to $P(CjS)$. Instead of looking at the probability distribution for the sensitive attribute, given the class, they observe the probability distribution for the class attribute, given the sensitive attribute. $P(CjS)$ is modified until the discrimination score is equal to 0.0.

Algorithm 1 Modifying Naive Bayes

Require: a probabilistic classi er M that uses distribution $P(CjS)$ and a data-set D
Ensure: M is modi ed such that it is (almost) non-discriminating, and the number of positive labels assigned by M to items from D is (almost) equal to the number of positive items in D

Calculate the discrimination *disc* in the labels assigned by M to D
while *disc* > 0:0 **do**
  numpos is the number of positive labels assigned by M to D
  if *numpos* < the number of positive labels in D **then**
    $N(C_+; S_-) = N(C_+; S_-) + 0{:}01 \quad N(C_-; S_+)$
    $N(C_-; S_-) = N(C_+; S_-) \quad 0{:}01 \quad N(C_-; S_+)$
  else
    $N(C_-; S_+) = N(C_-; S_+) + 0{:}01 \quad N(C_+; S_-)$
    $N(C_+; S_+) = N(C_-; S_+) \quad 0{:}01 \quad N(C_+; S_-)$
  end if
end while

	White	**Non-White**
Hired	7000	5000
Not Hired	3000	5000

Table 1: Example hiring data (d = 0.200)

For example, Table 1 contains hiring data for White and Non-White workers at a fictional company. Using the difference between the probability of White people to be hired and the probability of non-White people to be hired as a measure of discrimination, we have a discrimination score of 0.200. If we run through the algorithm once, assuming the classifier has assigned fewer positive labels than were present in the training data, we can see the results in Table 2. Alternatively, if we run the algorithm once, assuming the classifier has assigned more positive labels than were present in the training data, we can see the results in Table 3.

After one pass of the algorithm, the probability of non-White people getting hired has gone up and the probability of non-White people not receiving the positve "Hired" classifier has gone down. In addition, the discrimination score has been recalculated and now sits at 0.197. While it is not hugely different from the original discrimination score, it serves to demonstrate that the algorithm can achieve a discrimination score of 0.0 after a number of iterations. It should be noted that this algorithm changes the data, thus lowering the accuracy.

	White	**Non-White**
Hired	7000	5030
Not Hired	3000	4970

Table 2: First pass of algorithm (d = 0.197)

A first pass of the algorithm with the "too many positives" assumption changes the results, but still lowers the discrimination score. This time, the probability of White people getting hired has gone down and the probability of White people not receiving the positive "Hired" classifier has gone up. The discrimination score has dropped from 0.200 to 0.195.

	White	**Non-White**
Hired	6950	5000
Not Hired	3050	5000

Table 3: Second pass of algorithm (d = 0.195)

Algorithm 1 leads to what is *technically* discrimination-free classification in that it does not directly discriminate by making decisions based on the sensitive attribute. However, it does not take into account the red-lining effect wherein the classifier uses attributes correlated to the sensitive attributes to make decisions. Then, *indirect* discrimination is still a problem using this model.

To solve the problem found with the first approach, a second approach is explored, one involving two Naive Bayes models. Certainly, the simplest thing to do would be to remove the correlated attributes from the data set, but this leads to a significant loss in accuracy. Instead, Calders and Verwer separate the data set into two separate sets, one for S and one for S_+. From these two data sets, similar to the CND approach, two models M_+ and M are learned and chosen by the classifier based on the value of the sensitive attribute S.

The third and final approach discussed in the Calders and Verwer paper is the latent variable model. This model recreates the discrimination process to find the class labels the data set would have contained if it were "fair". The approach uses a hidden variable L to represent the actual class labels. In order to use this L, two assumptions must be made. The first is that L is independent of S and the second is that C is determined by discriminating the L labels using S uniformly at random. Essentially, the first of these assumptions ensures that the actual labels *are* discrimination-free and those labels are not dependent on any one attribute. The second assumption makes it so that every tuple in the data set has an equal chance of being discriminated against.

3.3 Regularization Model

Another approach to the problem of prejudiced algorithms relies on the idea of regularization, a process involving the introduction of additional information to solve certain problems. Kamishima et al. [8] focused on building

regularizers into logistic regression models. In these models, X, Y, and S are random variables that represent class, non-sensitive attributes, and a sensitive attribute, respectively. The conditional probability of a class given both sensitive and non-sensitive features is represented by $M[YjX, S; \Theta]$, where Θ is the set of parameters estimated based on the maximum likelihood principle which maximizes the log-likelihood ($\ell(D; \Theta)$) of each parameter.

Two types of regularizers were introduced, both of which cater to different issues affecting the loss function, a function that models the cost of inaccuracy of predictions. A regularlizer is an addition to the loss function that affects the overall cost. The first regularizer is a simple L_2 regularizer (denoted $k\Theta k_2^2$) which is standard in preventing over-fitting. Kamishima et al. also adopt a second regularizer which uses few resources (denoted $R(D, \Theta)$) to apply fair classification. The loss function we are looking to minimize then becomes $\ell(D; \Theta) + \eta R(D, \Theta) + \frac{}{2} k\Theta k_2^2$ where λ and η are positive parameters that define the level of regularization. The prejudice removal regularizer ($R(D, \Theta)$) grows when individuals are classified based mostly on sensitive attributes, increasing the cost of the predictions. As the cost rises, the algorithm learns to avoid those types of classifications.

4 Case Studies

4.1 AAE Study

In this section, I will analyze a case study performed by Su Lin Blodgett, Lisa Green, and Brendan O'Connor [2]. Using Twitter, the group attempted to improve upon language classifiers, focusing specifically on African-American English (AAE). In their paper, they discuss their methods for identifying AAE, ensure that their approach remains true to linguistic rules, examine existing natural language processing tools to test their classifier against, and develop a corpus of 830,000 tweets associated with the African-American population.

4.1.1 Identifying AAE

The first challenge for the group was gathering data for their study. Twitter seemed the best option as most posts contain instances of casual speech and social media has been proven to be useful in studying language. Still, there are hundreds of thousands of tweets posted and they needed a very specific type. Twitter doesn't have information like a user's race, so other methods had to be used to identify tweets that can be classified as AAE.

Instead of getting the user's information directly from Twitter, a user was assigned a racial group based on geo-location. Census data was used to determine the demographics of the neighborhoods, which made it easier to

place individuals in categories (non-Hispanic whites, non-Hispanic blacks, Hispanics, and Asians). What is interesting about this approach is that the researchers seem to be implementing discriminatory measures to acquire the reprensentation they need for each group. In determining a user's race based on their location and the demographics in that area, the researchers are making assumptions that may or may not be true about the user.

The tweets that were selected for use in the study met certain parameters to ensure that the focus was on conversational text. To that end, tweets containing too many hashtags, the word "follow" or "mention", the string "rt" or "http", retweets, and those by authors with 1000 or more followers were omitted from the data. Users in the data set were associated with a grouping of messages they had posted as well as their calculated demographics (represented as a vector $\pi_u^{(census)}$).

Once they had gathered data, the group analyzed that data in two different ways. The first method was the Direct Word-Demographic Analysis. This method of analysis essentially determines the average demographics per word using the $\pi_u^{(census)}$. Those words with the highest $\pi_{w;AA}$ values were associated with AAE. To construct their corpus, the researchers created a list of the n highest words that occurred at least m times in the data set. Next, tweets by specific users who tend to use the previously identified words were added to the corpus if at least $p\%$ of their posts contained one or more of the words on the list.

This first approach is useful in that it can identify a group of users who use a particular dialect often. However, Direct Word-Demographic Analysis makes use of specific thresholds n, m, and p, and fails to take users who use a dialect infrequently into account. For better analysis, researchers turned to a Mixed-Membership Demographic-Language Model which linked each of the demographic variables (non-Hispanic whites, non-Hispanic blacks, Hispanics, and Asians) to a unigram language model over the vocabulary. The model assumes that a user's mixture of unigram language models will correspond to their demographic weights. It also looks at each message as a separate entity, ensuring that authors who don't frequently use the dialect do not get overlooked. In essence, the model determines topic probabilities for each message and represents them as a vector θ_m.

4.1.2 Existing Language Identifiers

Identifying languages on social media is incredibly complicated. One language can have a number of dialects and short-hands, making it difficult for computers to identify the overall language of the post. In this paper, AAE is considered a dialect of English, so the classification the researchers were looking for was English.

The most popular language identification software, called *langid.py*, was

developed by Lui and Baldwin [11]. This algorithm was trained on a number of languages over a number of different texts, both formal (i.e. Wikipedia) and informal (i.e. Twitter). However, when tested on both AA-aligned and white-aligned tweets, *langid.py* was approximately 5.6% more likely to classify AA-aligned tweets as non-English. These results were compared with the results of Twitter's two language identifiers whose predictions could be found in a tweet's metadata. *Twitter-1*, which does not consider missing values to be a non-English prediction, was about 2.5% more likely to identify the language of AA-aligned tweets as non-English. *Twitter-2*, which considers missing values to be non-English predictions, returned more non-English predictions for both AAE and white-aligned tweets, but 6.8% more for AAE than for white-aligned.

	AAE	**White-Aligned**
langid.py	13.2%	7.6%
Twitter-1	8.4%	5.9%
Twitter-2	24.4%	17.6%

Table 4: % tweets in corpora classified as non-English [2]

Of tweets classified as non-English, most were found to be different dialects of English and thus misclassified. Additionally, it was found that an individual message was more likely to be classified as non-English if it had a higher probability of being an AA-aligned tweet according to the intial analysis.

4.1.3 Ensemble Solution

Most of the messages for which the model predicted a high probability of AAE were, in fact, written in English. The paper proposes a solution referred to as the Ensemble Classifier which works with *langid.py*. For each message $\vec{w}$, the demographic-language proportions $\hat{\theta}$ are predicted by the trained model. The prediction of *langid.py* $\hat{y}$ is calculated and, if it is English, the message is accepted as English. If not, the classifier only returns a non-English $\hat{y}$ decision if the AA, Hispanic, and White posterior probabilities calculated in $\hat{\theta}$ are less than 0.9 for each of the message's tokens.

The Ensemble Classifier was tested on 2.2 million geolocated tweets sent in the U.S. in 2014, all of which were outside of the training set. The researchers confirmed that the classifier was able to correct a number of false negatives for AAE messages. Additionally, since the classifier successfully identified all tweets that were high AA and high White (a message's posterior probability of using AA- or White-associated terms was above 0.8) as English for 100% of high AA or high white tweets tested, all high AA

or high white tweets were assumed to be English in measuring recall. The recall, then, for the Ensemble Classifier was estimated to be $\frac{n+m}{N}$ where m is the expected number of correctly changed classifications (non-English to English) by the Ensemble Classifier and n is the precision, estimated as 1.0. The differences in recall for high AA and high White tweets observed using *langid.py* was solved by the classifier.

While the data in this study was acquired in ways that were not quite unbiased, it is an excellent example of proper data-handling once data has been collected. The researchers were very specific in the data points they dropped and those they kept, explaining their reasoning for these decisions. The care taken to clean the data ultimately led to a better classifier.

