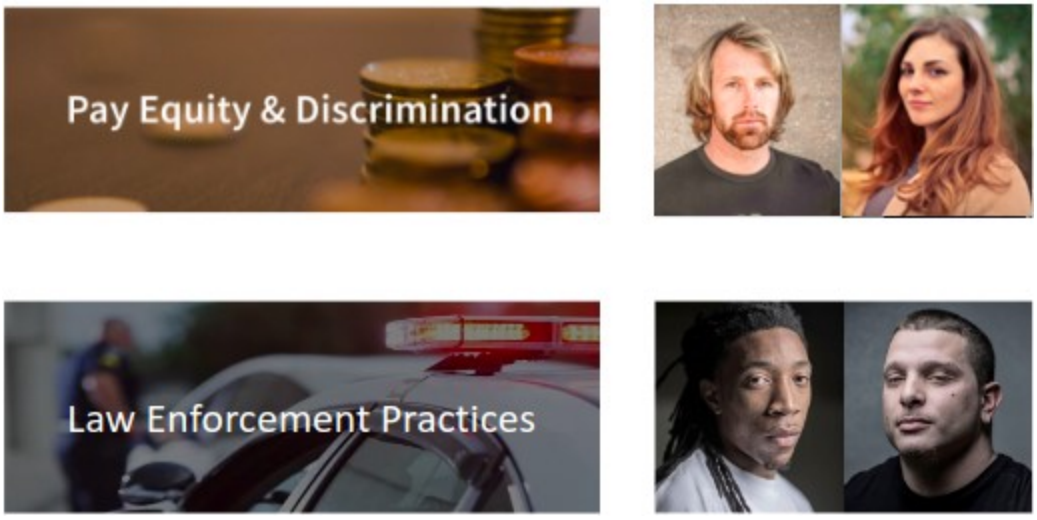
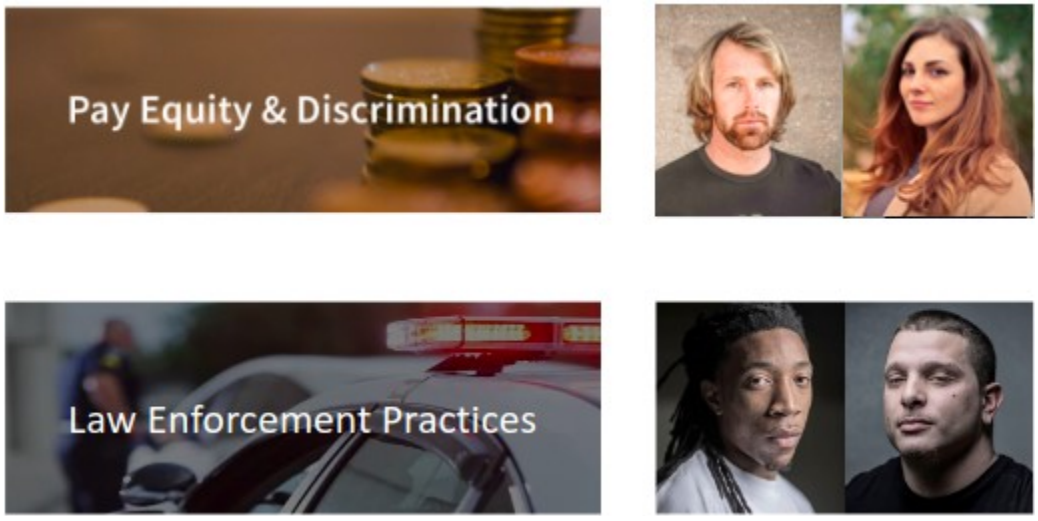


