

Ethical Implications of AI Bias as a Result of Workforce Gender Imbalance



THE UNIVERSITY OF
MELBOURNE



Preamble

To contextualise the findings provided within, this report should be read in conjunction with the Literature Review deliverable dated 26th June 2020 (attached within the Appendix for convenience). The terms resumé and CV will be used interchangeably throughout the report.

The authors have also provided a Python Jupyter Notebook¹ (titled *UniBank Project Source Code*) which contains the Python source code, technical documentation, experimental methodology (including internal tests and calculations), and results. This notebook is provided in the native computer-readable IPYNB format - with a print-friendly PDF version also included in the Appendix. The current report contains cross-references to relevant Jupyter Notebook sections as relevant.

¹ The programming language of choice is Python, due to its widespread use in data science and machine learning applications, as well as its extensive library of specialist functions (e.g. *numpy* for mathematical operations). It is provided in the Jupyter Notebook file format, which integrates both code and technical documentation. For a brief introduction on the Jupyter Notebook file format, please refer to https://en.wikipedia.org/wiki/Project_Jupyter.



Authors

Marc Cheong

*School of Computing and
Information Systems*

Reeva Lederman

*School of Computing and
Information Systems*

Aidan McLoughney

*School of Computing and
Information Systems*

Sheilla Njoto

The Policy Lab

Leah Ruppanner

The Policy Lab

Anthony Wirth

*School of Computing and
Information Systems*

A network diagram consisting of numerous white circular nodes connected by thin grey lines, forming a complex web-like structure. The nodes are distributed across the upper half of the page, with a higher density in the top left and right corners. The background is a solid dark blue color at the top, which transitions into a white background for the rest of the page.

Contents

Preamble	2
Authors	3
Contents	4
Executive Summary	5
Gender Roles, Gendered Judgements	7
Designing research: gender biases in panellists' recruitment	8
Quantifying gender bias in panellists' recruitment decisions	9
Patterns of heuristic judgements	10
Algorithms: eradicating or exacerbating gender bias?	11
Lessons from the Amazon Case	12
Data Science Approach: Machine Learning	14
Design Decisions	14
Experimental Design and Key Tasks	15
Experimental Results	17
Can the computer match human decision-making?	17
Machine Bias Could Influence Algorithmic Outcomes in Three Meaningful Ways	19
Human Bias Could Influence Algorithms in Two Meaningful Ways	24
Discussion and Future Work	29
Recommendations and Conclusion	31
Bibliography	33
Appendices	35

A network diagram with nodes and connecting lines, overlaid on a dark blue header bar. The nodes are represented by circles of varying sizes, and the lines are thin and light-colored, creating a complex web-like structure.

Executive Summary

This report summarises the key findings from the *Exploring the Ethical Implications of AI Bias as a Result of Workforce Gender Imbalance* project by University of Melbourne researchers (from the Policy Lab and the School of Computing and Information Systems (CIS)/Centre for AI and Digital Ethics (CAIDE). This interdisciplinary project draws upon sociological research and human panel experiments from Policy Lab, and industry-standard data science approaches for machine learning and algorithmic development and evaluation from CIS/CAIDE.

This report will first discuss background research in gender bias with respect to hiring; in order to contextualise results of the hiring decisions (and rationales) by the Policy Lab panel when given resumés characteristic of three different industries, controlling for applicants' names (as markers of gender).

Next, the report focuses on design decisions and justifications for the proposed machine learning algorithms (including transparency, participatory design, and use of open-source software); before describing the prototype machine learning algorithms to automate (and replicate) human judgements; as well as a critical investigation of their outputs. We use the term 'classifier' for an algorithm which attempts to classify a data point (in our case, e.g. 'hired' versus 'not hired'). The term 'predictor' or 'estimator' is used for an algorithm to, as its name implies, predict an unknown value (in our case, e.g. candidate is ranked '2nd' or '5th'). Hence, throughout the report, the terminology used reflects the context in which the algorithm in question is discussed².

The report finds that both human bias and machine bias can influence hiring outcomes, by illustrating three hypotheses on how algorithms affect hiring decisions and entrench stereotypes, and two hypotheses on how human factors play a role as well. Recommendations to reduce bias include training programs for human resource professionals; audits of hiring and gender discrimination across all positions; creating established quota systems for hiring; and creating proprietary hiring algorithms that are transparent and trained with the express aim of reducing gender bias (with regular audits).

² Further reading: Han,, Pei & Kamber 2011.





Gendered Roles, Gendered Judgements

Throughout history, women's position in society has been relegated⁵ to the role of homemaker. Although this gender-role norm has somewhat weakened today, its impact on lingering prejudice⁶ still elicits subconscious gender bias in the modern-day recruitment arena. As women are often still associated with domestic work, they are presumed to be less productive in the workplace than men. Despite identical qualifications and skills, employers often favour male candidates as they describe them as 'competitive', 'experienced' and 'ambitious' in comparison to women who are considered.⁷ It remains unclear, however, how these gender associations have been generated and why these specific traits became part of the recruitment success metrics.

These descriptions are not limited to employers. In fact, men and women often describe themselves with these gendered adjectives⁸. Women often use more communal, social and expressive vocabularies in comparison to men who use language that is more managerial and directive. Moreover, these gender differences are apparent in how society describes itself. Descriptions about men contain words that highlight 'prominence', such as 'outstanding' or 'unique', whilst women are often described more with 'social' connotations, such as 'warm' and 'collaborative'⁹.

Gender bias occurs when specific traits tied to broader expectations of gender are applied to individuals regardless of whether an individual exhibits these traits³. Gender bias often extends into job recruitment when hiring panels rely on existing heuristics to rank men and women as qualified for positions⁴.

³ Sczesny et al. 2016

⁴ Cohen 1976; Russo 1976

⁵ Bailey LaFrance, and Dividio 2018

⁶ Burgess 2013

⁷ Cohen 1976

⁸ Gaucher et al. 2011

⁹ Sczesny et al. 2016

Designing research: gender biases in panelists' recruitment

For each occupational role, we provide hiring panellists with a set of real resumés, with a balanced ratio between male and female candidates. Half of the panellists were given the original resumés, without the gender of the candidate manipulated. The other half were given the *exact same* resumés, but with the gender changed (male to female and female to male – indicated by names as gender variable; for instance, 'Mark' to 'Sarah', or 'Rachel' to 'John'). The hiring panellists within each group were then instructed to rank each resumé individually: with a lower value denoting a better rank (i.e., their favoured candidate has rank 1). They were asked to provide their ranked lists to the researchers prior to the group meeting. Then, the panellists deliberated in groups of three to collectively decide on the Top 3 and Bottom 3 resumés for each occupational role. Finally, we used these discussions and rankings to create a hiring algorithm to create a consensus ranking of candidates for the position.

Our research first investigates how gender bias is incorporated into employment recruitment. We observe hiring behaviours for three specific occupational roles, selected for their gender ratios, to include: *male-dominated*, *gender-balanced* and *female-dominated* industries. The three occupational roles are *Data Analyst*, *Finance Officer* and *Recruitment Officer* respectively (Figure 1).

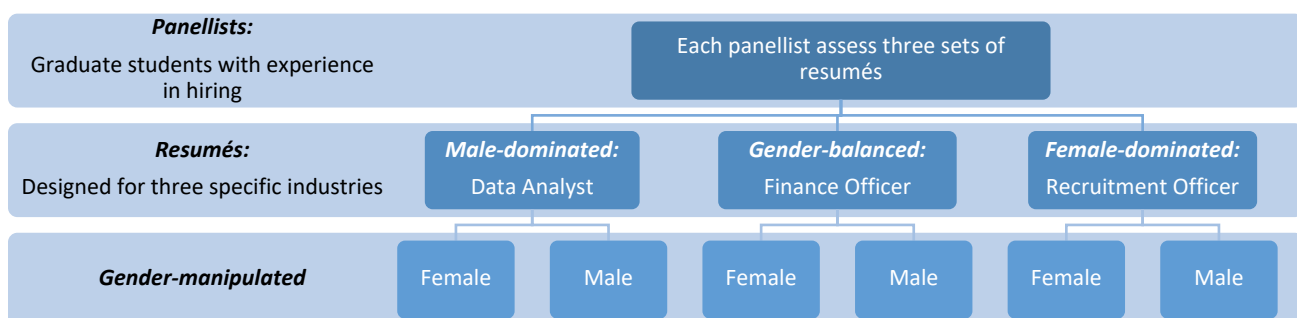
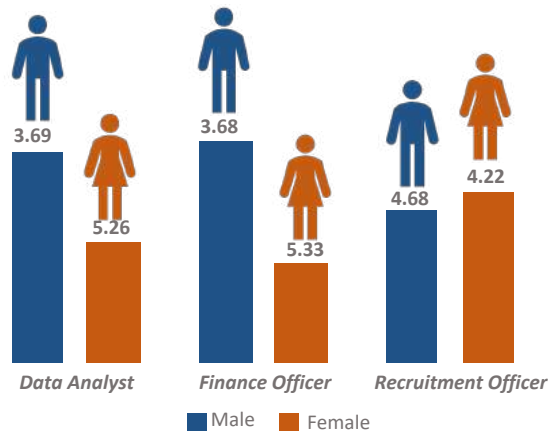


Figure 1. Experimental setup for human panellists in assessing resumés

Quantifying gender bias in panelists' recruitment decisions

Individual Ranking Average

Note: the lower the value, the higher the ranking for 40 panellists



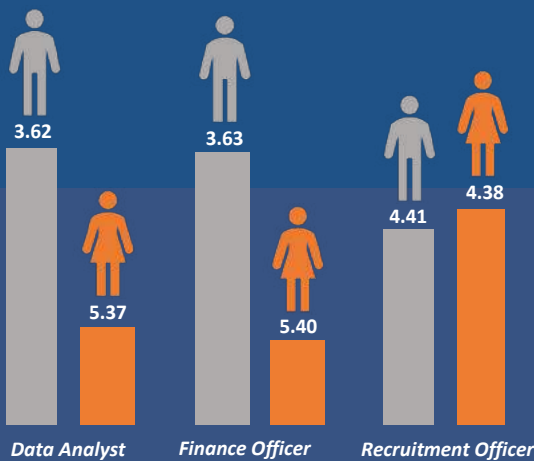
The findings in Figure 2 demonstrate that panellists of both genders ranked female resumes lower than male resumes in both *male-dominated* (Data Analyst) and *gender-balanced* (Finance Officer) roles, on average.

On the other hand, female resumes are favoured more in the *female-dominated* (Recruitment Officer) role, being almost half a rank better than male resumes on average.

Figure 2a. Average rankings of individual panellists

Average Ranking by Female Panellists

Note: the lower the value, the higher the ranking for 40 panellists



Average Ranking by Male Panellists

Note: the lower the value, the higher the ranking for 40 panellists

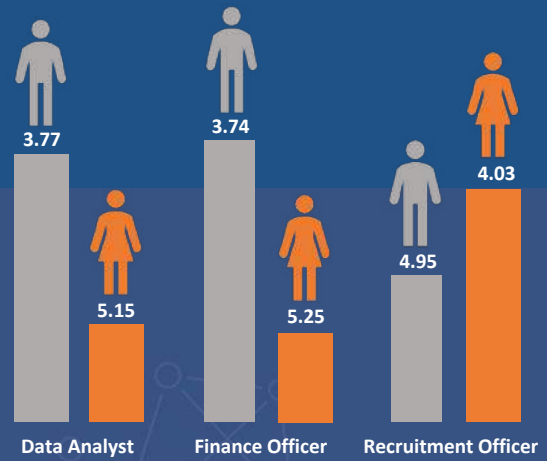


Figure 2b. Average rankings of (a) female panellists and (b) male panellists

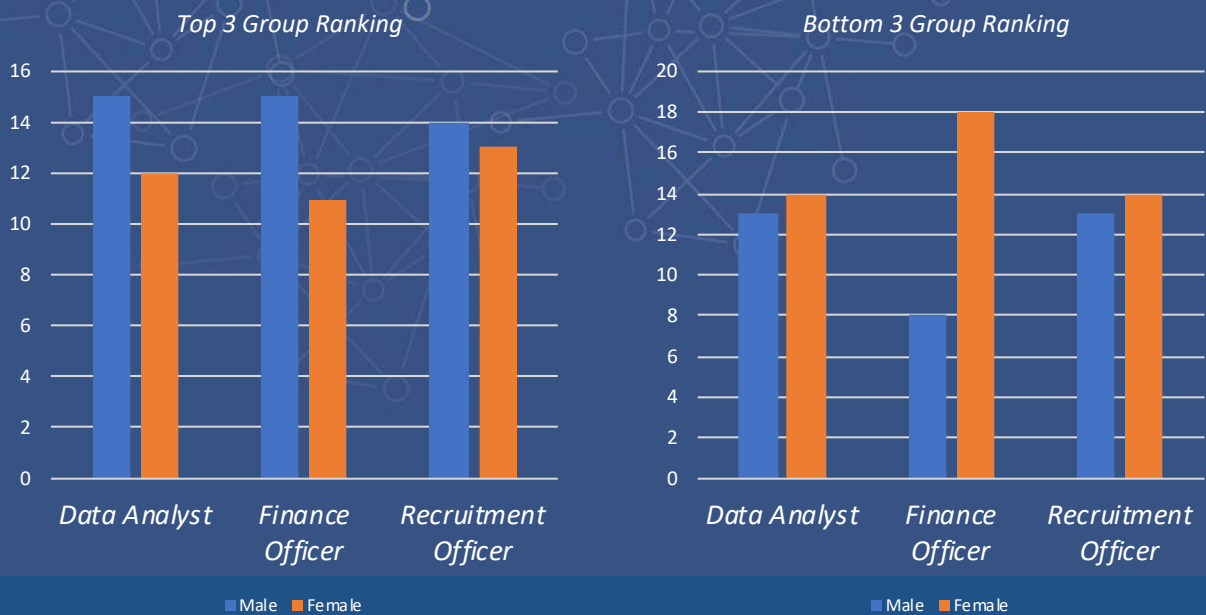


Figure 3. Count of topmost (ranks 1, 2 and 3) and bottommost (ranks 6, 7 and 8) rankings for resumes by gender, per industry type

As shown in Figure 3, our male candidates were more often ranked in the Top 3 for all jobs listed. By contrast, our female candidates were more often ranked in the Bottom 3. The figures also demonstrate that the gender bias occurs the most in the Bottom 3 candidates for the Finance Officer role, with male resumes half as likely to be ranked at the bottom (9 male vs. 18 female resumes). As a close second, the Top 3 ranked resumes for Finance Officer also illustrate a possible gender bias, with more male resumes ranking in the top 3 resumes than female (15 male vs. 11 female).

Patterns of heuristic judgements

Our study uncovered several themes during deliberation. We identified subjective assessments from the panellists that may increase the risk of introducing gender biases in hiring decisions. These findings are coherent with extant research on decision-making heuristic biases¹⁰.

1

First impression as a ruling lens

During each deliberation, panellists express their first impression towards the candidates' 'self-presentation'. This first impression emerged from the structure of the resume, layout and word choices in self-description. Most panellists believe that these components signify a candidate's *passion, likeability* and *resilience*. These assumptive associations are often referred to as a *first impression bias*, in which one's judgement is influenced by societal and cultural connotations and personal encounters.¹¹

2

Personal experience as a benchmark of quality

Panellists assess the resumes against the given job descriptions, including qualifications, skills and educational backgrounds; however, most panellists describe the term 'experienced' subjectively, by comparing against their own background. Studies have identified this pattern as an *affinity bias*, which has the potential to exclude women and other minority groups when underrepresented on hiring panels.¹² This includes the panellists' varying judgements over what 'job churn' means.

Whilst some favour this component as they believe it indicates *versatility* and *agility* based on their experience, others perceive this component as undesirable as it may indicate an undervaluing of growth and persistence. Given that females often have gaps in employment for childrearing, these differences in opinions may introduce gender bias.

¹⁰ Lim, Benbasat & Ward 2000; Lindsay & Norman 2013; Rivera 2012

¹¹ Judge & Cable 2004; Rivera 2012

¹² Lewicki et al. 2016

Algorithms: Eradicating or exacerbating gender bias?

These discussions and group rankings guided us to develop a suite of automated and semi-automated algorithms to rate the candidates' suitability for each of these jobs. The idea of adopting hiring algorithms was initially grounded on the premises that, firstly, the absence of human intervention gives rise to *purported* impartiality and neutrality. Secondly, technology enables optimal efficiency and accuracy in sorting a massive volume of applications with minimum cost and for maximum benefit of the company¹³.

Theoretically, hiring algorithms *should* be able to create an optimum amalgam of excellent candidates based on pure meritocracy. However, decision-making algorithms -- such as these are designed to mimic how a human would, in this case, chose a potential employee -- ultimately work with associations, just as our human brains. Without careful risk mitigation, algorithms are not immune to gender bias; in fact, in some cases, they may instead exacerbate gender bias¹⁴.

Gender bias in hiring algorithms can occur in three forms:



Bias in datasets

The limitation of the datasets is a major factor in the algorithms' proneness to discrimination as it builds the scope of benchmarking a candidate against others.¹⁵ If the algorithms are fed with limited datasets to assess and rank candidates, the algorithms will transform their benchmark based on this *specific* set of data. Without proportional representation of gender and other protected attributes, datasets can introduce bias to algorithmic judgements. Simply, if the data lacks enough female candidates, the algorithm will make hiring decisions based largely on male attributes.

Bias in the system

Correlational bias remains a major concern in algorithms, including those designed for hiring management¹⁶. One of the most prominent reasons for this is the use of proxy attributes to represent a parameter or an individual. For instance, algorithms make correlations between 'creativity' and the individual's length of employment within the same job¹⁷. They also make associations between higher levels of 'inquisitiveness' and their likelihood of finding other opportunities. When these correlations are made based on demographic traits, such as neighbourhood, race or gender, the algorithms are at risk for bias that can influence the whole corporate culture¹⁸. Therefore, when these proxies are embedded within the algorithms' judgement of suitability for employment, it will repeat the same societal bias. For women, this may lead to discrimination based on schooling, certain types of extracurricular activities, employment gaps for parental leave, and/or other correlated gendered characteristics. Another example will be the models used in natural language processing: these models, *trained* on large corpora of language data (from real-world news sites to webpages) will pick up any biased language usage, however subtle. As a result, these biases, in one form or another, will manifest themselves statistically in the language model¹⁹.

Bias in human decisions

Algorithms are generally trained to *record* and *memorise* past decisions and *learn* from them²⁰. This implies that the algorithms memorise the patterns of previous decisions and *can* replicate the patterns of these decisions. Without concrete mitigation plans, algorithms adopt human decision patterns and replicate them as a predictor of success metrics. Here, humans *may* rely on internalized gender bias to make hiring decisions or hiring panels may fail to include sufficient female representation, leading to gender bias codified in the algorithms. Ultimately, algorithms trained with human interference can replicate human bias.

¹³ Preuss 2017; Kulkarni & Che 2017

¹⁴ O'Neil 2016; Costa et al 2020; Kim 2019

¹⁵ Costa et al. 2020

¹⁶ Kim 2019

¹⁷ O'Neil 2016

¹⁸ O'Neil 2016

¹⁹ Chang, Prabhakaran & Ordóñez 2019

²⁰ O'Neil 2016



Lessons from the Amazon Case

Recall from the Literature Review document that in 2014, Amazon generated hiring algorithms to predict the suitability of applicants. The algorithms were trained using internal company data over the past 10 years²¹. Years after, it was then found that Amazon's hiring algorithms discriminated against female applicants.²² This bias was not introduced by the algorithms; rather, it was a consequence of the biased datasets that mirror the existing gender inequality in the workplace²³.

As the majority of Amazon's employees were Caucasian men, their hiring algorithms used this pattern as a determining factor of success, and therefore, discriminating against female candidates²⁴. Keywords such as "all-women's college" and "female" served as proxies that ranked female applicants lower²⁵.

Information Systems theory can also help explain the Amazon case. Research suggests that there is a reciprocal relationship between technologies, the organisational environment and organisational agents²⁶. When ranking algorithms for recruitment are trained with biased data sets, the technology impacts the organisation in a way that reflects the organisational operation, while at the same time influencing the way it operates. This means hiring algorithms trained with biased data can replicate existing inequalities while *also* introducing new ones.

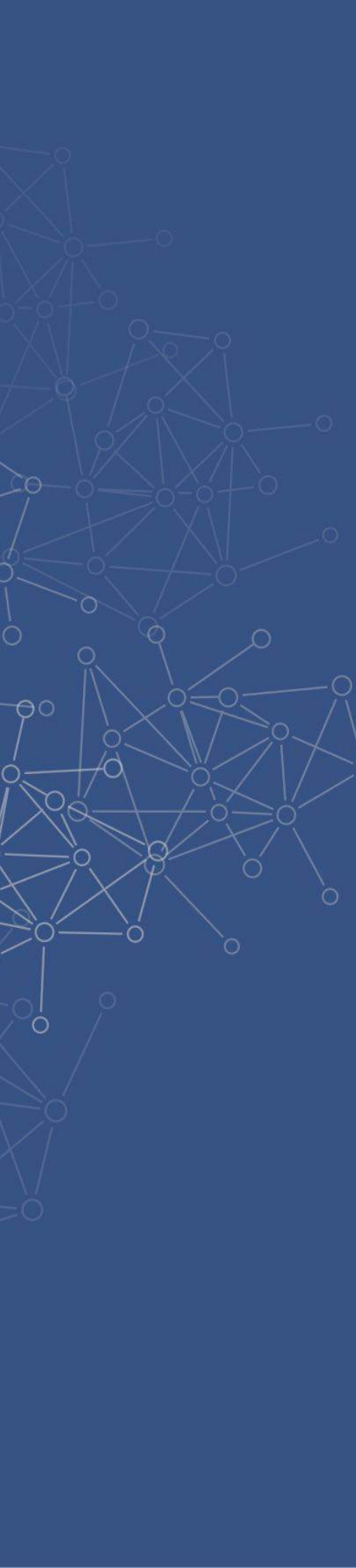
²¹ Costa et al. 2020

²² Bogen 2019; Dastin 2018

²³ Costa et al. 2020; O'Neil 2016

²⁴ Costa et al. 2020; Faragher 2019

²⁵ Orlikowski 1991



Data Science Approach: Machine Learning

Design decision

1

Design decision

2

Design decision

3

Prioritising explainability and interpretability over commercial accuracy

The choice of actual machine-learning algorithms used are to be as transparent and as *explainable*²⁸ as possible. Many existing algorithms do not show the math, logic or programming behind them. We prioritised transparency which allows us to scrutinise the algorithmic judgements— with emphasis on gender bias propagated from humans²⁹ – over the higher degrees of accuracy needed in a commercial or production-ready system. Hence, we refrain³⁰ from using complex techniques such as deep neural networks.

User-centred design and domain expert knowledge

To simulate a real-world user-centred systems development process, our data scientist programmer (co-author McLoughney) is *embedded* within the panel sessions conducted at Policy Lab (by co-authors Ruppanner and Njoto). Our data scientist is required to observe, ask questions and solicit input from panel participants – including assumptions, justifications, and clarifications – in order to gain domain knowledge of the task at hand³¹. This meant the hiring panel rather than the data scientist drove the logic of the algorithm specifications, while the data scientist's role is on the technical facets - such as techniques for data wrangling, choice of model, and programming decisions.

Use of open source tools as opposed to black-boxing

In our data science pipeline (from encoding candidates' experiences against the desired job qualifications, to production of the final model), existing external programming libraries and packages are used in the algorithm. These *de facto* standard tools are commonly used in data science and machine learning. In the spirit of *open source*, they are developed by a community of Python programmers who frequently review the code for issues and actively maintain the code to be of a high standard³². This means we are using well-accepted techniques for language processing in the field.

²⁷ Refer Footnote 1 for context

²⁸ Miller 2018; Cheong & Leins 2020

²⁹ Simply put, the decisions made by the human panel are the 'truth', as far as the algorithm is concerned, during its training phase to recognise and provide a correct judgement

³⁰ Rudin 2018

³¹ See <https://www.usability.gov/how-to-and-tools/methods/focus-groups.html>. Note that the remaining co-authors are careful not to interfere in the design process, to preserve independence from experimenter bias (such as a priori or pre-established hypotheses) but are on-hand to provide technical ideas and guidance

³² Take, for instance, the scikit-learn library used for machine learning projects – it is actively developed and maintained in an open source fashion on GitHub. A full list of such external libraries is before Section 1 in the Jupyter Notebook Pauli et al. 2020; Honnibal & Montani 2017; McKinney 2010

Experimental Design and Key Tasks

A high-level overview of the key tasks performed by our algorithm are as follows. To preserve clarity, technical details are abstracted from the explanation. (References to the Jupyter Notebook are provided for completeness).



Task 1

To ascertain differences between sets of rankings, a standardised method to mathematically compare sets of candidate rankings is pre-defined. To elaborate, we need a way to “aggregate the data to get an overall ranking because the data points are not statistically independent” (Jupyter Notebook Section 1) due to the nature of the ranking exercise (there can only be a single rank of e.g., ‘1’ per set of rankings), which causes techniques such as average calculation to fail. This aggregation is therefore required for our proposed machine learning techniques³³.



Task 2

To compare a candidate’s suitability with a job listing’s key selection criteria, we need to firstly define processes to calculate resumé suitability. This includes analysing key terms in the resumé, previous job durations, and education, amongst others. (Jupyter Notebook Section 2). Highlights from this task include:

- Keyword detection and matching to extract and compare key terms in a candidate’s resumé to the key selection criteria. For example, a job in “human resource[s]” must require its applicant to have either a degree containing the qualifier “bachelor” or “master” to be ranked as higher for the position.
- Calculate a *weighted metric of relevant experience* by weighting the time a candidate spent in a previous role (total number of months) with that role’s similarity to the currently advertised position. This similarity is calculated using a technique called *word vectorization*³⁴. Word vectorization intuitively groups together similar words, as these words are mathematically more likely to co-occur close to each other, based on common statistical patterns in existing real-world texts (for our implementation, we use the Python *spaCy* library³⁵). For example, refer to Figure 4, which is a graphical illustration to show how the technique -- given a list of words -- intuitively groups together similar words (colours, occupations, and vehicles) in common clusters.

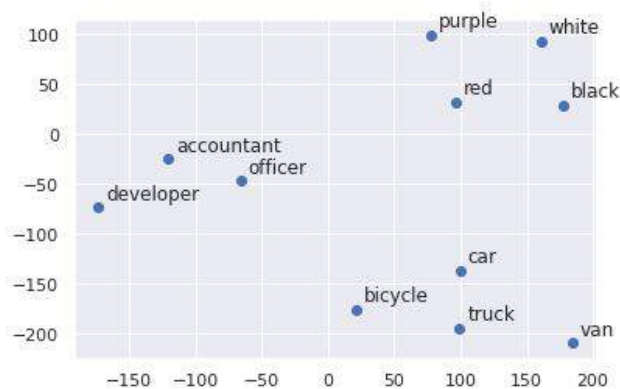


Figure 4. A graphical illustration of the word vectorisation technique in Experimental Task 4. Note that words which are *semantically* similar are grouped together.

³³ Ailon, Charikar & Newman 2008

³⁴ A technical exposition and simple example of the technique is illustrated in Jupyter Section 2.2.1. A brief primer on the technique with concrete examples is available at

< <https://towardsdatascience.com/understanding-nlp-word-embeddings-text-vectorization-1a23744f7223>>

³⁵ The exact language model is *en_core_web_md* (per Jupyter Notebook), and documentation on the language model is at < <https://spacy.io/models/en/>>.

- c. As common resumé formats (such as Microsoft Word's DOCX) are catered towards human readability (rather than machine analysis), a manual extraction process is required to convert them into JSON³⁶, which can be used directly by our algorithm. (Note that a manual approach is preferable to an automated one: the implementation of an automated process is beyond the scope of this project due to the time and resources required). This allows us to scrape the information in the CVs into our program for analysis.

Task 3

The processes in the prior step are applied to the resúmes studied by the human panel (Jupyter Notebook Section 3), and sample statistics of the key features e.g., distribution of relevant experience are calculated (Jupyter Notebook Section 4). The calculation of sample statistics helps us discover key features of the data – such as statistical properties of the distribution of human judgements – and helps us check our code for validity.

Task 4

Linear regression is applied to the rankings by human panellists, accounting for different permutations of gender (Jupyter Notebook Section 5). Linear regression is chosen as our predictor as it fits the key design decisions (it is more explainable and an efficient open source implementation – *scikit-learn* - is widely-used for Python), and as it is one of the most popular supervised ML algorithms used³⁷. In statistics, it is a classical technique (going back to Gauss & Legendre), with a track record of over 200 years.

Task 5

Finally, an unsupervised classification algorithm is applied onto the outputs of Task 2 above, in an experiment to illustrate how human panellists' influence can be removed from a hiring algorithm by removing the panellists' rankings altogether from the training data. In other words, an experiment in using purely statistical grouping techniques to find commonalities between candidates. The *k*-means clustering algorithm is chosen for this task: in a nutshell, given a number *k*, it attempts to find *k* 'clusters' of common data points (individual resúmes) which are statistically like each other. (Jupyter Notebook Section 6). The *k*-means algorithm is selected again based on the key design decisions (it is explainable, widely used, and efficiently implemented), and due to its long history (over 35 years old) of use in data mining and data science.

³⁶ JavaScript Object Notation is a common format used in machine learning and data science. See < <https://www.json.org/json-en.html> >

³⁷ Refer to a review by Doring (2018) in KD Nuggets < <https://www.kdnuggets.com/2018/12/supervised-learning-model-popularity-from-past-present.html> >.

Experimental Results

Can the computer match human decision-making?

Our first question is important because it helps us determine whether our supervised algorithm can match the human rankings - *can we replicate the human panellists' behaviour?* We review the findings from Task 4, i.e., *linear regression*. We calculate the R-squared statistic (Jupyter Notebook Section 6), in other words, the statistic which indicates how much the abstract model is capturing the human behaviour.

Figure 5 shows the R-squared value for each of the predictor models, which we express as percentages. When the value reaches 100%, it means that the human judgements and the predictor judgement overlap completely, i.e., a perfect fit between the model and the human decisions.

This means the higher the percentage, the more consistent the predictor is to mimicking decision-making in our human panel.

We found that the linear predictor could accurately match human panellists' decision-making in two jobs – the data analyst position and the recruitment officer position. For the recruitment officer position, the machine and the human only produced similar rankings when the candidates were matched to their original gender. Once the genders were flipped, the predictor underperformed.

As seen on figure 5, we have identified several key questions: *What is going on here – why does the algorithm become less predictive of the human panel? Is the bias in the algorithm itself or the humans?*

We categorise our hypotheses in two broad categories – Machine Bias (MB) and Human Bias (HB) – contextualised in our understanding of how such classification or prediction automation techniques can cause gender bias, in a broader sense.