4.2 Bias and Fairness in Authorial Gender Attribution

In 2017, Koolen and van Cranenburgh [9] looked into the ways in which existing classifiers use stereotypes to discern authorial gender in texts. In their study, they used two datasets of Dutch novels. The first, which I will refer to as the Riddle corpus, contains 401 Dutch-language novels published between 2007 and 2012 that were bestsellers or that were most often lent from libraries between 2009 and 2012. It is 46.4% suspense novels, 36.9% general fiction novels, 10.2% romance novels, and 6.5% other genres. There are also approximately the same number of female and male authors in the corpus as well as a small percentage of "other" gender which accounts for duos of differing gender and unknown or nontraditional gender. This corpus notably contains both untranslated and translated novels.

The second corpus, called the Nominee corpus, is a set of fifty novels nominated for one of two well-known literary prizes in the Netherlands. All novels in the Nominee corpus are not present in the Riddle corpus. Additionally, all are original Dutch-language novels of the same genre. This second data set is significantly smaller than the first as there are many more male authors than female authors nominated for the awards. In order to combat this skewness, the data set had to be reduced in size so that both male and female authors were equally represented.

4.2.1 Experiments with LIWC

To extract gender differences, the researchers used Linguistic Inquiry and Word Count, a text analysis tool used mostly for sentiment mining [13]. The tool looks at categorized word lists (i.e. personal) and calculates the frequency of each word list in a given text.

In the Riddle corpus, it was found that 48 out of 66 LIWC categories exhibit significant difference between male and female authors. The researchers concluded that gender stereotype-confirming differences were present,

independent of genre. Interestingly, female authors scored higher in categories such as Affect, Pronoun, Home, Body, and Social while male authors scored higher in categories such as Articles, Prepositions, Numbers, and Occupation. There was one result that countered gender stereotypes as female authors tended to score higher on Cognitive Processes, a category most closely associated with science-fiction.

The authors point out that the results from the Riddle corpus should not be taken at face value as a number of factors were not considered. The first of these is whether or not the text was translated. Translated texts can use different words than the author intended, leading to changes in gender prediction. Another factor is the existence of subgenres, some of which are famously written by one gender or another. Lastly, a writer's readership was not considered in the data set. As popular writers tend to write for a specific audience, they may use language more closely associated with one gender.

The Nominee corpus proved a bit more challenging to study. Its size compared to the Riddle corpus made it difficult to compare the LIWC scores as the comparison used a *t-test* which compared the mean of each group. To avoid this problem, the researchers chose two categories from the Riddle corpus for which the difference in means from the Nominee corpus is trivial, Anger and Occupation. When graphed, the overlap between the two corpora was almost perfect, indicating that the aformentioned confounding variables might have contributed to the differences between the Riddle and Nominee scores. Additionally, it was observed that in certain categories differences between individuals were more significant than differences between group means. For example, in the Body category there were a number of male authors who used more Body words than female authors as well as a number of male authors who used fewer Body words than female authors. The authors suggest that perhaps comparing the means of groups is not the best way to draw conclusions.

4.2.2 Machine Learning Experiments

After their initial examination of the data using LIWC, Koolen et al. [9] developed a classifier that predicted an author's gender with reasonable accuracy using features that could not be easily tied to either gender classification. This model was unable to confirm the results in the previous section, likely because of the small sample and the fact that the model is affected by unique aspects of the data set.

Instead of the predictive model, the researchers decided to use a descriptive model to analyze gender differences. A "topic model" was developed with 50 topics. Next, the mean topic weights for texts of each gender were compared. The results of this model show that the largest differences are be-

tween texts containing gender stereotype-confirming topics such as military (stereotypically associated with males) and settling down (stereotypically associated with females). This result was expected as some texts are going to be extreme in their differences. Perhaps more interesting is that the topics that were found to be the most similar also contained gender stereotype-confirming topics as well. The topic 'looks and parties' might be expected to have a higher topic score for females than males, but both genders scored similarly according to the model.

This study was an excellent example of a case study in general. The authors explained the data they acquired in great detail as well as the motivations for their experiments. Confounding variables were discussed rather than glossed over and the results of the experiments were analyzed at the end of the paper.

4.3 E ects of Job Testing on Minority Workers Study

In 2008, Autor and Scarborough [1] examined the impact of job testing on minority workers in the private sector. Their study used application, hiring, and employment outcome data from a large firm in the private sector with outlets in 47 states in the U.S. The data from multiple outlets was comparable because the sites all employed similar workers who performed similar tasks and offered similar products. Each outlet employed 10 to 20 workers, ages 18 to 30, who held their jobs for an average of a little under 6 months. About 70% of the hires are White (non-Hispanic), 19% are Black (non-Hispanic), and 12% are Hispanic.

In June of 1999, the firm began to use electronic application kiosks which asked applicants to complete a questionnaire. One aspect of the electronic process is a personality test that looks at five of what psychologists view as core traits. Employers receive a report regarding an applicant's customer service test score which places applicants in a "red" group (lowest quartile), a "yellow" group (second lowest quartile), or "green" group (two highest quartiles). According to the data, a higher percentage of white workers were placed in the "green" group than either Black or Hispanic workers. Additionally, it was found that a higher percentage of male applicants were hired.

Most of the data is taken from company personnel records, containing demographic information, date of hire, date of termination, and reason for termination for each worker hired during the sample period. The sample contains employment data from January, 1999 through May, 2000, allowing for conclusions to be drawn as to whether or not the electronic job testing negatively impacted minority workers. Each item in the data set has a flag that shows whether or not the data was collected before testing was implemented. It should be noted that only employment records (and not

application records) are available for workers hired after the introduction of job tests. Additionally, observations containing incomplete gender or race reporting were dropped from the sample. This missing data could be significant as it is more likely that applicants who incompletely reported race or gender are part of minority groups which might have been affected by the job testing.

In the end, the sample contains the records of 33,924 workers hired, 25,561 hired without job testing and 8,363 hired after receiving a test. Because wage data was omitted, wage variation was handled by controlling for year and month of hire as most of the positions paid minimum wage. Analysis of applicant test scores showed race gaps in test scores, as predicted by the researchers.

4.3.1 Equality-E ciency Trade-O

The so-called *equality-efficiency trade-off* is defined by Autor and Scarborough [1] as the situation wherein productivity gains from testing come at the cost of fewer minority hires. This measure is based on the discrepancy between information employers receive from the job tests and information employers can gain through normal hiring practices. This discrepancy is referred to as the *relative bias* of tests. To determine whether or not there is an equality-efficiency trade-off, both the hiring rates and the productivity must be examined.

One indicator of an equality-efficiency trade-off is a *hiring gap*, or the difference between the hiring rates of majority and minority workers. This measure is extremely similar to the definition Calders and Verwer gave for their discrimination score in their paper on the three Naive Bayes approaches to reducing algorithmic bias. Here, the hiring rate of a group of applicants x who have received an interview is

$$E\ [Hirejx] = 1 \quad \phi(z_I(x)). \tag{1}$$

The hiring rate of group x applicants who have received an interview and the test is

$$E\ [Hirejx] = 1 \quad \phi(z_S(x)). \tag{2}$$

Without the use of the job tests, the hiring decision (denoted by 0 or 1) is entirely based on the firm's interview process and information gathered about applicants in the traditional way and the resulting hiring gap is denoted by γ_I:

$$\Delta\gamma_I \quad E\ [Hirejx_1] \quad E\ [Hirejx_2]. \tag{3}$$

With the tests, however, the employer takes both the interviews and the test results into consideration and makes a hiring decision based on all of the available information and the resulting hiring gap is denoted by γ_T:

$$\Delta\gamma_I \quad E_{;S}[Hirejx_1] \quad E_{;S}[Hirejx_2]. \tag{4}$$

The effect of testing on the hiring gap is defined by the following formula:

$$\Delta\gamma \quad \gamma_T \quad \gamma_I. \tag{5}$$

Another indicator of an equality-efficiency trade-off is a *productivity gap*, or the difference between the mean productivity of majority workers and the mean productivity of minority workers. Testing affects productivity of workers through selectivity and screening precision. When there is an increase in selectivity (i.e. a *decrease* in hiring) for group x, the productivity of group x increases. An increase in screening precision leads to more accurate estimates of worker productivity, which generally leads to better hires for all groups.

The mean productivity for a group of workers x if only interviews are used (π_I) is

$$E\ [yjHire = 1, x] \quad \mu_0(x) + \sigma_0\rho_I\lambda(z_I(x)). \tag{6}$$

Comparatively, the mean productivity for a group of workers x if both tests and interviews are used (π_T) is

$$E_{;S}[yjHire = 1, x] \quad \mu_0(x) + \sigma_0\rho_T\lambda(z_T(x)). \tag{7}$$

The effect of testing on the productivity gap is defined by the following formula:

$$\Delta\pi \quad \pi_T \quad \pi_I. \tag{8}$$

4.3.2 Simulation Procedure

In order to determine whether or not an equality-efficiency trade-off was present in the data acquired from the retailer, the authors simulated the hiring process for each applicant and calculated job spell durations based on ability and averages. They did this once for interview-based hiring and once for test-based hiring before comparing race composition and productivity of hires under both circumstances. This process was repeated six times, allowing the researchers to look at different situations (i.e. interviews were biased toward White people, interviews were biased toward Black people, tests were biased towards one group or the other, etc.).

4.3.3 Results

Prior research suggests that job testing improves selection at the expense of minority hiring, meaning an equality-efficiency trade-off is present. From their study, Autor and Scarborough [1] conclude that both the job test and

the interview were unbiased and that there was no equality-efficiency trade-off as a result. Essentially, as long as job tests remain unbiased relative to the interviews they augment, there will not be a measurable equality-efficiency trade-off.

For the firm being studied, it was found that job testing improved productivity and increased the length of time employees stayed with the firm by 10% to 12%. In addition, tests lowered frequency of terminations for cause and did not change the racial composition of hires in any detectable way. The job test and screening interviews examined were both found to be unbiased by the authors.

Readers must take into consideration the fact that the study looks at only one, albeit large, retailer. Though a competitive economy suggests that retailers will behave similarly, this cannot be empirically stated. Too, the difference in test scores between groups is smaller for the job test being studied than for more rigorous job tests like those used by the military. When those more rigorous tests are examined, it is possible that disparate impact will become more obvious.

5 Experimental Setup

5.1 Algorithms

I used the same data used in the Autor and Scarborough paper, focusing on the columns for store site, race, gender, birthdate, hiring decision, position, zip code, and customer service test score. The data set only includes two genders (male and female) and three races (White, Black, and Hispanic). I then compared those results to those of three different data sets. Before running any of the experiments, the data had to be cleaned and preprocessed in certain ways. All data sets were preprocessed in the same ways to allow for accurate comparisons to be made between algorithms. The consistently cleaned data could be passed to the algorithms as input where it would be split into training and test sets depending on the algorithm itself.

