

THE BIAS TRADE-OFF FOR GRANTMAKING ALGORITHMS

February 2018



An initiative of:
ourcommunity.com.au
Where not-for-profits go for help



The bias trade-off for grantmaking algorithms

An Our Community Innovation Lab report,

February 2018

Report author: Joost van der Linden, data scientist

This report is produced by the Our Community Innovation Lab – the engine room for mobilising data to drive social change.

Our team of data scientists, IT engineers and domain knowledge experts is working to bring to life ideas to do old things better and new things first.

www.ourcommunity.com.au/innovationlab

Suppose we wish to develop an algorithm that automatically shortlists grant applications. The grantmaker may receive thousands of applications and aim to reduce their workload by asking the algorithm to reduce the entire set of applications to the most promising ones, to be further assessed by the human assessors. Immediately, some alarm bells should start to ring. How do we ensure that our algorithm is fair? How do we ensure that all promising applications make the shortlist, without missing out on any good ones? How do we identify those promising applications in the first place? Can we make sure no applications are unfairly denied a spot on the shortlist?





In this white paper, we focus on the fairness questions and leave the question on how to identify promising applications for another note. It turns out fairness is a really challenging problem for any algorithm that operates in the real world. The real world is messy. Algorithms are rarely 100% accurate, and the data that our algorithms use to make decisions is never truly unbiased. Under these circumstances, it has been shown mathematically (see [paper](#) by Kleinberg, Mullainathan and Raghavan) that *algorithmic bias is a trade-off*. By adjusting an algorithm to address a bias against one group of people, there may be another group of people that are disadvantaged. It is ultimately up to us, as the algorithm developers, to decide *how* we choose to make the trade-off. Let's see how this works out with an example.



Say a national grantmaker runs a hypothetical annual grant round for applicants from Sydney and Melbourne. Moreover, for some reason, applications from Melbourne-based applicants in previous years have been of better quality, and are better aligned with the grantmaker's objectives than applications from Sydney. Think of this difference as a difference in "base rate performance." Perhaps the grantmaker prefers arts applications, more likely to come from Melbourne, or perhaps the grantseeker's location is a proxy for other, underlying factors that influence the quality of the applications and/or the grantmaker's decision.

As an example, let's assume about 50% of past Melbourne applications were actually good applications, versus only 30% for applications from Sydney. Let's also assume the pool of applicants for this year's round is roughly the same as in previous years, in which case we can expect, again, 50% of applications from Melbourne and 30% of applications from Sydney to be actually promising.

Let's turn these percentages into actual numbers (derived from [this example](#) by Cathy O'Neil). Say we get 10,000 applications from Melbourne and 10,000 applications from Sydney. We also assume we know the "truth" for these applications, namely, whether or not each application is actually promising (and should be shortlisted by our algorithm later).





For example, we may get:

| | Bad applications (not promising) | Good applications (promising) |
|-----------------------------|--|---|
| Applications from Sydney |  |  |
| Applications from Melbourne |  |  |

 = 1,000 bad applications  = 1,000 good applications

Observe that our assumptions regarding base rates are indeed satisfied in this example. 3,000 out of 10,000 applications from Sydney (30%) are promising this year, as expected from previous years' performance. Similarly, 50% of applications from Melbourne were shortlisted, as expected.

Having defined our hypothetical grant round and the rates of promising applications, the next table shows some hypothetical output that our shiny new shortlisting algorithm might produce. "Shortlisted by the algorithm" means that the algorithm decided this application was promising and should be shortlisted (and vice versa for "not shortlisted by the algorithm").

| | Not shortlisted | Shortlisted |
|-----------------------------|--|---|
| | by the algorithm | by the algorithm |
| Applications from Sydney |  |  |
| Applications from Melbourne |  |  |

Observe that, like most algorithms in the real world, the shortlisting predictions are not perfect: there are 1,000 applications from Melbourne shortlisted by the algorithm, for example, that were not actually promising (one thumbs down in bottom-right corner).