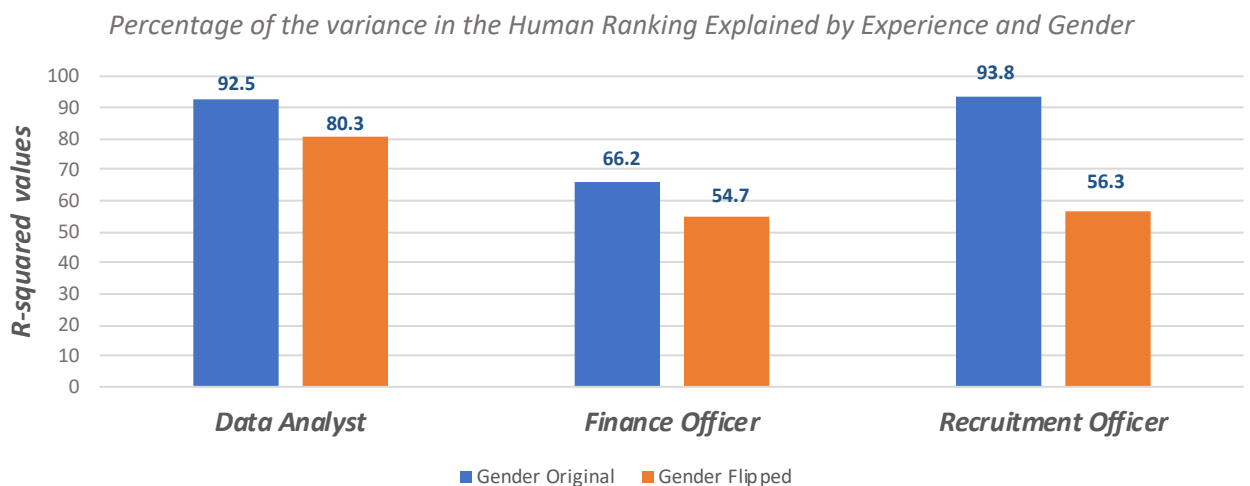


Figure 5. R-squared values for the linear regression models, according to CV gender, per industry. The R-squared values determine correlation between human rankings and linear regression outputs and are expressed as percentages for convenience.



Machine Bias Could Influence Algorithmic Outcomes in Three Meaningful Ways

For the following hypotheses, *let's assume that the machine is working alone without any human influence*. Questions are: *Can it introduce bias? And, if so, do we have any indication that our classifier/predictor is doing this for the sample CVs provided by UniBank?*



Hypothesis MB 1

Keyword model may have inherent gender bias

We model our candidates' work experiences based on a keyword matching technique (Experimental Task 2) that captures relevant work experience that includes: (1) time in relevant work and (2) experience that matches search criteria.

Although we are using industry standard classifier and predictor algorithms (linear regression and *k*-means respectively) and natural language processing techniques (word vectorization in *SpaCy*), these may introduce gender bias *if the data used to train these algorithms are biased*.

To elaborate this point, we are *not focusing on the ranking data from the human panel*, but rather the data model used for mapping words in resumés to an experience score. Specifically, used in Task 2 for computerised grouping of statistically similar terms are trained on 'standard' de rigueur data science corpora.

These corpora, *OntoNotes/GloVe*³⁸, were trained on "...various genres of text (news, conversational telephone speech, weblogs, usenet³⁹ and a large collection of web data archived over the past few years (including Wikipedia)⁴⁰". The potential for gender bias can arise from human tendency to use gendered language in daily communicues, which in turn can be 'encoded' – in some shape or form – in the resulting model used for word vectorisation.



Evidence for MB 1

Model is introducing gender bias for UniBank Sample

Not enough information; we cannot specifically identify these biases as state-of-the-art language models are trained across big data sets with billions of cases, but we must acknowledge that they exist⁴¹. A thorough dissection of the biases in language models are an ongoing topic of research in the field of natural language processing and are far beyond the scope of this research. It will require a large multi-year project to understand how and why these biases are encoded, and to understand how we can build 'big data' corpora that minimises such biases.

Another possibility is to build our own algorithm with concrete specifications to mitigate gender bias, while acknowledging that bias exists in language models. The latter point is, however, a possible direction for future research (see Future Research).

³⁸ According to < <https://spacy.io/models/en> >, the sources used are specifically *OntoNotes 5* < <https://catalog.ldc.upenn.edu/LDC2013T19> > and *GloVe Common Crawl*
³⁹ Weischedel, Ralph et al. 2013
⁴⁰ Pennington, Socher & Manning 2014
⁴¹ Bender & Friedman 2018



Hypothesis MB 2

Algorithm is applying a different ranking for experience and education for our original and flipped CVs

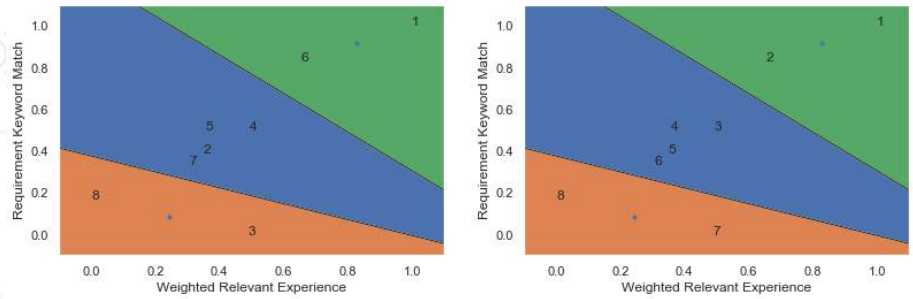
A second way the algorithm could introduce bias into our sample is if it weights experience and education *differently* for the original and gendered flipped CVs. We apply the same keyword ranking logic for both types of resumés, but it is possible that a classifier/predictor starts to register gender differently across these CVs based on the name of the candidate.

To illustrate this hypothesis, we use the unsupervised *k*-means algorithm (Experimental Task 5) which tries to cluster 'similar' CVs based on the dimensions of relevant experience and requirement keyword matching. Compared to linear regression, *k*-means *does not know the human panellist's rankings at all*, and instead tries to assign each CV (data point) to one of several clusters (analogous to how Figure 3 groups together 'alike' ranks). The value of *k* is chosen by the programmer to deliver an optimal clustering⁴²; in this experiment, a value of *k* of 3 or 4 is chosen.

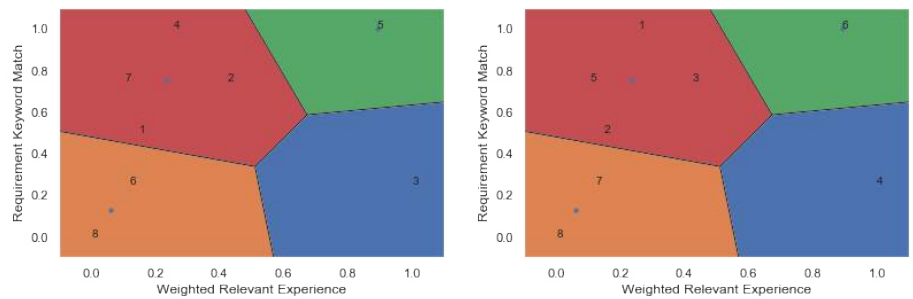
*Please note that this is only a proof-of-concept created purely as a value-add to the project, for the sake of discussion. It is not suitable for a production-ready nor commercial application and has not been rigorously tested. A thorough examination / investigation of the *k*-means method is beyond the scope of this project.*

Figure 6 illustrates the clustering result of *k*-means in two-dimensional space, with the numbers illustrating the different CV rankings, provided solely for information (and to reiterate, is *not* used in the training of the algorithm). If this hypothesis - *experience and education are weighted differently for the original and gendered flipped CVs* - were true, we would instead see in Figure 6 that the classifier ends up contributing a disproportionate share of experience to the different models, i.e., the data points will have a totally different distribution depending on gender.

Data Analyst (Left: Original Genders; Right: Flipped Genders)



Financial Officer (Left: Original Genders; Right: Flipped Genders)



Recruitment Officer (Left: Original Genders; Right: Flipped Genders)

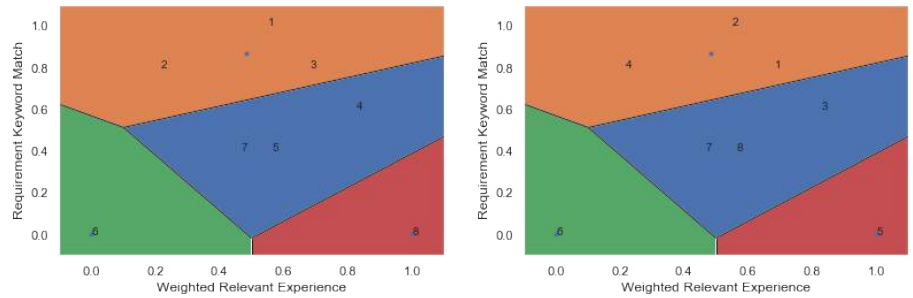


Figure 6. The Voronoi maps for outputs of *k*-means classification, purely on requirement keyword matching and weighted experience matching (Experimental Task 2).



Evidence for MB 2

Algorithm is applying a different ranking for experience and education for our original and flipped CVs for UniBank sample

There is no evidence to support this claim, per Figure 6, as *k*-means performs the same clustering (illustrated by the different coloured polygons) for both the original and flipped samples. The distribution of data points (calculated statistics) does not differ either.

⁴² There are many techniques to select appropriate values of *k*, the details of which are beyond the scope of this report. A good reference is Pham, Dimov & Nguyen 2005



Hypothesis MB 3

Women and men bring different levels of experience that, over time become amplified in the algorithm to discriminate against women

A third way hiring algorithms can introduce gender bias is if the type of data that were originally used to train the algorithm have gender differences. Over time, the machine reinforces and amplifies these gender differences *if they are identified as important for hiring a successful candidate.*

Women's disproportionate share of caregiving can lead women to reduce or exit employment. This gender difference is an integral way that women can be disadvantaged in hiring as women may exhibit: (1) less relevant experience; and (2) fewer employment skills to match selection criteria. These gender differences used to initially develop the hiring algorithm can become amplified over time leading men to hold greater hiring advantage.

To extrapolate the idea above, a commercial production algorithm will be trained on many data points of human judgements, and often uses sophisticated algorithms such as deep neural networks (implemented using e.g., Tensorflow or Keras) to *optimise for accuracy* at the expense of transparency, explainability, and interpretability. Even the slightest bias present into the initial set of data points will be 'ingrained', or 'encoded', in resulting neural network models. This means the bias may be even more entrenched in these algorithms than our simple classifier/predictor. Our experiment illustrates how bias in a model can be introduced at inception and will slowly build over time.

To further elaborate, even though it takes way more time for a human to review a CV (in a magnitude of minutes, compared to, say, a magnitude of seconds for a computer), in the process of manually reading and ranking CVs, a human has the added advantage of reflection and review. For example, a human can review their past judgements in light of new evidence (e.g. feedback from colleagues on a new hiring policy, or strong referee reports for an otherwise borderline candidate); whereas a machine's model will simply use the existing pre-trained model unless otherwise instructed.

Figure 7 shows that men in our sample hold more relevant experience measured in months than women (cf Python Notebook Section 4). Men and women are similarly qualified for the position based on our keyword model with men slightly less qualified at the low ends but slightly more represented at higher values. These gender differences are being picked up by our algorithm, across all industries, even for a small sample of UniBank CVs. Over time, a classifier/predictor trained on it may amplify gender differences advantaging men in their relevant experience. Also, based on our observations, it is pertinent to note that algorithmic classifiers/predictors for hiring may exacerbate and ‘bake in’ a broad spectrum of gender biases, especially if they are touted as a ‘one size fits all approach’. A hypothetical example of such a worst-case is when a single hiring algorithm uses training data from many CVs from an assortment of different industries, rather than optimising for hiring outcomes within specific industries.



Evidence for MB 3

Women and men bring different levels of experience that, over time become amplified in the algorithm to discriminate against women

There is partial evidence that men will be favoured by the classifier/predictor in terms of relevant experience, and that women may hold higher keyword qualifications. Hence, we identify that there is a potential these could be amplified over time and a direction for future research to identify magnitude of impact for UniBank hiring process (see Future Research Directions later in the report).

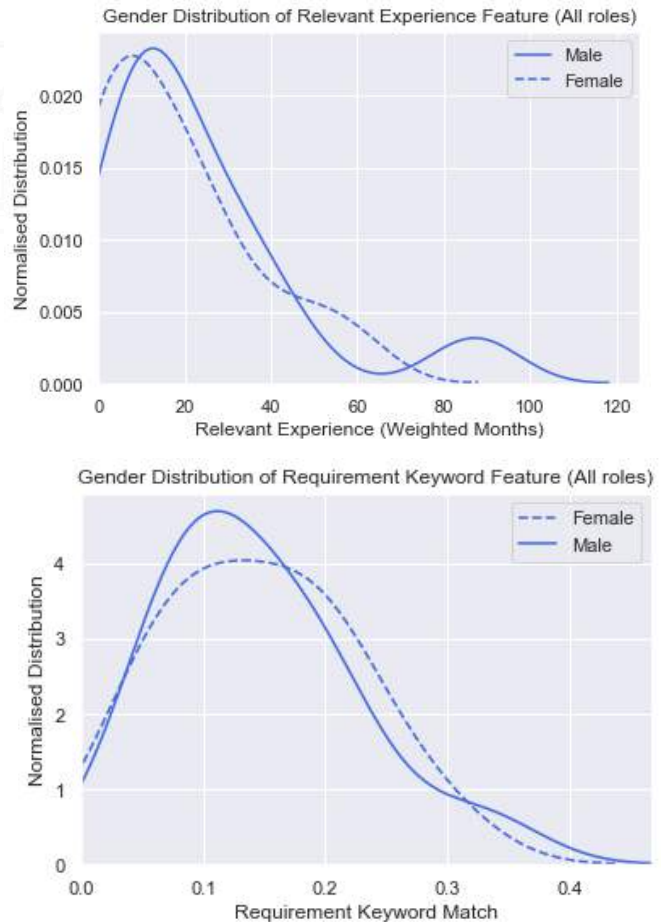
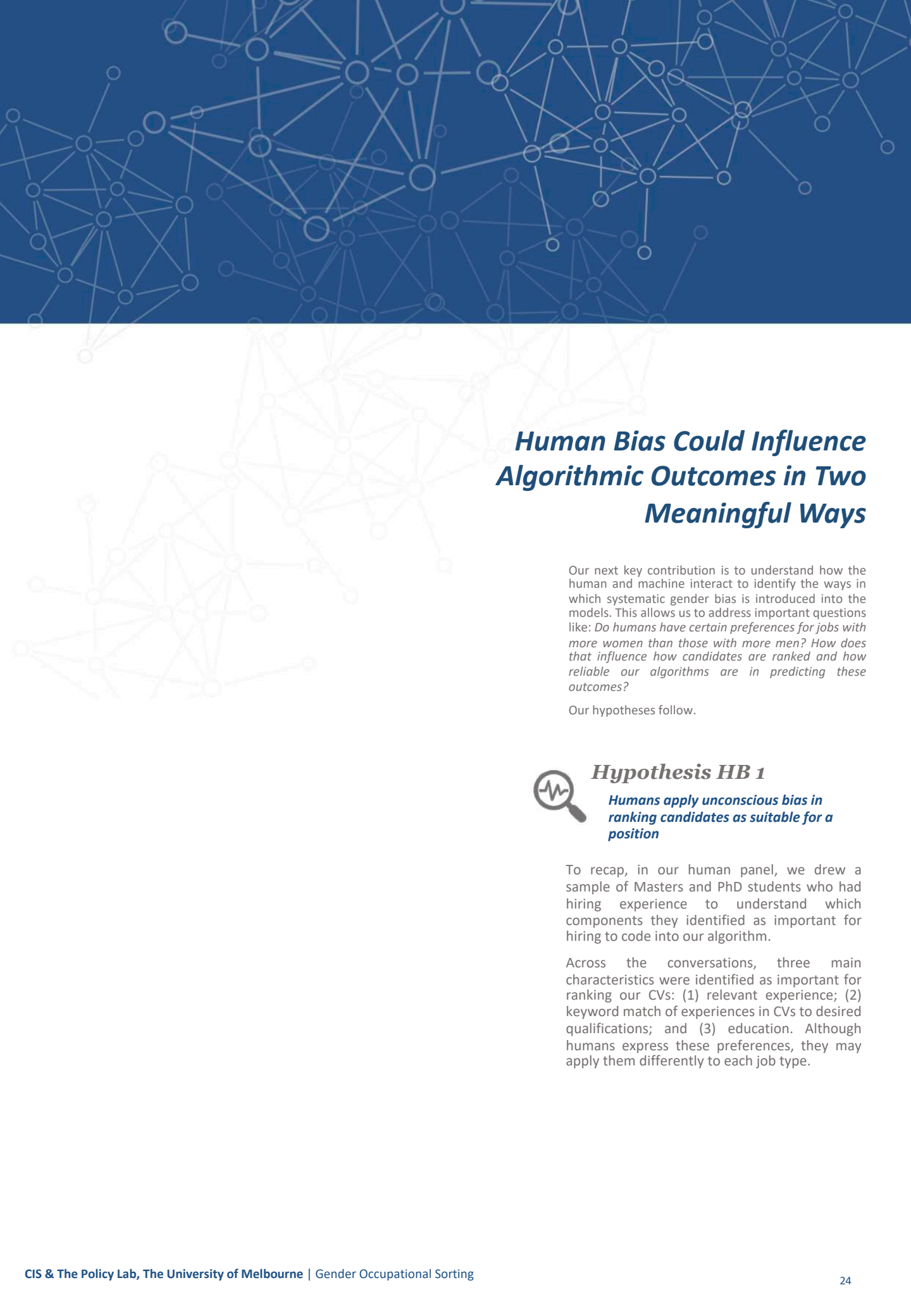


Figure 7. Statistical distribution of (left) relevant experience features and (right) requirement keyword features found in CV data, based on the outputs of Experimental Tasks 2 and 3.



A network diagram consisting of numerous white circles of varying sizes connected by thin white lines, set against a dark blue background at the top and a light blue background at the bottom. The circles and lines form a complex, interconnected web.

Human Bias Could Influence Algorithmic Outcomes in Two Meaningful Ways

Our next key contribution is to understand how the human and machine interact to identify the ways in which systematic gender bias is introduced into the models. This allows us to address important questions like: *Do humans have certain preferences for jobs with more women than those with more men? How does that influence how candidates are ranked and how reliable our algorithms are in predicting these outcomes?*

Our hypotheses follow.



Hypothesis HB 1

Humans apply unconscious bias in ranking candidates as suitable for a position

To recap, in our human panel, we drew a sample of Masters and PhD students who had hiring experience to understand which components they identified as important for hiring to code into our algorithm.

Across the conversations, three main characteristics were identified as important for ranking our CVs: (1) relevant experience; (2) keyword match of experiences in CVs to desired qualifications; and (3) education. Although humans express these preferences, they may apply them differently to each job type.

Table 1 shows which of these three components were significant and improved the linear model's fit in our regression analyses. These differences show that, although our recruitment panel discussed these three components as important, they only used a *subset* of them to rank candidates for the specific jobs, as identified by summary statistics from the analyses.

For the *data analyst* position, relevant experience and keyword match ultimately determined the human panellists' ranking; education level was not important. For the financial officer position, none of the characteristics the human panel indicated were important predicted their ranking of candidates. For the recruitment officer position, keyword requirements and educational qualifications predicted their rankings, but relevant experience proved non-significant.

These differences are important because they indicate that the human panel is *inconsistent* in matching stated desires to actual rankings which suggests these biases are unconscious.

These biases have implications for women in multiple ways. First, women hold more educational qualifications compared to men. Our results suggest these may not matter for industries with more men (i.e., data analyst and finance recruitment office jobs). Second, women may have less relevant experience due to carer-driven career disruptions which may make them less appointable to male dominated jobs like Data Analyst, but more appointable to a female-dominated industry like Recruitment Officers. Finally, our panel ranked our candidates for the Financial Officer job based on criteria not captured by education, experience or keyword qualifications. This means the hiring committee liked something else about these resumé that we did not capture in our rank predictor. This job appears to have the most potential to be gender biased which is of note for a financial institution like UniBank.

Table 1. Characteristics of Candidates Human Panellists Used to Rank Applicants

	<i>Relevant</i>	<i>Not Relevant</i>
Data Analyst	1) Relevant Experience; 2) Keyword Match	1) Education
Finance Officer		1) Relevant Experience; 2) Keyword Match 3) Education
Recruitment Officer	1) Keyword Match 2) Education	1) Relevant Experience



Evidence for HB 1

Humans apply unconscious bias in ranking candidates as suitable for a position

Strong. We find that the human panel was inconsistent in applying equivalent logics to rank candidates. If hiring algorithms use expert-driven human rankings as one component of their development (as seen in e.g. Experimental Task 4), unconscious bias with potential to discriminate against women will be introduced. We have evidence that women will be disadvantaged in the male dominated professions – Data Analyst and Finance Officer.



Hypothesis HB 2

Humans exhibit a stronger preference for male candidates

Existing sociological research shows men are advantaged in employment *even when they work in female-typed jobs with more women*. We may find our human panel expresses a strong preference for the male candidate regardless of their education, experience or keyword match.

Our regression results (available in the Jupyter Notebook Section 5) show that gender bias for the male candidate is evident in the two positions employing more men: (1) Data Analyst and (2) Financial Officer.

When the gender was flipped or presented with a female name but male experience. What is more, when we flipped the genders, relevant experience and keyword match were no longer predictive of our candidates' success. This means our panel preferred CVs with women's names only when their resumé had men's experiences.

As Figure 8 shows, CVs with a man's characteristics but a woman's name were 2.5 places better ranked than a man's name with a woman's experience. This suggests that when gender expectations are violated, our panel gave preference to characteristics in the men's CVs even if they had a woman's name.

For our *Finance Officer* position, we found that our panel preferred men over women for this role by 4 ranks (Figure 9). What is notable about this job is that experience, keyword match and education did not predict the ranking of candidates (see Hypothesis HB1 above). This means there is something distinct about the men's resumé that made our panel rank them higher, beyond experience, qualification and education. This forms the most alarming dimension of gender bias, as we are not capturing what gives men the edge in these positions.



Figure 8. An average woman with male characteristics ranked 2.5 places higher than average men in Data Analyst job



Figure 9. An average woman with average experience and education ranked 4 places lower than average men in Finance Officer job

For the *Recruitment Officer* position, we do not find the gender of the applicant is significantly associated with higher or lower ranks. But we do find that the predictability of the flipped gender linear model reduces dramatically, compared to those of the other roles. Figure 10 illustrates the difference between R-squared values, when the genders are flipped, for all three industries: the values indicate the net loss of R-squared values, derived from the raw data in Figure 5 earlier.

Difference in R-squared values for linear regression, after gender is flipped

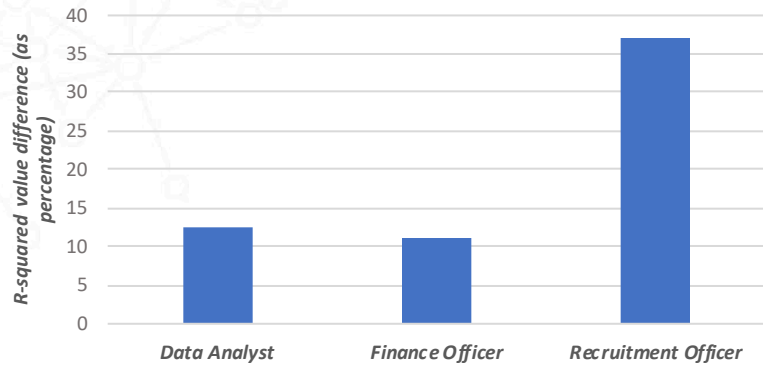


Figure 10. Difference in R-squared values for linear regression models, after the gender is flipped. Note that the Recruitment Officer role’s linear model has more than three times the loss of prediction accuracy, compared to the other two models’ R-squared loss.



Evidence for HB 2

Humans exhibit a stronger preference for male candidates

Strong, but only in male dominated professions. We find that the characteristics of the male CVs were strongly favoured by our recruitment panel in positions that employed more men. We find flipping the gender for the female dominated job – Recruitment Officer – dramatically reduced model fit which suggests our panel was impacted by this change.



Discussion and Future Work

This project provided preliminary insight into how gender bias is introduced into hiring algorithms. By bringing together a hiring panel to rank real world CVs, we developed an algorithm to match their preferences. This included the strong specification to match experience against desired qualifications, and 'embedding' of the data scientist into the Policy Lab. Even with this small dataset and simple algorithmic logic, we found gender bias was introduced into some of the hiring outcomes.

We found the results for the Financial Officer position proved the most troubling. Our panel preferred men for this position regardless of their educational attainment, relevant experience and keyword match to hiring specifications. The size of this effect was large – our men were ranked 4 points higher than women as suitable for this job. This means that our panel had a strong preference for men in this position regardless of training, ability or match to the job. If human decision-making logics were built into a hiring algorithm, women would be much less successful in accessing an interview for this job. Our argument is that any amount of initial bias – no matter how little -- will perpetuate in the algorithmic models used, further continuing the bias, and thus reinforces it by keeping human oversight out of the loop. We are relying on a small sample of data, but this type of gender bias has been documented in large companies like Amazon. For UniBank, for whom financial positions are core business, this may lead to gendered job sorting based on human-preference rather than individual ability. This is a drain on productivity and human capital maximization, especially since the Australian Bureau of Statistics shows this occupation to have the greatest gender balance.

We also found gender bias in the male dominated position of Data Analyst, whereby women's resumés were ranked higher only when they had male characteristics. This suggests a degree of gender bias against women in this space. Also, our human panel valued relevant experience and keyword match but not educational qualifications in ranking our candidates for this role. Since women are more likely to experience career disruptions which lead to less relevant experience, women may be particularly disadvantaged in these job types.

The Recruitment Officer job, which heavily employs women, exhibited some interesting variations with implications for women's employability. First, only keyword requirements and educational attainment significantly predicted the ranking of candidates. Relevant experience, which captures employment disruptions, did not predict rankings even though our panel viewed it as important. This means women caregivers should experience less disadvantage in hiring in these jobs. Second, we found the model fit weakened by half when we flipped the genders. This means something about the experience of reading a woman's resumé with a male name led the hiring panel make less consistent decisions tied to keyword requirements. This suggests that our panel wants men and women with keywords that match their gender for this position.

Our results clearly indicate that the human panel holds unconscious bias that introduces gender bias into the models. The machine also has potential to compound this disadvantage by ranking keywords and experiences against gendered language. We also show that men had slightly more experience and women better match to keyword requirements. Each has the potential, if extrapolated over a larger sample of data, to introduce, replicate and reinforce gender bias in hiring. These trends are particularly troubling given that, for commercial systems, these decisions are often made in "black boxes" meaning we cannot see, measure or understand how and the scope of gender bias in existing commercial hiring algorithms. An organization like UniBank should be aware of this potential bias: we show here how simply humans can introduce bias into an algorithm using three reasonable ranking dimensions: education, keyword requirements, and relevant experience.

Our research project provides an initial snapshot into the process through which gender bias is introduced into algorithms. It is an integral first step that indicates clear directions for future research as outlined below:



Future direction 1
Mapping bias across all roles

Expanding analyses to estimate bias in hiring across a comprehensive dataset of occupations within the UniBank corporate structure. We are drawing upon three key roles to understand gender bias in algorithms for those that are *male-dominated*, *gender equal*, and *female dominated* but a comprehensive study with resumés from all occupations would illuminate the scale of the problem.



Future direction 2
Accounting for corporate culture

Our project provided a “mock” hiring panel to contextualize decision-making in this process. Embedding within the UniBank hiring panels would allow us to more accurately estimate existing bias and corporate culture.



Future direction 3
Linking hiring algorithms to actual hiring decisions

Hiring algorithms are only one component of a recruitment process and shouldn't be touted as a ‘one size fits all’ approach. The initial ranking of candidates can introduce bias – as we have seen in our experiments -- but so can the proceeding processes which further entrench algorithmic bias. Providing a whole of process evaluation that includes final hiring decision would allow us to more accurately isolate the role of algorithmic against human judgements.



Future direction 4
Legal, social and policy analysis to ‘automated’ discrimination

Expanding social and legal theory to which *decisions* made on the basis of gender are considered *discrimination* on the basis of gender. This analysis will allow a concrete formulation of policy response that is coherent in its assessment and procedures against an ‘automated discrimination’. It offers a policy basis to a strategic algorithmic model that balances both fairness and accuracy in a decision-making.



Future direction 5
Rethinking learning techniques for data and computer scientists

We need approaches that avoid the economic loss and injustice of women not being invited to participate in the workforce to the right extent. Just as a society evolves and, tends to question its biases, conscious and unconscious, we can design our learning models to “check their biases” over time. Automating oversight, self-reflection, input from moderators, and incorporating established techniques such as quotas should not only mitigate the effect of accelerating the effect of bias, but also support society in hiring more fairly. Like any improvement, it is unlikely to be easy or consistently successful, but it is achievable⁴³.

⁴³ Kleinberg, J. & Raghavan 2018



Conclusion

Recommendation 1

Provide training programs to introduce human resource professionals to the potential of gender bias in algorithmic judgements in hiring process.

Recommendation 2

Complete regular audits of hiring by gender across all positions to identify potential roles that are vulnerable to gender discrimination.

Recommendation 3

Create established quota systems for hiring to ensure women are not excluded from male-dominated or gender balanced professions based on hiring biases.

Recommendation 4

Create proprietary hiring algorithms that are transparent and trained with the aim of reducing gender bias in hiring, with regular audits of algorithm output and models (both trained on human judgment as well as unsupervised).

In conclusion, this report summarises the factors – from both the sociological and the technical perspectives – on how and why gender bias can be exacerbated when algorithmic systems form part of the decision-making process.

Moving forward, our analysis has uncovered several recommendations and best practices to reduce gender bias – both existing and anticipated – resulting from the interplay between human and algorithmic judgement.