5.1.1 Preprocessing

Much of the data I used in my experiments was the same data used by Evan Hamilton [6] in his experiments. Hamilton found it difficult to compare algorithmic results because of the different data cleaning schemes and so developed his own scheme that is in essence a modified version of Zafar's cleaning scheme. To avoid experiencing the same difficulty, I chose to use the cleaning scheme Hamilton developed on all data sets so that all algorithms would be receiving the same input. Using Hamilton's cleaning scheme, each data set had attributes, sensitive attributes, attributes to ignore, and a class

attribute already defined. Those were pulled out at the start of the cleaning process and saved in order to preserve them. Next, the data was loaded from a CSV and translated into a pandas dataframe. Rows with missing data were removed and will be discussed in detail per data set in Section 5.4. In the preprocessing stage, dimensionality of features was reduced and converted to binary integer values via one-hot encoding, also discussed in greater detail in Section 5.4. All of the data looked looked the same before it was passed to each algorithm. Each data set was cleaned and output to a CSV that could be read by any of the algorithms, with all variables encoded such that each column contained only integers or floats. The data sets contained both continuous numerical data and binary numerical data. All data sets were input to all algorithms as numerical data with categorical variables one-hot encoded so that string attributes were converted to integers.

5.1.2 Calders' Two Naive Bayes

The Calders algorithm required a sensitive attribute along with training and test sets as input. Training and test data is split by sensitive feature into sensitive and non-sensitive groups as discussed in Section 3.2. A standard naive Bayes classifier is trained on each group and used to predict that group's results. Following prediction, a Calders-Verwer *discrimination score* is calculated as explained in Section 6.2. If the score is above zero, the algorithm perturbs the training data in one of two ways based on the number of estimated positive values versus actual positive values. It may be possible to expand this algorithm to work with other metrics besides the Calders-Verwer score (e.g. Disparate Impact score).

The model is retrained with the new data and the process is repeated until the discrimination score reaches zero. Separating the data by sensitive feature ensures that the sensitive variable is removed from the classification process and the calculated changes to the data keep the total number of positive classifications and the total number of sensitive and nonsensitive individuals relatively stable.

5.1.3 Feldman's Repair Scheme

Feldman's scheme [5] takes place mostly in the preprocessing of the data. It relies on the ordering of numeric and categorical features to find the median value for a feature. Additionally, Feldman's repair scheme removes all sensitive features from the data before training a model or making any classifications, preventing a model from learning from the sensitive attributes. It takes several additional parameters not used by the other algorithms such as a repair level between zero and one, a class attribute, attributes to ignore, and a sensitive attribute.

A higher repair value leads to more perturbation of the data. As the repair level gets higher, the algorithm relabels protected individuals as unprotected individuals, storing the original data and sensitive features for later comparisons. The repair scheme takes the sensitive attribute as well as any attributes to ignore as input and increases fairness using the sensitive feature. Then, a model is trained on the repaired data. The predictions for the model are compared with the original data when the metrics are calculated.

5.1.4 Kamishima's Regularization Model

As discussed in Section 3.3, I began by implementing Kamishima's regularization model [8] which takes only numerical data and an eta value. This model builds two regularizers into a standard logistic regression model. The first of the two regularizers is in place to prevent over-fitting, but the second is much more relevant to the topic at hand. This regularizer adds to the loss function as individuals are classified based on the sensitive attribute, increasing the cost for that classification so the algorithm will not repeat it. The algorithm takes a parameter eta which represents the amount of accuracy one is willing to trade for increased fairness. At higher eta values, the algorithm tends to fail on smaller data sets as the group ratios grow smaller.

The Kamishima algorithm required data to be numeric, necessitating the conversion of strings to integers in the preprocessing of the data. I was also unable to get it to work on data sets without a binary sensitive attribute. Another problem with the Kamishima algorithm was its parameter. At different eta values, different levels of fairness are achieved. I added to the algorithm my own code that auto-tunes the variable so the algorithm can be run as is. It works very similarly to the Calders algorithm in that it simply repeats to maximize fairness.

There are three variations of my maximization/minimization algorithm. The first variation finds the eta value that gives results such that $min(\frac{j1 \quad DIj}{acc})$. In this case, we are looking for the value that maximizes accuracy and minimizes the distance between DI and 1, giving us a fair algorithm that is also accurate. The second variation maximizes accuracy only and the third maximizes fairness based on the DI score by minimizing the distance between DI and 1 ($j1 \quad DIj$). My maximization/minimization algorithm finds the score for the midpoint and then checks the scores at the minimum and maximum of the range of eta values. If the midpoint's score is greater than or less than the endpoint's score, the function is called again over whichever half maximized or minimized the appropriate score. This eventually finds the optimal score as the scores converge at a maximum or minimum over all values of eta (1 to 1000 in increments of 1).

Tables 5-8 show the eta values the algorithm chose for each variant per data set. Figures 1-12 show the scores per eta value for each data set and each variant of the algorithm. In each case, the algorithm minimized or maximized the appropriate score and chose the eta value that corresponded with that score. The graphs do not show all eta values up to 1000 as the score flattens well before that point. All graphs show up to the point at which the lines flatten.

	$\frac{j1 \quad DIj}{Acc}$	Acc	$j1$	DIj
1	68	1000		70
2	72	1000		55
3	67	1000		69
4	51	1000		68
5	54	1000		47
6	56	1000		51
7	71	1000		70
8	53	1000		68
9	67	1000		52
10	68	1000		54

Table 5: Adult Eta per Run

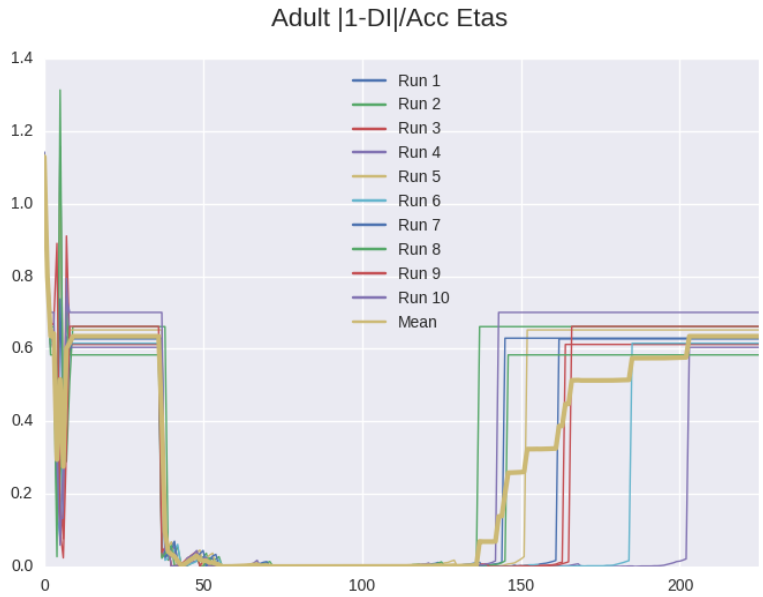

Figure 1: Eta values for first variant of Kamishima algorithm on Adult data

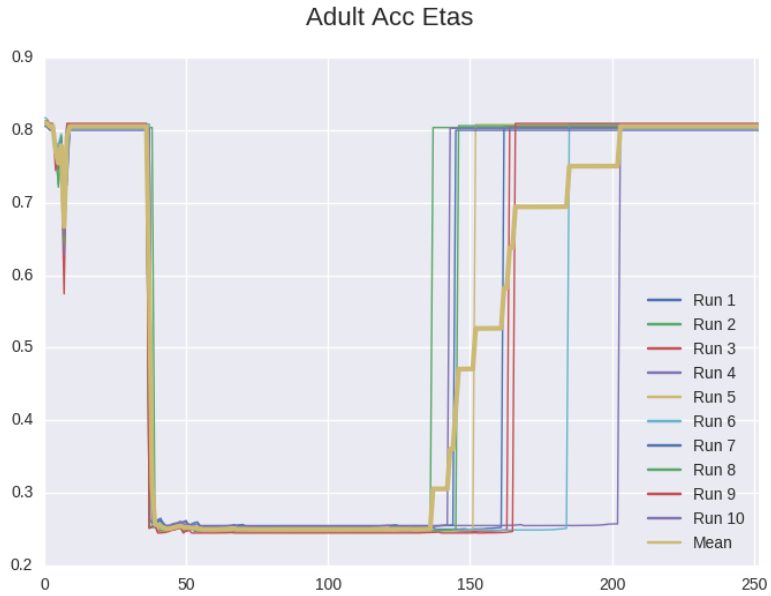

Figure 2: Eta values for second variant of Kamishima algorithm on Adult data

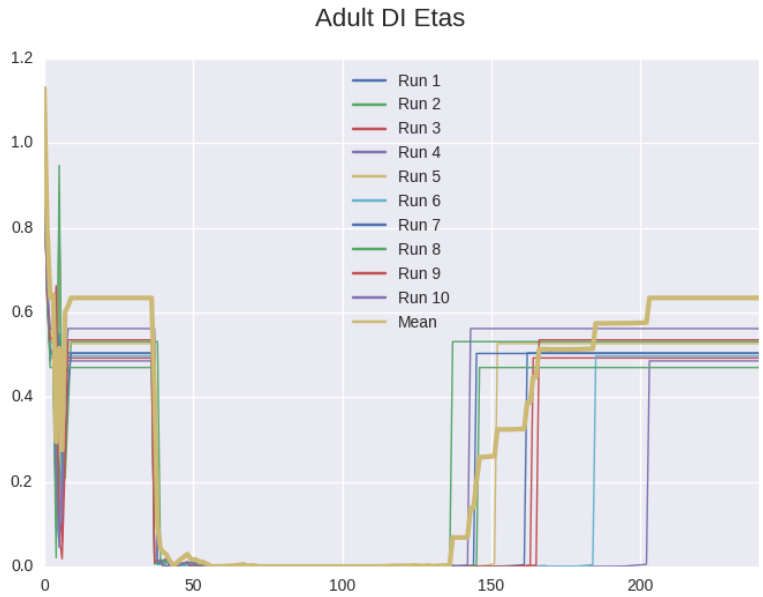

Figure 3: Eta values for third variant of Kamishima algorithm on Adult data

	$\frac{j1 \quad DIj}{Acc}$	Acc	$j1$	DIj
1	5	1000	7	
2	5	1000	1	
3	6	1000	5	
4	5	1000	6	
5	5	1000	1	
6	4	1000	3	
7	6	1000	6	
8	5	1000	5	
9	5	1000	7	
10	3	1000	5	