Tackling Bias in Machine Learning

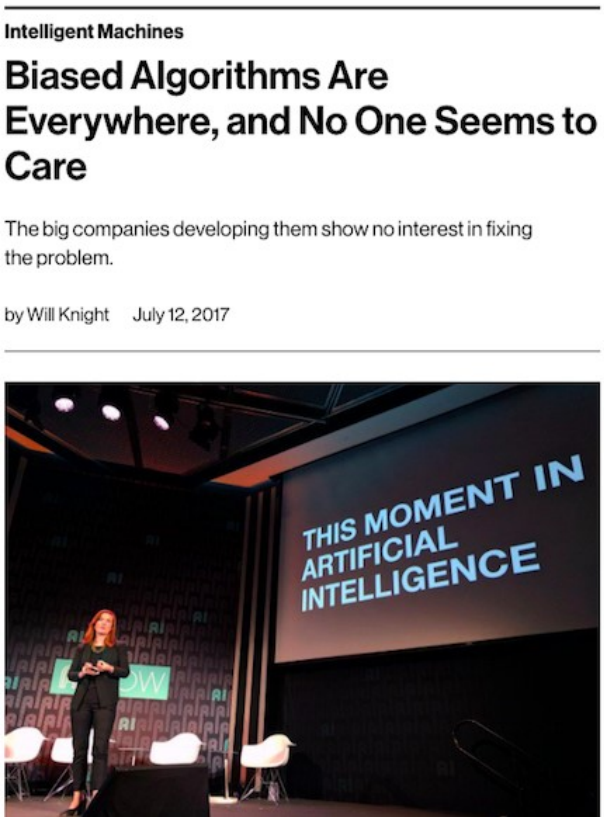
blog.insightdatascience.com



Bias in Machine Learning Algorithms (Bottom Photos Source: ProPublica; Top Photos Source: Pexels.com)

Biases in predictive modeling are a widespread issue

Machine learning and AI applications are used across industries, from recommendation engines to self-driving cars and more. When machine learning is used in automated decision-making, it can create issues with transparency, accountability, and equity. For example, last year it came to light that the AI tool Amazon built to automate their hiring process had to be shut down because it was discriminating against women.



Biased algorithms are a widespread problem (Source: MIT Technology Review).

As data scientists, machine learning engineers, and AI practitioners, we should be aware of the behaviors of the models we build and check them for biases toward certain attributes, such as race or sex.

In 2016, ProPublica published an article stating that the software used across the country to predict future criminals is biased towards certain people of color. Looking at this, I wondered if we could do something to address this problem using existing solutions.

Ensuring Equity in Machine Learning Models

Equality and equity are important in decision-making, but can produce very different results. In striving for *equality*, the same opportunity may be provided to everyone without taking into consideration the fact that there are societal factors that give some groups a head start. These groups have an automatic boost built-in. In these cases, equality is promoted at the expense of fairness. However, we can better promote fairness by ensuring *equity*, and leveling the playing field so that all groups have opportunities to succeed.

To address this issue of fairness, I've built a python package called **fairNN**, which quantifies the fairness of a model and uses an adversarial network to help mitigate biases in machine learning models. In using adversarial debiasing, our motivation is to bring fairness to the prediction while minimally sacrificing the prediction accuracy.

Defining a fairness metric

To define fairness in this context, we used a p% rule, a generic version of the 80% rule used to quantify the fairness of a model (there are other methods proposed by researchers). The rule is based on quantifying the disparate impact on a group of people with a protected attribute (race, sex, age, etc). A canonical example, where fairness metric can apply, is mortgage lending, where redlining in the 1930s continues to disproportionally affect communities today.

Disparate impact occurs when policies and rules that appear to be neutral don't behave that way in practice. Quantifying the disparate impact on the dataset will tell us whether or not the model we are building is fair. As this article explains in more detail:

"The p% rule states that the ratio between the probability of a positive outcome, given the sensitive attribute being true, and the same probability, given the sensitive attribute being false, is no less than p:100. So, when a classifier is completely fair it will satisfy a 100%-rule. In contrast, when it is completely unfair it satisfies a 0%-rule."

This rule is what we will use to judge the fairness of our work.

Techniques for mitigating biases

There are several processing techniques commonly used to mitigate the bias of predictive models. These processing techniques are grouped into three methods: pre-processing, in-processing and post-processing.

Pre-processing

The first mitigation approach is *pre-processing,* which occurs before the creation of the model. Using these techniques, the dataset is transformed with the aim of removing the underlying bias from the data before modeling. This is the most basic approach of mitigating the bias wherein we remove the underlying sensitive attribute from the data and then feed the model with the transformed data. This approach does not always work because of correlated attributes in the data which can surface underlying bias to the model. For example, if you were to remove ethnicity from a dataset, an individual's zip code can inadvertently encode partial information of the likelihood of a certain ethnicity.