Bibliography

- Ailon, N., Charikar, M. and Newman, A. (2008). Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)* 55(5): 1-27. Available at: http://dimacs.rutgers.edu/~alantha/papers2/aggregating_journal.pdf.
- Bailey, A. H., LaFrance, M., and Dividio (2018). Is Man the Measure of All Things? A Social Cognitive Account of Androcentricism. *Personality and Social Psychology Review*, 23(4): 307–331. DOI: 10.1177/1088868318782848.
- Bender, E. M., & Friedman, B. (2018). Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6: 587-604. DOI: doi.org/10.1162/tacl_a_00041.
- Bogen, M. (2019). All the ways Hiring Algorithms Can Introduce Bias. *Harvard Business Review*. Available at: <https://hbr.org/2019/05/all-the-ways-hiring-algorithms-can-introduce-bias>.
- Burgess, N. (2013). The Motherhood Penalty: How Gender and Parental Status Influence Judgements of Job-Related Competence and Organizational Commitment. *Seminar Research Paper Series*, paper 32. Available at: https://digitalcommons.uri.edu/cgi/viewcontent.cgi?article=1035&context=lrc_paper_series.
- Cheong, M. and Leins, K. (2020). Who Oversees the Government? Modernising Regulation and Review of Australian Automated Administrative Decision-making', in Boughey, J. and Miller, K. (eds) *Government Automation and Public Law Project Workshop* (forthcoming).
- Chang, K. W., Prabhakaran, V. and Ordonez, V. (2019). Bias and fairness in natural language processing, presented at the Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): Tutorial Abstracts, available at: <https://www.aclweb.org/anthology/D19-2004/> (accessed 17 September 2020).
- Cohen, S. L. (1976). The basis of sex-bias in the job recruitment situation. *Human Resource Management*, 15(3). DOI: 10.1002/hrm.3930150303.
- Costa, A., Cheung, C. and Langenkamp, M. (2020). *Hiring Fairly in the Age of Algorithms*. Research Paper, Cornell University.
- Dastin, J. (2018). Amazon scraps secret AI recruiting tool that showed bias against women. *Reuters*. Available at: <https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G>.
- Faragher, J. (2019). Is AI the enemy of diversity?. *People Management*, 6 June, viewed 20 April 2020. Available at: <https://www.peoplemanagement.co.uk/long-reads/articles/is-ai-enemy-diversity>.
- Gaucher, D., Friesen, J. and Kay, A. (2011). Evidence That Gendered Wording in Job Advertisements Exists and Sustains Gender Inequality. *Journal of Personality and Social Psychology*, 101(1): 109–128. DOI: 10.1037/a0022530.
- Han, J., Kamber, M., and Pei, J. (Eds). (2012). *Data Mining: Concepts and Techniques (3rd Edition)*. Elsevier. DOI: 10.1016/B978-0-12-381479-1.00017-4.
- Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. Judge, T. A., and Cable, D. M. (2004). The Effect of Physical Height on Workplace Success and Income: Preliminary Test of a Theoretical Model. *Journal of Applied Psychology*, 89(3), 428–441. DOI: 10.1037/0021-9010.89.3.428.
- Kleinberg, J. and Raghavan, M. (2018). Selection Problems in the Presence of Implicit Bias', in *Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany*. DOI: 10.4230/LIPICS.ITCS.2018.33. Available at: <https://drops.dagstuhl.de/opus/volltexte/2018/8323/pdf/LIPICS-ITCS-2018-33.pdf>.
- Kim, P. T. (2019). Big Data and Artificial Intelligence: New Challenges for Workplace Equality. *University of Louisville Law Review*, 57: 313–328. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3296521.
- Kulkarni, S. and Che, X. (2017). Intelligent Software Tools for Recruiting. *Journal of International Technology & Information Management*, 28(2): 1–16. Available at: <https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=1398&context=jitim>.
- Lewicki, R., Polin, B. and Lount Jr., R. B. (2016). An Exploration of the Structure of Effective Apologies. *Negotiation and Conflict Management Research*, 9(2). DOI: 10.1111/ncmr.12073.

- Lim, K. H., Benbasat, I. and Ward, L. M. (2000). The Role of Multimedia in Changing First Impression Bias. *Information Systems Research*, 11(2): 115–136. Available at: <https://www.jstor.org/stable/23015878>.
- Lindsay, P. H., Norman, D. A. (1977). *Human Information Processing: An Introduction to Psychology*, Academic Press.
- McKinney, W. (2010). Data structures for statistical computing in python, *Proceedings of the 9th Python in Science Conference*, 445.
- Miller, T. (2018). Explanation in Artificial Intelligence: Insights from the Social Sciences. *Artificial Intelligence*, 267: 1–38. DOI: 10.1016/j.artint.2018.07.007.
- O’Neil, C. (2016). *Weapons of Math Destruction*. Crown Books.
- Orlikowski, W.J. (1991). Integrated information environment or matrix of control? The contradictory implications of information technology. *Accounting, Management and Information Technologies*, 1(1): 9-42. DOI: 10.1016/0959-8022(91)90011-3.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perot, M., Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830. Available at: <https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf>.
- Pennington, J., Socher, R. and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *EMNLP*: 1532–1543. Available at: <https://nlp.stanford.edu/pubs/glove.pdf>.
- Pham, D. T., Dimov, S. S., & Nguyen, C. D. (2005). Selection of K in K-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1): 103–119. DOI: 10.1243/095440605X8298.
- Preuss, A. (2017). Airline pilots: the model for intelligent recruiting?. *Recruiter*, 12–13.
- Rivera, L. A. (2012). Hiring as Cultural Matching: The Case of Elite Professional Service Firms. *American Sociological Review*, 77(5): 999–1022. DOI: 10.1177/0003122412463213.
- Rudin, C. (2018). Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Natural Machine Intelligence*, 1: 206–215. Available at: <http://arxiv.org/abs/1811.10154>.
- Russo, N. F. (1976). The Motherhood Mandate. *Journal of Social Issues*, 32(3). DOI: 10.1111/j.1540-4560.1976.tb02603.x.
- Sczesny, S., Formanowicz, M., Moser, F. (2016). Can Gender-Fair Language Reduce Gender Stereotyping and Discrimination. *Frontiers in Psychology*, 7(25): 1–11. DOI: 10.3389/fpsyg.2016.00025.
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, E. J., Polat, I., Feng, Y., Moore, E., VanderPlas, J., Laxalde, D., Perktold, Cimrman, J. R., Henriksen, I., Quintero, E.A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17(3), 261-272.
- Weischedel, R., et al. (2013). OntoNotes Release 5.0 LDC2013T19. Web Download. Philadelphia: Linguistic Data Consortium.



Appendices



CIS & Policy Lab, The University of Melbourne
Interim Report for UniBank (Teachers Mutual Bank Limited).

Literature Review on Gender Occupational Sorting

The Role of Artificial Intelligence in Exacerbating
Human Bias in STEM Employment

26 June 2020

Table of Contents

Executive Summary	3
1.0. Current State of Hiring	4
1.1. Gendered Roles, Gendered Judgements.....	4
1.2. Hiring in STEM.....	5
2.0. Artificial Intelligence, Algorithms, and Bias	6
2.1. How do Decision-making and Classification Algorithms Work?.....	6
2.2. Algorithmic Fairness and Bias: A Primer	7
2.3. Contemporary Critiques on Automated Decision Making.....	8
3.0. Risks of Bias in Algorithmic Hiring	10
3.1. Case Study: Amazon’s Experiment Gone Wrong.....	10
3.2. Job sorting: translating history into future outcomes	11
3.2. Algorithmic Hiring: Risks for Women in STEM	12
3.3. Hiring and Protected Attributes.....	13
4.0. Research Directions	14
4.1. Hypotheses and Methods.....	14
4.2. Future Research	15
References	16
Project Contact	19

Table of Figures

Figure 1: Model Creation Process (Costa et al., 2020).....	6
--	---

Gender Occupational Sorting: The Role of Artificial Intelligence in Exacerbating Human Bias in STEM Employment

M. Cheong, R. Lederman, A. McLoughney, S. Njoto, L. Ruppner, A. Wirth¹

Executive Summary

This project investigates the extent to which algorithms that classify resumes can be biased against women. A key contribution is the development of a proof-of-concept algorithm to rank resumes. We use this to identify how AI techniques analyse patterns and perform classifications for new resumes, given a set of pre-established decisions on a prior set of resume data (i.e. *training data*). We introduce humans into the design to understand how bias – both conscious and unconscious - can become part of the decision-making process.

We use an innovative research design, drawing upon a panel of graduate students with experience in recruitment to rate fictitious job candidates against a job advertisement. The fictitious job candidates are represented by both simulated CVs and/or anonymised CVs provided by UniBank that are populated with fictitious biographic data. The panel members first rate the candidates alone and then as a combined panel. Based on the discussion and outcomes from the panel, we adjust our algorithm to their preferences to draw a new list of potential candidates. This allows us to compare multiple job candidate rankings – those done by our panellists without group input, those done by the group, and the group's outcomes which we will integrate into our AI system. This innovative approach allows us to understand where and how bias against women is introduced.

This approach is novel and innovative but builds upon a robust literature base on AI and gendered hiring that is outlined in more detail below.

¹ Authors are listed in surname-alphabetical order.

1.0. Current State of Hiring

1.1. Gendered Roles, Gendered Judgements

It is evident that women's position in society has shaped the way in which women are socially perceived (Sczesny *et al.* 2016). This translates into androcentric biases as a norm in describing both men and women (Hegarty & Buechel 2006). Androcentric pronouns and references are more generally used to describe humans in data and documentations. Men are perceived to be a more typical member and the standard of 'human' in general (Sczesny *et al.* 2016). On the contrary, in categories where women are overrepresented, women are perceived to be the typical member of the respective group; for instance, when it comes to describing 'parenting' (Hegarty & Buechel 2006). These inaccuracies and generalisation of data become problematic when trained into algorithms (Section 2.0) as they may generate results that are equally inaccurate or worse, expand the inaccuracies by making decisions blindly based on these data.

Key literature on the fairness of hiring often mentions the impact of language in gender discrimination, from the use of gendered words in job advertisements to the use of certain words in describing oneself in job applications (Stout & Dasgupta 2011; Gaucher, Friesen & Kay 2011). Gaucher *et al.* (2011) have found that there are gender differences in the use of language and psychological traits between men and women. It has been reported that women would refer to more communal words and utilise social and expressive words in comparison to men (Gaucher *et al.* 2011). Not only do men and women tend to use different adjectives to describe themselves, they also use different adjectives to describe other people according to gender. For instance, Schmader, Whitehead & Wysocki (2007, cited by Gaucher *et al.* 2011) have found that letters of recommendations that describe men contain words that describe 'prominence', such as 'outstanding' or 'unique'. On the contrary, letters that describe women suggest words that are contain more social and less directive connotations, such as 'warm' and 'collaborative' (Sczesny *et al.* 2016).

As evidenced by a landmark case study on Amazon's automated decision-making experiment for hiring (Section 3.1), the use of keywords that describe gender, such as 'women's chess club captain' or 'women's college' is highly critical (Dastin 2020). However, as indicated, it is also evident that similar patterns occur in the feminine and masculine choice of words when explaining oneself. As hiring algorithms are designed by those in charge and based on existing data, they can limit themselves to a narrower candidate demography (Faragher 2019): bearing in mind that protected groups haven't been in charge or been in high frequency. Consequently, as women have historically been underrepresented in the workforce, especially in senior positions, the language differences can be a significant variable. In fact, numerous algorithmic hiring tools have filtered out CVs that incorporate feminine words, such as 'collaborative' or 'supportive', in comparison to those that include more masculine words, such as 'execute' or 'competitive' (Faragher 2019).

1.2. Hiring in STEM

Lambrecht and Tucker (2020) investigate algorithmic judgement in microtargeting audiences for advertising STEM (Science, Technology, Engineering and Math) job vacancies. With its intention in responding to policymakers' concerns about the underrepresentation of women within the field, careful consideration was put into gender-neutral targeting. Their job advertisement was tested in 191 countries with individuals in their most productive years and shown to 20% more male than female audiences. Lambrecht and Tucker (2020), however, suggest that this is not based on the predictions on the likelihood of engagement with the ad; in fact, it is found that women were more likely to engage with the ads in comparison to men. However, it is suggested that instead, because of this, women are perceived to be a high-quality demographic and therefore are a higher-priced population as the object of advertisements. In adjusting to the cost effectiveness, therefore, they are shown these advertisements, not according to gender balance, but cost and profit balance. It is, therefore, concluded that, with the capitalistic principle, algorithms are prone to generate discriminating outcomes even in relatively and intentionally the most 'neutral' setting (Lambrecht & Tucker 2020).

These issues cannot remain unchallenged. The absence of information on opportunities is a highly significant hindrance to individuals who are pursuing them (Kim, as cited by Bogen 2019). This should reflect the current flaw in the implementation of equal opportunity and equal access for all, especially for women (Dalenberg 2018). When it comes to automated hiring and candidate selection systems, the complex nature of AI, however, creates another layer of concern, as tracking the microtargeting decisions within the algorithm cannot be fully tested (Raub 2018). When decision making is determined by a system that further adds a layer of opacity to the decision-making process (Section 2.1), the accountability of the decision becomes questionable.

2.0. Artificial Intelligence, Algorithms, and Bias

2.1. How do Decision-making and Classification Algorithms Work?

Big Data and Machine Learning are the two pillars that are pivotal to defining modern AI. Big Data refers to the massive volume of data, with datasets characterised by massive volume, high velocity in generation, its vast variety of sources and types, and its validity (Nersessian 2018). Although the sources of these data vary, most data are involuntarily donated by digital users, such as social media activities, website traffic, sensor data and online platforms, which provide insights on real-time societal behaviour and networks (Seely-Gant & Frehill 2015; McFarland & McFarland 2015).

As Raub (2018) defines it, Machine Learning (ML) is a family of procedures that can automatically generate results from datasets and improve at tasks with experience. This is only a subset of what constitutes AI. Modern ML techniques such as deep neural networks consist of models which improve in accuracy as more cases or data samples are used in ‘training’ it. For AI to work, its algorithms need to be supplied with massive sizes of datasets and corresponding patterns, fed into its algorithmic models to generate assumptions and predictions. Costa *et al.* (2020) succinctly described this modelling process in a diagram, as shown in **Figure 1**.

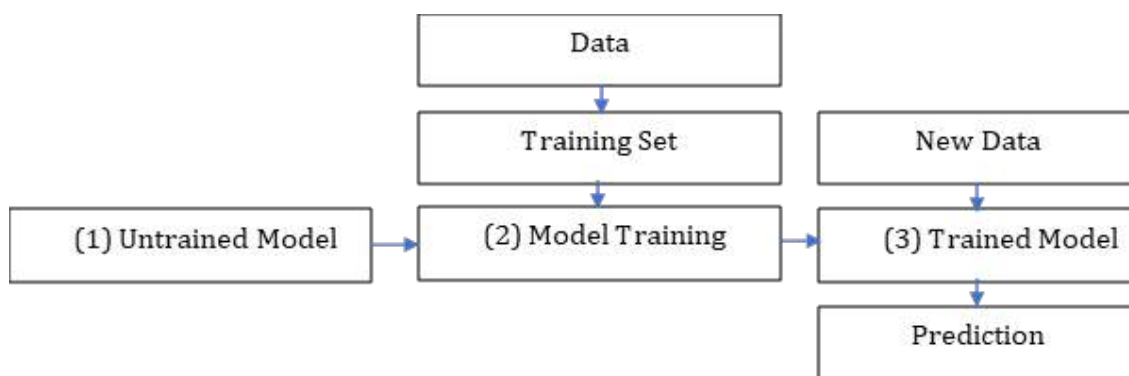


Figure 1: Model Creation Process (Costa *et al.*, 2020)

An AI system generates desired outputs by answering a question using the value of a target variable that programmers have installed. AI is fed with a variety of datasets; however, most of the time these data are sourced from Big Data, due to its volume and velocity. As Big Data is often incomplete due to gaps in data (e.g., random noise from sensors, simple errors in data entry) as well as the need for privacy and anonymity (e.g., redaction of personally-identifiable attributes to comply with laws and policy), there is a need to fill in these gaps. In doing this, programmers usually use a *proxy* to represent a characteristic; for instance, sexual orientation, political affiliations, religious convictions and race are often represented by Facebook “likes” or similar proxies, and these have been proven to be considerably accurate (Dattner *et al.* 2019). This is because not every demographic trait is explicitly recorded in digital prints; therefore, segmenting the digital citizens into categories requires a level of generalisation and representation of values. These proxies will then be rendered into algorithms that act as the ‘engine’ of the AI systems (Raub 2018). Domingos (2012) suggests that algorithms work in amalgamations of three components: representation, evaluation and optimisation. According to Domingos (2012), representation refers to the alteration of ‘language’ of the

classifier for the computer to access and learn; evaluation points to the sorting out the invalid classifiers from the valid ones; and optimisation refers to the examining of the highest-scoring classifiers.

A key criticism of modern machine learning techniques, such as neural networks, lies in its incomprehensibility (i.e., non-explainability) due to its technical complexity, and lack of mathematical proof of behaviour. Simply put, such models have mechanics which "may be hard to comprehend for experts and can be virtually incomprehensible to non-experts" (Kolkman 2020). This risk is overshadowed by the fact that "cloud technology providers such as Amazon's AWS come with 'out-of-the-box' services [for complex classification and prediction]... without [the operator] even needing to know how to construct such a [decision-making neural] network" (Cheong & Leins 2020, forthcoming). To concisely summarise why these algorithms are low in explainability (Miller 2017), two key factors are: (i) the complex model structure (Cheong & Leins 2020, forthcoming), modelled upon numerous connections, statistical weights, and virtual 'neurons' akin to a human brain; and (ii) the sheer volume of data needed to train these models means that oftentimes, the particular mathematical idiosyncrasies of a model cannot be explained. For a thorough and accessible exposition of the technical and practical implications of constructing, training, (and more controversially) basing decisions upon generated models, refer to Barocas & Selbst (2016), Tolan (2018), Boscoe (2019), and Mittelstadt *et al* (2016).

2.2. Algorithmic Fairness and Bias: A Primer

The field of Algorithmic Fairness is relatively new, studying the inherent design flaws and bias introduced to AI and other automated systems. The field initially focused on mathematical definitions of bias, treating it like another parameter to optimize. More recently, the field is moving away from the strict mathematical definitions and towards cross-discipline work with experts in other fields, for example the fields of philosophy and law.

To elaborate on the definition of algorithmic biases, Kim (2019) proposed three core algorithmic biases, which are record error bias, intentional bias and statistical bias. Record error bias refers to the bias that occurs in incomplete or misrepresentative datasets. This greatly affects the accuracy of prediction and decision making by AI, especially for the underrepresented groups. Intentional bias is the discriminatory action that is intentionally ingrained within the algorithms in order to explicitly box some groups out of the equation. However, this can also be done for the opposite reason, to battle the existing structural inequality. Lastly, Kim refers to statistical bias. This refers to the associations that algorithms make to demographic traits to make decisions based on what is recorded in statistical data.

In academic papers discussing the notion of fairness – centring upon a case study of a controversial criminal risk-assessment system (Section 2.3) – researchers have found that different ideas of fairness can co-exist when assessing an automated decision-making system or model (Chouldechova 2017; Kleinberg *et al.* 2016). Importantly, these different notions of fairness are known in some scenarios to be incompatible: a single model cannot meet every reasonable or accepted definition of fairness, and therefore bias must exist in one way or another inside the model. This restriction doesn't hold if the model is extremely trivial or, at the other end of spectrum, where the model is perfect – neither of these extremes is feasible. Therefore, we must carefully select which definitions of fairness we value and design our model towards meeting those definitions. (Narayanan 2018; Verma & Rubin 2018). This is not a trivial matter: different parties may value the (inconsistent) notions of fairness differently, depending on their involvement and how they are

impacted by the model. Further, the ethics and values of the model generally originate from the people designing the model, where implicit bias can be introduced unintentionally, and even the problem or data given by the client can frame a particular viewpoint or solution (Barabas *et al.* 2020).

Although this paper realises the importance of the three biases, more attention will be given to statistical bias in terms of the research study. Statistical bias is inextricably linked to arguments in algorithmic fairness: as seen earlier, one can argue that an AI model is ‘algorithmically fair’ given the statistics are unbiased, despite its outcome which is unfair to stakeholders. In particular, a key deliverable from this research project is the statistical identification of human biases² which are captured in even a simple, naïve classifier prototype.

2.3. Contemporary Critiques on Automated Decision Making

Before moving on to study how bias and (un)fairness exists in algorithmic hiring practices, we briefly survey contemporary issues associated with automated decision making.

Since the early 2000s, the value of computer-based, automated, decision making (ADM hereinafter) has been recognised by administrators and decision-making bodies. The Australian Government’s *Automated Assistance in Administrative Decision-making* report to the Attorney-General was “considered to be ahead of its time in anticipating the usage of expert programs to help human administrators in their decision making” processes (Cheong & Leins 2020, forthcoming). The report has identified algorithms – when implemented as part of complete systems (Boscoe 2019; Cheong & Leins 2020, forthcoming) – can be used for purposes such as giving legal advice and “profil[ing] risk factors amongst taxpayers” (Administrative Review Council, 2004). Per Cheong & Leins (2020, forthcoming), the general algorithms employed in such systems *circa* 2000s are now easily and readily implementable – and in common modern use by data scientists – with pre-built toolkits and packages in modern programming languages (such as Python). Guided with this ethos, this project will deliver a similar working prototype as part of its outcomes.

When it comes to automated decision-making disadvantaging certain sections of the population, two modern cases in algorithmic fairness and ethics come to mind. The first case is in risk assessment for criminal recidivism (Tolan 2018; Boscoe 2019). A 2016 ProPublica paper (Angwin *et al.*, 2016), on the COMPAS Risk Assessment instrument by Northpointe Inc., sparked a significant amount of academic and public debate and discussion. The argument that ProPublica made against COMPAS was that it was biased against African American defendants, giving them on average a higher risk score compared to their white American counterparts. This argument was met with criticism from both Northpointe and the academic community, mainly due to ProPublica’s choice of bias (fairness) definition. With Northpointe’s choice of fairness definition, it could be argued that COMPAS was indeed fair; however, a defendant is rightly concerned with the probability that they will be incorrectly classified as high-risk, due to their ethnic background. Researchers have shown that it was mathematically impossible for both fairness definitions chosen by ProPublica and Northpointe to hold simultaneously (Chouldechova 2017; Kleinberg *et al.*, 2016).

² Please note that ‘bias’ has a specific meaning when it comes to statistics and data science; for the purposes of this paper, ‘bias’ is used interchangeably with ‘human bias’, unless otherwise specified. In a nutshell, “statistical bias is defined as the difference between the parameter to be estimated and the mathematical expectation of the estimator” (King *et al.*, 2018). See: King *et al.* (2018). “4.3 - Statistical Biases | STAT 509”. The Pennsylvania State University. <<https://online.stat.psu.edu/stat509/node/29/>>

There are inherent incompatibilities between these definitions of 'fairness', of the algorithm itself versus the people impacted by it. In this case study, we find tensions in a system's purported outcome – should it prioritise mathematical efficiency by trading off ethical and legal notions of fairness and equality?

Another ethical case study of automated decision making is the use of predictive-policing software (Byrne & Cheong 2017; Ferguson 2017). In a thorough analysis of predictive-policing software, Ferguson (2017) identified that the modern application of predictive-policing does not align with the original intention of predictive-policing (which they term Predictive Policing 1.0, using platforms such as *PredPol*). The wrong assumption had been used in building the model: the original version ("1.0") tried to deter "place-based property crimes" because it focuses "...only on crimes that were regularly and rather consistently reported (burglary, auto theft, and theft from auto) ... [and] avoids many of the data-collection problems of a broader crime focus" (Ferguson 2017). However, with this assumption for "1.0" holding (which is subject to debate), modern predictive-policing (termed version "3.0") focusing on "person-based crimes" cannot depend on the existing models used (Ferguson 2017). Here, instead of *preventing crime to property* in certain locations, the wrong assumptions are used in *predicting individuals' propensity to offend* within those locations. With these wrong assumptions, police departments will act by sending more resources to the area predicted to have more crime; and hence more of that crime is reported; creating a positive feedback loop. In other words, this can be characterised as the model building a self-fulfilling prophecy in entrenching prejudice against minorities – the more police resources deployed in an area, the more people of colour will be identified by police as suspicious, leading to a justification for prolonged police presence (Ferguson 2017; Byrne & Cheong 2017). This case study highlights how the feedback loop above and *overfitting* the model to the data (not discounting the wrong assumptions in model building) – amongst others covered in Ferguson (2017) - can have negative implications to civil liberties and exacerbates unjust treatment based on race and class.

3.0. Risks of Bias in Algorithmic Hiring

3.1. Case Study: Amazon's Experiment Gone Wrong

Before moving into a detailed survey of how algorithmic hiring poses risks to gender equality, particularly in the STEM fields, we review how such a system has gone wrong³.

According to Reuters, Amazon Inc.'s Machine Learning team has "been building computer programs since 2014 to review job applicants' resumes with the aim of mechanizing the search for top talent" (Dastin 2018):

"The company's experimental hiring tool used artificial intelligence to give job candidates scores ranging from one to five stars - much like shoppers rate products on Amazon, some of the people said. ..."

"They literally wanted it to be an engine where I'm going to give you 100 resumes, it will spit out the top five, and we'll hire those. ..."

"But by 2015, the company realized its new system was not rating candidates for software developer jobs and other technical posts in a gender-neutral way..."
(Dastin, 2018).

The system was known to "systematically downgrade women's CV's for technical jobs such as software developer", due to "patterns in resumes submitted to the company over a 10-year period" (Lavanchy 2018), which led to the system regarding male dominance as a factor for shortlisting (Lavanchy 2018; Dastin 2018).

This led to the eventual scrapping of the system; a "much-watered down version" version has been repurposed "...to help with some rudimentary chores, including culling duplicate candidate profiles from databases" (Dastin 2018).

As seen earlier in Section 1.1, the selection of gendered words used in CVs led to gendered outcomes in a supposedly neutral algorithm. This is a well-known issue in such classification algorithms (Ferguson 2017; Boscoe 2019), where implicit or hidden bias is captured in the result of classification or prediction tasks. This parallels the outcomes raised in Section 2.3, which were biased along ethnic lines. Section 4.0 will detail design ideas for a prototype algorithm that intends to investigate what could go wrong in such a system.

³ This case study draws upon media reporting by Dastin (2018) <<https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scrap-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G>> and Lavanchy (2018) <<https://phys.org/news/2018-11-amazon-sexist-hiring-algorithm-human.html>>

3.2. Job Sorting: Translating History into Future Outcomes

The idea of adopting hiring algorithms was initially grounded on the premises that, firstly, the absence of human intervention gives rise to impartiality; and secondly, that technology enables efficiency and accuracy in sorting a massive volume of applications with minimum cost and for maximum benefit of the company (Preuss 2017; Kulkarni & Che 2017). Theoretically, hiring algorithms *should* be able to create an optimum amalgam of excellent candidates based on pure meritocracy. This paper, however, argues that the market-driven algorithms are radically prone to replicate societal biases. In making this case, this subsection will explain the theoretical logic on the proneness of the algorithms to correlational and statistical bias.

It is, however, important that this paper first elucidate the various tools and different functions of hiring algorithms. Generally, different tools would perform similar processes in running different tasks. It would typically source and screen data, from either direct input (including tests, assessments, video interviews and submitted resumés), or from automated input (usually from Big Data). These data would then be used to grade and rank candidates' suitability to the role, based on weighted components such as assessment metrics, qualifications, test scoring, keywords and other parameters (Roy 2017). However, different software tools would perform differently as it is designed for different objectives and for different sources. Although there are various categorisations of automated hiring software, this paper focuses on recruitment tools that function as *Candidate Assessment Software (CAS)* and *Applicant Tracking Software (ATS)*. CAS usually generates personality tests, case studies, logic tests, automated interviews and other assessment tools and evaluates candidates based on the outcome of the test in comparison to a predetermined set of criteria. Generally, the tests can be categorised into three broad types: intelligence, personality and mental or clinical health (Dattner *et al.* 2019). *HireVue* is one of the examples of CAS tools. This software assesses candidates based on virtual interviews, among many others. The algorithmic model would assess candidates by voice and facial recognition to rank candidates for their suitability and predict their 'success' (Raub 2018). Software tools such as this can evaluate the choice of words, for instance, to predict the levels of empathy, hospitality, and other characteristics, and weigh these in accordance to the company's culture (Alsever 2017). A similar idea applies to logic tests, where candidates are asked to perform assessments to be ranked based on suitability (see similar tools such as *ARYA*, *CEIPA* and *HackerRank*).

ATS, on the other hand, performs resumé parsing, *Customer Relationship Management (CRM)*, background and social screening, candidates sourcing, and success prediction. It is noteworthy, however, that not all ATS tools would include all functions. Some can only perform one specific function whilst others may be able to perform all functions, and a few other ones may include both CAS and ATS operations. One of the most prominent examples to this is *Gild*. *Gild* performs beyond resumé parsing and extends its rendition from Big Data (O'Neil 2016). It implies that the assessment of candidates' suitability is not only based on qualifications or meritocracy, but it also rates applicants based on their 'social capital', generated from their digital activities. An identical approach applies to other tools, especially those involving background screening from Big Data, such as *Ascendify*, *BambooHR* and others. Major companies with a massive amount of applications are reliant on these tools for their hiring process (O'Neil 2016). In fact, seventy-two per cent of resumés are never seen by hiring officers at all (O'Neil 2016). However, given the numerous ways in which these different tools source datasets and perform tasks, the degree of risks may vary.

3.2. Algorithmic Hiring: Risks for Women in STEM

This section focuses on the theoretical implications of potential problematic risks that can impact hiring outcomes due to correlational and statistical biases.

Firstly, the limitation of the datasets is a major factor in the algorithms' proneness to discrimination as it builds the scope of benchmarking (Costa *et al.* 2020). If the algorithms are fed with limited datasets to assess and produce decisions, the algorithms will transform their benchmark based on this specific set of data (Costa *et al.* 2020). For instance, an experiment was done using a facial recognition software to determine its accuracy when applied to various races. Due to a larger number of inputs from Caucasian individuals, the software performs higher accuracy when observing Caucasians, and lower accuracy to the races of minority (Costa *et al.* 2020). As discussed in prior sections, digital population is key in determining accurate representation. In the same manner, considering excluded demography within a dataset is equally significant as those who are represented.

Secondly, as evidenced in the previous chapters, correlational bias remains a major concern in algorithms, including those designed for hiring management (Kim 2019). One of the prominent reasons for this is the use of proxy attributes to represent an individual or a parameter. For instance, algorithms make correlations between 'creativity' and the individual's length of employment within the same job (O'Neil 2016). They also make associations between higher levels of 'inquisitiveness' and their likeliness to find other opportunities. When these correlations are made based on demographic traits, such as neighbourhood, race or gender, this becomes severely problematic and it could control the change of the whole corporate culture (O'Neil 2016). Therefore, when these proxies are embedded within the algorithms' judgement of suitability for employment, it will repeat the same societal bias. In background screening, for instance, as the algorithms search through the Big Data, the decisions will be more sensitive in making racial biases, especially in relation to various discriminatory records across the history (O'Neil 2016; Raub 2018). A medical school in the United Kingdom, St. George's Hospital, has used a hiring program for its admission decision since the 1980s (Raub 2018). It is found that the algorithms did not introduce new biases, but captured the pertaining biases as its target variables, and therefore imitated the pertaining biases in its decisions. Consequently, the software refused approximately 60 applicants annually based on race and gender.

These premises should question the legitimacy of the parameters of 'suitability'; which implies that there is room for algorithmic bias in selecting attributes as candidates' criteria for suitability assessment. In other words, if each candidate is to be assessed with the same set of criteria, where will this model of 'ideal' candidate come from? How much of the algorithmic judgement would be determined by 'objective' measures in comparison to correlational variables? The inputs of these criteria can be severely problematic (O'Neil 2016). This can be applied in a similar manner both in CAS and ATS tools. Firstly, the defining of an 'ideal' employee by algorithms is highly ambiguous and, therefore, cannot be held entirely accountable. Naturally, when recruiters set certain criteria, they will ideally settle on measurable outcomes, such as GPA results, longer employment, and so on (Raub 2018). However, it has been noted that subjective decisions and human biases could still occur; hence, the cases for AI in the first place (Raub 2018). Nevertheless, this paper argues that the use of AI does not in any way eliminate these biases based on two reasons. Firstly, algorithms are generally trained to record and memorise past decisions and *learn* from them (O'Neil 2016). This implies that the algorithms memorise the patterns of previous decisions and *can* replicate the patterns of these decisions. Secondly, in the cases where AI's judgement becomes central to the hiring decisions, the objective measures become questionable as the algorithms adapt to the sourced data and preceding decisions that they have previously collected and memorised, and transformed their calculations against

the candidates based on what the algorithms perceive as effective and efficient (Raub 2018). Coming back to the marketised principles of corporate hiring algorithms, however, this is deemed to produce the most efficiency of cost, as it predicts the success of a candidate and their length of retention, at the expense of impartial judgements (Kulkarni & Che 2017).