Table 6: German Eta per Run

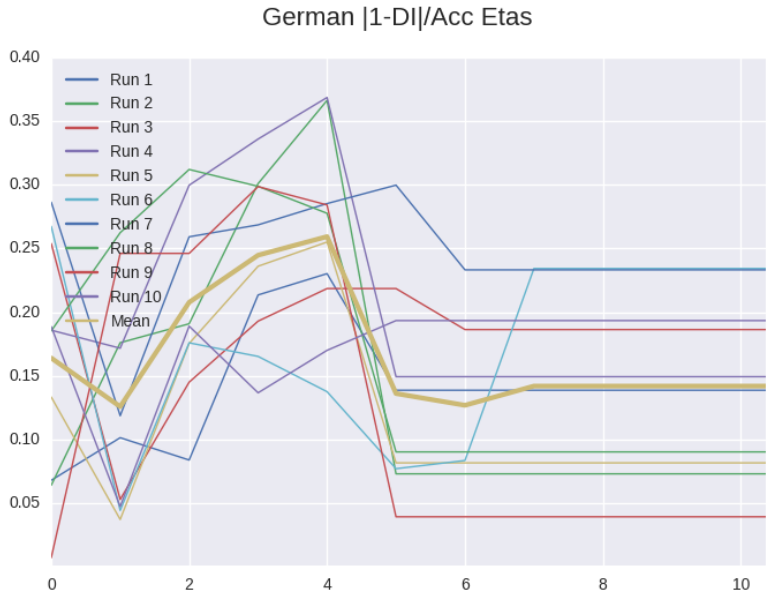

Figure 4: Eta values for first variant of Kamishima algorithm on German data

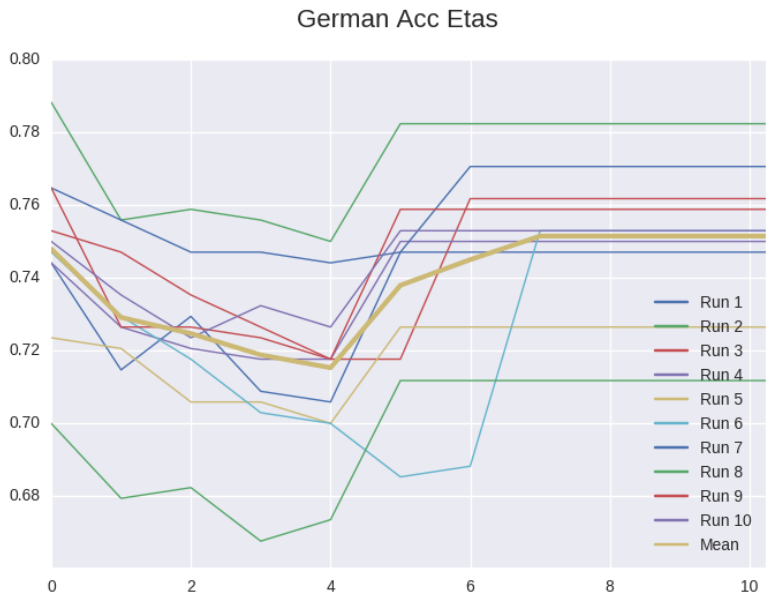

Figure 5: Eta values for second variant of Kamishima algorithm on German data

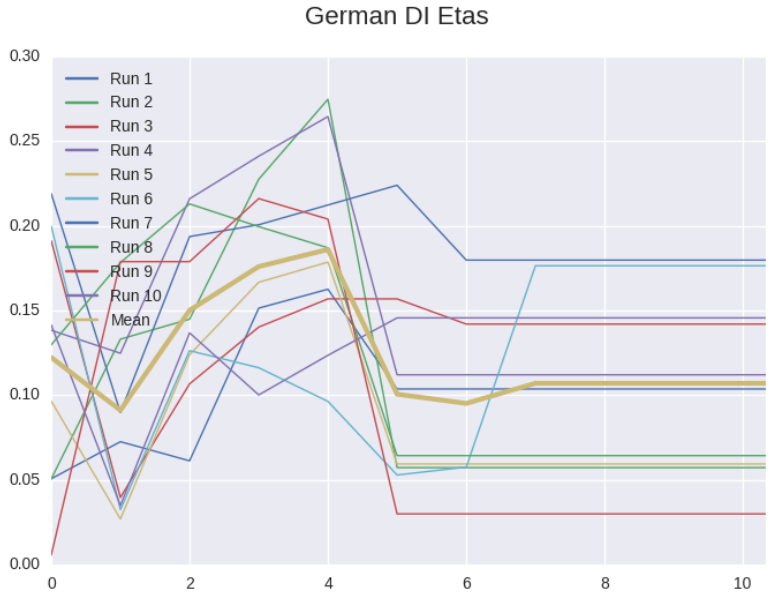

Figure 6: Eta values for third variant of Kamishima algorithm on German data

	$\frac{j1 \quad DIj}{Acc}$	Acc	$j1$	DIj
1	1	1000	1	
2	1	1000	1	
3	1	1000	1	
4	1	1000	1	
5	1	1000	1	
6	1	1000	1	
7	1	1000	1	
8	1	1000	1	
9	1	1000	1	
10	1	1000	1	

Table 7: Retailer Eta per Run

In the case of the retailer data, the graphs demonstrate that there is no measurable difference in score when the eta value is changed. I expect this has to do with the size of the data set in that eta represents the amount of accuracy one is willing to trade for fairness. With data sets as large as this, trading any amount of accuracy will yield the same results.

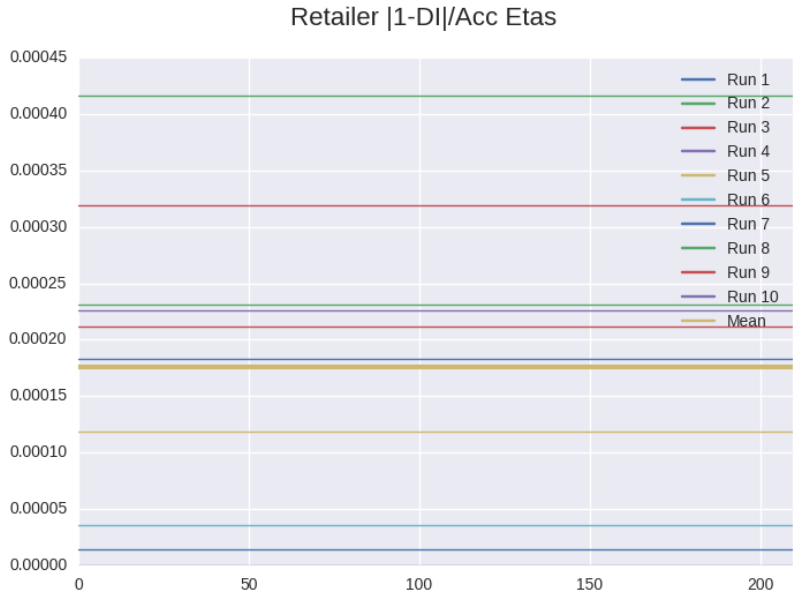

Figure 7: Eta values for first variant of Kamishima algorithm on Retailer data

Figure 8: Eta values for second variant of Kamishima algorithm on Retailer data

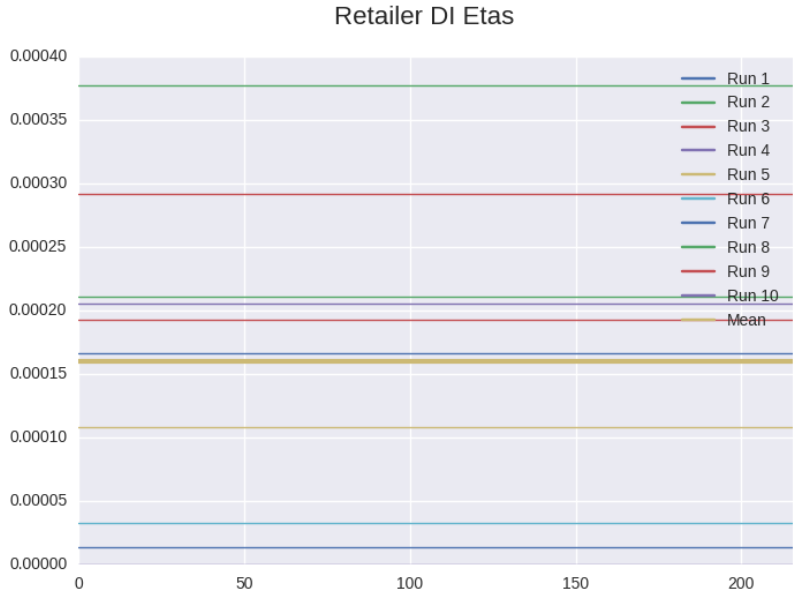

Figure 9: Eta values for third variant of Kamishima algorithm on Retailer data

	$\frac{j1 \quad DIj}{Acc}$	Acc	$j1$	DIj
1	43	50	4	
2	6	50	9	
3	7	50	10	
4	2	50	1	
5	3	50	12	
6	48	50	2	
7	18	50	3	
8	11	50	6	
9	2	50	2	
10	13	50	3	