In-processing

In-processing techniques can be considered as modifications to the traditional learning algorithms to address discrimination during the model training phase. An example of this technique is modifying the cost function, to include a "fairness" regularizer for training the model. These regularizers take into account differences in how the learning algorithm classifies protected versus non-protected classes

(i.e., *S1* vs. *S0*) and penalizes the total loss based on the extent of the difference. Another example of the in-processing technique is to oversample the dataset wherein the features with minor representation are oversampled and the dataset is augmented. This augmented dataset is then fed into the model.

Post-processing

In *post-processing* techniques, the processing of the dataset is done post-model training. AIF360 also lists several other state-of-the-art mitigation algorithms. The approach we will use, called adversarial debiasing, is an example of a post-processing approach wherein bias is mitigated after the training takes place.

My Motivation & Approach

I used adversarial networks in mitigating the bias of the predictive model. An adversarial network, first introduced by Ian Goodfellow, is attached on top of the predictive classifier. The framework of my project is to:

- Quantify the fairness of a predictive model using a fairness metric
- After calculating fairness, determine whether or not the model satisfies a certain threshold value for fairness
- If the model is not fair, then the classifier is debiased using an adversarial network.

Overview of what this project aims to build.

Data Used: Recidivism Risk

I used a dataset compiled and released by ProPublica that labels how likely a criminal is to re-offend in the future based on his score. To make it a binary class problem, I re-labeled the dataset so that risk_score > 5 is labeled as high and risk_score ≤ 5 is labeled as low.

The network tries to label whether a criminal is a high risk or low risk to re-offend in the future. The goal here is to check whether or not a predictive model is biased toward a certain race. A common assumption is that, if we remove the sensitive attribute from training data, then the prediction of our model will be bias-free. **However, removing sensitive attributes still results in a biased model of criminal recidivism!**

To illustrate, I have divided my dataset into three sets: features, targets, and sensitive attributes.

- *Features* are the dataset that is used to train our model.
- *Targets* are the binary class labels which our model has to predict.
- S*ensitive attributes* are the attributes we want to prevent our model from being biased against.

With this training, I achieved a classification accuracy of ~77%. But, in order to check whether or not the model is biased, we need to quantify fairness. Calculating the value of fairness using p% rule for the sensitive attribute (race) gave the model a value of 38, which is much less than it should have been in a fair classifier. Using the most basic approach of removing sensitive attributes still results in a biased model.

Results with and without sensitive attributes

Implementing adversarial debiasing

The input to the adversarial network is the output, i.e. the predicted class labels from the previous classifier. The adversarial network then tries to predict the sensitive attribute. The two networks train, turn by turn, with the objective to minimize the loss of the prediction of both networks.

With this technique implemented, calculating the p% rule gives us the value of 70, almost twice as likely to be fair than that of the previous approach.

Implementation of fairNN: Fairness value almost doubled, but accuracy did decrease.

However, the accuracy of classification dropped around ~6%. Important to note, is the reduction in accuracy can be controlled! The figure below shows a plot (with confidence intervals) of fairness vs accuracy. We can see the trend, as we move towards gaining fairness, there is a slight drop in the accuracy of the model. For ML problems from predicting criminal recidivism, issuing loans, hiring and more; the tradeoff between accuracy and fairness can be quantitatively controlled using FairNN!

95% confidence interval plot for accuracy vs fairness

Analysis

Using adversarial networks, I was able to significantly increase the fairness of our model. However, I have identified some important considerations when using this approach:

- At maximal fairness, we sacrifice ~6% accuracy to make the model ~200% more fair!
- We need to specify which sensitive attribute we should check for fairness. The model itself is unable to determine the particular attribute for which predictions should be fair.
- When we increase fairness at the cost of accuracy, we can look at a confusion matrix (see figure below) to see how our model's decisions have changed in practice. In our case, our model has increased its chances of false positives, which

helps counteract the inherent bias in the data, but leads to lower accuracy for the “advantaged class” (within the context of our biased dataset).

■