This should raise several questions on the impact of hiring algorithms to gender equality in relation to statistical bias. Firstly, if the male population has dominated the formal labour force throughout history, how would that translate into the algorithms' benchmark of success? Secondly, if women who are currently in the workforce now are underrepresented in full-time labour, and occupy most of the casual, part-time and domestic work, how would this impact the future of other women (Australian Government 2020)?

3.3. Hiring and Protected Attributes

In their research into bias in hiring, Raghavan *et al.* (2020) identified that the requirements set out by US law for hiring require a fine balance between disparate impact and disparate treatment (Barocas & Selbst 2016). Due to a longstanding legal tension between the two, techniques that control and reduce disparate impact can require the use of protected attributes, which can potentially cause disparate treatment (and not to mention an invasion of privacy).

Based on their research of industry solutions to algorithmic hiring (Sections 2.0 and 3.0), Raghavan *et al.* (2020) found that most companies focused on explicit legal requirements, such as the US "4/5 rule", where the selection rate for each group must be at least 4/5ths of the dominant group, rather than evaluating the entire system as a whole. One of the key points that Raghavan *et al.* (2020) make in their study is that while it is important for companies to protect their clients by meeting legal requirements, strictly meeting the 4/5 rule for US hiring does not substitute for critical analysis into the potential bias in recruitment assessment. In particular, it is difficult for companies to minimize the disparate impact to different groups based on protected attributes that they don't collect data on, this can be important as it has been shown that strictly excluding protected data attributes does not imply that the model will be intrinsically fair (Narayanan 2018; Kusner & Loftus 2020).

4.0. Research Directions

4.1. Hypotheses and Methods

One of the deliverables of this project depends on the construction of a prototype classifier -- i.e., a simple machine learning model -- to rank CVs. As with all models, we will need to train it, based on a set of pre-defined judgements.

Therefore, a panel at Policy Lab at The University of Melbourne will remotely⁴ perform a mock hiring process, given fictitious CVs with attributes reflecting gender. One of the project RAs (co-author Njoto) will facilitate, amongst others, the generation of simulated CVs and/or synthesis on anonymised real-world CVs (Refer Executive Summary above).

To train this classifier, we use judgements collected from the panel to form the basis of the training data. One of the project RAs (co-author McLoughney) will attend the panel sessions in order to elicit user requirements for the classifier and to come up with core design decisions. To avoid problems arising from low explainability of complex classifiers such as deep neural networks (Sections 2.1-2.2), classical techniques will be favoured instead (see e.g., those featured in Section 2.3).

Techniques which may be used include, but are not limited to:

- Semi-automated techniques to convert CVs into a corresponding mathematical representation (e.g. one-hot encoding)
- Natural language processing (NLP) techniques such as stemming, lemmatising, removal of stopwords.
- Classification techniques including k-nearest neighbours, decision trees.
- Analysis on keywords captured in the classifier as basis for discrimination.

The outcome would be a report on the gender balance/disparity found in both the human panel and the classifier approach. This includes an analysis of the similarities and differences between a statistical model, which may learn discriminatory bias from the data set collected, versus human judgements, which would reflect the bias encoded based on the values and on the assumptions made by humans.

⁴ Due to the transition of The University of Melbourne to a 'Virtual Campus' model, per <https://about.unimelb.edu.au/newsroom/news/2020/march/accelerating-our-transition-to-a-virtual-campus>

4.2. Future Research

The underrepresentation of women in the labour force data, then, should be seriously considered in the algorithmic data training. This paper, thus, contends that, firstly, if the algorithms are fed with the existing internal workforce data, algorithms can generate correlational bias between meritocratic principles and gender, and therefore, would discriminate against women. Secondly, if the algorithms are trained using Big Data in relation to the submitted resumés, the accountability of the validity of data becomes questionable, especially when extended to a candidate's social data. As a hypothetical example, if the algorithms capture a female candidate as being married, would they, then, assume that the candidate would be more prone to exiting the workforce because of motherhood? The principle of meritocracy in conjunction to the correlational bias, then, bears a very significant risk for boxing women out of workforce. Thirdly, as an implication of the two points above, if algorithms assume that the model of a successful candidate is male, then the determinants of 'suitability' itself need to be questioned and reconsidered.

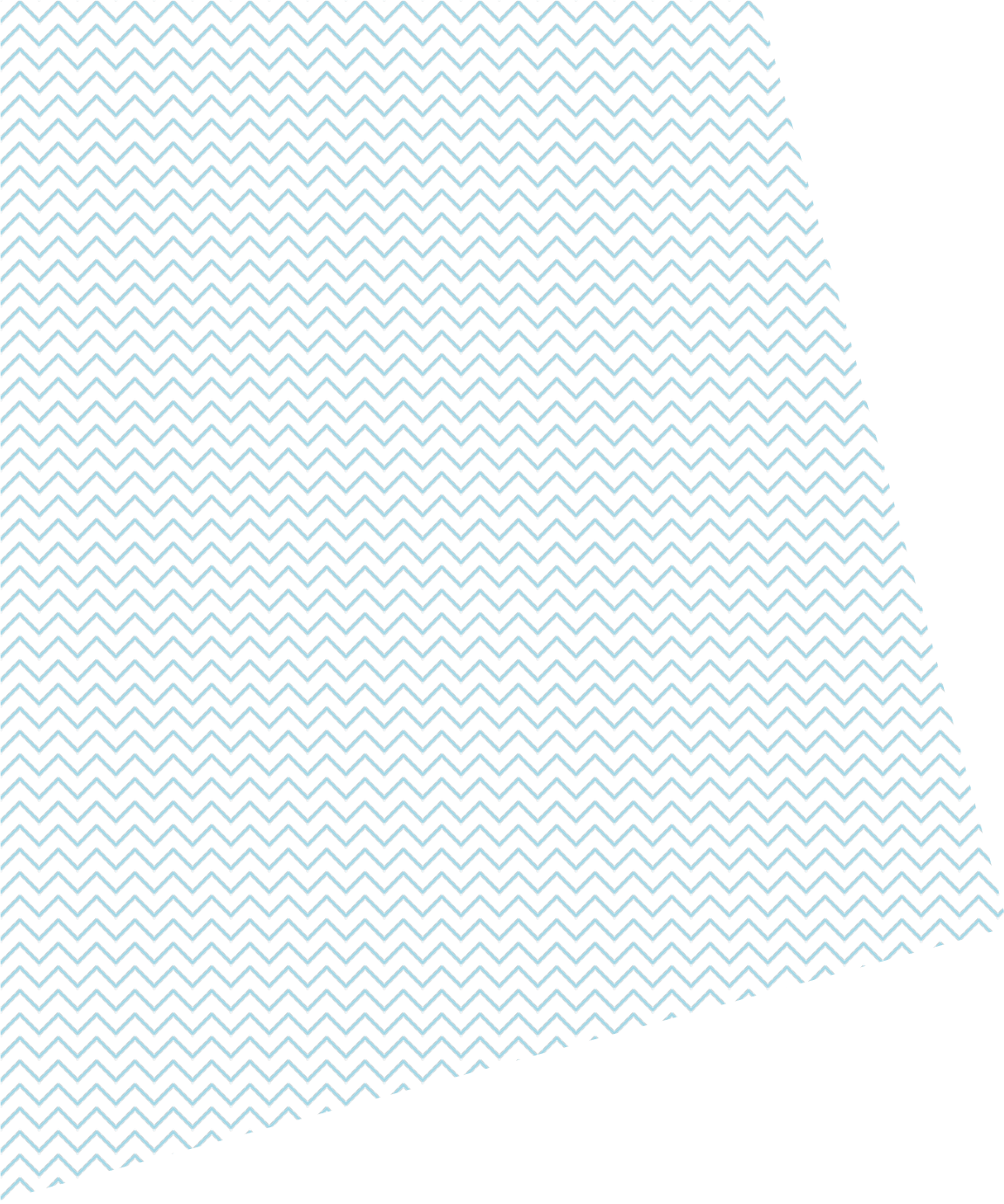
This project provides an impetus for developing AI systems that are less biased, perhaps by mimicking the way humans become less biased. Looking forward, an idea for extending the current project is a thorough survey with HR professionals to understand how they view gender in the hiring process; how gender can bias their views; and how there has been progress (through e.g. training and diversity awareness initiatives) in mitigating bias by human evaluators. These can potentially be fed back into the design of future HR AI systems as well as currently-used CAS and ATS tools to mitigate such biases.

References

- Administrative Review Council (2004) Automated assistance in administrative decision-making: Report to the Attorney-General. Attorney-General's Department, Australian Government. Available at: <https://www.ag.gov.au/LegalSystem/AdministrativeLaw/Pages/practice-guides/automated-assistance-in-administrative-decision-making.aspx> (Accessed: 9 April 2020).
- Alsever, J. (2017). Where Does the Algorithm See You in 10 Years?. *Fortune*, 1 June. Retrieved from <https://fortune.com/2017/05/19/ai-changing-jobs-hiring-recruiting/>.
- Angwin, J. et al. (2016) Machine Bias, ProPublica. Available at: <https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing> (Accessed: 11 June 2020).
- Australian Government (2020). Work and family: Facts and Figures: Report. Available at: <https://aifs.gov.au/facts-and-figures/work-and-family>.
- Barabas, C. et al. (2020) 'Studying up: reorienting the study of algorithmic fairness around issues of power', in Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. New York, NY, USA: Association for Computing Machinery (FAT* '20), pp. 167–176.
- Barocas, S. and Selbst, A. (2016) 'Big Data's Disparate Impact', *Calif. L. Rev. clr*, 104(671). doi: 10.15779/Z38BG31.
- Bogen, M. (2019). All the ways Hiring Algorithms Can Introduce Bias. *Harvard Business Review*. 6 May. Retrieved from <https://hbr.org/2019/05/all-the-ways-hiring-algorithms-can-introduce-bias>.
- Boscoe, B. (2019) 'Creating transparency in algorithmic processes', *Delphi*. HeinOnline, 2, p. 12.
- Byrne, J. and Cheong, M. (2017) 'The Algorithm as Human: a cross-disciplinary discussion of anthropology in an increasingly data-driven world', in AAS/ASA/ASAANZ 2017 Shifting States Conference. University of Adelaide.
- Cheong, M. and Leins, K. (2020) 'Who Oversees the Government? Modernising Regulation and Review of Australian Automated Administrative Decision-making', in Boughey, J. and Miller, K. (eds) *Government Automation and Public Law Project Workshop* (forthcoming). Government Automation and Public Law Project Workshop (forthcoming).
- Chouldechova, A. (2017) 'Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments', *Big data*, 5(2), pp. 153–163.
- Costa, A., Cheung, C. and Langenkamp, M. (2020). *Hiring Fairly in the Age of Algorithms* (Research Paper). Retrieved from <https://arxiv.org/abs/2004.07132>.
- Dalenberg, D. J. (2018). Preventing discrimination in the automated targeting of job advertisements. *Computer Law & Security Review*, 34, pp. 615–627. <https://doi.org/10.1016/j.clsr.2017.11.009>.
- Dastin, J. (2018) Amazon scraps secret AI recruiting tool that showed bias against women, Reuters. Reuters. Available at: <https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G> (Accessed: 11 June 2020).
- Dattner, B., Chamorro-Premuzic, T., Buchband, R. and Schettler, L. (2019). The Legal and Ethical Implications of Using AI in Hiring. *Harvard Business School*. 25 April. Retrieved from <https://hbr.org/2019/04/the-legal-and-ethical-implications-of-using-ai-in-hiring>.

- Domingos, P. (2012). A Free Useful Things to Know about Machine Learning. *Communications of the ACM*, 55(10), pp. 78–87. doi:10.1145/2347736.2347755.
- Faragher, J. (2019). Is AI the enemy of diversity?. *People Management*, 6 June. Retrieved from <https://www.peoplemanagement.co.uk/long-reads/articles/is-ai-enemy-diversity>.
- Ferguson, A. G. (2017) 'Policing predictive policing', Wash. UL Rev. HeinOnline, 94(5). Available at: https://heinonline.org/hol-cgi-bin/get_pdf.cgi?handle=hein.journals/walq94§ion=35.
- Gaucher, D., Friesen, J. and Kay A. (2011). Evidence That Gendered Wording in Job Advertisements Exists and Sustains Gender Inequality. *Journal of Personality and Social Psychology*, 101(1), pp. 109–128. doi: 10.1037/a0011530.
- Hegarty, P. and Buechel, C. (2006). Androcentric Reporting of Gender Differences in APA Journals: 1965–2004. *Review of General Psychology*, 10(4), pp. 377–389. doi.org/10.1037/1089-2680.10.4.377.
- Kim, P. T. (2019). Big Data and Artificial Intelligence: New Challenges for Workplace Equality. *University of Louisville Law Review*, 57, pp. 313–328. Retrieved from <https://ssrn.com/abstract=3296521>.
- Kleinberg, J., Mullainathan, S. and Raghavan, M. (2017) 'Inherent Trade-Offs in the Fair Determination of Risk Scores'. Available at: <https://arxiv.org/abs/1609.05807> (Accessed: 11 June 2020).
- Kolkman, D. (2020) 'The (in)credibility of algorithmic models to non-experts', *Information, Communication and Society*. Routledge, pp. 1–17.
- Kulkarni, S. and Che, X. (2017). Intelligent Software Tools for Recruiting. *Journal of International Technology & Information Management*, 28(2), pp. 1–16. Retrieved from https://scholarworks.lib.csusb.edu/jitim/?utm_source=scholarworks.lib.csusb.edu%2Fjitim%2Fvol28%2Fiss2%2F1&utm_medium=PDF&utm_campaign=PDFCoverPages.
- Kusner, M. J. and Loftus, J. R. (2020) 'The long road to fairer algorithms', *Nature*, 578(7793), pp. 34–36.
- Lambrecht, A. and Tucker, C. (2020). Algorithmic Bias? An Empirical Study of Apparent Gender-Based Discrimination in the Display of STEM Career Ads. *Management Science*, pp. 2966–2981. <http://dx.doi.org/10.2139/ssrn.2852260>.
- Lavanchy, M. (2018) Amazon's sexist hiring algorithm could still be better than a human, *Phys.org*. Available at: <https://phys.org/news/2018-11-amazon-sexist-hiring-algorithm-human.html> (Accessed: 11 June 2020).
- McFarland, D., and McFarland, R. (2015). Big Data and the danger of being precisely inaccurate. *Big Data & Society*, Jul–Dec, pp. 1-4. doi: 10.1177/2053951715602495.
- Miller, T. (2017) 'Explanation in Artificial Intelligence: Insights from the Social Sciences', arXiv [cs.AI]. Available at: <http://arxiv.org/abs/1706.07269>.
- Mittelstadt, B. D. et al. (2016) 'The ethics of algorithms: Mapping the debate', *Big Data & Society*. SAGE Publications Ltd, 3(2), p. 2053951716679679.
- Narayanan, A. (2018) Tutorial: 21 fairness definitions and their politics, YouTube. Available at: <https://www.youtube.com/watch?v=jlXluYdnyyk> (Accessed: 11 June 2020).
- O'Neil, C. (2016). *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. United Kingdom: Penguin Random House.
- Preuss, A. (2017). Airline pilots: the model for intelligent recruiting?. *Recruiter*, pp. 12–13. Retrieved from <https://www.recruiter.co.uk/trends/2017/08/airline-pilots-model-intelligent-recruiting>.

- Raghavan, M. et al. (2020) 'Mitigating bias in algorithmic hiring: evaluating claims and practices', in Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. New York, NY, USA: Association for Computing Machinery (FAT* '20), pp. 469–481.
- Raub, M. (2018). Bots, Bias and Big Data: Artificial Intelligence, Algorithmic Bias and Disparate Impact Liability in Hiring Practices. *Arkansas Law Review*, 71(2), pp. 529–570. Retrieved from <https://scholarworks.uark.edu/alr/vol71/iss2/7>.
- Roy, R. (2017). Corporate recruiting opening its doors to AI: the performance opportunity?. *Performance Improvement*, 56(10), November–December, pp. 43–44. <https://doi.org/10.1002/pfi.21747>.
- Sczesny, S., Formanowicz, M. and Moser, F. (2016). Can Gender-Fair Language Reduce Gender Stereotyping and Discrimination?. *Frontiers in Psychology*, 7(25), pp. 1–11. <https://doi.org/10.3389/fpsyg.2016.00025>.
- Seely-Gant, K. and Frehill, L. (2015). Exploring Bias and Error in Big Data Research. *Journal Washington Academy of Sciences*, 101(3), pp. 29–37.
- Stout, J. G. and Dasgupta, N. (2011). When *He* Doesn't Mean *You*: Gender-Exclusive Language as Ostracism. *Personality and Social Psychology Bulletin*, 37(6), pp. 757–769. doi: 10.1177/0146167211406434.
- Tolan, S. (2018) Fair and Unbiased Algorithmic Decision Making. 2018-10. European Commission, Joint Research Centre Technical Reports.
- Verma, S. and Rubin, J. (2018) 'Fairness definitions explained', Proceedings of the International Workshop on Software Fairness - FairWare '18. doi: 10.1145/3194770.3194776.



Project Contact

Marc Cheong

<marc.cheong@unimelb.edu.au>



```
In [255]: import warnings
warnings.filterwarnings('ignore')
from scipy.spatial import Voronoi, voronoi_plot_2d
import itertools
import math
import docx2txt
import nltk
import spacy
import json
from datetime import datetime
import os
import csv
import statistics
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import f_regression
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
from sklearn.model_selection import RepeatedKFold
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from nltk.tokenize import word_tokenize
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from scipy import stats
from patsy import dmatrices
sns.set() # for plot styling
```

```
In [2]: nlp = spacy.load("en_core_web_md")
```

"en_core_web_md": English multi-task CNN trained on OntoNotes, with GloVe vectors trained on Common Crawl. Assigns word vectors, POS tags, dependency parse and named entities.

1 Rankings

In this section we compare two different methods of aggregating rankings. We need to aggregate the data to get an overall ranking because the data points are not statistically independent, and this is a requirement for our machine learning techniques. The response variables with the lower case (e.g. `y_da_c`) are calculated with the mean method, and response variables with upper case (e.g. `Y_da_c`) are calculated with the distance method.

Throughout the notebook we will be displaying the rankings data as a list of numbers, where the first number represents the rank of resume #1, the second number represents the rank of resume #2, and so on.

For example, if the rankings was [2,3,1] then resume #3 is ranked as the 1st (Best) candidate, while resume #1 is ranked 2nd, and resume #2 is ranked 3rd.

```
In [3]: # Creates an ordering based on the average rank of each resume
def rank_list(mean_list):

    mean_sort = np.sort(mean_list)
    order = []

    for mean in mean_list:

        order.append(np.where(mean_sort == mean)[0][0]+1)

    return order
```

Based on the current literature, the most optimal method for aggregating multiple rankings is to quantify the difference between two different rankings by looking at how pairs of candidates are ordered. [1]

This based on the idea that if List 1 ranks Resume A higher than Resume B, and List 2 ranks Resume B higher than Resume A then there is a difference/distance of one between the two lists, assuming the two lists are the same in other rankings. E.g. List1 = [1,2,3,4], List 2 = [2,1,3,4] the distance/difference is one.

```
In [4]: def rank_distance(list1, list2):
        temp_list = list2[:]

        distance = 0

        for rank1 in list1:

            for rank2 in temp_list:

                if rank1 == rank2:
                    break

                distance +=1

            temp_list.remove(rank1)

        return distance
```

Rather than spending lots of time on creating a complex optimisation algorithm, it's much faster to just brute force calculate all of the permutations of the data, to find the ranked list that is closest to the dataset.

```
In [5]: def brute_rank_aggregation(rankings):  
    perms = list(itertools.permutations(['1','2','3','4','5','6','7','8']))  
    min_distance = 1000  
    for perm in perms:  
        distance = 0  
        for ranking in rankings:  
            distance += rank_distance(perm, ranking)  
        if distance < min_distance:  
            min_perm = perm  
            min_distance = distance  
    return min_perm
```

To optimise the distance calculations, it's more efficient to have the rank lists in the format where the numbers represent the resume and position represents the rank, rather than having the numbers represent the rank and the position represent the resume.

```
In [6]: # This function converts the standard ranking list format into the more optimi  
sed format for the distance calculations  
def order_to_ranks(order_rank):  
    temp = []  
    for rank in ['1','2','3','4','5','6','7','8']:  
        temp.append(order_rank.index(rank) + 1)  
    return temp
```

The data has been preprocessed and optimised for distance calculations, where the files are labeled as "orderedrankings..."

1.1 Data Analyst Job

As part of our experiment, half of our participants were given resumes where the gender had been flipped. Two separate datasets have been organised based on whether the gender was flipped or not. This was under the hypothesis that the perceived gender could affect the rankings. We also combined the two datasets into a single dataset to get a more general sense of the data.

```
In [7]: # Original Gender
with open("CV Json Research Data/Job 1 - Data Analyst/rankings.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    da_ranks_o = np.asfarray(ranks, float)

# Preprocessed data for distance calculations
with open("CV Json Research Data/Job 1 - Data Analyst/ordered_rankings_original.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    da_ordered_ranks_o = list(rankings)

# Flipped Gender
with open("CV Json Research Data/Job 1 - Data Analyst/rankings_gender_flip.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    da_ranks_f=np.asfarray(ranks,float)

# Preprocessed data for distance calculations
with open("CV Json Research Data/Job 1 - Data Analyst/ordered_rankings_flipped.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    da_ordered_ranks_f = list(rankings)

# Combined (Original and Flipped Gender data)
with open("CV Json Research Data/Job 1 - Data Analyst/rankings_combined.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    da_ranks_c=np.asfarray(ranks,float)

# Preprocessed data for distance calculations
with open("CV Json Research Data/Job 1 - Data Analyst/ordered_rankings_combined.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    da_ordered_ranks_c = list(rankings)
```

1.1.1 Original Gender

1.1.1.1 Aggregated ranks based on mean

We calculated the mean ranks for each of the resumes, and used the means to give aggregated ranks for the resumes. We also calculated the variance for the ranks because we were curious to see whether some resumes more consistently received the same ranking.

```
In [8]: y_da_o = rank_list(np.mean(da_ranks_o, axis=0))

print("Means: ", np.mean(da_ranks_o, axis=0))
print("Variance: ", np.var(da_ranks_o, axis=0))
print("Aggregated ranks based on mean values: ", y_da_o)

Means: [7.0952381  3.04761905 4.23809524 5.57142857 4.19047619 3.76190476
 4.          4.0952381 ]
Variance: [2.46712018 5.28344671 1.99092971 3.4829932  5.29705215 3.80045351
 4.          4.56235828]
Aggregated ranks based on mean values: [8, 1, 6, 7, 5, 2, 3, 4]
```

Just as a reminder, the first number in the list is the ranking of resume #1.

E.g. based on these results, resume #1 has an aggregated rank of 8th.

1.1.1.2 Ranks calculated from ordered pair distance

Using the optimal ordered pair distance method, we calculate the aggregated ranks for the data analyst original gender dataset.

```
In [9]: da_original_ranks = brute_rank_aggregation(da_ordered_ranks_o)
Y_da_o = order_to_ranks(da_original_ranks)

print("Aggregated ranks based on ordered pair distance: ", Y_da_o)

Aggregated ranks based on ordered pair distance: [8, 1, 5, 6, 7, 4, 2, 3]
```

Comparing the results of the two different methods, the top and bottom candidates are consistent while the remaining candidates are shuffled in their respective top and bottom halves.

1.1.2 Flipped Genders

Repeating the same process on the flipped gender data, we receive similar results to the original gender data except that candidates #3 and #7 have swapped ranks.

1.1.2.1 Aggregated ranks based on mean

```
In [10]: y_da_f = rank_list(np.mean(da_ranks_f, axis=0))

print("Means: ", np.mean(da_ranks_f, axis=0))
print("Variance: ", np.var(da_ranks_f, axis=0))
print("Aggregated ranks based on mean values: ", y_da_f)
```

```
Means: [7.05555556 2.11111111 3.72222222 5.77777778 4.27777778 3.83333333
 4.77777778 4.16666667]
Variance: [2.7191358 3.20987654 3.75617284 2.17283951 4.3117284 3.02777778
 3.50617284 3.58333333]
Aggregated ranks based on mean values: [8, 1, 2, 7, 5, 3, 6, 4]
```

1.1.2.2 Ranks calculated from ordered pair distance

```
In [11]: da_flipped_ranks = brute_rank_aggregation(da_ordered_ranks_f)
Y_da_f = order_to_ranks(da_flipped_ranks)

print("Aggregated ranks based on ordered pair distance: ", Y_da_f)
```

```
Aggregated ranks based on ordered pair distance: [8, 1, 2, 7, 3, 5, 6, 4]
```

1.1.3 Combined Results

1.1.3.1 Aggregated ranks based on mean

```
In [12]: y_da_c = rank_list(np.mean(da_ranks_c,axis = 0))

print("Means: ",np.mean(da_ranks_c,axis = 0))
print("Variance: ",np.var(da_ranks_c,axis = 0))
print("Aggregated ranks based on mean values: ",y_da_c)
```

```
Means: [7.07692308 2.61538462 4.          5.66666667 4.23076923 3.79487179
 4.35897436 4.12820513]
Variance: [2.58382643 4.5443787 2.87179487 2.88888889 4.84418146 3.44510191
 3.92241946 4.11176857]
Aggregated ranks based on mean values: [8, 1, 3, 7, 5, 2, 6, 4]
```

1.1.3.2 Ranks calculated from ordered pair distance

```
In [13]: da_combined_ranks = brute_rank_aggregation(da_ordered_ranks_c)
Y_da_c = order_to_ranks(da_combined_ranks)

print("Aggregated ranks based on ordered pair distance: ", Y_da_f)
print(Y_da_c)
```

```
Aggregated ranks based on ordered pair distance: [8, 1, 2, 7, 3, 5, 6, 4]
[8, 1, 3, 7, 6, 4, 5, 2]
```

1.2 Finance Officer

```
In [14]: # Original Gender
with open("CV Json Research Data/Job 2 - Finance Officer/rankings.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    fo_ranks_o=np.asfarray(ranks,float)

# Preprocessed data for distance calculations
with open("CV Json Research Data/Job 2 - Finance Officer/ordered_rankings_original.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    fo_ordered_ranks_o = list(rankings)

# Flipped Gender
with open("CV Json Research Data/Job 2 - Finance Officer/rankings_gender_flip.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    fo_ranks_f=np.asfarray(ranks,float)

# Preprocessed data for distance calculations
with open("CV Json Research Data/Job 2 - Finance Officer/ordered_rankings_flipped.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    fo_ordered_ranks_f = list(rankings)

# Combined (Original and Flipped Gender data)
with open("CV Json Research Data/Job 2 - Finance Officer/rankings_combined.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    fo_ranks_c=np.asfarray(ranks,float)

# Preprocessed data for distance calculations
with open("CV Json Research Data/Job 2 - Finance Officer/ordered_rankings_combined.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    fo_ordered_ranks_c = list(rankings)
```

1.2.1 Original Gender

```
In [15]: y_fo_o = rank_list(np.mean(fo_ranks_o,axis = 0))

print("Means: ",np.mean(fo_ranks_o,axis = 0))
print("Variance: ",np.var(fo_ranks_o,axis = 0))
print("Aggregated ranks based on mean values: ",y_fo_o)
```

```
Means: [3.22727273 4.54545455 6.95454545 5.18181818 3.77272727 4.22727273
 4.40909091 3.68181818]
Variance: [4.81198347 6.0661157 1.86157025 5.14876033 3.90289256 3.35743802
 4.42355372 3.03512397]
Aggregated ranks based on mean values: [1, 6, 8, 7, 3, 4, 5, 2]
```

```
In [16]: fo_original_ranks = brute_rank_aggregation(fo_ordered_ranks_o)
Y_fo_o = order_to_ranks(fo_original_ranks)

print("Aggregated ranks based on ordered pair distance: ", Y_fo_o)
```

```
Aggregated ranks based on ordered pair distance: [1, 5, 8, 7, 4, 6, 3, 2]
```

1.2.2 Flipped Gender

```
In [17]: y_fo_f = rank_list(np.mean(fo_ranks_f,axis = 0))

print("Means: ",np.mean(fo_ranks_f,axis = 0))
print("Variance: ",np.var(fo_ranks_f,axis = 0))
print("Aggregated ranks based on mean values: ",y_fo_f)
```

```
Means: [3.1875 4.625 6.875 5.5 3.375 4.25 4.6875 3.5 ]
Variance: [3.90234375 4.984375 2.859375 6.375 3.609375 3.1875
 2.96484375 3.375 ]
Aggregated ranks based on mean values: [1, 5, 8, 7, 2, 4, 6, 3]
```

```
In [18]: fo_flipped_ranks = brute_rank_aggregation(fo_ordered_ranks_f)
Y_fo_f = order_to_ranks(fo_flipped_ranks)

print("Aggregated ranks based on ordered pair distance: ", Y_fo_f)
```

```
Aggregated ranks based on ordered pair distance: [2, 6, 8, 5, 1, 7, 4, 3]
```

1.2.3 Combined Results

```
In [19]: y_fo_c = rank_list(np.mean(fo_ranks_c,axis = 0))
```

```
print("Means: ",np.mean(fo_ranks_c,axis = 0))
print("Variance: ",np.var(fo_ranks_c,axis = 0))
print("Aggregated ranks based on mean values: ",y_fo_c)
```

```
Means: [3.21052632 4.57894737 6.92105263 5.31578947 3.60526316 4.23684211
4.52631579 3.60526316]
```

```
Variance: [4.42936288 5.61218837 2.283241 5.68975069 3.81786704 3.28601108
3.82825485 3.18628809]
```

```
Aggregated ranks based on mean values: [1, 6, 8, 7, 2, 4, 5, 2]
```

```
In [20]: fo_combined_ranks = brute_rank_aggregation(fo_ordered_ranks_c)
Y_fo_c = order_to_ranks(fo_combined_ranks)
```

```
print("Aggregated ranks based on ordered pair distance: ", Y_fo_c)
```

```
Aggregated ranks based on ordered pair distance: [2, 5, 8, 7, 1, 6, 4, 3]
```

1.3 Recruitment Officer


```
In [21]: # Original Gender
with open("CV Json Research Data/Job 3 - Recruitment Officer/rankings.csv") as
rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ro_ranks_o=np.asfarray(ranks,float)

# Preprocessed data for distance calculations
with open("CV Json Research Data/Job 3 - Recruitment Officer/ordered_rankings_
original.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    ro_ordered_ranks_o = list(rankings)

# Flipped Gender
with open("CV Json Research Data/Job 3 - Recruitment Officer/rankings_gender_f
lip.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ro_ranks_f=np.asfarray(ranks,float)

# Preprocessed data for distance calculations
with open("CV Json Research Data/Job 3 - Recruitment Officer/ordered_rankings_
flipped.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    ro_ordered_ranks_f = list(rankings)

# Combined (Original and Flipped Gender data)
with open("CV Json Research Data/Job 3 - Recruitment Officer/rankings_combine
d.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ro_ranks_c=np.asfarray(ranks,float)

# Preprocessed data for distance calculations
with open("CV Json Research Data/Job 3 - Recruitment Officer/ordered_rankings_
combined.csv") as rankings_file:
    rankings = csv.reader(rankings_file)
    ro_ordered_ranks_c = list(rankings)
```

1.3.1 Original Gender

```
In [22]: y_ro_o = rank_list(np.mean(ro_ranks_o,axis = 0))

print("Means: ",np.mean(ro_ranks_o,axis = 0))
print("Rank Variance: ",np.var(ro_ranks_o,axis = 0))
print("Aggregated ranks based on mean values: ",y_ro_o)

Means: [5.45 5.65 4.25 6.9 2.8 6.05 2.35 2.55]
Rank Variance: [4.0475 3.5275 2.0875 1.69 2.06 1.8475 3.5275 1.4475]
Aggregated ranks based on mean values: [5, 6, 4, 8, 3, 7, 1, 2]
```

```
In [23]: ro_original_ranks = brute_rank_aggregation(ro_ordered_ranks_o)
Y_ro_o = order_to_ranks(ro_original_ranks)

print("Aggregated ranks based on ordered pair distance: ", Y_ro_o)

Aggregated ranks based on ordered pair distance: [5, 7, 4, 8, 2, 6, 1, 3]
```

1.3.2 Flipped Gender

```
In [24]: y_ro_f = rank_list(np.mean(ro_ranks_f,axis = 0))

print("Means: ",np.mean(ro_ranks_f,axis = 0))
print("Variance: ",np.var(ro_ranks_f,axis = 0))
print("Aggregated ranks based on mean values: ",y_ro_f)

Means: [5.47058824 5.70588235 4.29411765 5.70588235 3.82352941 5.17647059
3.05882353 2.76470588]
Variance: [3.5432526 3.03114187 2.79584775 4.79584775 3.55709343 7.67474048
4.05536332 2.65051903]
Aggregated ranks based on mean values: [6, 7, 4, 7, 3, 5, 2, 1]
```

```
In [25]: ro_flipped_ranks = brute_rank_aggregation(ro_ordered_ranks_f)
Y_ro_f = order_to_ranks(ro_flipped_ranks)

print("Aggregated ranks based on ordered pair distance: ", Y_ro_f)

Aggregated ranks based on ordered pair distance: [8, 7, 3, 5, 4, 6, 2, 1]
```

1.3.3 Combined Results

```
In [26]: y_ro_c = rank_list(np.mean(ro_ranks_c,axis = 0))

print("Means: ",np.mean(ro_ranks_c,axis = 0))
print("Variance: ",np.var(ro_ranks_c,axis = 0))
print("Aggregated ranks based on mean values: ",y_ro_c)

Means: [5.45945946 5.67567568 4.27027027 6.35135135 3.27027027 5.64864865
2.67567568 2.64864865]
Variance: [3.81592403 3.30021914 2.41344047 3.47114682 3.00803506 4.71439007
3.89481373 2.01168736]
Aggregated ranks based on mean values: [5, 7, 4, 8, 3, 6, 2, 1]
```

```
In [27]: ro_combined_ranks = brute_rank_aggregation(ro_ordered_ranks_c)
Y_ro_c = order_to_ranks(ro_combined_ranks)

print("Aggregated ranks based on ordered pair distance: ", Y_ro_c)
```

Aggregated ranks based on ordered pair distance: [7, 6, 4, 8, 2, 5, 1, 3]

2 Feature functions

2.1 Requirement Keyword match

Uses the word tokenizer from NLTK to create two sets of tokens, one for the CV and one for the requirements in the job description. Tokens are typically the individual words from a sentence, with the punctuation removed. The Requirement Keyword match feature returns the ratio of matched tokens against the number of requirement tokens.