Next, we'll outline three "fairness conditions" and demonstrate that we can *only satisfy two at a time*. This is the trade-off we have to make.

Fairness condition 1: Calibration

The first fairness condition for our shortlisting algorithm is that the fraction of automatically shortlisted applications should be equal to the fraction of promising applications (50% for Melbourne and 30% for Sydney). This fairness condition is called *calibration*. It would be unfair, for example, if the algorithm shortlisted only 10% of applications from Sydney, knowing that the actual fraction of good applications is closer to 30%.

You can also see our algorithm is indeed well-calibrated: for Sydney applications in the first row, there are 3 thumbs up and 7 thumbs down (30% of applications are actually promising) and the algorithm also shortlisted 3,000 out of 10,000 applications (= 30%). Similarly, the algorithm is well-calibrated for Melbourne: there are 5 thumbs up and 5 thumbs down (50% promising applications) and also 5,000 out of 10,000 applications are shortlisted (= 50%).

Another way to think about a calibrated algorithm is that it needs to trade mistakes. For the 1,000 incorrectly shortlisted applications in the bottom-right corner, the algorithm also made 1,000 mistakes in the bottom-left corner: 1,000 promising applications from

Melbourne were not shortlisted. Keep this calibration requirement of "trading mistakes" in mind as we go forward.

Fairness condition 2: False shortlisting rate

Our second fairness condition is for the *false shortlisting rate* to be equal for Melbourne and Sydney applicants. The false shortlisting rate is the fraction of applications that were shortlisted, but are not actually promising.

To understand what this means, let's quantify and define some of the errors our algorithm made in relation to this fairness condition:

- 1,000 applications out of the 3,000 applications from Sydney that were shortlisted were not actually promising (top right corner in the previous table). This means we have a "false shortlisting rate" for Sydney applications equal to:

$$1,000 / 3,000 = 33\%.$$

- 1,000 applications out of the 5,000 applications from Melbourne that were shortlisted were not actually promising (bottom right corner). This means we have a false shortlisting rate for Melbourne applicants equal to:

$$1,000 / 5,000 = 20\%.$$

The false shortlisting rate is higher for Sydney (33%) than for Melbourne (20%), even though they both have only 1,000 misclassified applications each. That's unfair: it means that applications from Sydney are more likely to be incorrectly shortlisted than applications from Melbourne. While that may seem as an (unfair) advantage initially, recall that the algorithmic shortlisting will be followed by a human assessment of the shortlist. The assessor will likely decline the bad applications, reducing the fraction of approved applications from Sydney.

Fairness condition 3: False denial rate

The third fairness condition is for the *false denial rate* to be equal for Melbourne and Sydney applicants. The false denial rate is the fraction of applications that were not shortlisted, but are actually promising.

In our example:

- 1,000 applications out of the 7,000 applications from Sydney that were not shortlisted were actually promising (top left corner). This means we have a "false denial rate" for Sydney applications equal to:

$$1,000 / 7,000 = 14.3\%.$$

- 1,000 applications out of the 5,000 applications from Melbourne that were not shortlisted were actually promising (bottom left corner). This means we have a false denial rate for Melbourne applicants equal to:

$$1,000 / 5,000 = 20\%.$$

Adjusting for fairness

We can see that under fairness condition 2, Sydney applicants are disadvantaged by our algorithm. Luckily, we can do something about this. We can adjust the algorithm for Sydney applicants to lower the false shortlisting rate from 33% to 20%, equal to the false shortlisting rate for Melbourne. *How* we make this adjustment in the algorithm is beyond the scope of this note.