Table 8: Ricci Eta per Run

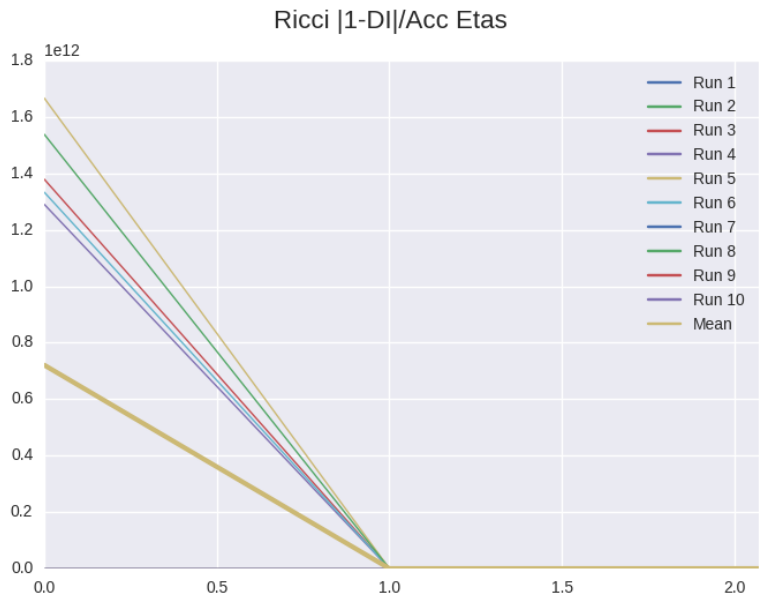

Figure 10: Eta values for first variant of Kamishima algorithm on Ricci data

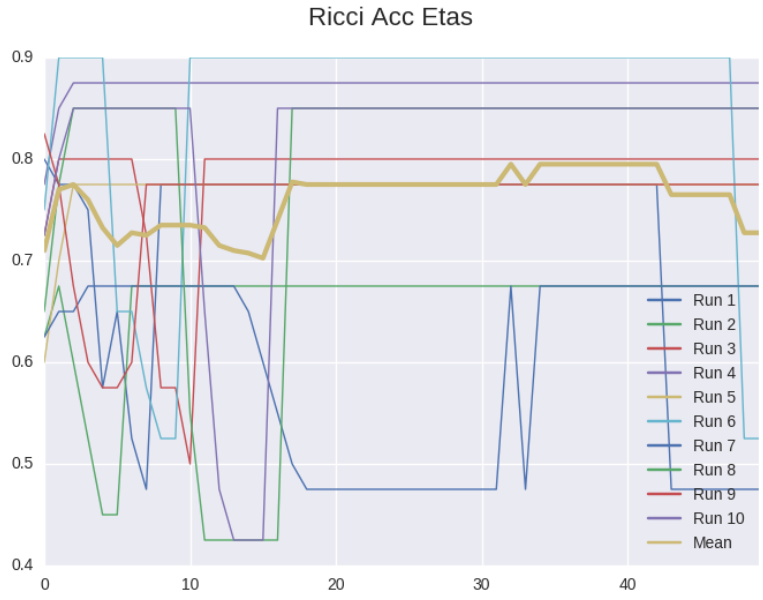

Figure 11: Eta values for second variant of Kamishima algorithm on Ricci data

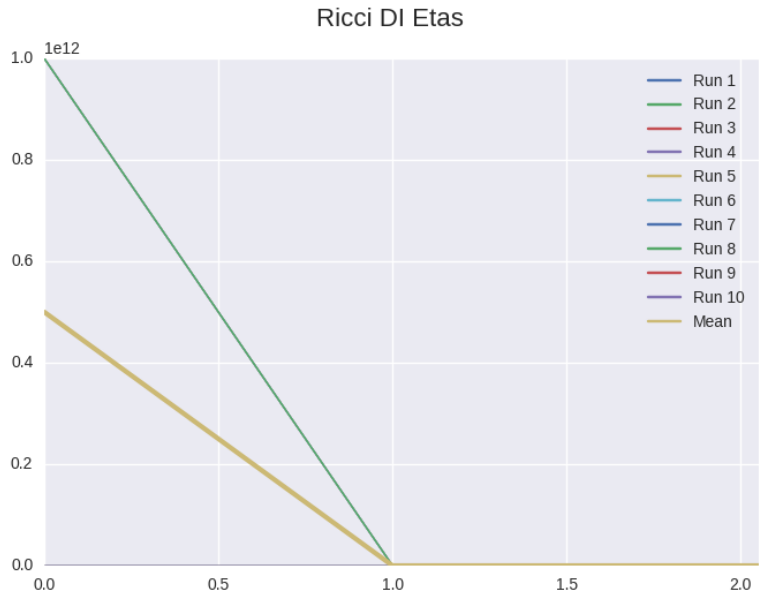

Figure 12: Eta values for third variant of Kamishima algorithm on Ricci data

5.1.5 Zafar's Fair Logistic Regression Classi er

The Zafar classifier [14] is similar to Kamishima's in that it takes X and y where X is basically the data with the sensitive feature and classification column removed and y is that classification column. The sensitive feature is passed into the algorithm as $x\_control$, a dictionary with the feature as a key and its values *in order* as values. All of these variables must be kept in order as data items are referred to with an index that should remain the same across X, y, and $x\_control$.

The classifier takes a gamma value that acts as an inverse to Kamishima's eta with higher values corresponding to a lower level of acceptable accuracy loss for increased fairness. To run the algorithm without constraint, gamma is set to None. In addition to gamma, the algorithm takes values that determine which variant of the algorithm to use. There are three variations of the algorithm, each focusing on a particular thing. When Zafar's algorithm is run with only fairness constraints applied and no gamma value, it maximizes fairness. When accuracy constraints are applied, accuracy is similarly maximized. Zafar chose a suitably small gamma (0.5) that maximizes accuracy. Finally, the application of the *sep_constraints* variable ensures a different gamma for each individual to minimize the number of false positives in the data.

I ran Zafar's algorithm four times on each data set, once unconstrained, once with fairness constraints applied and a specified sensitive attribute, once with accuracy constraints applied and a gamma of 0.5, and once with the third variant. The different gamma values were used to compare the results of the algorithm with differing levels of acceptable losses in accuracy in the same way different eta values were used in the Kamishima tests and were chosen by Zafar to maximize either fairness or accuracy and to eliminate false positives.

Originally, I planned to auto-tune gamma in the Zafar algorithm as I auto-tuned eta in Kamishima's. However, it soon became clear that each variant specified in Zafar's paper [14] had gamma values associated with it (described above) that maximized fairness or accuracy and minimized the number of false positives. In the end, no auto-tuning was necessary for this algorithm.

5.2 Metrics

Six metrics were used to measure the performance of each of the algorithms. Using all of these measures, it is possible to compare the stability of an algorithm across data sets and to compare the overall performance of the algorithms in each metric category. Each metric took the actual classification values denoted as A, predicted classification values denoted as P, and protected values denoted as X as input. Total number of sensitive and non-

sensitive individuals as well as true negatives, false negatives, true positives, and false positives were extrapolated from that data. A true positive or negative occurs when the actual and predicted labels are the same whereas a false positive or negative occurs when the classification model labels incorrectly. For each metric and algorithm, the standard deviation was also calculated. Accuracy was determined based on the actual versus predicted labels for each data set while fairness was determined based on the number of those in the protected group who got the positive classification versus the number of those in the nonprotected group who got the positive classification. The simplest measure is time and serves to highlight the faster algorithms.

In the following subsections, I will denote the number of nonprotected individuals with a predicted negative classification with NP_{neg}, the number of nonprotected individuals with a predicted positive classification with NP_{pos}, the number of protected individuals with a predicted negative classification with P_{neg}, and the number of protected individuals with a predicted positive classification with P_{pos}. The total number of people will be denoted T with T_p and T_n representing the total number of protected and unprotected individuals respectively. True positives and negatives will be denoted V_p and V_n while false positives and negatives will be denoted F_p and F_n.

5.2.1 Accuracy

Accuracy is measured using sklearn's *accuracy_score* function which computes subset accuracy. The function calculates a Jaccard similarity coefficient score

$$J(A, P) = \frac{jA \setminus Pj}{jA \cup Pj} = \frac{jA \setminus Pj}{jAj + jPj \quad jA \setminus Pj}$$

The more similar the model's predictions to the original classifications, the higher the accuracy for that algorithm and the closer to 1 the score gets.

For example, if $A = [1, 1, 0, 1]$ and $P = [0, 1, 0, 1]$, we have

$$J(A, P) = \frac{3}{4 + 4 \quad 3} = 0.6$$

Whereas if $A = [1, 1, 0, 1]$ and $P = [1, 1, 0, 1]$, we have

$$J(A, P) = \frac{4}{4 + 4 \quad 4} = 1.0$$

5.2.2 Balanced Classi cation Rate

Balanced Error Rate is based on the confusion matrix. It measures the average proportion of incorrect classifications for a class. The Balanced Classification Rate is simply the opposite of BER and is calculated extremely simply $(1 \quad BER)$.

Let $R_{NP} = \frac{NP_{neg}}{T}$ and $R_P = \frac{P_{pos}}{T}$. Then, we have that

$$BCR = 1 \quad \frac{R_{NP} + R_P}{2}$$

5.2.3 Calders-Verwer Score & Disparate Impact Score

Given the variables described above, we have that

$$CV\_score = (\frac{P_{pos}}{T_p} \quad \frac{NP_{pos}}{T_n})$$

The closer this score is to zero, the less discrimination is present in the data. In the Results & Analysis section, the column with the CV score results is actually 1 CV_score to make it easier to compare fairness values in that all columns are trying to maximize values.

The Disparate Impact score is calculated similarly using the variables described above. Additionally, let $R_P = \frac{P_{pos}}{T_p}$ and let $R_{NP} = \frac{NP_{pos}}{T_n}$. Then, we have that

$$DI\_score = \frac{R_P}{R_{NP}}$$

The closer the DI score is to one, the less discrimination is present in the data. Scores below one indicate discrimination in favor of the unprotected group and scores above one indicate discrimination in favor of the protected group. So, when the majority group is more likely to get jobs than the minority group, the DI score will be less than one. If the algorithm has been over-adjusted for fairness and is actually giving preference to the minority group, the DI score will be greater than one.

5.3 Matthews Correlation Coe cient

The Matthews Correlation Coefficient (MCC) is a measure of how well a classification model is working. Scores closer to 1 are the best.

$$MCC = \frac{V_p(V_n) \quad F_p(F_n)}{\sqrt{(V_p + F_p)(V_p + F_n)(F_n + V_n)(F_p + V_n)}}$$

For example, suppose that $A = [1, 1, 1, 0, 0, 1, 0, 1, 0, 1]$ and that $P = [0, 1, 0, 1, 0, 1, 1, 0, 0, 0]$. Then, we have $V_p = 2$, $V_n = 3$, $F_p = 1$, and $F_n = 4$. Consequently,

$$MCC = \frac{2(3) \quad 1(4)}{\sqrt{(2+1)(2+4)(4+3)(1+3)}} = \frac{2}{504} = 0.089$$

Whereas if $A = [1, 1, 1, 0, 0, 1, 0, 1, 0, 1] = P$, we have $V_p = 6$, $V_n = 4$, $F_p = 0$, and $F_n = 0$. In this case,

$$MCC = \frac{6(4) \quad 0(0)}{\sqrt{(6+0)(6+0)(0+4)(0+4)}} = \frac{24}{576} = 1.0$$

5.3.1 Time

In addition to the different measures of fairness, the run time for each algorithm was also measured. A time was taken when the algorithm started running and saved. Then, when the algorithm finished, it subtracted the start time from the current time to get the run time of the algorithm for each test. All experiments were run on a machine with an average CPU speed of 3700 MHz and 128905 MB of available memory. Generally, shorter run times are preferred, but there might be tradeoffs that will be discussed in the Results & Analysis section.

5.4 Experiments

Experiments were run on four different data sets and the results were averaged over 10 runs. Each algorithm received the same cleaned data as detailed in the Preprocessing section and metrics were calculated for each algorithm's results per data set.

5.4.1 Adult Census Data

The Adult census data [1] [10] is used in a number of papers on fairness-aware machine learning and has been analyzed by multiple authors. As a result of the commonality of its use, it is much easier to compare results from this data set across algorithms and papers. The data was originally taken in a 1994 census. There are 30,162 items in the data set, each with fourteen features, which are either numeric or categorical. About 14.0% of that data set were in the protected group while approximately 86.0% of the data came from individuals in the unprotected group. Also important to note is the 25.0% success rate for the entire data set, though the protected group only had a 15.8% success rate while the unprotected group had a 26.4% success rate. For the experiments on this data set, I used a 0.66 split over 10 runs.

18,682 of the original 48,844 items were missing data and were removed from the data set. Dimensionality of the attribute *native_country* was reduced to two values: *United-States* or *Non-United-States*. The first thing to do was to combine those of the same country of origin into one variable. Then, the values were converted to integer values of 0 for the protected group and 1 for the unprotected (white) group. The sensitive attribute for the census data is gender with females marked with 0s and males marked with 1s. The class attribute for this data set is yearly income and an income less than or equal to $50,000 is denoted with a 0 while incomes greater than $50,000 are denoted with a 1.

[1] https://archive.ics.uci.edu/ml/datasets/Adult

5.4.2 German Credit Data

The German Credit data set [2] came from the same repository [10] as the Adult census data and is also widely used in papers exploring fairness-aware machine learning. It is not as commonly used as the Adult data and the German data set is significantly smaller with only 1,000 items. Each item represents an individual classified by a German bank in 1994 as a credit risk or not. Credit, then, is the class attribute where 1 denotes good credit and 0 denotes bad credit. I use an encoded version of the original data set produced by Strathclyde University in which all values are numerical. The numerical version of the data was also used in Hamilton's paper [6], though it is not used in every paper that cites this data set, which further complicates analysis.

There are 1,000 items in the data set with none removed due to missing data. Gender is chosen as the sensitive attribute and the data set includes data for 690 men (69.0% of the data) and 310 women (31.0% of the data). In the original data, gender and marital status were combined in one column (ex: 'male: divorced/separated'). The numerical version of the data separates gender and marital status into separate columns and marks males with 1 and females with 0. The overall success rate was 70.0%, but it was significantly lower for females at 64.8% than males at 81.8%. For the experiments on this data set, I used a 0.66 split over 10 runs.