Sets(unique words) are used for the ratio calculation because we want to get a naive sense of how many requirements the candidate meets rather than how well the requirements are met.

```

In [28]: def requirement_keyword_match(cv, requirements_path):
cv_text = []
all_stopwords = nlp.Defaults.stop_words

# collect all of the words from the resume's experience section
for experience in cv['experience']:
    for responsibility in experience['responsibilities']:
        tokens = word_tokenize(responsibility)

        for token in tokens:
            if not token in all_stopwords:
                cv_text.append(token)

# collect all of the words from the resume's general skills
for skill in cv['skills']['general']:
    tokens = word_tokenize(skill)

    for token in tokens:
        if not token in all_stopwords:
            cv_text.append(token)

# collect all of the words from the resume's technical section
for skill in cv['skills']['technical']:
    tokens = word_tokenize(skill)

    for token in tokens:
        if not token in all_stopwords:
            cv_text.append(token)

require_tokens = []
# collect all of the words from the job's requirements
with open(requirements_path) as requirements_file:
    requirements = csv.reader(requirements_file)

    for requirement in requirements:

        tokens = word_tokenize(requirement[0])

        for token in tokens:
            if not token in all_stopwords:
                require_tokens.append(token)

# filter out punctuation
require_tokens_filtered = [word for word in require_tokens if not word in
[' ', '(', ')', '/', '.', '&']]
cv_text_filtered = [word for word in cv_text if not word in [' ', '(', ')',
 '/', '.', '&']]

requirement_keyword_match = set(require_tokens_filtered).intersection(cv_t
ext_filtered)

return len(requirement_keyword_match)/len(require_tokens_filtered)

```

2.2 Relevant Experience Feature

2.2.1 Word Vector Visualisation

t-distributed stochastic neighbor embedding/vector

```
In [ ]: embeddings = []
key_words = ['purple', 'white', 'red', 'black', 'accountant', 'officer', 'developer',
             'car', 'bicycle', 'truck', 'van']

for word in key_words:
    embeddings.append(nlp.vocab[word].vector)

tsne = TSNE(n_components=2, random_state=0, perplexity=5)
Y = tsne.fit_transform(embeddings)
plt.scatter(Y[:, 0], Y[:, 1])
for i, txt in enumerate(key_words):
    plt.annotate(txt, (Y[i,0], Y[i,1]), textcoords="offset points", xytext=(5,5))
))
```

The shortest amount of time for a previous job on a resume is a few months. To make the granularity consistent, we convert all job lengths into units of months.

```
In [30]: # expected format of inputs is datetime
def months_length(start_datetime, end_datetime):

    return (end_datetime.year - start_datetime.year) * 12 + end_datetime.month
    - start_datetime.month
```

The doc similarity feature converts the resume and the job description into word vectors, and then compares how similar the word vectors are. As a preprocessing step, the common/stop words (e.g 'a','and','the') and numbers are removed because they provide no information on how similar two sentences are. The spaCy library was used to calculate the word vectors.

Through the testing of the features, we found that the lowest similarity rating for jobs that were completely irrelevant to the Unibank role was ~75%, while the best matches were 90-95% rating. The small difference between the lowest and highest similarity is due to the poor performance word vectors have when comparing large amounts of words. Despite this flaw, we used word vectors because the complexity of the technique is low and we have a clear understanding of how bias can exist in the method.

Because the difference between the lowest and highest similarity rating is relatively small, this causes the issue where for example 5 years experience in a irrelevant job is valued higher than two years experience in an ideal role. To address this issue we included a power variable, which is used to increase the difference between the best and worst match by taking the power of the similarity based on the input variable. E.g. if the 'power_weight' is set to 20, then the worst match would equal $0.75^{20}=0.003$ and the best match would equal $0.9^{20}=0.122$.

```
In [31]: def doc_similarity(doc1, doc2, power_weight):
    all_stopwords = nlp.Defaults.stop_words

    doc1_tokens = []

    tokens = word_tokenize(doc1)

    for token in tokens:
        if (not token in all_stopwords) and (token.isalpha()) and (str(token).
lower() not in doc1_tokens):
            doc1_tokens.append(str(token).lower())

    doc1_nlp = nlp(' '.join(doc1_tokens))

    doc2_tokens = []

    tokens = word_tokenize(doc2)

    for token in tokens:
        if (not token in all_stopwords) and (token.isalpha()) and (str(token).
lower() not in doc2_tokens):
            doc2_tokens.append(str(token).lower())

    doc2_nlp = nlp(' '.join(doc2_tokens))

    return (doc1_nlp.similarity(doc2_nlp))**power_weight
```

Calculates the number of months the candidate spent in each previous job, and the weights the total number of months based on document similarity via word vectorization with spaCy.

```
In [32]: def relevant_experience(cv, responsibilities_path):  
  
    # combines the job responsibilities into a single string for the word vect  
    or  
    require_string = ""  
    with open(responsibilities_path) as responsibilities_file:  
        responsibilities = csv.reader(responsibilities_file)  
  
        for responsibility in responsibilities:  
  
            require_string += responsibility[0] +'. '.  
  
    total = 0  
    for job in cv['experience']:  
  
        start_datetime = datetime.strptime(job['start-date'], '%B %Y')  
  
        if job['end-date'] == 'Present':  
  
            end_datetime = datetime.now()  
  
        else:  
  
            end_datetime = datetime.strptime(job['end-date'], '%B %Y')  
  
        job_length = months_length(start_datetime, end_datetime)  
  
        relevance = doc_similarity(' '.join(job['responsibilities']),require_s  
tring,20)  
  
        total += job_length*relevance  
  
    return total
```

2.3 Education Requirements

This feature simply checks whether the candidate has a degree that meets the role requirements, and returns true if they do. As part of the preprocessing, we have set the degree level, and we had to hard-code relevant fields for the roles to reduce complexity.

```
In [33]: def education_requirement(fields, levels, cv):  
    for degree in cv['education']:  
        if degree['degree-level'].lower() in levels:  
            for field in fields:  
                if field in degree['field'].lower():  
                    return 1  
  
    return 0
```

3 Feature Computation

We stored the genders of the candidates in CSV files, so we could easily load and reference them when checking gender bias.

```
In [34]: with open("CV Json Research Data/Job 1 - Data Analyst/gender.csv") as gender_file:
          da_gender = list(csv.reader(gender_file))

          with open("CV Json Research Data/Job 2 - Finance Officer/gender.csv") as gender_file:
              fo_gender = list(csv.reader(gender_file))

          with open("CV Json Research Data/Job 3 - Recruitment Officer/gender.csv") as gender_file:
              ro_gender = list(csv.reader(gender_file))
```

The resume data is loaded from the JSON files and passed through each of the feature functions to calculate the inputs for the models. The Data Analyst does not have the education feature as it was not a requirement in the job description. The gender feature is labelled as feature #3 because it was developed after the education feature.


```
In [228]: X_0_da = [] # Relevant Experience Feature
X_1_da = [] # Requirement Keyword Match Feature
X_3_da = [] # Gender Feature

files = os.listdir("CV Json Research Data/Job 1 - Data Analyst/")

for file in files:

    if file.split('.')[-1] == "json":

        path = "CV Json Research Data/Job 1 - Data Analyst/" + file

        with open(path, encoding="utf8") as json_file:
            cv = json.load(json_file)

            X_0_da.append(relevant_experience(cv, "CV Json Research Data/Job 1 - D
ata Analyst/job_requirements.csv"))
            X_1_da.append(requirement_keyword_match(cv, "CV Json Research Data/Job
1 - Data Analyst/job_requirements.csv"))

for gender in da_gender[0]:
    if gender == 'Female':
        X_3_da.append(0)
    else:
        X_3_da.append(1)

X_da = np.column_stack((X_0_da,X_1_da,X_3_da))
```

```
In [229]: X_0_fo = [] # Relevant Experience Feature
X_1_fo = [] # Requirement Keyword Match Feature
X_2_fo = [] # Education Feature
X_3_fo = [] # Gender Feature

required_fields = ['finance', 'accounting', 'accountancy']
required_levels = ['bachelor', 'master', 'graduate conversion course']
files = os.listdir("CV Json Research Data/Job 2 - Finance Officer/")

for file in files:

    if file.split('.')[-1] == "json":

        path = "CV Json Research Data/Job 2 - Finance Officer/" + file

        with open(path, encoding="utf8") as json_file:
            cv = json.load(json_file)

            X_0_fo.append(relevant_experience(cv, "CV Json Research Data/Job 2 - F
inance Officer/job_requirements.csv"))
            X_1_fo.append(requirement_keyword_match(cv, "CV Json Research Data/Job
2 - Finance Officer/job_requirements.csv"))
            X_2_fo.append(education_requirement(required_fields, required_levels, c
v))

for gender in fo_gender[0]:
    if gender == 'Female':
        X_3_fo.append(0)
    else:
        X_3_fo.append(1)

X_fo = np.column_stack((X_0_fo, X_1_fo, X_2_fo, X_3_fo))
```

```
In [230]: X_0_ro = [] # Relevant Experience Feature
X_1_ro = [] # Requirement Keyword Match Feature
X_2_ro = [] # Education Feature
X_3_ro = [] # Gender Feature

required_fields = ['human resource']
required_levels = ['bachelor', 'master', 'graduate conversion course', 'diploma',
'certificate iv', 'post graduate diploma']
files = os.listdir("CV Json Research Data/Job 3 - Recruitment Officer/")

for file in files:

    if file.split('.')[-1] == "json":

        path = "CV Json Research Data/Job 3 - Recruitment Officer/" + file

        with open(path, encoding="utf8") as json_file:
            cv = json.load(json_file)

            X_0_ro.append(relevant_experience(cv, "CV Json Research Data/Job 3 - R
ecruitment Officer/job_requirements.csv"))
            X_1_ro.append(requirement_keyword_match(cv, "CV Json Research Data/Job
3 - Recruitment Officer/job_requirements.csv"))
            X_2_ro.append(education_requirement(required_fields, required_levels, c
v))

for gender in ro_gender[0]:
    if gender == 'Female':
        X_3_ro.append(0)
    else:
        X_3_ro.append(1)

X_ro = np.column_stack((X_0_ro, X_1_ro, X_2_ro, X_3_ro))
```

We've combined the three datasets from the different roles to see whether there is a more general relationship between the features and the rankings.

```
In [232]: X_0 = X_0_da + X_0_fo + X_0_ro
X_1 = X_1_da + X_1_fo + X_1_ro
X = np.column_stack((X_0, X_1))
y = Y_da_c + Y_fo_c + Y_ro_c
```

3.1 Feature Z-scores

We tested the idea of using relative feature values, to see whether this improves the models. I.e. if one of the candidates have much less experience compared to the average experience of all of the the candidates then you would expect the candidate to be ranked lower. But we found that this z-score version of the features did not improve the model.

```
In [225]: scaler = StandardScaler()
X_0_da_zscore = scaler.fit_transform(np.array(X_0_da).reshape(-1,1))
X_1_da_zscore = scaler.fit_transform(np.array(X_1_da).reshape(-1,1))
X_da_zscore = np.column_stack((X_0_da_zscore,X_1_da_zscore,X_3_da))
```

```
In [226]: scaler = StandardScaler()
X_0_fo_zscore = scaler.fit_transform(np.array(X_0_fo).reshape(-1,1))
X_1_fo_zscore = scaler.fit_transform(np.array(X_1_fo).reshape(-1,1))
X_fo_zscore = np.column_stack((X_0_fo_zscore,X_1_fo_zscore,X_2_fo,X_3_fo))
```

```
In [227]: scaler = StandardScaler()
X_0_ro_zscore = scaler.fit_transform(np.array(X_0_ro).reshape(-1,1))
X_1_ro_zscore = scaler.fit_transform(np.array(X_1_ro).reshape(-1,1))
X_ro_zscore = np.column_stack((X_0_ro_zscore,X_1_ro_zscore,X_2_ro,X_3_ro))
```

4 Feature Analysis

4.1 All jobs Analysis

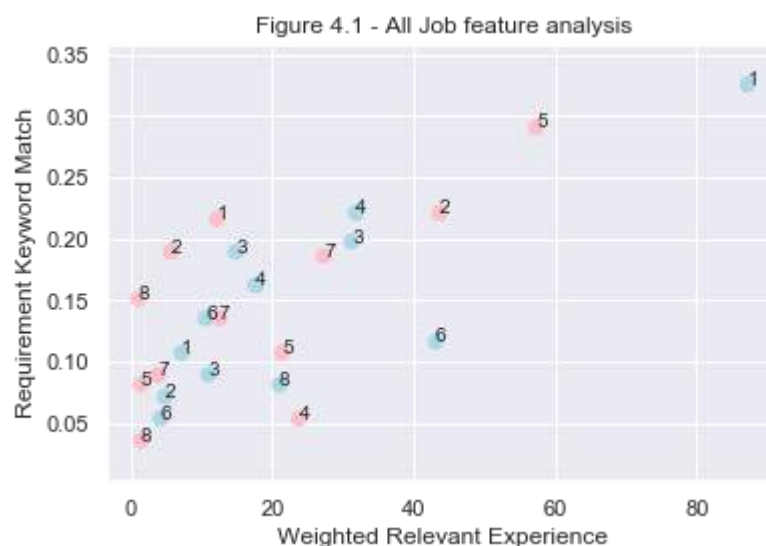
The idea behind this initial analysis was to check whether there was any underlying pattern/relation between the features and the ranks, regardless of the type of job being applied for. I.e. Is the value of meeting requirements and having experience universal across all jobs?

The numbers on the data points in Figure 4.1 represent the rank the resume received.

```
In [233]: # hard-coded colours to represent the genders of the candidates
da_colours = ['pink', 'lightblue', 'lightblue', 'pink', 'lightblue', 'lightblue',
'pink', 'pink']
fo_colours = ['lightblue', 'pink', 'pink', 'pink', 'lightblue', 'lightblue', 'pink', 'lightblue']
ro_colours = ['pink', 'lightblue', 'lightblue', 'lightblue', 'pink', 'pink', 'pink', 'lightblue']

colours = da_colours + fo_colours + ro_colours
plt.scatter(X_0, X_1, c=colours, s=50, cmap='viridis')
plt.xlabel("Weighted Relevant Experience")
plt.ylabel("Requirement Keyword Match")
plt.title("Figure 4.1 - All Job feature analysis")

for i, txt in enumerate(y):
    plt.annotate(txt, (X_0[i], X_1[i]))
```



```
In [235]: reg = LinearRegression().fit(X, y)
print("Regression R^2 score:", reg.score(X, y))
```

Regression R² score: 0.18658346115655233

As we can see from Figure 4.1, there is no clear pattern between the features and ranks of the resumes. This is confirmed by the low R² value from our regression model. This suggests the experience and met requirements of the candidates are valued differently across the different jobs; that the differences between fields is too great to develop a general model.

4.2 Gender Distributions of Relevant Experience

To check whether there is any significant bias towards either of the genders for our computed features, we plotted distribution plots as a high level test. Some of the distributions have unusual shapes due to the size of the dataset.

```

In [241]: X_0_female = []
X_0_male = []

# go through each feature vector for the roles and split them based on gender
for x_0, gender in zip(X_0_da,da_gender[0]):
    if gender == "Male":
        X_0_male.append(x_0)
    else:
        X_0_female.append(x_0)

print("Female Experience",X_0_female)
print("Male Experience",X_0_male)

print("Average Female Experience",np.mean(X_0_female, axis=0))
print("Average Male Experience",np.mean(X_0_male, axis=0))

sns.color_palette("Set2")

sns.distplot(X_0_female, hist = False,
             label = "Female", color="royalblue",kde_kws={'linestyle':'--'})
).set(xlim=(0))

sns.distplot(X_0_male, hist = False,
             label = "Male", color="royalblue").set(xlim=(0))
plt.title("Figure 4.2.1: Gender Distribution of Relevant Experience Feature (Data Analyst)")
plt.xlabel("Relevant Experience (Weighted Months)")
plt.ylabel("Normalised Distribution")

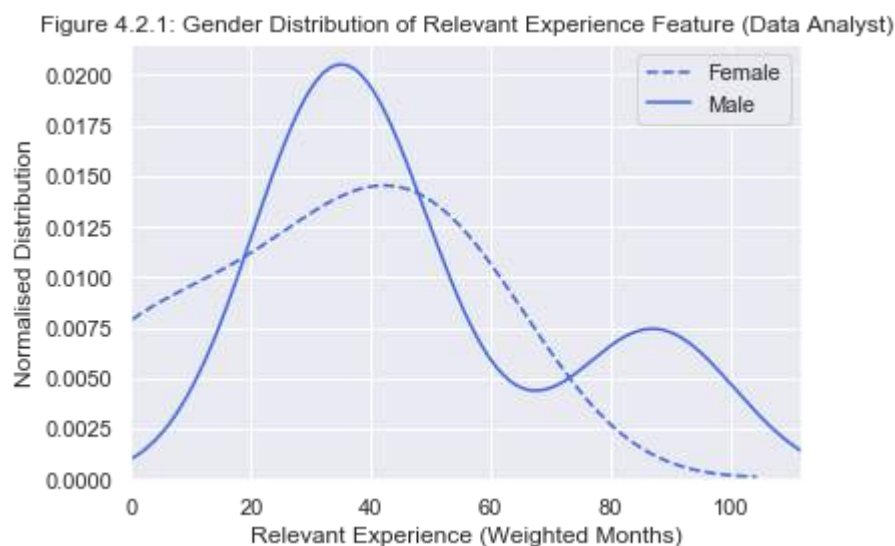
```

```

Female Experience [1.0629323104052746, 27.270870156624024, 57.31516723095554
5, 43.610573199880065]
Male Experience [87.2138513474456, 31.23897839674321, 43.11847853305852, 31.8
7005233462274]
Average Female Experience 32.31488572446622
Average Male Experience 48.360340152967524

```

Out[241]: Text(0, 0.5, 'Normalised Distribution')



```

In [243]: X_0_female = []
X_0_male = []

# go through each feature vector for the roles and split them based on gender
for x_0, gender in zip(X_0_fo,fo_gender[0]):
    if gender == "Male":
        X_0_male.append(x_0)
    else:
        X_0_female.append(x_0)

print("Female Experience",X_0_female)
print("Male Experience",X_0_male)

print("Average Female Experience",np.mean(X_0_female, axis=0))
print("Average Male Experience",np.mean(X_0_male, axis=0))

sns.distplot(X_0_female, hist = False,
              label = "Female",color="royalblue",kde_kws={'linestyle':'- -'
              }).set(xlim=(0))

sns.distplot(X_0_male, hist = False,
              label = "Male",color="royalblue").set(xlim=(0))

plt.title("Figure 4.2.2: Gender Distribution of Relevant Experience Feature (F
inance Officer)")
plt.xlabel("Relevant Experience (Weighted Months)")
plt.ylabel("Normalised Distribution")

```

Female Experience [21.383032371633487, 1.4829252248607538, 3.842338536636174, 23.81311702962811]

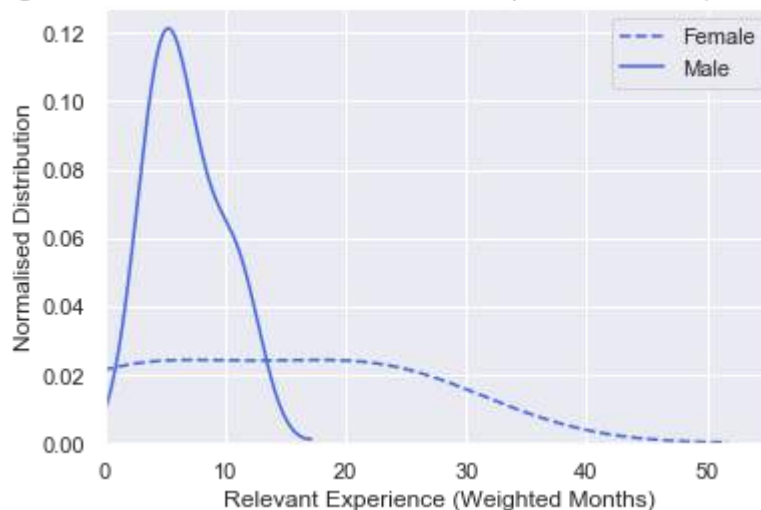
Male Experience [4.844861364879119, 7.204622754851076, 4.198750909885769, 10.973704127688382]

Average Female Experience 12.63035329068963

Average Male Experience 6.805484789326086

Out[243]: Text(0, 0.5, 'Normalised Distribution')

Figure 4.2.2: Gender Distribution of Relevant Experience Feature (Finance Officer)



```

In [245]: X_0_female = []
X_0_male = []

# go through each feature vector for the roles and split them based on gender
for x_0, gender in zip(X_0_ro,ro_gender[0]):
    if gender == "Male":
        X_0_male.append(x_0)
    else:
        X_0_female.append(x_0)

print(X_0_female)
print(X_0_male)

print("Average Female Experience",np.mean(X_0_female, axis=0))
print("Average Male Experience",np.mean(X_0_male, axis=0))

sns.distplot(X_0_female, hist = False,
              label = "Female",color="royalblue",kde_kws={'linestyle':'--'
})

sns.distplot(X_0_male, hist = False,
              label = "Male",color="royalblue").set(xlim=(0))
plt.title("Figure 4.2.3: Gender Distribution of Relevant Experience Feature (R
ecruitment Officer)")
plt.xlabel("Relevant Experience (Weighted Months)")
plt.ylabel("Normalised Distribution")

```

```

[12.509130667169947, 5.655672128623767, 1.392999145238279, 12.20264435468282
8]

```

```

[10.608242768666603, 17.66933189635241, 21.04882082518529, 14.83590493428790
1]

```

```

Average Female Experience 7.9401115739287045

```

```

Average Male Experience 16.04057510612305

```

```

Out[245]: Text(0, 0.5, 'Normalised Distribution')

```

Figure 4.2.3: Gender Distribution of Relevant Experience Feature (Recruitment Officer)




```
In [246]: X_0_female = []
X_0_male = []

# go through each feature vector for the roles and split them based on gender
for x_0, gender in zip(X_0_da,da_gender[0]):
    if gender == "Male":
        X_0_male.append(x_0)
    else:
        X_0_female.append(x_0)

for x_0, gender in zip(X_0_fo,fo_gender[0]):
    if gender == "Male":
        X_0_male.append(x_0)
    else:
        X_0_female.append(x_0)

for x_0, gender in zip(X_0_ro,ro_gender[0]):
    if gender == "Male":
        X_0_male.append(x_0)
    else:
        X_0_female.append(x_0)

print("Female Experience",X_0_female)
print("Male Experience",X_0_male)

print("Average Female Experience",np.mean(X_0_female, axis=0))
print("Average Male Experience",np.mean(X_0_male, axis=0))

sns.distplot(X_0_male, hist = False,
              label = "Male",color="royalblue").set(xlim=(0))
sns.distplot(X_0_female, hist = False,
              label = "Female", color="royalblue",kde_kws={'linestyle':'--'
}).set(xlim=(0))

plt.title("Figure 4.2.4: Gender Distribution of Relevant Experience Feature (A
ll roles)")
plt.xlabel("Relevant Experience (Weighted Months)")
plt.ylabel("Normalised Distribution")
```

Female Experience [1.0629323104052746, 27.270870156624024, 57.315167230955545, 43.610573199880065, 21.383032371633487, 1.4829252248607538, 3.842338536636174, 23.81311702962811, 12.509130667169947, 5.655672128623767, 1.392999145238279, 12.202644354682828]

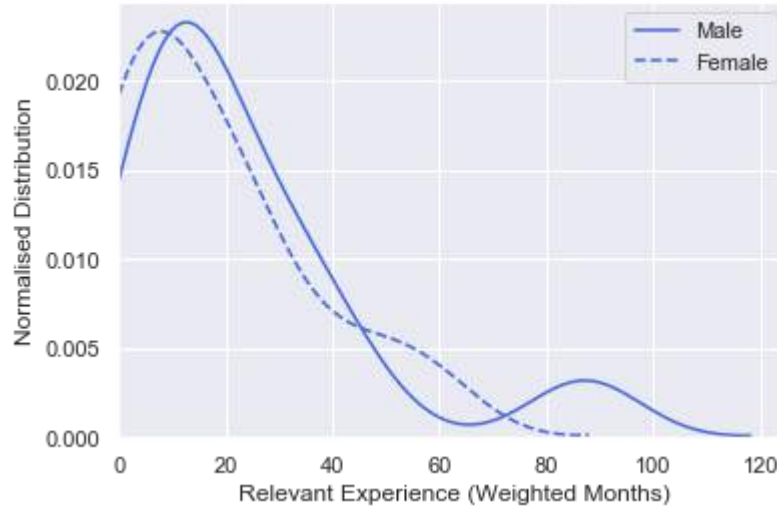
Male Experience [87.2138513474456, 31.23897839674321, 43.11847853305852, 31.87005233462274, 4.844861364879119, 7.204622754851076, 4.198750909885769, 10.973704127688382, 10.608242768666603, 17.66933189635241, 21.04882082518529, 14.835904934287901]

Average Female Experience 17.628450196361523

Average Male Experience 23.73546668280555

Out[246]: Text(0, 0.5, 'Normalised Distribution')

Figure 4.2.4: Gender Distribution of Relevant Experience Feature (All roles)



Based on Figure 4.2.4, we can see the male candidates have a slightly higher average level of experience compared to the female candidates, and one of the male candidates is an outlier.