Here's the catch. Remember, for the algorithm to meet fairness condition 1 and remain *well-calibrated*, we need to trade mistakes. If we lower the false shortlisting rate for Sydney to 1,000 out of every 5,000 applications (20%, to match the false shortlisting rate for Melbourne), this implies that for the 3,000 shortlisted applications from Sydney, $(1,000/5,000) * 3,000 = 600$ applications are incorrectly shortlisted. To keep the algorithm well-calibrated, we also need to make 600 "mistakes" for the 7,000 applications in the top left corner of the table. In other words, 600 promising applications would have to be incorrectly not shortlisted. This would result in a $600 / 7,000 = 8.6\%$ false denial rate.

Because we didn't change the false shortlisting rate for Melbourne, the false denial rate for Melbourne remains the same at 20%.

Before the correction, the false denial rate for Sydney was $1,000 / 7,000 = 14.3\%$. After our tweaks to make the false shortlisting rate equal, the false denial rate dropped to 8.6%. As a result, the difference in the false denial rate between Melbourne and Sydney increased from $20\% - 14.3\% = 5.7\%$ to $20\% - 8.6\% = 11.4\%$.

Do you see what happened? By equalizing the false shortlisting rate, we made the difference in the false denial rate *larger* (more unfair!). A high false denial rate for Melbourne (compared to Sydney) implies that good applications from Melbourne are more likely to be incorrectly denied a spot on the shortlist by our algorithm. Just like a higher false shortlisting rate, that's a disadvantage.

Conclusion

In conclusion, we outlined three fairness conditions and showed that by making two of them fair (calibration and the false shortlisting rate) we made the third condition worse (false denial rate). Similarly, we could have made the algorithm fair in terms of calibration and the false denial rate, but that would have made it more unfair in terms of the false shortlisting rate. [This paper](#) shows that, indeed, it is mathematically impossible to achieve all three fairness conditions simultaneously if the base rates are not equal and our algorithm makes imperfect predictions.

So, how should we proceed with our shortlisting algorithm? We have to make a choice: which fairness conditions will we focus on? One potential strategy could be:

- Ensure, first and foremost, that a minimum (and equal) fraction of good applications is denied a spot on the algorithmic shortlist, for Sydney and Melbourne applicants (addressing fairness conditions 1 and 3).
- Allow a slight bias against Sydney applicants, in terms of the fraction of bad applications making it onto the shortlist, to be knocked off by the human assessor (making some concessions for fairness condition 2).

Obviously, these are not easy decisions. If we proceeded with this strategy, Sydney applicants already have a lower success rate (for whatever reason) which is further impacted by our desire to minimize the number of good applications slipping through the cracks. We could further compensate for this issue by having the human assessor being slightly more lenient in assessing shortlisted Sydney applications. That raises a fairness issue of its own, however, because we would be applying different standards to Melbourne and Sydney applicants.

Another consideration is that, at times, it may be warranted to accept (or introduce) some bias into an algorithm if there is evidence to suggest a group faces genuine barriers that have resulted in a difference in base rates. For example, under an affirmative action approach, perhaps the algorithm should shortlist an equal fraction (40% for Sydney and 40% for Melbourne) or even an equal number (4,000 for Sydney and 4,000 for Melbourne) of applications.

In conclusion, for our grantmaking algorithm, and for many other algorithms out there in the real world (some of which we encounter in our own work with grantmakers and beyond), the choice is rarely simple. It is ultimately up to the developers of the algorithm, the people using the algorithm, and the people impacted by the algorithm, to understand the trade-offs, and jointly decide where to strike the balance.

Sources

- Example adapted from: Cathy O'Neil (2017). Two out of three "fairness" criteria can be satisfied ([link](#)).
- Original paper: Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan (2017). Inherent Trade-Offs in the Fair Determination of Risk Scores ([link](#)).
- Discussion of the paper: Cody Marie Wild (2017). Fair and Balanced? Thoughts on Bias in Probabilistic Modeling ([link](#)).

Author

Joost van der Linden – Data Scientist, Our Community

With the help of:

- Sarah Barker – Director of Data Intelligence, Our Community
- Paola Oliva-Altamirano – Data Scientist, Our Community
- Kathy Richardson – Executive Director, Our Community