5.4.3 Retailer Data

This data set was taken directly from Autor and Scarborough [1]. It started as described in Section 4.3. However, in order to run the algorithms on this data set, I used Hamilton's one-hot encoding data cleaning scheme to encode all feature values as numeric values. Since not every column in the data set was necessary for my exploration of the data, I focused on columns containing data on the store site, race, gender, zip code, hiring test score, position, and hiring outcome. Initially, the retailer data did not have an 'age' column. It had instead a date of birth column from which age was extrapolated through simple subtraction from the hiring date in preprocessing. The age column then represents a candidate's age as of the date they were hired. Both race and gender could be used as sensitive features in this data set, but I chose to focus on race such that the protected group has a race value of 0 and the unprotected group has a race value of 1.

There were a total number of 824,821 items in the data set (232,688 items were removed due to missing data), 446,469 in the majority group (54.0%) and 378,352 in the minority group (46.0%). The total hiring rate was extremely low at 6.00% and lower for members of the minority group at

[2] https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

5.10% than members of the majority group at 7.50%. For the experiments on this data set, I used a 0.66 split over 10 runs.

5.4.4 Ricci Data

Data from a case *Ricci v. DeStefano* regarding disparate impact of firefighters' promotion exams in New Haven, CT [12] was consolidated in this data set. It contains 118 items, 68 in the majority group (58.0%) and 50 in the minority group (42.0%). The overall passing rate for the promotion exam was 47.0%, though it was lower for the minority group at 30.0% than the majority group at 60.3%. For the experiments on this data set, I used a 0.66 split over 10 runs.

There are six columns: Position, Oral, Written, Race, Combine, and Class. Class was not in the original data, but was extrapolated from it depending on whether the firefighter passed the test with a combined . A 1 in the Class column denotes a passing score while a 0 denotes a failure. A 1 in the Position column denotes a Captain while a 0 denotes a Lieutenant. The Oral and Written columns contain scores on the oral and written parts of the promotion exam for each firefighter while Combine is made up of total test scores. Race is the sensitive feature and marks a firefighter in the protected class with a 0 and a white firefighter with a 1.

6 Results & Analysis

Each of the data sets is represented by a table with rows representing different algorithms. The values in each column representing a metric are an average of 10 separate runs. Similarly, the standard deviation column values were the standard deviation for a specific metric taken over 10 runs.

6.1 Adult Data Results

Table 9 shows the results per metric for each of the different algorithms and their variations along with the standard deviation for each metric per algorithm. Because of its consistently low accuracy scores, we will disregard the Zafar algorithm when considering the best algorithms for maximizing fairness and efficiency.

Looking at the data, the standard logistic regression algorithm begets the highest accuracy rating of 0.806 as well as a decently small standard deviation of 0.003. The Kamishima algorithm optimized for accuracy achieves a similarly high score of 0.805. However, the SVM and Naive Bayes algorithms have the smallest standard deviations of 0.002 as well as a comparatively high accuracy scores of 0.798 and 0.790 respectively, arguably making them better choices. Using BCR as a measure for performance, the Feldman SVM

variation comes out on top with a BCR of 0.828 and a reasonably low standard deviation of 0.002. MCC similarly determines performance based on accuracy of predictions and the MCCs indicate that the logistic regression algorithm is the best choice to maximize accuracy. That said, all of the MCCs are fairly low with the highest scoring at 0.416.

Taking fairness into account, we first look at the DI scores. The first and third variants of the Kamishima algorithm both achieve 1.000, but their extremely low accuracy scores make them less than ideal as choices. The Feldman algorithm's SVM variant stands out with a DI score of 0.938, all other choices barring the first and third Kamishima variants and the Feldman Decision Tree variant significantly further from 1. The Feldman algorithm's SVM variant has a relatively small standard deviation, though it is about twice as large as the smallest standard deviation. Comparatively, the Feldman DT variant has a slightly lower DI score, but a much smaller standard deviation, making it the better choice. Turning to another measure of fairness, the Feldman algorithm's SVM variant emerges on top once again with a CV score of 1.003 and a standard deviation of 0.003, only 0.001 larger than the minimum standard deviation. Again, the first and third Kamishima variants achieve CV scores of 1.000, but their low accuracy scores make the Feldman SVM variant a better choice.

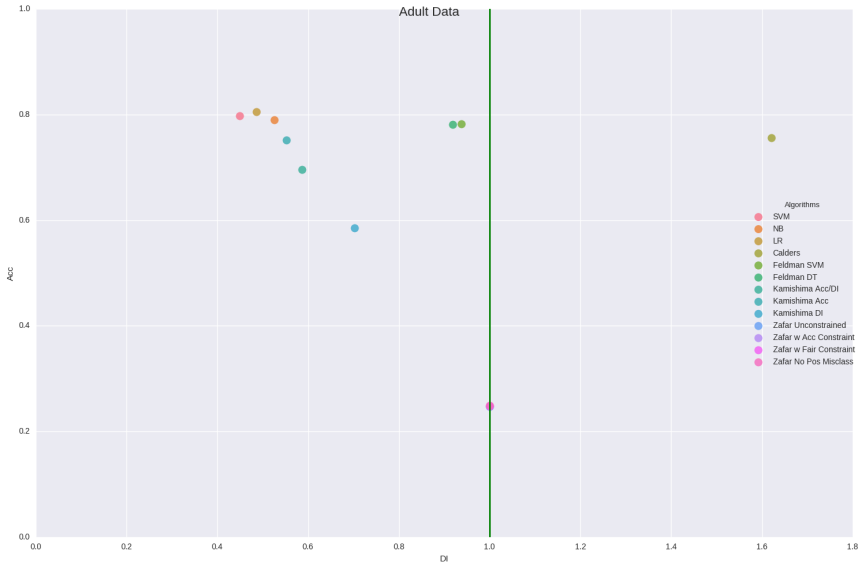

Figure 13: Results for all algorithms on Adult data

Because of the size of the Adult data set, run time is an important factor in deciding which algorithm is "best" to use for a similar data set. The Naive Bayes algorithm is the fastest, running at 0:00:00.014 while the Feldman algorithm took about an hour to run over one split and the first

variant of the Kamishima algorithm took over 4 hours. Taking all of the information into account, I would recommend logistic regression on a similar data set as it scores highly on accuracy and CV score. More importantly for a larger data set, the logistic regression algorithm is nearly as fast as the Naive Bayes algorithm, making it far more efficient than some of the other choices.

This data set was quite large, which made running through the algorithms very slow. Specifically, the Feldman algorithm and third variant of the Kamishima algorithm take about an hour or more to run for one split, which makes them unappealing when working with larger data sets. Another issue I came across with the Adult data was that all variations of the Zafar algorithm and the first and third variants of the Kamishima algorithm got consistently low accuracy scores. These results combined with the results from the Retailer data, another larger data set, lead me to believe that these algorithms do not handle large data sets as well as they handle smaller ones like the German data. However, when the data set is too small (as in the case of the Ricci data set), the Zafar algorithm has similar trouble making predictions while the Kamishima algorithm does decently well.

	Acc	Acc_SD	BCR	BCR_SD	MCC	MCC_SD	DI	DI_SD	CV	CV_SD	Run Time
SVM	0.798	0.002	0.817	0.002	0.373	0.007	0.449	0.031	1.044	0.003	0:00:12.392
NB	0.790	0.002	0.804	0.002	0.352	0.006	0.526	0.033	1.065	0.005	0:00:00.014
LR	0.806	0.003	0.796	0.003	0.416	0.007	0.486	0.031	1.083	0.007	0:00:00.044
Calders	0.756	0.031	0.828	0.021	0.282	0.049	1.621	0.886	0.914	0.127	0:00:01.539
Feldman SVM	0.782	0.003	0.828	0.001	0.295	0.013	0.938	0.059	1.003	0.003	0:55:53.142
Feldman DT	0.782	0.002	0.829	0.002	0.294	0.013	0.919	0.039	1.004	0.002	0:54:25.575
Kamishima $\frac{\|1-DI\|}{Acc}$	0.250	0.003	0.663	0.002	0.000	0.000	1.000	0.000	1.000	0.000	4:10:50.725
Kamishima Acc	0.805	0.003	0.797	0.003	0.414	0.008	0.494	0.023	1.080	0.005	0:08:09.126
Kamishima DI	0.250	0.005	0.662	0.001	0.000	0.000	1.000	0.000	1.000	0.000	0:08:05.415
Zafar Unconstrained	0.247	0.005	0.663	0.003	0.000	0.000	1.000	0.000	1.000	0.000	0:00:00.341
Zafar w Acc Constraint	0.248	0.002	0.663	0.002	0.000	0.000	1.000	0.000	1.000	0.000	0:00:00.179
Zafar w Fair Constraint	0.248	0.003	0.663	0.002	0.000	0.000	1.000	0.000	1.000	0.000	0:00:01.495
Zafar No Pos Misclass	0.249	0.003	0.662	0.002	0.000	0.000	1.000	0.000	1.000	0.000	0:00:41.416

Table 9: Results on Adult Data

6.2 German Data Results

Table 10 shows the results per metric for each of the different algorithms and their variations along with the standard deviation for each metric per algorithm. Based on the data, the Kamishima algorithm optimized for accuracy is the most accurate in terms of a straight-forward accuracy score with a score of 0.764 and the lowest standard deviation. Note that the other variations of the Kamishima algorithm as well as the logistic regression scored similarly on accuracy. That said, depending on the chosen measure of accuracy, the Kamishima algorithm might not actually be the "best" for this data set as the Calders algorithm has the highest BCR of 0.758. Once again, though the Calders algorithm has the highest score in the column, it is the Decision Tree variant of the Feldman algorithm that has the smallest standard deviation at 0.008. Given that the Calders standard deviation is significantly higher than either of the Feldman variants while the Calders BCR is only 0.054 higher than both Feldman BCRs, either Feldman variant might be a better choice. Still another measure of accuracy is the MCC. The Kamishima algorithm optimized for accuracy and the Kamishima algorithm optimized for both accuracy and fairness have extremely similar MCCs, but the accuracy optimization is a slightly better choice as its standard deviation is a bit smaller.

In the case where the "best" algorithm is based on its fairness score, we must consider DI and CV scores. For the German data, both the DI and CV scores were consistently high, the lowest DI score a 0.826 and the lowest CV score a 0.853. In terms of fairness, all of the algorithms are relatively safe choices when working with a data set similar to the German data. That said, the Zafar algorithm with no constraints applied achieves a DI score of 1.000 and a standard deviation of 0.000. The Zafar algorithm also scores 1.000 or - extremely close to 1.000 - in CV score in all of its variations, making it a decent choice if fairness is the focus.

This data set was reasonably small compared to some of the others, but time is still important to consider when choosing the algorithm that is most appropriate for a situation. The Naive Bayes algorithm is the fastest, clocking in at 0:00:00.001. Both the logistic regression and Zafar algorithms seemed to do well with this data set. They are not the fastest, but the data set is not large enough for the millisecond difference to matter very much. Both the Kamishima and Feldman algorithms were considerably slower than the rest, leading me to believe that they would not be ideal for use with larger data sets.