4.3 Gender Distribution of Keyword Match

We repeat the same process as the previous section to check for significant bias in the keywords feature. Once again, some of the distributions have unusual shapes due to the size of the dataset.

```

In [247]: X_1_female = []
X_1_male = []

# go through each feature vector for the roles and split them based on gender
for x_1, gender in zip(X_1_da, da_gender[0]):
    if gender == "Male":
        X_1_male.append(x_1)
    else:
        X_1_female.append(x_1)

print(X_1_female)
print(X_1_male)

print("Average Female Keywords", np.mean(X_1_female, axis=0))
print("Average Male Keywords", np.mean(X_1_male, axis=0))

sns.distplot(X_1_female, hist = False, bins=50,
              label = "Female", color="royalblue", kde_kws={'linestyle':'--'
}).set(xlim=(0))

sns.distplot(X_1_male, hist = False, bins=50,
              label = "Male", color="royalblue").set(xlim=(0))
plt.title("Figure 4.3.1: Gender Distribution of Requirement Keyword Feature (D
ata Analyst)")
plt.xlabel("Requirement Keyword Match")
plt.ylabel("Normalised Distribution")

```

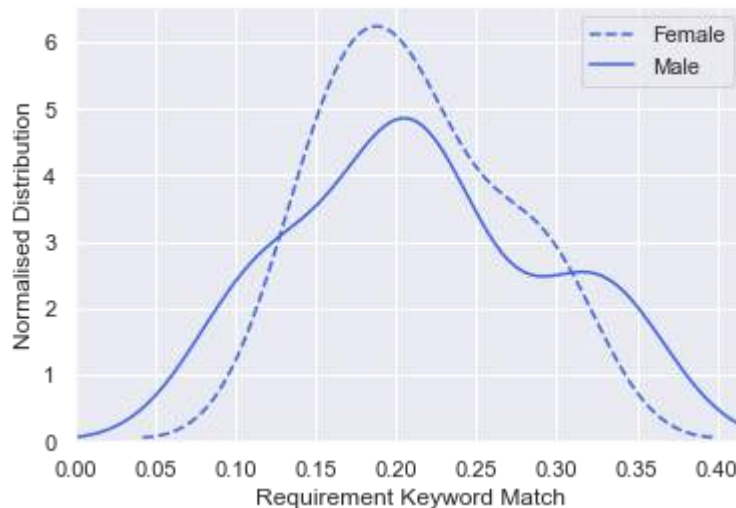
```

[0.1511627906976744, 0.18604651162790697, 0.29069767441860467, 0.220930232558
13954]
[0.32558139534883723, 0.19767441860465115, 0.11627906976744186, 0.22093023255
813954]
Average Female Keywords 0.21220930232558138
Average Male Keywords 0.21511627906976744

```

Out[247]: Text(0, 0.5, 'Normalised Distribution')

Figure 4.3.1: Gender Distribution of Requirement Keyword Feature (Data Analyst)



```

In [248]: X_1_female = []
X_1_male = []

# go through each feature vector for the roles and split them based on gender
for x_1, gender in zip(X_1_fo,fo_gender[0]):
    if gender == "Male":
        X_1_male.append(x_1)
    else:
        X_1_female.append(x_1)

print(X_1_female)
print(X_1_male)

print("Average Female Keywords",np.mean(X_1_female, axis=0))
print("Average Male Keywords",np.mean(X_1_male, axis=0))

sns.distplot(X_1_female, hist = False,bins=50,
              label = "Female",color="royalblue",kde_kws={'linestyle':'--'})
).set(xlim=(0))

sns.distplot(X_1_male, hist = False, bins=50,
              label = "Male",color="royalblue").set(xlim=(0))
plt.title("Figure 4.3.2: Gender Distribution of Requirement Keyword Feature (F
inance Officer)")
plt.xlabel("Requirement Keyword Match")
plt.ylabel("Normalised Distribution")

```

```

[0.10714285714285714, 0.03571428571428571, 0.08928571428571429, 0.05357142857
142857]

```

```

[0.07142857142857142, 0.10714285714285714, 0.05357142857142857, 0.08928571428
571429]

```

```

Average Female Keywords 0.07142857142857142

```

```

Average Male Keywords 0.08035714285714285

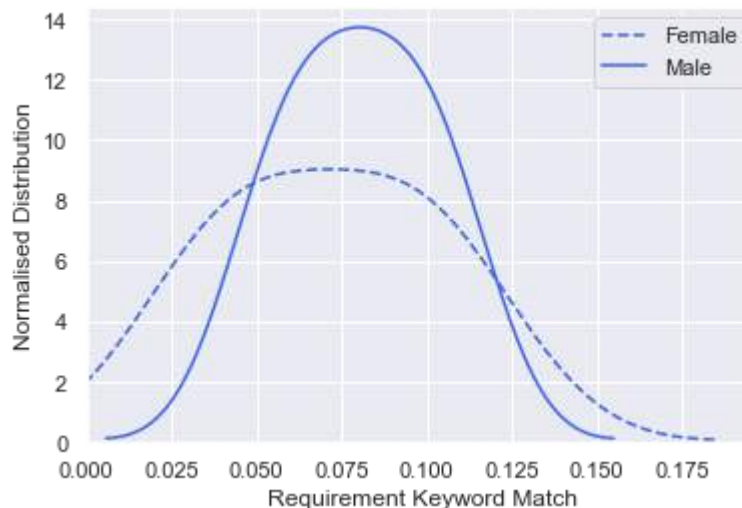
```

```

Out[248]: Text(0, 0.5, 'Normalised Distribution')

```

Figure 4.3.2: Gender Distribution of Requirement Keyword Feature (Finance Officer)



```

In [249]: X_1_female = []
X_1_male = []

# go through each feature vector for the roles and split them based on gender
for x_1, gender in zip(X_1_ro,ro_gender[0]):
    if gender == "Male":
        X_1_male.append(x_1)
    else:
        X_1_female.append(x_1)

print(X_1_female)
print(X_1_male)

print("Average Female Keywords",np.mean(X_1_female, axis=0))
print("Average Male Keywords",np.mean(X_1_male, axis=0))

sns.distplot(X_1_female, hist = False, bins=50,
              label = "Female", color="royalblue", kde_kws={'linestyle':'--'
}).set(xlim=(0))

sns.distplot(X_1_male, hist = False, bins=50,
              label = "Male", color="royalblue").set(xlim=(0))
plt.title("Figure 4.3.3: Gender Distribution of Requirement Keyword Feature (R
ecruitment Officer)")
plt.xlabel("Requirement Keyword Match")
plt.ylabel("Normalised Distribution")

```

```
[0.13513513513513514, 0.1891891891891892, 0.08108108108108109, 0.216216216216
21623]
```

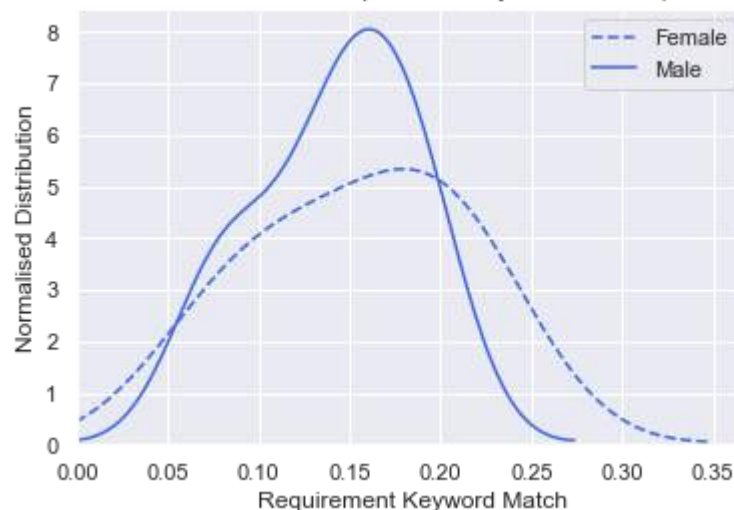
```
[0.13513513513513514, 0.16216216216216217, 0.08108108108108109, 0.18918918918
91892]
```

```
Average Female Keywords 0.15540540540540543
```

```
Average Male Keywords 0.14189189189189189
```

Out[249]: Text(0, 0.5, 'Normalised Distribution')

Figure 4.3.3: Gender Distribution of Requirement Keyword Feature (Recruitment Officer)



```
In [250]: X_1_female = []
X_1_male = []

for x_1, gender in zip(X_1_da,da_gender[0]):
    if gender == "Male":
        X_1_male.append(x_1)
    else:
        X_1_female.append(x_1)

for x_1, gender in zip(X_1_fo,fo_gender[0]):
    if gender == "Male":
        X_1_male.append(x_1)
    else:
        X_1_female.append(x_1)

for x_1, gender in zip(X_1_ro,ro_gender[0]):
    if gender == "Male":
        X_1_male.append(x_1)
    else:
        X_1_female.append(x_1)

# print(X_1_female)
# print(X_1_male)

print("Average Female Keywords",np.mean(X_1_female, axis=0))
print("Average Male Keywords",np.mean(X_1_male, axis=0))

sns.distplot(X_1_female, hist = False, bins=50,
              label = "Female", color="royalblue", kde_kws={'linestyle':'--'
}).set(xlim=(0))

sns.distplot(X_1_male, hist = False, bins=50,
              label = "Male", color="royalblue").set(xlim=(0))
plt.title("Figure 4.3.4: Gender Distribution of Requirement Keyword Feature (A
ll roles)")
plt.xlabel("Requirement Keyword Match")
plt.ylabel("Normalised Distribution")
```

Average Female Keywords 0.14634775971985275
Average Male Keywords 0.14578843793960075

Out[250]: Text(0, 0.5, 'Normalised Distribution')

Figure 4.3.4: Gender Distribution of Requirement Keyword Feature (All roles)

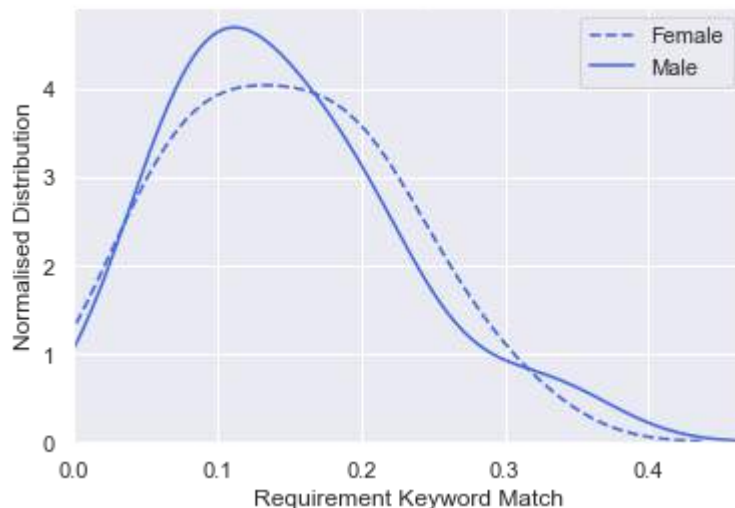


Figure 4.3.4 shows that the female candidates have a slightly high average of meeting the keyword requirements, but this doesn't appear to be significantly higher.

5 Linear Regression

Using the features calculated in the previous section, we used a linear regression model to model the rankings of the candidates. We used the statsmodel library to model the regression, as it provides accurate p-values for the significant of the features, and provides a regression table output for easier interpretation of the results. The features were converted to matrices because it provides clear labelling of the features in the regression table output.

As we will see below, not all of the features are significant/relevant for the rankings for each dataset. To find the optimal model we use a basic and manual step method, where we iteratively remove non-significant features until the model has only significant features.

5.1 Data Analyst

5.1.1 Original Genders

All features (Experience, Keywords, Gender)

```
In [256]: df = pd.DataFrame({'Experience':X_0_da,'Keywords':X_1_da,'Gender':X_3_da,'Rank':Y_da_o})
y, X = dmatrices('Rank ~ Experience + Keywords + Gender', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```


OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.92
6
Model:                  OLS    Adj. R-squared:        0.87
0
Method:                 Least Squares    F-statistic:           16.6
7
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):    0.010
0
Time:                   11:50:05    Log-Likelihood:        -7.572
2
No. Observations:      8    AIC:                   23.1
4
Df Residuals:          4    BIC:                   23.4
6
Df Model:               3
Covariance Type:       nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept    11.1018      1.289      8.614    0.001      7.523    14.68
0
Experience   -0.0296      0.024     -1.226    0.287     -0.097     0.03
7
Keywords    -25.4283      8.250     -3.082    0.037    -48.334    -2.52
3
Gender       0.0484      0.724      0.067    0.950     -1.961     2.05
8
=====

```

```

=====
=
Omnibus:          0.795    Durbin-Watson:         1.13
5
Prob(Omnibus):    0.672    Jarque-Bera (JB):      0.52
1
Skew:             -0.524    Prob(JB):              0.77
1
Kurtosis:         2.317    Cond. No.               1.25e+0
3
=====
=

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.25e+03. This might indicate that there are

strong multicollinearity or other numerical problems.



Based on the results we see that the Gender feature is the least significant feature, so we remove it from the model and re-fit the model with just the Experience and Keywords features.

Significant features (Experience, Keywords)

```
In [257]: df = pd.DataFrame({'Experience':X_0_da,'Keywords':X_1_da,'Rank':Y_da_o})
y, X = dmatrices('Rank ~ Experience + Keywords', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.92
6
Model:                  OLS    Adj. R-squared:         0.89
6
Method:                 Least Squares    F-statistic:           31.2
2
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):    0.0015
0
Time:                   11:50:12    Log-Likelihood:        -7.576
7
No. Observations:      8    AIC:                   21.1
5
Df Residuals:          5    BIC:                   21.3
9
Df Model:               2
Covariance Type:       nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
Intercept    11.1400      1.034     10.775     0.000      8.482    13.79
8
Experience   -0.0288      0.019     -1.546     0.183     -0.077     0.01
9
Keywords    -25.6486      6.770     -3.788     0.013    -43.052    -8.24
5
=====

```

```

=====
=
Omnibus:              0.999    Durbin-Watson:         1.07
0
Prob(Omnibus):        0.607    Jarque-Bera (JB):      0.53
0
Skew:                 -0.564    Prob(JB):              0.76
7
Kurtosis:             2.437    Cond. No.               1.14e+0
3
=====
=

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.14e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Generally the convention is to use a threshold of 0.05 for determining whether a feature is significant or not. Therefore Experience is not significant, and we refit the model with just Keywords.

Significant features (Keywords)

```
In [258]: df = pd.DataFrame({'Keywords':X_1_da,'Rank':Y_da_o})
y, X = dmatrices('Rank ~ Keywords', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:          0.89
0
Model:                  OLS    Adj. R-squared:    0.87
2
Method:                 Least Squares    F-statistic:       48.7
5
Date:                   Fri, 25 Sep 2020    Prob (F-statistic): 0.00042
9
Time:                   11:50:13    Log-Likelihood:    -9.140
4
No. Observations:      8    AIC:               22.2
8
Df Residuals:          6    BIC:               22.4
4
Df Model:               1
Covariance Type:       nonrobust
=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept    11.6955    1.076    10.869    0.000     9.062    14.32
9
Keywords    -33.6771    4.823    -6.982    0.000   -45.479   -21.87
5
=====
=
Omnibus:           0.278    Durbin-Watson:      1.25
4
Prob(Omnibus):     0.870    Jarque-Bera (JB):    0.09
0
Skew:              0.130    Prob(JB):            0.95
6
Kurtosis:          2.551    Cond. No.            16.
3
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Z-score - All features (Experience, Keywords, Gender)

We fitted the model with using the z-score versions of the Experience and Keywords features but we found that we get the same results as the standard model. Z-score versions do not add anything to the model, so we just use the basic version.

```
In [259]: df = pd.DataFrame({'Experience':np.transpose(X_0_da_zscore)[0], 'Keywords':np.t  
ranspose(X_1_da_zscore)[0], 'Gender':X_3_da, 'Rank':Y_da_o})  
y, X = dmatrices('Rank ~ Experience + Keywords + Gender', data=df, return_type  
='dataframe')  
mod = sm.OLS(y,X)  
res = mod.fit()  
print(res.summary())
```

OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.92
6
Model:                  OLS    Adj. R-squared:         0.87
0
Method:                 Least Squares    F-statistic:           16.6
7
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):    0.010
0
Time:                   11:50:14    Log-Likelihood:        -7.572
2
No. Observations:      8    AIC:                   23.1
4
Df Residuals:          4    BIC:                   23.4
6
Df Model:               3
Covariance Type:       nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept      4.4758      0.478      9.369      0.001      3.149      5.80
2
Experience     -0.6913      0.564     -1.226      0.287     -2.257      0.87
4
Keywords       -1.6325      0.530     -3.082      0.037     -3.103     -0.16
2
Gender         0.0484      0.724      0.067      0.950     -1.961      2.05
8
=====

```

```

=====
=
Omnibus:           0.795    Durbin-Watson:         1.13
5
Prob(Omnibus):     0.672    Jarque-Bera (JB):      0.52
1
Skew:              -0.524    Prob(JB):              0.77
1
Kurtosis:          2.317    Cond. No.              4.0
7
=====
=

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



5.1.2 Flipped Genders

For the following dataset and different jobs, we repeated the same step process to get the best fit model for the data, given the features.

All features (Experience, Keywords, Gender)

```
In [261]: df = pd.DataFrame({'Experience':X_0_da,'Keywords':X_1_da,'Gender':X_3_da,'Rank':Y_da_f})
y, X = dmatrices('Rank ~ Experience + Keywords + Gender', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.80
4
Model:                  OLS    Adj. R-squared:        0.65
7
Method:                 Least Squares    F-statistic:           5.46
5
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):    0.067
2
Time:                   11:50:16    Log-Likelihood:        -11.46
9
No. Observations:      8    AIC:                   30.9
4
Df Residuals:          4    BIC:                   31.2
5
Df Model:               3
Covariance Type:      nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept      6.8792      2.098      3.280     0.031      1.056     12.70
3
Experience     -0.0635      0.039     -1.617     0.181     -0.172      0.04
6
Keywords       6.6982     13.427      0.499     0.644     -30.580     43.97
7
Gender        -2.5013      1.178     -2.123     0.101     -5.772      0.77
0
=====

```

```

=====
=
Omnibus:          1.779    Durbin-Watson:         1.76
7
Prob(Omnibus):    0.411    Jarque-Bera (JB):      1.10
6
Skew:             -0.788    Prob(JB):              0.57
5
Kurtosis:         2.085    Cond. No.               1.25e+0
3
=====
=

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.25e+03. This might indicate that there are

strong multicollinearity or other numerical problems.



Significant features (Experience, Gender)

```
In [263]: df = pd.DataFrame({'Experience':X_0_da, 'Keywords':X_1_da, 'Gender':X_3_da, 'Rank':Y_da_f})
y, X = dmatrices('Rank ~ Experience + Gender', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:                0.79
2
Model:                  OLS    Adj. R-squared:         0.70
8
Method:                 Least Squares    F-statistic:           9.50
0
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):    0.019
8
Time:                   11:50:17    Log-Likelihood:        -11.71
0
No. Observations:      8    AIC:                   29.4
2
Df Residuals:          5    BIC:                   29.6
6
Df Model:               2
Covariance Type:       nonrobust
=====
=
                    coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept             7.7892     0.955     8.159     0.000     5.335    10.24
3
Experience            -0.0476     0.021    -2.236     0.076    -0.102     0.00
7
Gender                -2.7357     0.996    -2.747     0.040    -5.296    -0.17
6
=====
=
Omnibus:               0.794    Durbin-Watson:         1.92
3
Prob(Omnibus):         0.672    Jarque-Bera (JB):      0.62
9
Skew:                  -0.517    Prob(JB):              0.73
0
Kurtosis:              2.095    Cond. No.               10
7.
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [264]: df = pd.DataFrame({'Experience':X_0_da, 'Keywords':X_1_da, 'Gender':X_3_da, 'Rank':Y_da_f})
y, X = dmatrices('Rank ~ Gender', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:                0.58
3
Model:                  OLS    Adj. R-squared:         0.51
4
Method:                 Least Squares    F-statistic:           8.40
0
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):    0.027
4
Time:                   11:50:18    Log-Likelihood:        -14.48
3
No. Observations:      8    AIC:                   32.9
7
Df Residuals:          6    BIC:                   33.1
2
Df Model:               1
Covariance Type:       nonrobust
=====
=

```

	coef	std err	t	P> t	[0.025	0.97
Intercept	6.2500	0.854	7.319	0.000	4.161	8.33
Gender	-3.5000	1.208	-2.898	0.027	-6.455	-0.54

```
=====
=
Omnibus:                1.452    Durbin-Watson:         1.71
4
Prob(Omnibus):          0.484    Jarque-Bera (JB):      0.44
4
Skew:                   0.000    Prob(JB):              0.80
1
Kurtosis:               1.846    Cond. No.              2.6
2
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Z-score - All features (Experience, Keywords, Gender)

Once again we found that z-score features gave the same results, so we expect this is very likely to be the case for all datasets, and will just use the original features.

```
In [265]: df = pd.DataFrame({'Experience':np.transpose(X_0_da_zscore)[0], 'Keywords':np.t
ranspose(X_1_da_zscore)[0], 'Gender':X_3_da, 'Rank':Y_da_f})
y, X = dmatrices('Rank ~ Experience + Keywords + Gender', data=df, return_type
='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```


OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.80
4
Model:                  OLS    Adj. R-squared:         0.65
7
Method:                 Least Squares    F-statistic:           5.46
5
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):    0.067
2
Time:                   11:50:19    Log-Likelihood:        -11.46
9
No. Observations:      8    AIC:                    30.9
4
Df Residuals:          4    BIC:                    31.2
5
Df Model:               3
Covariance Type:       nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept      5.7506      0.777        7.397    0.002      3.592      7.90
9
Experience     -1.4834      0.918       -1.617    0.181     -4.031      1.06
4
Keywords       0.4300      0.862        0.499    0.644     -1.963      2.82
3
Gender        -2.5013      1.178       -2.123    0.101     -5.772      0.77
0
=====

```

```

=====
=
Omnibus:          1.779    Durbin-Watson:         1.76
7
Prob(Omnibus):    0.411    Jarque-Bera (JB):      1.10
6
Skew:             -0.788    Prob(JB):              0.57
5
Kurtosis:         2.085    Cond. No.               4.0
7
=====
=

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



5.1.3 Combined

All features (Experience, Keywords, Gender)

```
In [267]: df = pd.DataFrame({'Experience':X_0_da,'Keywords':X_1_da,'Gender':X_3_da,'Rank':Y_da_c})
y, X = dmatrices('Rank ~ Experience + Keywords + Gender', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.64
3
Model:                  OLS    Adj. R-squared:         0.37
5
Method:                 Least Squares    F-statistic:           2.40
0
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):    0.20
8
Time:                   11:50:20    Log-Likelihood:        -13.86
6
No. Observations:      8    AIC:                   35.7
3
Df Residuals:          4    BIC:                   36.0
5
Df Model:               3
Covariance Type:       nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept      9.8117      2.831      3.466    0.026      1.953    17.67
1
Experience     -0.0253      0.053     -0.478    0.658     -0.172     0.12
2
Keywords      -16.4646     18.119     -0.909    0.415    -66.771    33.84
2
Gender        -1.5461      1.590     -0.972    0.386     -5.960     2.86
8
=====

```

```

=====
=
Omnibus:           6.556    Durbin-Watson:         1.95
1
Prob(Omnibus):     0.038    Jarque-Bera (JB):      2.24
0
Skew:              -1.278    Prob(JB):              0.32
6
Kurtosis:          3.433    Cond. No.               1.25e+0
3
=====
=

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.25e+03. This might indicate that there are

strong multicollinearity or other numerical problems.



Significant features (Experience)

```
In [269]: df = pd.DataFrame({'Experience':X_0_da, 'Keywords':X_1_da, 'Gender':X_3_da, 'Rank':Y_da_c})
y, X = dmatrices('Rank ~ Experience', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:                0.53
0
Model:                  OLS    Adj. R-squared:         0.45
1
Method:                 Least Squares    F-statistic:           6.75
5
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):    0.040
7
Time:                   11:50:21    Log-Likelihood:       -14.96
8
No. Observations:      8    AIC:                   33.9
4
Df Residuals:          6    BIC:                   34.0
9
Df Model:               1
Covariance Type:       nonrobust
=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept          7.3772      1.280      5.766      0.001      4.246     10.50
8
Experience         -0.0713      0.027     -2.599      0.041     -0.138     -0.00
4
=====
=
Omnibus:            2.346    Durbin-Watson:         2.53
7
Prob(Omnibus):      0.310    Jarque-Bera (JB):      0.87
4
Skew:               -0.284    Prob(JB):              0.64
6
Kurtosis:           1.483    Cond. No.               93.
0
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



5.2 Finance Officer

5.2.1 Original Genders

All features (Experience, Keywords, Education, Gender)

```
In [271]: df = pd.DataFrame({'Experience':X_0_fo,'Keywords':X_1_fo,'Education':X_2_fo,'Gender':X_3_fo,'Rank':Y_fo_o})
y, X = dmatrices('Rank ~ Experience + Keywords + Gender + Education', data=df,
return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```


OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.66
4
Model:                 OLS     Adj. R-squared:        0.21
7
Method:               Least Squares    F-statistic:          1.48
5
Date:                 Fri, 25 Sep 2020    Prob (F-statistic):   0.38
8
Time:                 11:50:23    Log-Likelihood:       -13.61
7
No. Observations:    8    AIC:                   37.2
3
Df Residuals:        3    BIC:                   37.6
3
Df Model:             4
Covariance Type:     nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept      8.0397      2.591      3.103     0.053     -0.205     16.28
4
Experience     -0.1782      0.133     -1.339     0.273     -0.602      0.24
5
Keywords       6.2902     39.083      0.161     0.882    -118.089     130.66
9
Gender        -3.4318      2.093     -1.640     0.200     -10.093      3.22
9
Education     -0.6503      3.739     -0.174     0.873     -12.550     11.25
0
=====

```

```

=====
=
Omnibus:          1.462    Durbin-Watson:        1.71
5
Prob(Omnibus):    0.481    Jarque-Bera (JB):     0.01
6
Skew:             -0.069    Prob(JB):             0.99
2
Kurtosis:         3.171    Cond. No.             64
2.
=====

```

=

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



Significant* features (Experience, Gender)

```
In [273]: df = pd.DataFrame({'Experience':X_0_fo, 'Keywords':X_1_fo, 'Education':X_2_fo, 'Gender':X_3_fo, 'Rank':Y_fo_o})
y, X = dmatrices('Rank ~ Experience + Gender', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:                0.66
0
Model:                  OLS    Adj. R-squared:         0.52
4
Method:                 Least Squares    F-statistic:           4.85
8
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):    0.067
3
Time:                   11:50:24    Log-Likelihood:       -13.66
6
No. Observations:      8    AIC:                   33.3
3
Df Residuals:          5    BIC:                   33.5
7
Df Model:               2
Covariance Type:       nonrobust
=====
=

```

	coef	std err	t	P> t	[0.025	0.97
Intercept	8.1234	1.330	6.108	0.002	4.704	11.54
Experience	-0.1879	0.081	-2.310	0.069	-0.397	0.02
Gender	-3.5946	1.285	-2.797	0.038	-6.898	-0.29

```
=====
=
Omnibus:                1.123    Durbin-Watson:         1.77
0
Prob(Omnibus):          0.570    Jarque-Bera (JB):     0.11
6
Skew:                   -0.292    Prob(JB):              0.94
4
Kurtosis:               2.925    Cond. No.              35.
9
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [274]: df = pd.DataFrame({'Experience':X_0_fo, 'Keywords':X_1_fo, 'Education':X_2_fo, 'Gender':X_3_fo, 'Rank':Y_fo_o})
y, X = dmatrices('Rank ~ Gender', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:          0.29
8
Model:                  OLS    Adj. R-squared:    0.18
1
Method:                 Least Squares    F-statistic:       2.54
2
Date:                   Fri, 25 Sep 2020    Prob (F-statistic): 0.16
2
Time:                   11:50:24    Log-Likelihood:    -16.57
1
No. Observations:      8    AIC:               37.1
4
Df Residuals:          6    BIC:               37.3
0
Df Model:               1
Covariance Type:       nonrobust
=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept          5.7500      1.109      5.186      0.002      3.037      8.46
3
Gender             -2.5000      1.568     -1.594      0.162     -6.337      1.33
7
=====
=
Omnibus:              1.531    Durbin-Watson:      1.66
1
Prob(Omnibus):        0.465    Jarque-Bera (JB):    0.67
8
Skew:                 0.000    Prob(JB):            0.71
2
Kurtosis:             1.573    Cond. No.             2.6
2
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

5.2.2 Flipped Genders

All features (Experience, Keywords, Education, Gender)

```
In [276]: df = pd.DataFrame({'Experience':X_0_fo,'Keywords':X_1_fo,'Education':X_2_fo,'Gender':X_3_fo,'Rank':Y_fo_f})
y, X = dmatrices('Rank ~ Experience + Keywords + Gender + Education', data=df,
return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.54
8
Model:                  OLS     Adj. R-squared:        -0.05
5
Method:                 Least Squares    F-statistic:          0.909
5
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):   0.55
4
Time:                   11:50:26    Log-Likelihood:       -14.80
8
No. Observations:      8     AIC:                   39.6
2
Df Residuals:          3     BIC:                   40.0
1
Df Model:               4
Covariance Type:      nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept      9.2344      3.006      3.072      0.054      -0.333     18.80
1
Experience     -0.0491      0.154     -0.318      0.771      -0.541      0.44
3
Keywords      -32.5231     45.354     -0.717      0.525     -176.858    111.81
2
Gender         -2.3153      2.429     -0.953      0.411     -10.045      5.41
5
Education     -0.7214      4.339     -0.166      0.879     -14.531     13.08
8
=====

```

```

=====
=
Omnibus:          1.173    Durbin-Watson:         3.04
4
Prob(Omnibus):   0.556    Jarque-Bera (JB):      0.82
4
Skew:             0.616    Prob(JB):              0.66
2
Kurtosis:        2.024    Cond. No.               64
2.
=====

```

=

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



Significant* features (Keywords, Gender)

```
In [278]: df = pd.DataFrame({'Experience':X_0_fo, 'Keywords':X_1_fo, 'Education':X_2_fo, 'Gender':X_3_fo, 'Rank':Y_fo_f})
y, X = dmatrices('Rank ~ Keywords + Gender', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:          0.50
8
Model:                  OLS    Adj. R-squared:    0.31
1
Method:                 Least Squares    F-statistic:       2.57
7
Date:                  Fri, 25 Sep 2020    Prob (F-statistic): 0.17
0
Time:                  11:50:27    Log-Likelihood:    -15.15
1
No. Observations:      8    AIC:               36.3
0
Df Residuals:          5    BIC:               36.5
4
Df Model:              2
Covariance Type:      nonrobust
=====
=

```

	coef	std err	t	P> t	[0.025	0.97
Intercept	8.8167	2.334	3.778	0.013	2.817	14.81
Keywords	-42.9333	29.408	-1.460	0.204	-118.529	32.66
Gender	-2.1167	1.462	-1.448	0.207	-5.875	1.64

```
=====
=
Omnibus:              0.205    Durbin-Watson:      2.97
1
Prob(Omnibus):        0.902    Jarque-Bera (JB):    0.36
6
Skew:                 0.060    Prob(JB):            0.83
3
Kurtosis:             1.959    Cond. No.            47.
0
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

5.2.3 Combined

All features (Experience, Keywords, Education, Gender)


```
In [280]: df = pd.DataFrame({'Experience':X_0_fo,'Keywords':X_1_fo,'Education':X_2_fo,'Gender':X_3_fo,'Rank':Y_fo_c})
y, X = dmatrices('Rank ~ Experience + Keywords + Education + Gender', data=df,
return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.75
5
Model:                  OLS    Adj. R-squared:        0.42
8
Method:                 Least Squares    F-statistic:          2.30
8
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):   0.25
9
Time:                   11:50:28    Log-Likelihood:       -12.36
3
No. Observations:      8    AIC:                   34.7
3
Df Residuals:          3    BIC:                   35.1
2
Df Model:               4
Covariance Type:       nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept      9.2637      2.215      4.183    0.025      2.216    16.31
2
Experience     -0.1519      0.114     -1.334    0.274     -0.514     0.21
0
Keywords      -29.0788     33.411     -0.870    0.448    -135.407    77.24
9
Education      0.9751      3.197      0.305    0.780     -9.198    11.14
8
Gender        -3.8687      1.789     -2.162    0.119     -9.563     1.82
6
=====

```

```

=====
=
Omnibus:           1.042    Durbin-Watson:         2.98
7
Prob(Omnibus):     0.594    Jarque-Bera (JB):      0.60
6
Skew:              0.142    Prob(JB):              0.73
9
Kurtosis:          1.682    Cond. No.               64
2.
=====
=

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Significant* features (Experience, Gender)

```
In [282]: df = pd.DataFrame({'Experience':X_0_fo, 'Keywords':X_1_fo, 'Education':X_2_fo, 'Gender':X_3_fo, 'Rank':Y_fo_c})
y, X = dmatrices('Rank ~ Experience + Gender', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:          0.69
0
Model:                  OLS    Adj. R-squared:    0.56
6
Method:                 Least Squares    F-statistic:       5.57
2
Date:                  Fri, 25 Sep 2020    Prob (F-statistic): 0.053
4
Time:                  11:50:28    Log-Likelihood:    -13.29
6
No. Observations:      8    AIC:              32.5
9
Df Residuals:          5    BIC:              32.8
3
Df Model:              2
Covariance Type:      nonrobust
=====
=

```

	coef	std err	t	P> t	[0.025	0.97
Intercept	8.0163	1.270	6.313	0.001	4.752	11.28
Experience	-0.1596	0.078	-2.056	0.095	-0.359	0.04
Gender	-3.9299	1.227	-3.203	0.024	-7.084	-0.77

```
=====
=
Omnibus:              1.540    Durbin-Watson:      2.67
0
Prob(Omnibus):        0.463    Jarque-Bera (JB):    0.31
4
Skew:                 0.484    Prob(JB):            0.85
5
Kurtosis:             2.925    Cond. No.            35.
9
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [283]: df = pd.DataFrame({'Experience':X_0_fo, 'Keywords':X_1_fo, 'Education':X_2_fo, 'Gender':X_3_fo, 'Rank':Y_fo_c})
y, X = dmatrices('Rank ~ Gender', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:                0.42
9
Model:                  OLS    Adj. R-squared:         0.33
3
Method:                 Least Squares    F-statistic:           4.50
0
Date:                  Fri, 25 Sep 2020    Prob (F-statistic):    0.078
1
Time:                  11:50:29    Log-Likelihood:       -15.74
6
No. Observations:      8    AIC:                   35.4
9
Df Residuals:          6    BIC:                   35.6
5
Df Model:               1
Covariance Type:       nonrobust
=====
=

```

	coef	std err	t	P> t	[0.025	0.97
Intercept	6.0000	1.000	6.000	0.001	3.553	8.44
Gender	-3.0000	1.414	-2.121	0.078	-6.460	0.46

```
-----
-
Omnibus:                0.992    Durbin-Watson:         3.04
2
Prob(Omnibus):          0.609    Jarque-Bera (JB):      0.70
4
Skew:                   0.433    Prob(JB):              0.70
3
Kurtosis:               1.833    Cond. No.               2.6
2
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

5.3 Recruitment Officer

5.3.1 Original Genders

All features (Experience, Keywords, Education, Gender)

```
In [285]: df = pd.DataFrame({'Experience':X_0_ro,'Keywords':X_1_ro,'Education':X_2_ro,'Gender':X_3_ro,'Rank':Y_ro_o})
y, X = dmatrices('Rank ~ Experience + Keywords + Gender + Education', data=df,
return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.93
9
Model:                 OLS     Adj. R-squared:         0.85
7
Method:               Least Squares    F-statistic:           11.4
7
Date:                 Fri, 25 Sep 2020    Prob (F-statistic):    0.036
6
Time:                 11:50:30          Log-Likelihood:        -6.821
1
No. Observations:     8      AIC:                   23.6
4
Df Residuals:         3      BIC:                   24.0
4
Df Model:              4
Covariance Type:      nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept      9.7053      1.388      6.993      0.006      5.289     14.12
2
Experience      0.0098      0.079      0.124      0.909     -0.241      0.26
1
Keywords     -38.3517      9.455     -4.056      0.027     -68.443     -8.26
0
Gender         1.2411      0.960      1.293      0.287     -1.813      4.29
5
Education     -0.6457      0.946     -0.683      0.544     -3.655      2.36
4
=====

```

```

=====
=
Omnibus:          1.155    Durbin-Watson:         1.78
1
Prob(Omnibus):    0.561    Jarque-Bera (JB):     0.30
3
Skew:             0.462    Prob(JB):              0.85
9
Kurtosis:         2.759    Cond. No.              39
0.
=====
=

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



Significant features (Keywords, Gender)

```
In [287]: df = pd.DataFrame({'Experience':X_0_ro, 'Keywords':X_1_ro, 'Education':X_2_ro, 'Gender':X_3_ro, 'Rank':Y_ro_o})
y, X = dmatrices('Rank ~ Keywords + Gender', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:          0.92
9
Model:                  OLS    Adj. R-squared:    0.90
1
Method:                 Least Squares    F-statistic:      32.7
5
Date:                  Fri, 25 Sep 2020    Prob (F-statistic): 0.0013
4
Time:                  11:50:31    Log-Likelihood:   -7.399
7
No. Observations:      8    AIC:              20.8
0
Df Residuals:          5    BIC:              21.0
4
Df Model:               2
Covariance Type:       nonrobust
=====
```

```
=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept    10.1064    0.994    10.172    0.000    7.552    12.66
0
Keywords     -42.5106    5.891    -7.216    0.001   -57.654   -27.36
7
Gender        1.4255    0.552     2.585    0.049     0.008     2.84
3
=====
```

```
=====
=
Omnibus:          1.746    Durbin-Watson:    1.78
0
Prob(Omnibus):    0.418    Jarque-Bera (JB): 0.86
1
Skew:             0.759    Prob(JB):         0.65
0
Kurtosis:         2.472    Cond. No.         25.
2
=====
```

```
=====
=
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```



5 3 2 Flipped Genders

All features (Experience, Keywords, Education, Gender)

```
In [289]: df = pd.DataFrame({'Experience':X_0_ro,'Keywords':X_1_ro,'Education':X_2_ro,'Gender':X_3_ro,'Rank':Y_ro_f})
y, X = dmatrices('Rank ~ Experience + Keywords + Gender + Education', data=df,
return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.56
3
Model:                 OLS     Adj. R-squared:        -0.01
9
Method:                Least Squares    F-statistic:          0.968
1
Date:                  Fri, 25 Sep 2020    Prob (F-statistic):   0.53
2
Time:                  11:50:32    Log-Likelihood:       -14.66
9
No. Observations:      8     AIC:                   39.3
4
Df Residuals:          3     BIC:                   39.7
3
Df Model:              4
Covariance Type:      nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept      9.8680      3.701      2.666      0.076      -1.911     21.64
7
Experience     -0.0074      0.210     -0.035      0.974      -0.677      0.66
2
Keywords      -27.6381     25.217     -1.096      0.353     -107.891     52.61
5
Gender         -1.5702      2.560     -0.613      0.583      -9.716      6.57
6
Education     -1.0277      2.522     -0.407      0.711      -9.054      6.99
9
=====

```

```

=====
=
Omnibus:          0.853    Durbin-Watson:        0.79
9
Prob(Omnibus):    0.653    Jarque-Bera (JB):     0.64
2
Skew:            0.383    Prob(JB):             0.72
5
Kurtosis:        1.843    Cond. No.              39
0.
=====

```

=

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



Significant features (Keywords)

```
In [291]: df = pd.DataFrame({'Experience':X_0_ro,'Keywords':X_1_ro,'Education':X_2_ro,'G
ender':X_3_ro,'Rank':Y_ro_f})
y, X = dmatrices('Rank ~ Keywords', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:          0.43
7
Model:                OLS    Adj. R-squared:    0.34
4
Method:              Least Squares    F-statistic:      4.66
7
Date:                Fri, 25 Sep 2020    Prob (F-statistic): 0.074
1
Time:                11:50:33    Log-Likelihood:   -15.68
3
No. Observations:    8    AIC:              35.3
7
Df Residuals:        6    BIC:              35.5
2
Df Model:             1
Covariance Type:     nonrobust
=====
=

```

	coef	std err	t	P> t	[0.025	0.97
Intercept	9.3125	2.336	3.987	0.007	3.597	15.02
Keywords	-32.3750	14.987	-2.160	0.074	-69.046	4.29

```
-----
=
Omnibus:              0.914    Durbin-Watson:    0.98
7
Prob(Omnibus):        0.633    Jarque-Bera (JB): 0.70
2
Skew:                 0.522    Prob(JB):         0.70
4
Kurtosis:             1.992    Cond. No.         21.
8
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

5.3.3 Combined

All features (Experience, Keywords, Education, Gender)

```
In [293]: df = pd.DataFrame({'Experience':X_0_ro,'Keywords':X_1_ro,'Education':X_2_ro,'Gender':X_3_ro,'Rank':Y_ro_c})
y, X = dmatrices('Rank ~ Experience + Keywords + Gender + Education', data=df,
return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```

=====
=
Dep. Variable:          Rank    R-squared:              0.88
6
Model:                  OLS    Adj. R-squared:        0.73
5
Method:                 Least Squares    F-statistic:           5.84
2
Date:                   Fri, 25 Sep 2020    Prob (F-statistic):    0.089
4
Time:                   11:50:34    Log-Likelihood:        -9.290
5
No. Observations:      8    AIC:                   28.5
8
Df Residuals:          3    BIC:                   28.9
8
Df Model:               4
Covariance Type:       nonrobust
=====

```

```

=====
=
              coef    std err          t      P>|t|      [0.025    0.97
5]
-----
-
Intercept      8.3271      1.890      4.407      0.022      2.313     14.34
1
Experience      0.1643      0.107      1.530      0.223     -0.177      0.50
6
Keywords     -33.1979     12.875     -2.579      0.082     -74.171      7.77
5
Gender         -0.6411      1.307     -0.491      0.657     -4.800      3.51
8
Education     -1.4454      1.288     -1.123      0.343     -5.543      2.65
2
=====

```

```

=====
=
Omnibus:           2.550    Durbin-Watson:         0.80
0
Prob(Omnibus):     0.279    Jarque-Bera (JB):      0.88
8
Skew:              0.257    Prob(JB):               0.64
2
Kurtosis:          1.451    Cond. No.               39
0.
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Significant features (Experience, Keywords)

```
In [295]: df = pd.DataFrame({'Experience':X_0_ro, 'Keywords':X_1_ro, 'Education':X_2_ro, 'Gender':X_3_ro, 'Rank':Y_ro_c})
y, X = dmatrices('Rank ~ Experience + Keywords', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:          0.83
7
Model:                  OLS    Adj. R-squared:    0.77
1
Method:                 Least Squares    F-statistic:       12.8
1
Date:                   Fri, 25 Sep 2020    Prob (F-statistic): 0.010
8
Time:                   11:50:35    Log-Likelihood:    -10.73
4
No. Observations:      8    AIC:               27.4
7
Df Residuals:          5    BIC:               27.7
1
Df Model:               2
Covariance Type:       nonrobust
=====
=

```

	coef	std err	t	P> t	[0.025	0.97
Intercept	9.1864	1.596	5.755	0.002	5.083	13.28
Experience	0.1313	0.070	1.865	0.121	-0.050	0.31
Keywords	-42.1138	8.848	-4.760	0.005	-64.858	-19.36

```
=====
=
Omnibus:                2.836    Durbin-Watson:      0.79
3
Prob(Omnibus):          0.242    Jarque-Bera (JB):   1.33
8
Skew:                   0.975    Prob(JB):            0.51
2
Kurtosis:                2.538    Cond. No.            28
9.
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Significant features (Keywords)

```
In [296]: y, X = dmatrices('Rank ~ Keywords', data=df, return_type='dataframe')
mod = sm.OLS(y,X)
res = mod.fit()
print(res.summary())
```

OLS Regression Results

```
=====
=
Dep. Variable:          Rank    R-squared:          0.72
3
Model:                  OLS    Adj. R-squared:    0.67
7
Method:                 Least Squares    F-statistic:       15.6
8
Date:                   Fri, 25 Sep 2020    Prob (F-statistic): 0.0074
6
Time:                   11:50:36    Log-Likelihood:    -12.84
6
No. Observations:      8    AIC:              29.6
9
Df Residuals:          6    BIC:              29.8
5
Df Model:               1
Covariance Type:       nonrobust
=====
=

```

	coef	std err	t	P> t	[0.025	0.97
Intercept	10.6875	1.638	6.523	0.001	6.679	14.69
Keywords	-41.6250	10.513	-3.959	0.007	-67.349	-15.90

```
-----
-
Omnibus:                0.664    Durbin-Watson:      0.86
6
Prob(Omnibus):          0.717    Jarque-Bera (JB):   0.20
3
Skew:                   -0.342    Prob(JB):           0.90
3
Kurtosis:               2.625    Cond. No.           21.
8
=====
=
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

5.4 Train / Test Experiment

For some internal testing, we did some k-folds testing of the regression models.

```
In [298]: def prediction_to_rank(predictions):
ranks = []
temp = sorted(predictions)

for resume in predictions:
    index = temp.index(resume)
    ranks.append(index+1)

return ranks
```

```
In [297]: def repeated_regression_test(model, df, num_repeats):
count = 0
mean_total = 0
r2_total = 0
rkf = RepeatedKFold(n_splits=2, n_repeats=num_repeats, random_state=265212
4)
for train_index, test_index in rkf.split(X_da):
    train_df = df.iloc[train_index]
    test_df = df.iloc[test_index]
    y, X = dmatrices(model, data=train_df, return_type='dataframe')

    mod = sm.OLS(y,X)
    res = mod.fit()

    y, X = dmatrices(model, data=test_df, return_type='dataframe')
    prediction = res.predict(X)

    count += 1
    mean_total += mean_squared_error(prediction_to_rank(y.to_numpy()), pre
diction_to_rank(prediction))
    r2_total += r2_score(prediction_to_rank(y.to_numpy()), prediction_to_r
ank(prediction))

print('Average Mean squared error: %.2f'
      % (mean_total/count))

# The coefficient of determination: 1 is perfect prediction
print('Average Coefficient of determination (R^2): %.2f'
      % (r2_total/count))
```

We were a bit curious about the R^2 results, and wanted to check them with our own R^2 function.

```
In [98]: def r2_score_a(observed, predicted):
         mean = np.mean(observed)
         sum_sq_total = 0

         for y in observed:
             sum_sq_total += (y - mean)**2

         sum_sq_res = 0
         for y, f in zip(observed, predicted):
             sum_sq_res += (y - f)**2

         return (1 - (sum_sq_res/sum_sq_total))
```

5.4.1 Data Analyst

5.4.1.1 Original Gender

```
In [126]: df = pd.DataFrame({'Experience':X_0_da, 'Keywords':X_1_da, 'Gender':X_3_da, 'Rank':Y_da_o})
```

```
In [127]: repeated_regression_test('Rank ~ Experience + Keywords + Gender', df, 500)
```

Average Mean squared error: 1.05
Average Coefficient of determination (R²): 0.16

```
In [128]: repeated_regression_test('Rank ~ Keywords + Gender', df, 500)
```

Average Mean squared error: 0.36
Average Coefficient of determination (R²): 0.71

```
In [129]: repeated_regression_test('Rank ~ Keywords', df, 500)
```

Average Mean squared error: 0.16
Average Coefficient of determination (R²): 0.87

5.4.1.2 Flipped Gender

```
In [130]: df = pd.DataFrame({'Experience':X_0_da, 'Keywords':X_1_da, 'Gender':X_3_da, 'Rank':Y_da_f})
```

```
In [131]: repeated_regression_test('Rank ~ Experience + Keywords + Gender', df, 500)
```

Average Mean squared error: 1.41
Average Coefficient of determination (R²): -0.13

```
In [132]: repeated_regression_test('Rank ~ Experience + Gender', df, 500)
```

Average Mean squared error: 0.88
Average Coefficient of determination (R²): 0.30

```
In [133]: repeated_regression_test('Rank ~ Experience', df, 500)
```

Average Mean squared error: 1.65
Average Coefficient of determination (R²): -0.32

5.4.1.3 Combined Dataset

```
In [134]: df = pd.DataFrame({'Experience':X_0_da,'Keywords':X_1_da,'Gender':X_3_da,'Rank':Y_da_c})
```

```
In [135]: repeated_regression_test('Rank ~ Experience + Keywords + Gender', df, 500)
```

Average Mean squared error: 2.00
Average Coefficient of determination (R²): -0.60

```
In [136]: repeated_regression_test('Rank ~ Experience', df, 500)
```

Average Mean squared error: 1.19
Average Coefficient of determination (R²): 0.05

5.4.2 Finance Officer

5.4.2.1 Original Gender

```
In [137]: df = pd.DataFrame({'Experience':X_0_fo,'Keywords':X_1_fo,'Education':X_2_fo,'Gender':X_3_fo,'Rank':Y_fo_o})
```

```
In [138]: repeated_regression_test('Rank ~ Experience + Keywords + Gender + Education', df, 500)
```

Average Mean squared error: 2.35
Average Coefficient of determination (R²): -0.88

```
In [139]: repeated_regression_test('Rank ~ Experience + Gender', df, 500)
```

Average Mean squared error: 1.37
Average Coefficient of determination (R²): -0.09

5.4.2.2 Flipped Gender

```
In [140]: df = pd.DataFrame({'Experience':X_0_fo, 'Keywords':X_1_fo, 'Education':X_2_fo, 'Gender':X_3_fo, 'Rank':Y_fo_f})
```

```
In [141]: repeated_regression_test('Rank ~ Experience + Keywords + Gender + Education', df, 500)
```

Average Mean squared error: 2.60
Average Coefficient of determination (R²): -1.08

```
In [142]: repeated_regression_test('Rank ~ Keywords + Gender', df, 500)
```

Average Mean squared error: 1.71
Average Coefficient of determination (R²): -0.37

5.4.2.3 Combined Dataset

```
In [143]: df = pd.DataFrame({'Experience':X_0_fo, 'Keywords':X_1_fo, 'Education':X_2_fo, 'Gender':X_3_fo, 'Rank':Y_fo_c})
```

```
In [144]: repeated_regression_test('Rank ~ Experience + Keywords + Gender + Education', df, 500)
```

Average Mean squared error: 2.50
Average Coefficient of determination (R²): -1.00

```
In [145]: repeated_regression_test('Rank ~ Experience + Gender', df, 500)
```

Average Mean squared error: 1.28
Average Coefficient of determination (R²): -0.03

5.4.3 Recruitment Officer

5.4.3.1 Original Gender

```
In [146]: df = pd.DataFrame({'Experience':X_0_ro, 'Keywords':X_1_ro, 'Education':X_2_ro, 'Gender':X_3_ro, 'Rank':Y_ro_o})
```

```
In [147]: repeated_regression_test('Rank ~ Experience + Keywords + Gender + Education', df, 500)
```

Average Mean squared error: 1.78
Average Coefficient of determination (R²): -0.43

```
In [148]: repeated_regression_test('Rank ~ Keywords + Gender', df, 500)
```

Average Mean squared error: 0.35
Average Coefficient of determination (R²): 0.72

5.4.3.2 Flipped Gender

```
In [149]: df = pd.DataFrame({'Experience':X_0_ro,'Keywords':X_1_ro,'Education':X_2_ro,'Gender':X_3_ro,'Rank':Y_ro_f})
```

```
In [150]: repeated_regression_test('Rank ~ Experience + Keywords + Gender + Education',  
df, 500)
```

Average Mean squared error: 2.80
Average Coefficient of determination (R²): -1.24

```
In [151]: repeated_regression_test('Rank ~ Keywords', df, 500)
```

Average Mean squared error: 0.97
Average Coefficient of determination (R²): 0.22

5.4.3.3 Combined Dataset

```
In [152]: df = pd.DataFrame({'Experience':X_0_ro,'Keywords':X_1_ro,'Education':X_2_ro,'Gender':X_3_ro,'Rank':Y_ro_c})
```

```
In [154]: repeated_regression_test('Rank ~ Experience + Keywords + Gender + Education',  
df, 500)
```

Average Mean squared error: 2.01
Average Coefficient of determination (R²): -0.61

```
In [155]: repeated_regression_test('Rank ~ Keywords', df, 500)
```

Average Mean squared error: 0.49
Average Coefficient of determination (R²): 0.61

6 Clustering

One of the things we tried to do, was to see whether an algorithm could find patterns/groups of the top candidates based on the feature inputs without human input on what a good candidate is.

We attempted this by trying to use clustering to identify groups of candidates based on the features, and see whether these groups of candidates have similar rankings from the human panelists. We visualise this with a scatter plot, where the colour of the points represent the difference groups and the numbers are the rankings from the human panelists.

```
In [156]: def cluster_scatter(X, Y, num_clusters):
kmeans = KMeans(n_clusters=num_clusters)
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
plt.xlabel("Weighted Relevant Experience")
plt.ylabel("Requirement Keyword Match")

for i, txt in enumerate(Y):
    plt.annotate(txt, (X[i, 0], X[i, 1]),
                 textcoords="offset points", xytext=(0, 10))
```

We have also included a Voronoi graph, which shows the boundaries of the different groups calculated by the K-mean clustering algorithm. I.e. any point that lands in the blue area will be assigned to the blue group by the algorithm when predicting.

```
In [157]: def cluster_voronoi(X,Y,num_clusters,xlim,ylim):
kmeans = KMeans(n_clusters=num_clusters)
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
kmeans.fit(X)

# add 4 distant dummy points for the polygon colouring
kmeans.cluster_centers_ = np.append(kmeans.cluster_centers_, [[999,999], [-999,999], [999,-999], [-999,-999]], axis = 0)
vor = Voronoi(kmeans.cluster_centers_)

# plot
voronoi_plot_2d(vor, show_vertices = False)

## colorize
for region in vor.regions:
    if not -1 in region:
        polygon = [vor.vertices[i] for i in region]
        plt.fill(*zip(*polygon))

# fix the range of axes
plt.xlim(xlim), plt.ylim(ylim)

plt.xlabel("Weighted Relevant Experience")
plt.ylabel("Requirement Keyword Match")

for i, txt in enumerate(Y):
    plt.annotate(txt, (X[i,0], X[i,1]))

plt.show()
```

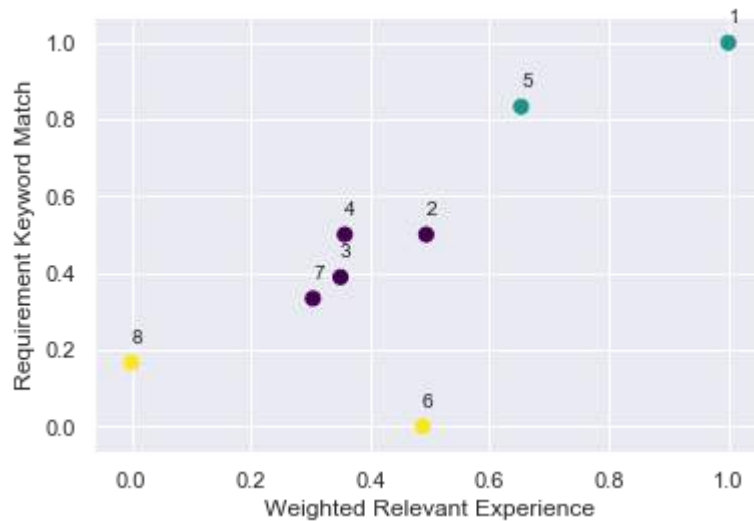
6.1 Data Analyst

We were mainly curious about how well the experience and keyword features captured the relationships in the data, so we reduced the dataset to just these two features, so it was easier to visualise.

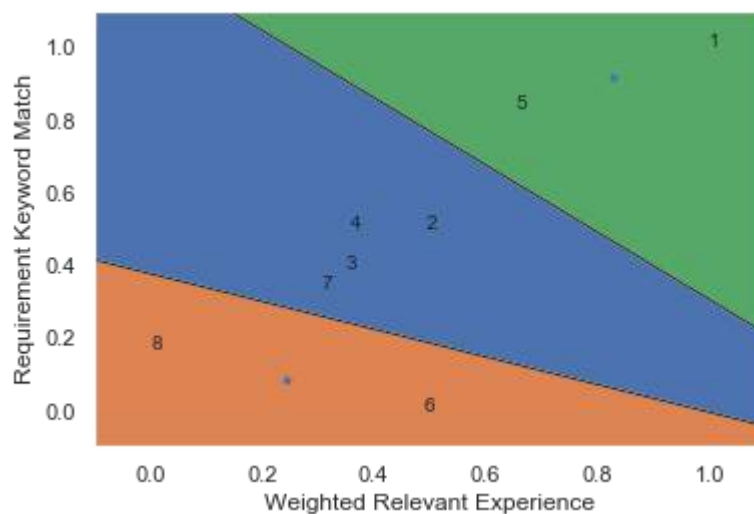
```
In [158]: X_da_cluster = np.column_stack((X_0_da,X_1_da))
```

6.1.1 Combined Results

```
In [159]: cluster_scatter(X_da_cluster,Y_da_c,3)
```

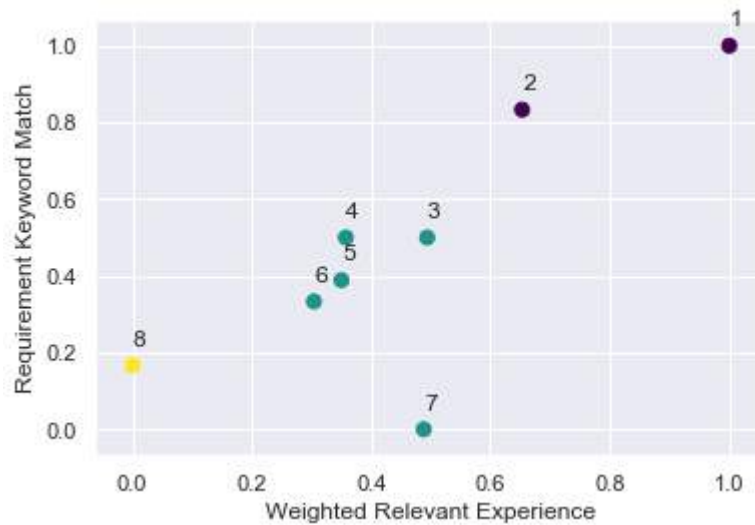


```
In [160]: cluster_voronoi(X_da_cluster,Y_da_c,3,[-0.1,1.1],[-0.1,1.1])
```



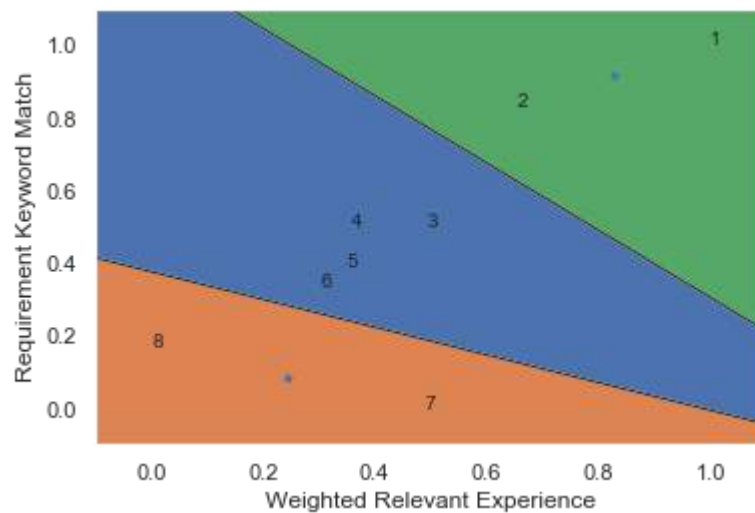
6.1.2 Original Genders


```
In [299]: cluster_scatter(X_da_cluster,Y_da_o,3)
```



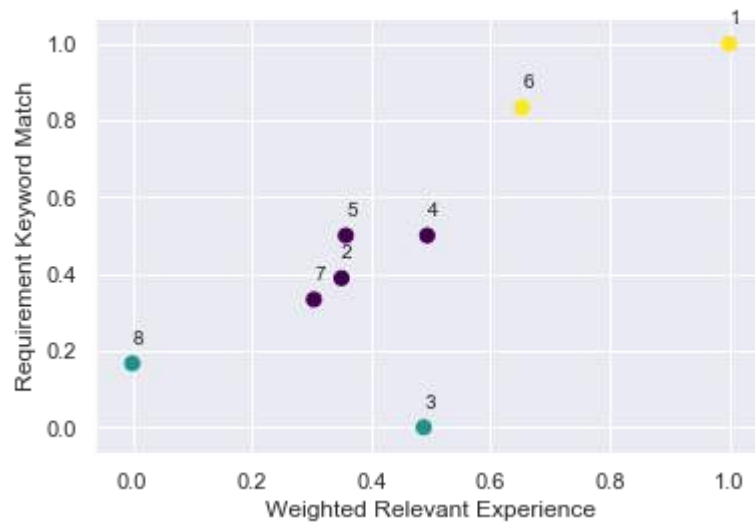
This clustering result is nearly ideal, where the purple group could represent the top candidates, and the yellow group represents the worst candidates. Of course, this is still dependent on what the human panelists view as a top candidate.

```
In [162]: cluster_voronoi(X_da_cluster,Y_da_o,3,[-0.1,1.1],[-0.1,1.1])
```

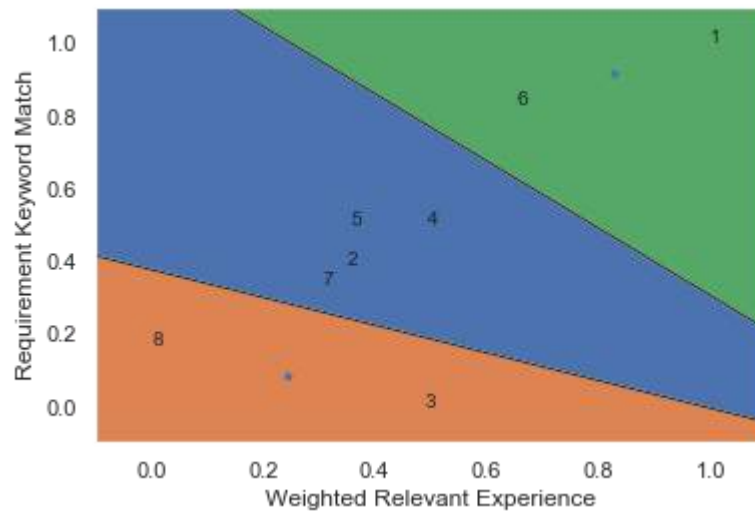


6.1.3 Flipped Genders

```
In [163]: cluster_scatter(X_da_cluster,Y_da_f,3)
```



```
In [164]: cluster_voronoi(X_da_cluster,Y_da_f,3,[-0.1,1.1],[-0.1,1.1])
```