The biggest roadblock in the research on the German data came with Zafar's algorithm. With very small training sets, the variation that ensures no positive misclassifications would run indefinitely, but not on every split. When I looked at the total credit rate, the minority credit rate, and the majority credit rate for the broken splits, I saw the same numbers every

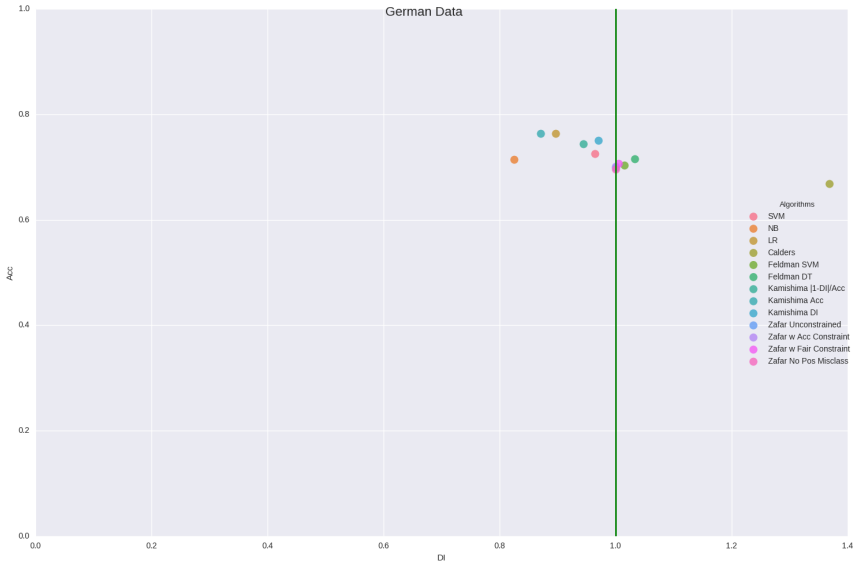

Figure 14: Results for all algorithms on German data

time. Confusingly, those numbers were extremely similar to the same rates for working splits, so it was difficult to tell exactly what the problem was.

I hypothesized that the algorithm was having issues differentiating between true and false positives when there were similar individuals in the training and test sets, so I ran some tests on a synthetic data set with duplicates of the same row. The Zafar algorithm ran perfectly on the tests when there were more protected individuals and when there were more unprotected individuals, but the test data did break the Calders algorithm which fails when there are many more individuals in one group (protected or unprotected) than another group.

	Acc	Acc_SD	BCR	BCR_SD	MCC	MCC_SD	DI	DI_SD	CV	CV_SD	Run Time
SVM	0.726	0.020	0.672	0.011	0.259	0.040	0.965	0.032	1.032	0.029	0:00:00.033
NB	0.714	0.030	0.694	0.010	0.352	0.047	0.826	0.047	1.119	0.028	0:00:00.001
LR	0.764	0.017	0.678	0.009	0.404	0.050	0.897	0.035	1.083	0.029	0:00:00.004
Calders	0.668	0.058	0.758	0.040	0.277	0.054	1.368	0.506	0.853	0.121	0:00:00.047
Feldman SVM	0.704	0.029	0.704	0.009	0.254	0.062	1.015	0.035	0.989	0.026	0:00:13.580
Feldman DT	0.716	0.013	0.704	0.008	0.280	0.053	1.033	0.032	0.975	0.024	0:00:13.618
Kamishima $\frac{j1\_DIj}{Acc}$	0.744	0.025	0.680	0.010	0.367	0.060	0.945	0.087	1.043	0.074	0:08:18.596
Kamishima Acc	0.764	0.012	0.675	0.009	0.399	0.041	0.870	0.050	1.106	0.041	0:00:15.927
Kamishima DI	0.751	0.034	0.681	0.008	0.335	0.117	0.970	0.089	1.022	0.073	0:00:20.064
Zafar Unconstrained	0.699	0.023	0.651	0.009	0.000	0.000	1.000	0.000	1.000	0.000	0:00:00.015
Zafar w Acc Constraint	0.701	0.022	0.655	0.011	0.008	0.025	1.000	0.001	0.991	0.001	0:00:00.212
Zafar w Fair Constraint	0.707	0.018	0.659	0.008	0.167	0.062	1.005	0.022	0.995	0.021	0:00:00.144
Zafar No Pos Misclass	0.696	0.023	0.649	0.009	0.000	0.000	1.000	0.000	1.000	0.000	0:00:09.867

Table 10: Results on German Data

6.3 Retailer Data Results

The results for the experiments with the Retailer data set are displayed in Table 11. All of the algorithms scored extremely similarly in all of the accuracy metrics. The Kamishima algorithm with my additions and the Zafar algorithm optimized for accuracy are about 0.001 more accurate than the rest with equal standard deviations 0.001 higher than the minimum. A number of the algorithms achieved the maximum BCR of 0.683, with only the SVM variant of the Feldman algorithm, the Kamishima algorithm optimized for accuracy, the Zafar algorithm with no constraints applied, and the third variant of the Zafar algorithm scoring the minimum at 0.682. As all of the standard deviations are similarly small, any algorithm would be an acceptable choice. Unlike the other two accuracy measures, MCC scores were consistently low with a high of 0.006. The Kamishima algorithm with my additions achieved the maximum MCC and a low standard deviation of 0.007, but MCC might not be the best choice when it comes to measuring a data set like this one.

Much like the accuracy metrics, the fairness metrics scored extremely closely. All variants of the Kamishima and Feldman algorithms achieved DI scores of 1.000 along with all variants of the Zafar algorithm except the one

optimizing fairness. The highest DI scores all had standard deviations of 0.000, making any of them a good choice. The CV scores were even more similar than the DI scores in that only the Calders algorithm did not score a 1.000. If CV is the measure, any algorithm will do, although Calders is slightly less optimal than the rest.

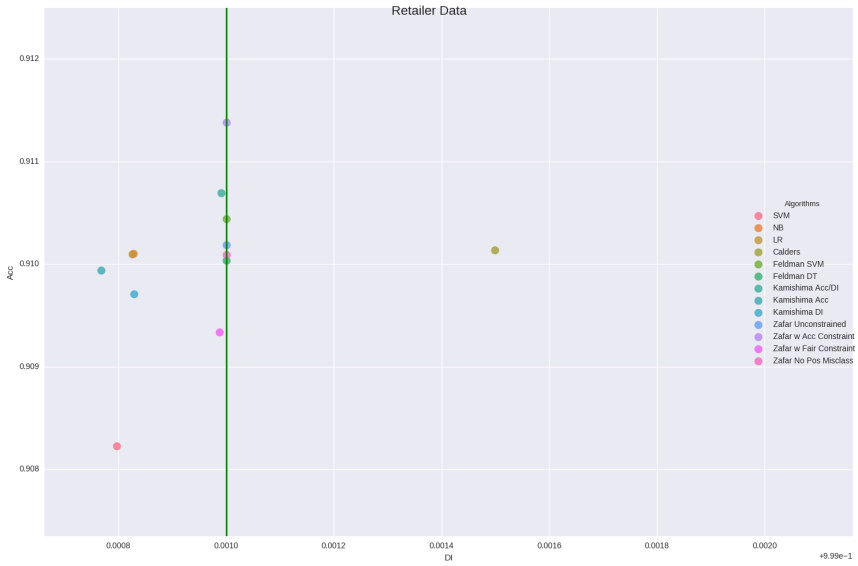

Figure 15: Results for all algorithms on Retailer data

Because all of the algorithms scored so similarly on all of the metrics, run time becomes a major factor in choosing which algorithm to use. The Retailer data set was also the largest data set tested, so a shorter run time is important. Table 11 shows that the Naive Bayes algorithm is faster than all of the others running in an average of 0:00:00.046 per split. Since its scores are all comparatively high, the Naive Bayes algorithm would be a good choice when running experiments on a data set like this one.

The size of the Retailer data set caused a number of issues when it came to running the algorithms on the data. Perhaps the most significant problem was time. Because of the size of the data set, it took hours and sometimes days to complete all 10 runs. The Feldman and Kamishima algorithms were particularly slow, suggesting that they might not be ideal for extremely large data sets like this one.

	Acc	Acc_SD	BCR	BCR_SD	MCC	MCC_SD	DI	DI_SD	CV	CV_SD	Run Time
SVM	0.908	0.002	0.683	0.001	-0.002	0.004	0.991	0.001	1.000	0.001	0:05:49.378
NB	0.910	0.002	0.683	0.001	-0.001	0.005	0.991	0.000	1.000	0.000	0:00:00.046
LR	0.910	0.002	0.683	0.001	0.004	0.007	0.991	0.000	1.000	0.000	0:00:00.145
Calders	0.910	0.001	0.683	0.002	-0.003	0.003	1.001	0.001	0.991	0.001	0:00:00.512
Feldman SVM	0.910	0.001	0.682	0.002	0.000	0.000	1.000	0.000	1.000	0.000	0:07:11.569
Feldman DT	0.910	0.002	0.683	0.002	0.000	0.000	1.000	0.000	1.000	0.000	0:07:14.092
Kamishima $\frac{j1\_DIj}{Acc}$	0.911	0.002	0.683	0.002	0.006	0.007	1.000	0.000	1.000	0.000	5:47:55.852
Kamishima Acc	0.910	0.001	0.682	0.001	-0.002	0.005	1.000	0.000	1.000	0.000	0:11:09.822
Kamishima DI	0.910	0.002	0.683	0.001	-0.001	0.005	1.000	0.000	1.000	0.000	0:11:31.727
Zafar Unconstrained	0.910	0.002	0.682	0.001	0.000	0.000	1.000	0.000	1.000	0.000	0:00:00.120
Zafar w Acc Constraint	0.911	0.002	0.683	0.002	0.000	0.000	1.000	0.000	1.000	0.000	0:00:03.746
Zafar w Fair Constraint	0.909	0.001	0.683	0.001	-0.001	0.001	0.991	0.000	1.000	0.000	0:00:09.781
Zafar No Pos Misclass	0.910	0.001	0.682	0.002	0.000	0.000	1.000	0.000	1.000	0.000	0:07:15.593

Table 11: Results on Retailer Data

6.4 Ricci Data Results

The results for the experiments with the Ricci data set are displayed in Table 12. For this data set, the standard Naive Bayes algorithm scored the highest in accuracy with a score of 0.940. Notably, it also has the lowest standard deviation. Using BCR as a measure to determine the "best" algorithm, the data shows that the Calders algorithm wins out over the other algorithms with a BCR of 0.763 and the smallest standard devation at 0.014. Considering the final measure of accuracy, MCC, the Naive Bayes comes up again with an MCC of 0.885, much higher than the other algorithms' scores. It also achieves the lowest standard deviation, making it the ideal choice if MCC is used as a measure of accuracy.

Once again, fairness must also be taken into account. The Feldman algorithm's SVM variant has the DI score closest to 1.000 (1.084), but its standard deviation is twice as high as the minimum standard deviation. Looking at DI score and disregarding all other metrics, the Zafar algorithm with no constraints applied seems the obvious choice with a perfect DI score of 1.000 and a low standard deviation of 0.000. However, this variant of the Zafar algorithm has very low accuracy and MCC scores. Additionally, the Feldman algorithm's SVM variant has the CV score closest to one at 0.970

and a fairly low standard deviation of 0.089, making it a good choice for experiments that focus on fairness.

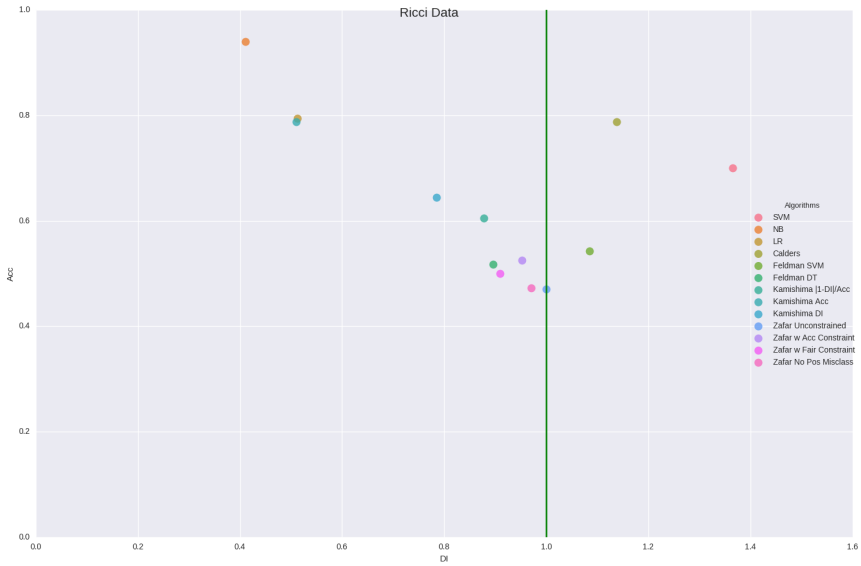

Figure 16: Results for all algorithms on Ricci data

The Ricci data set is extremely small with only 118 items, so run time isn't necessarily a huge factor when it comes to choosing the "best" algorithm. Still, the Feldman algorithm is considerably slower than the rest with a maximum run time of 0:00:05.666, followed by the Kamishima algorithm with a maximum run time of 0:00:01.494. Overall, the Naive Bayes algorithm seems like the best choice if accuracy is the most important as it scores highly in accuracy and MCC while the Calders algorithm is a good choice when fairness is the priority as it scores highly in both DI and CV. The Naive Bayes algorithm is the fastest with the Calders algorithm only 0.018 seconds behind.

On the whole, the Ricci data was fairly easy to work with simply because of its diminutive size. The only issue that arose came from the Zafar algorithm. As the gamma value rose, the Zafar algorithm in all of its variations failed to predict labels for items in the data set. This is most likely a consequence of the size of the data set as a high gamma value indicating a lower acceptable loss of accuracy paired with a small training set makes it much more difficult to make predictions. More testing could be done with the Zafar algorithm and the size of data sets.

	Acc	Acc_SD	BCR	BCR_SD	MCC	MCC_SD	DI	DI_SD	CV	CV_SD	Run Time
SVM	0.700	0.106	0.744	0.053	0.507	0.142	1.365	1.685	1.094	0.170	0:00:00.001
NB	0.940	0.041	0.644	0.026	0.885	0.076	0.410	0.138	1.419	0.115	0:00:00.001
LR	0.795	0.066	0.666	0.024	0.610	0.132	0.512	0.108	1.337	0.092	0:00:00.001
Calders	0.788	0.048	0.763	0.014	0.567	0.097	1.138	0.122	0.940	0.058	0:00:00.019
Feldman SVM	0.543	0.092	0.754	0.028	0.040	0.162	1.084	0.286	0.970	0.089	0:00:05.635
Feldman DT	0.518	0.065	0.729	0.042	0.038	0.193	0.896	0.251	1.074	0.139	0:00:05.666
Kamishima $\frac{j1\_DIj}{Acc}$	0.605	0.152	0.715	0.031	0.262	0.281	0.878	0.204	1.080	0.140	0:00:01.402
Kamishima Acc	0.788	0.104	0.661	0.050	0.614	0.145	0.511	0.259	1.358	0.207	0:00:01.494
Kamishima DI	0.645	0.232	0.700	0.040	0.393	0.356	0.784	0.260	1.146	0.188	0:00:01.455
Zafar Unconstrained	0.470	0.056	0.711	0.034	0.000	0.000	1.000	0.000	1.000	0.000	0:00:00.001
Zafar w Acc Constraint	0.525	0.064	0.706	0.030	0.066	0.120	0.952	0.097	1.047	0.068	0:00:00.039
Zafar w Fair Constraint	0.500	0.081	0.681	0.037	0.017	0.174	0.910	0.103	1.079	0.085	0:00:00.012
Zafar No Pos Misclass	0.473	0.083	0.700	0.022	0.042	0.081	0.970	0.047	1.030	0.047	0:00:00.439

Table 12: Results on Ricci Data

6.5 Maximizing Metrics

6.5.1 Accuracy

The data displayed in the previous subsections suggests that the logistic regression algorithm is the most accurate across data sets. While it does not always have the highest possible accuracy score, it does score consistently highly on accuracy for all of the data sets I looked into. I hypothesize that this more general algorithm is both well-tested and widely applied, keeping its accuracy score relatively high regardless of the size or nature of the data set. The Feldman algorithm does only slightly worse accuracy-wise.

6.5.2 Stability

Stability is another important factor to look into when discussing the pros and cons of each algorithm. It is important to ensure both stability over different training and test splits of the data as well as stability over metrics. If an algorithm is unstable over different splits, one person might run it and get an entirely different fairness level than another person who runs it slightly differently. Obviously, this is a problem when it comes to re-

producible results. Similarly, if an algorithm is unstable over a metric, the scores for that metric can differ greatly depending on input.

To measure stability of algorithms, I looked at the performance of each algorithm over all four data sets, using the average standard deviation over ten splits as an indicator of stability.

Overall, I would say that the Zafar algorithm optimized for fairness and its fourth variant (minimizing positive misclassifications) is the most stable as it scores similarly in the most columns for all of the tests I ran. I imagine this has something to do with the auto-tuning of the variable as the optimal gamma value is picked for each data set. The Kamishima algorithm optimized for accuracy, both Feldman variants, and the logistic regression algorithm were also decently stable.

Looking at specific metrics, all three of the more standard algorithms (logistic regression, Naive Bayes, and SVM) achieved consistently high accuracy scores. The SVM variant of the Feldman algorithm had the most stable BCR scores. None of the algorithms proved particularly stable in the case of MCC, which might make it a less than optimal metric to use. For both fairness metrics (DI and CV), both variants of the Feldman algorithm demonstrated stability.

6.5.3 Fairness

DI score and CV score are important measures of fairness. Across all of the data sets, the Feldman and Zafar algorithms got consistently high DI scores, suggesting that either would be an acceptable algorithm to use if the goal is to maximize fairness. They also achieved consistently high CV scores, further supporting the idea that either would be a good choice.

6.5.4 Other Metrics

The other metrics used in this study are not as often-used as accuracy to measure the performance of a decision-making algorithm. That said, BCR and MCC measure a slightly different kind of accuracy. Time is also important to consider, especially as data sets grow larger.

In terms of maximizing the other measures of accuracy, the logistic regression algorithm comes up again and again, doing well in MCC in almost all data sets examined. The Naive Bayes algorithm also scores highly in MCC for most of the data sets, though that might again be due to its nature as a more general algorithm. I would argue that the Feldman algorithm is the best choice as it maximizes a measure of accuracy as well as both measures of fairness.

Finally, we must consider efficiency of algorithms. The standard Naive Bayes algorithm emerges the victor, taking at most a few seconds to run a single split. That said, the Calders algorithm runs almost as fast. For larger

data sets, the logistic regression algorithm would likely be the idea choice as it is almost as fast as the Naive Bayes or the Calders algorithms and it scores well in accuracy and CV.

6.6 Trade-o s

In maximizing one metric such as accuracy or stability, there are certain trade-offs one makes in regards to the other metrics. For example, a more stable algorithm across data sets might be less accurate than an unstable one and vice versa. We can see in the data above that the logistic regression algorithm that maximizes accuracy is not the same as the Zafar variant that maximizes stability. A choice must be made, then, between the two. A more stable algorithm like the Zafar algorithm will be a better choice to run over multiple data sets while the very accurate logistic regression algorithm will be more useful to study a single data set.

Similarly, a decision must be made between accuracy and fairness as not all of the algorithms manage to achieve high scores in both. The Feldman algorithm seems the best choice in that it does well in both an accuracy and a fairness metric, but it is not the most accurate. The Feldman algorithm does consistently well, but take a lot of time to run. Similarly, the Kamishima algorithm does well for certain metrics (in particular, the variant optimizing accuracy achieves consistently high accuracy scores), but takes a very long time to run. In order to maximize other metrics, time must be sacrificed, which could be difficult with larger data sets.

6.7 Conclusion

Through my research, I have discovered that there are some fairness-aware machine-learning algorithms that are applicable across data sets, the most versatile being the logistic regression algorithm which has high scores in measures of accuracy and fairness as well as a decent average run time. Overall, the logistic regression algorithm seems the best to use in any case, while all variants of the Feldman algorithm are the ideal choices for smaller data sets.

In summary, it is clear that more work in this area needs to be done as this is still a very small number of data sets over which to compare the algorithms' performances. Adding more algorithms and more data sets of differing sizes and ratios of protected to unprotected individuals will likely produce more generalizable results.

References

[1] David Autor and David Scarborough. Does job testing harm minority workers? evidence from retail establishments. *The Quarterly Journal of Economics*, 123(1):219–277, 2008.

[2] Su Lin Blodgett, Lisa Green, and Brendan O'Connor. Demographic dialectal variation in social media: A case study of african-american english. *CoRR*, abs/1608.0, 2016.

[3] Toon Calders and Sicco Verwer. Three naive bayes approaches for discrimination-free classification. *Data Min. Knowl. Discov.*, 21(2):277–292, sep 2010.

[4] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S Zemel. Fairness through awareness. *CoRR*, abs/1104.3, 2011.

[5] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 259–268, New York, NY, USA, 2015. ACM.

[6] Evan Hamilton and Sorelle Friedler. Benchmarking four approaches to fairness-aware machine learning. 2017.

[7] F Kamiran and T Calders. Classifying without discriminating. In *2009 2nd International Conference on Computer, Control and Communication*, pages 1–6, feb 2009.

[8] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops*, ICDMW '11, pages 643–650, Washington, DC, USA, 2011. IEEE Computer Society.

[9] Corina Koolen and Andreas van Cranenburgh. These are not the stereotypes you are looking for: Bias and fairness in authorial gender attribution. In *Proceedings of the First Workshop on Ethics in Natural Language Processing*, pages 12–22, Stroudsburg, PA, USA, 2017. Association for Computational Linguistics.

[10] M. Lichman. UCI machine learning repository, 2013.

[11] Marco Lui and Timothy Baldwin. Langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 25–30, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[12] Weiwen Miao. Did the Results of Promotion Exams Have a Disparate Impact on Minorities? Using Statistical Evidence in Ricci v. DeStefano. *Journal of Statistics Education*, 18(3):1–26, 2010.

[13] James W. Pennebaker and Yla R. Tausczik. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of Language and Social Psychology*, 2010.

[14] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P. Gummadi. Fairness Constraints: Mechanisms for Fair Classification. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 962–970, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.