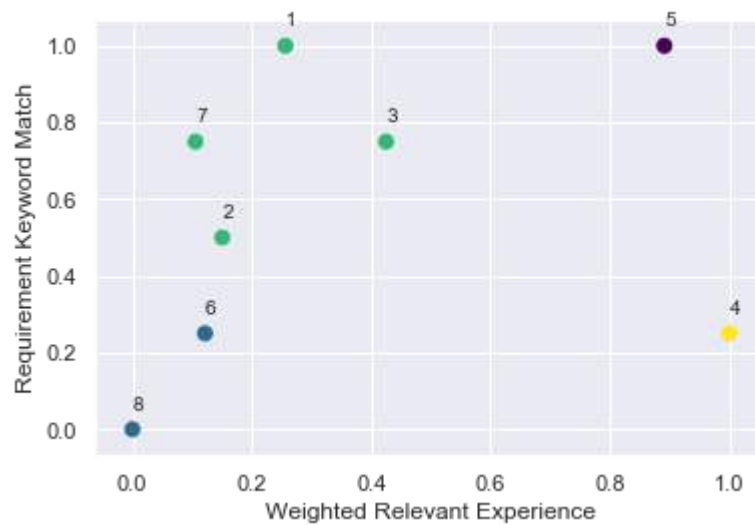


6.4 Finance Officer

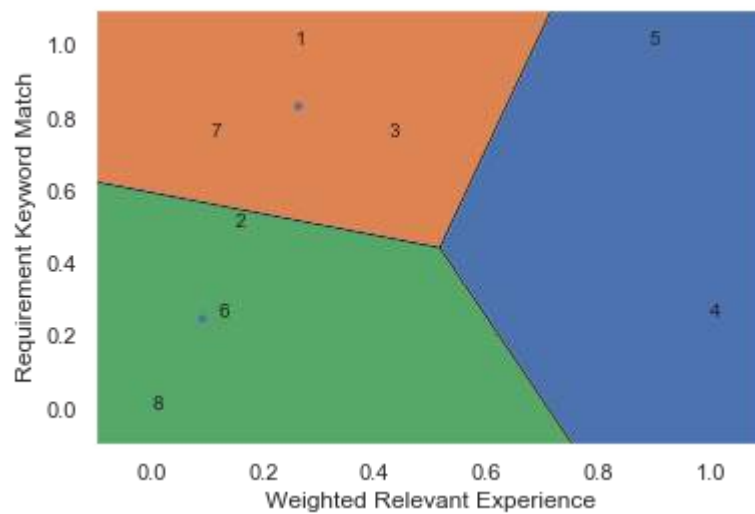
```
In [165]: X_fo_cluster = np.column_stack((X_0_fo,X_1_fo))
```

6.4.1 Combined Results

```
In [166]: cluster_scatter(X_fo_cluster,Y_fo_c,4)
```

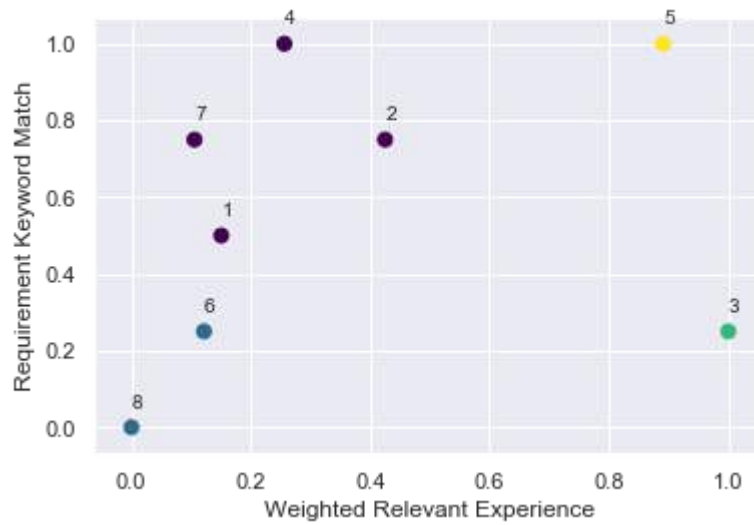


```
In [167]: cluster_voronoi(X_fo_cluster,Y_fo_c,3,[-0.1,1.1],[-0.1,1.1])
```

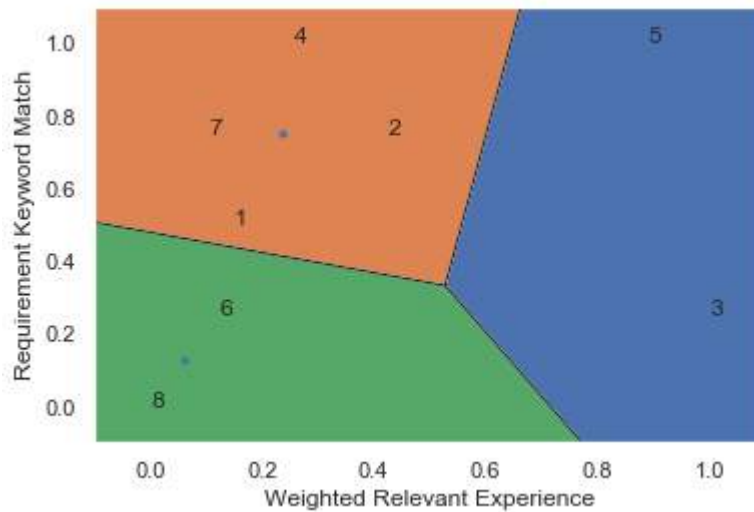


6.4.2 Original Genders

```
In [168]: cluster_scatter(X_fo_cluster,Y_fo_o,4)
```

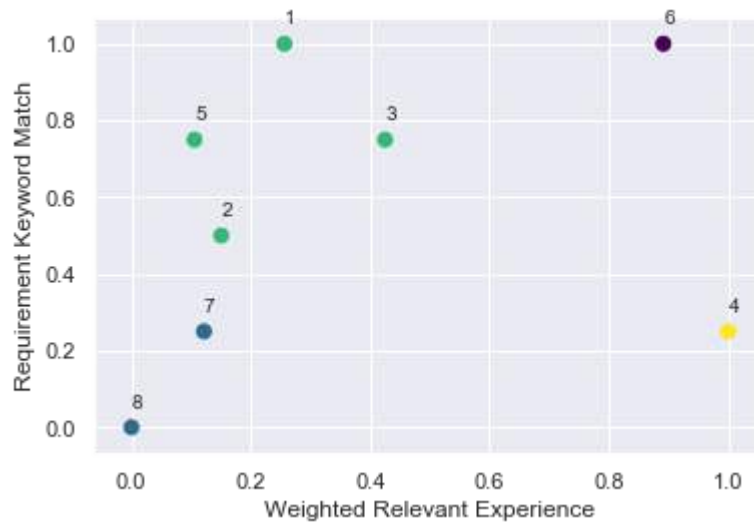


```
In [304]: cluster_voronoi(X_fo_cluster,Y_fo_o,3,[-0.1,1.1],[-0.1,1.1])
```

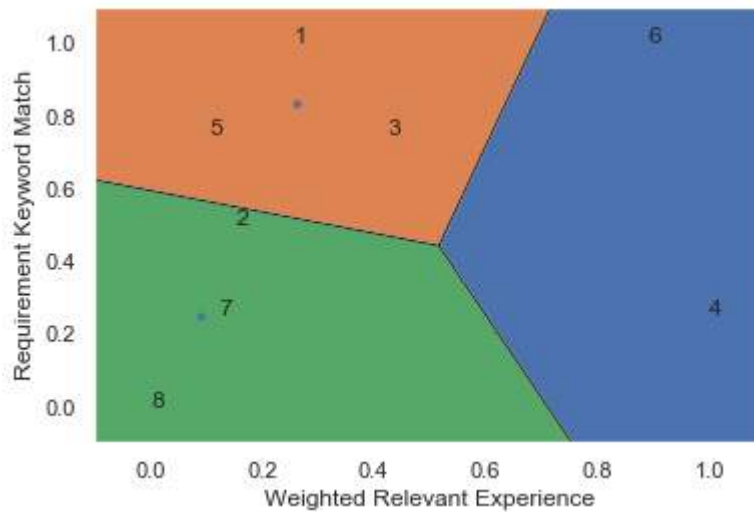


6.4.3 Flipped Genders

```
In [170]: cluster_scatter(X_fo_cluster,Y_fo_f,4)
```



```
In [303]: cluster_voronoi(X_fo_cluster,Y_fo_f,3,[-0.1,1.1],[-0.1,1.1])
```

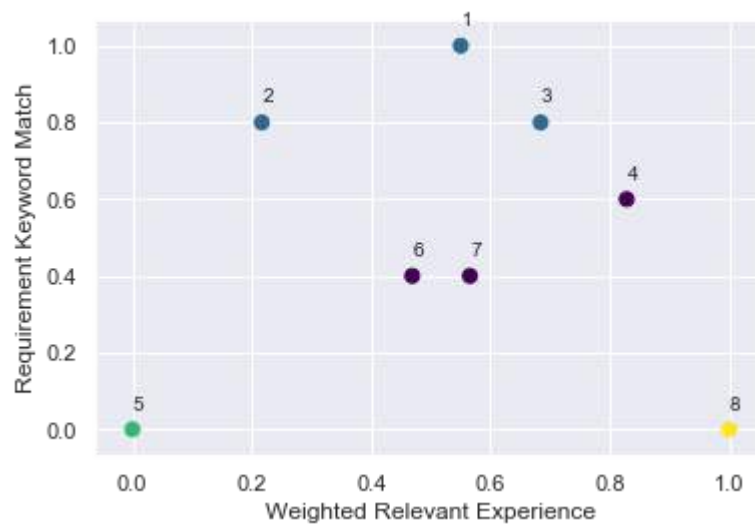


6.5 Recruitment Officer

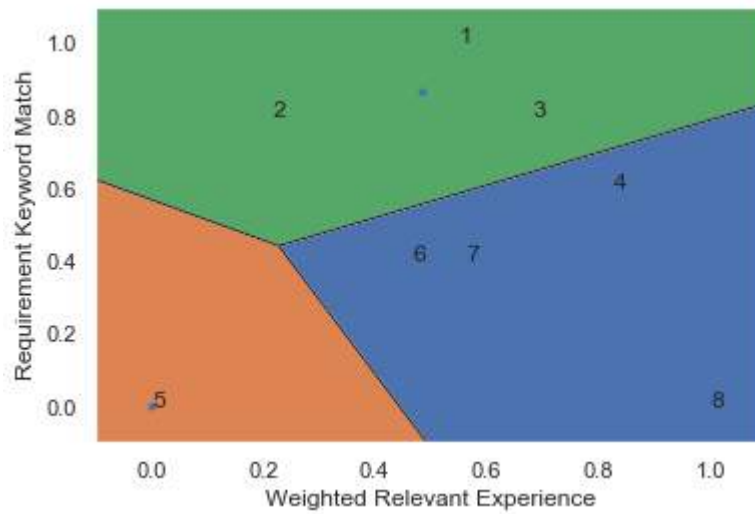
```
In [172]: X_ro_cluster = np.column_stack((X_0_ro,X_1_ro))
```

6.5.1 Combined Results

```
In [173]: cluster_scatter(X_ro_cluster,Y_ro_c,4)
```

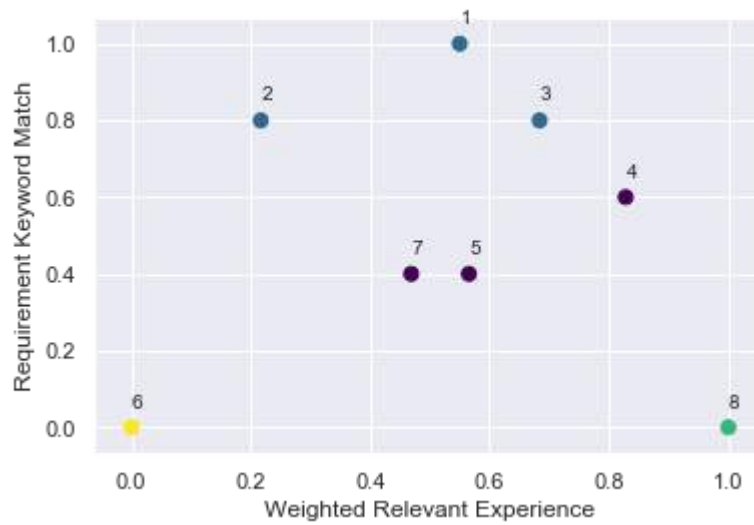


```
In [302]: cluster_voronoi(X_ro_cluster,Y_ro_c,3,[-0.1,1.1],[-0.1,1.1])
```

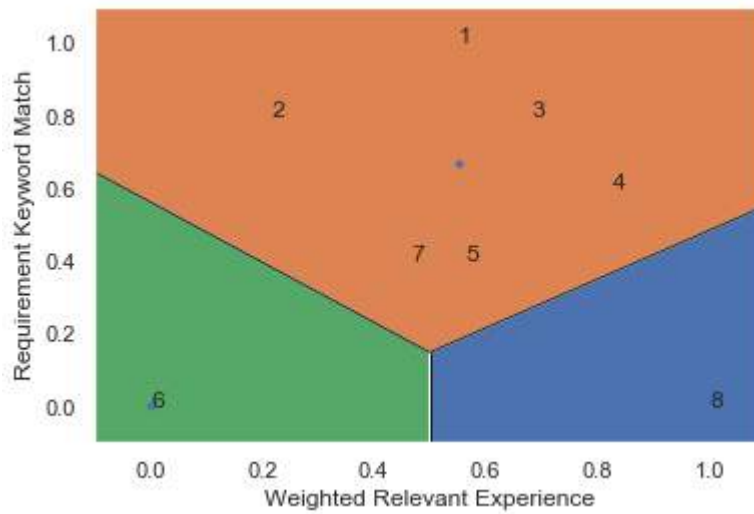


6.5.2 Original Genders

```
In [175]: cluster_scatter(X_ro_cluster,Y_ro_o,4)
```

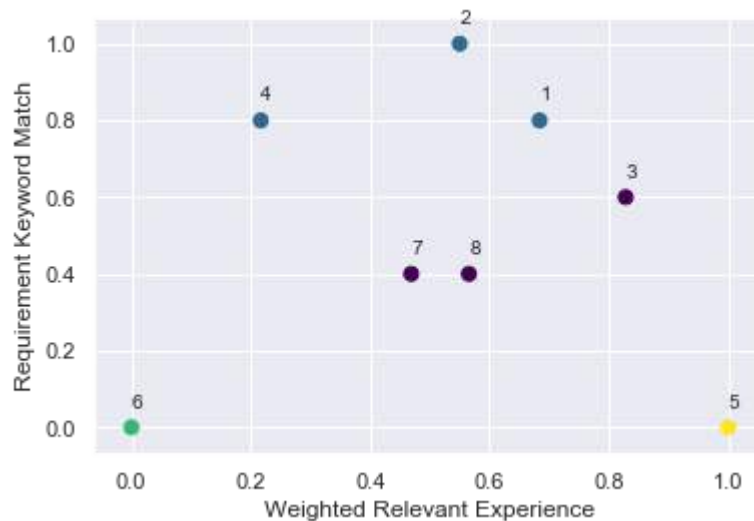


```
In [301]: cluster_voronoi(X_ro_cluster,Y_ro_o,3,[-0.1,1.1],[-0.1,1.1])
```

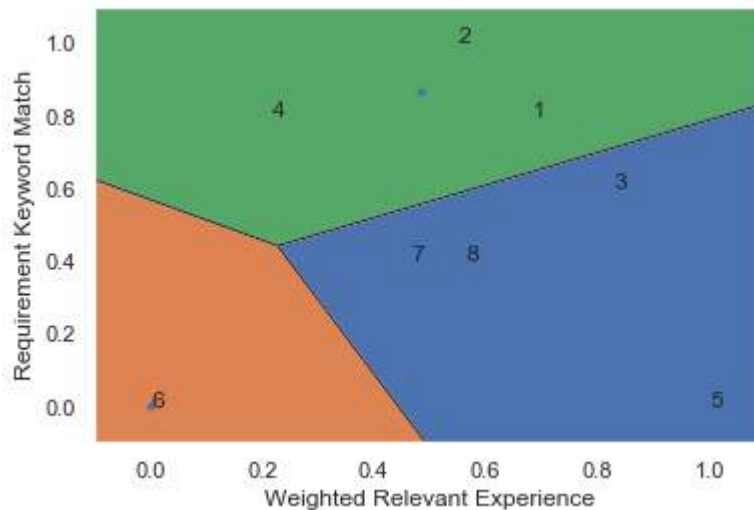


6.5.3 Flipped Genders

```
In [177]: cluster_scatter(X_ro_cluster,Y_ro_f,4)
```



```
In [300]: cluster_voronoi(X_ro_cluster,Y_ro_f,3,[-0.1,1.1],[-0.1,1.1])
```



7 Statistical Analysis of the rankings

For internal testing of the results, we computed a few statistics to compare against the results and interpretations. The testing of significant features was moved to Section 5.

7.1 Rank Boxplots

To confirm whether the rank aggregation methods make sense and work as expected, we created boxplots of the ranking data of the candidates and compared them to the aggregated rankings.

7.1.1 Data Analyst

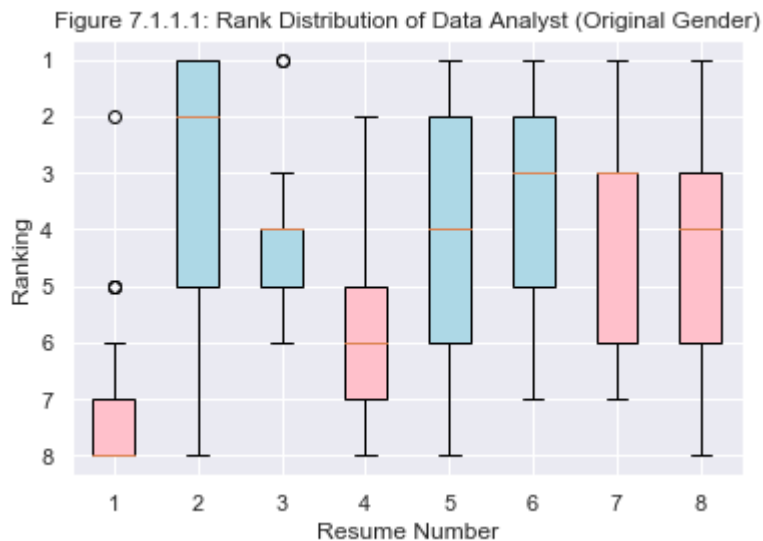
```
In [306]: bplot = plt.boxplot(da_ranks_o, patch_artist=True)
ax = plt.gca()
ax.set_ylim(ax.get_ylim()[::-1]) #flips the y-axis values
plt.xlabel("Resume Number")
plt.ylabel("Ranking")
plt.title("Figure 7.1.1.1: Rank Distribution of Data Analyst (Original Gender)")

print("Mean Aggregation Method:",y_da_o)
print("Ordered Pair Distance Aggregation Method:",Y_da_o)

# fill with colors to match gender
colors = ['pink', 'lightblue', 'lightblue','pink','lightblue','lightblue','pink', 'pink']

for patch, color in zip(bplot['boxes'], colors):
    patch.set_facecolor(color)
```

Mean Aggregation Method: [8, 1, 6, 7, 5, 2, 3, 4]
 Mean Aggregation Method: [8, 1, 5, 6, 7, 4, 2, 3]



```
In [309]: bplot = plt.boxplot(da_ranks_f, patch_artist=True)
ax = plt.gca()
ax.set_ylim(ax.get_ylim()[::-1]) #flips the y-axis values
plt.title("Figure 7.1.1.2: Rank Distribution of Data Analyst (Flipped Gender)"
)

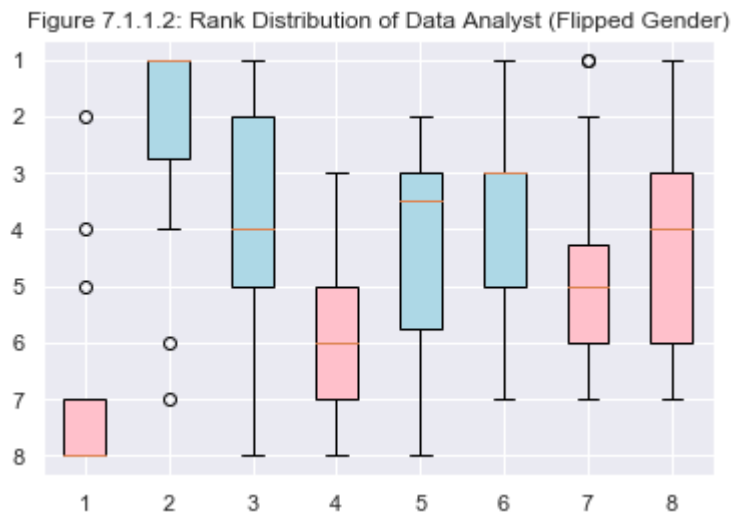
print("Mean Aggregation Method:",y_da_f)
print("Ordered Pair Distance Aggregation Method:",Y_da_f)

# fill with colors
colors = ['pink', 'lightblue', 'lightblue','pink','lightblue','lightblue','pink','pink']

for patch, color in zip(bplot['boxes'], colors):
    patch.set_facecolor(color)
```

Mean Aggregation Method: [8, 1, 2, 7, 5, 3, 6, 4]

Ordered Pair Distance Aggregation Method: [8, 1, 2, 7, 3, 5, 6, 4]



```
In [310]: bplot = plt.boxplot(da_ranks_c, patch_artist=True)
ax = plt.gca()
ax.set_ylim(ax.get_ylim()[::-1]) #flips the y-axis values

plt.xlabel("Resume Number")
plt.ylabel("Ranking")

plt.title("Figure 7.1.1.3: Rank Distribution of Data Analyst (Combined)")

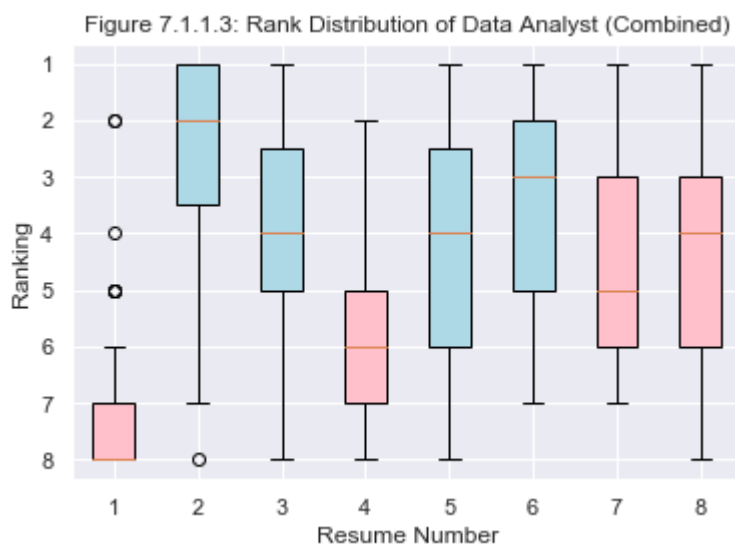
print("Mean Aggregation Method:",y_da_c)
print("Ordered Pair Distance Aggregation Method:",Y_da_c)

# fill with colors
colors = ['pink', 'lightblue', 'lightblue','pink','lightblue','lightblue','pink',
'pink']

for patch, color in zip(bplot['boxes'], colors):
    patch.set_facecolor(color)
```

Mean Aggregation Method: [8, 1, 3, 7, 5, 2, 6, 4]

Ordered Pair Distance Aggregation Method: [8, 1, 3, 7, 6, 4, 5, 2]



7.1.2 Finance Officer

Because the Data Analyst boxplot results met our expectations, we only did a single boxplot for Finance Officer and Recruitment Officer.

```
In [312]: bplot = plt.boxplot(fo_ranks_c, patch_artist=True)
ax = plt.gca()
ax.set_ylim(ax.get_ylim()[::-1]) #flips the y-axis values

plt.xlabel("Resume Number")
plt.ylabel("Ranking")
plt.title("Figure 7.1.2.1: Rank Distribution of Finance Officer (Combined)")

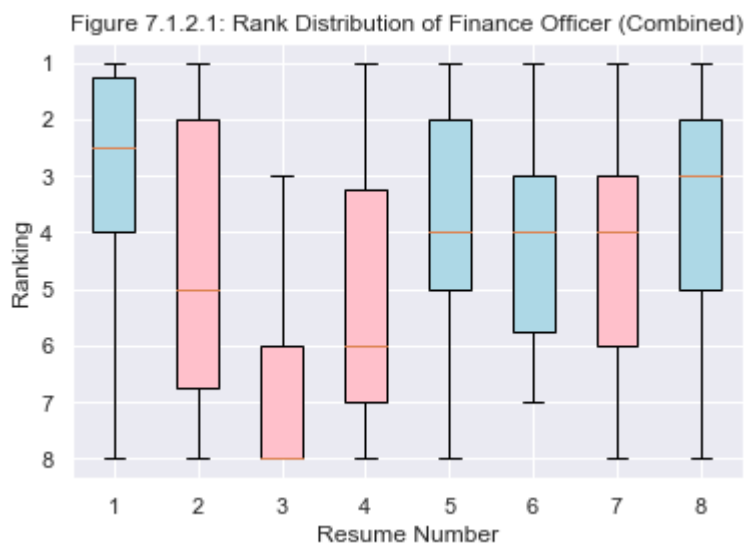
print("Mean Aggregation Method:",y_fo_c)
print("Ordered Pair Distance Aggregation Method:",Y_fo_c)

# fill with colors
colors = ['lightblue', 'pink', 'pink', 'pink', 'lightblue', 'lightblue', 'pink', 'lightblue']

for patch, color in zip(bplot['boxes'], colors):
    patch.set_facecolor(color)
```

Mean Aggregation Method: [1, 6, 8, 7, 2, 4, 5, 2]

Ordered Pair Distance Aggregation Method: [2, 5, 8, 7, 1, 6, 4, 3]



7.1.3 Recruitment Officer

```
In [314]: bplot = plt.boxplot(ro_ranks_c, patch_artist=True)
ax = plt.gca()
ax.set_ylim(ax.get_ylim()[::-1]) #flips the y-axis values

plt.xlabel("Resume Number")
plt.ylabel("Ranking")
plt.title("Figure 7.1.3.1: Rank Distribution of Recruitment Officer (Combined)")

print("Mean Aggregation Method:",y_ro_c)
print("Ordered Pair Distance Aggregation Method:",Y_ro_c)

# fill with colors
colors = ['pink', 'lightblue', 'lightblue','lightblue','pink','pink','pink','lightblue']

for patch, color in zip(bplot['boxes'], colors):
    patch.set_facecolor(color)
```

Mean Aggregation Method: [5, 7, 4, 8, 3, 6, 2, 1]

Ordered Pair Distance Aggregation Method: [7, 6, 4, 8, 2, 5, 1, 3]



7.4 Group Fairness Metric

We developed a function to calculate the group fairness of our model to check whether it was bias towards either gender. Group fairness is when both groups/genders have approximately equal probability of being predicted to be a top candidate. After calculating the results, we realised that the group fairness metric doesn't exactly align with the model design, so it doesn't identify the amount of bias in the model. So, we did not use the results in the final report.

```
In [202]: def group_fairness(ranks, genders, top3):

    # If the individual ranks are passed in, convert them to group format.
    if not top3:
        group_ranks = []

        for rank in ranks:
            temp_rank = []

            for resume in rank:

                if resume < 4:
                    temp_rank.append(1)
                elif resume > 5:
                    temp_rank.append(-1)
                else:
                    temp_rank.append(0)

            group_ranks.append(temp_rank)

        ranks = group_ranks

    total = 0
    top_male = 0
    top_female = 0
    middle_male = 0
    middle_female = 0
    bottom_male = 0
    bottom_female = 0

    for rank in ranks:
        for resume, gender in zip(rank, genders):
            total += 1

            if resume == 1:
                if gender == 'Male':
                    top_male += 1
                else:
                    top_female += 1

            elif resume == 0:
                if gender == 'Male':
                    middle_male += 1
                else:
                    middle_female += 1
            else:
                if gender == 'Male':
                    bottom_male += 1
                else:
                    bottom_female += 1

    print('Top 3 Male Proportion: %.4f' % (top_male/total))
    print('Top 3 Female Proportion: %.4f' % (top_female/total))
    print('Middle 2 Male Proportion: %.4f' % (middle_male/total))
    print('Middle 2 Female Proportion: %.4f' % (middle_female/total))
```

```
print('Bottom 3 Male Proportion: %.4f' % (bottom_male/total))
print('Bottom 3 Female Proportion: %.4f' % (bottom_female/total))
print()
```

```
In [203]: group_fairness(da_ranks_o,da_gender[0],False)
```

```
Top 3 Male Proportion: 0.2202
Top 3 Female Proportion: 0.1548
Middle 2 Male Proportion: 0.1667
Middle 2 Female Proportion: 0.0833
Bottom 3 Male Proportion: 0.1131
Bottom 3 Female Proportion: 0.2619
```

7.4.1 Data Analyst

All Participants

```
In [204]: print("Original Gender Ranks (#participants = %d)"%len(da_ranks_o))
group_fairness(da_ranks_o,da_gender[0],False)

print("Flipped Gender Ranks (#participants = %d)"%len(da_ranks_f))
group_fairness(da_ranks_f,da_gender[0],False)

print("Combined Gender Ranks (#participants = %d)"%len(da_ranks_c))
group_fairness(da_ranks_c,da_gender[0],False)
```

```
Original Gender Ranks (#participants = 21)
Top 3 Male Proportion: 0.2202
Top 3 Female Proportion: 0.1548
Middle 2 Male Proportion: 0.1667
Middle 2 Female Proportion: 0.0833
Bottom 3 Male Proportion: 0.1131
Bottom 3 Female Proportion: 0.2619
```

```
Flipped Gender Ranks (#participants = 18)
Top 3 Male Proportion: 0.2917
Top 3 Female Proportion: 0.0903
Middle 2 Male Proportion: 0.1111
Middle 2 Female Proportion: 0.1389
Bottom 3 Male Proportion: 0.0972
Bottom 3 Female Proportion: 0.2708
```

```
Combined Gender Ranks (#participants = 39)
Top 3 Male Proportion: 0.2532
Top 3 Female Proportion: 0.1250
Middle 2 Male Proportion: 0.1410
Middle 2 Female Proportion: 0.1090
Bottom 3 Male Proportion: 0.1058
Bottom 3 Female Proportion: 0.2660
```

Participant Group Ranks

```
In [205]: with open("CV Json Research Data/Job 1 - Data Analyst/group_rankings_original.
csv", encoding='utf-8-sig') as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ranks=np.asfarray(ranks,float)

print("Original Gender Ranks (#groups = %d)" %len(ranks))
group_fairness(ranks,da_gender[0],True)

with open("CV Json Research Data/Job 1 - Data Analyst/group_rankings_flipped.c
sv", encoding='utf-8-sig') as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ranks=np.asfarray(ranks,float)

print("Flipped Gender Ranks (#groups = %d)" %len(ranks))
group_fairness(ranks,da_gender[0],True)

with open("CV Json Research Data/Job 1 - Data Analyst/group_rankings_combined.
csv", encoding='utf-8-sig') as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ranks=np.asfarray(ranks,float)

print("Combined Gender Ranks (#groups = %d)" %len(ranks))
group_fairness(ranks,da_gender[0],True)
```

```
Original Gender Ranks (#groups = 2)
Top 3 Male Proportion: 0.1875
Top 3 Female Proportion: 0.1875
Middle 2 Male Proportion: 0.2500
Middle 2 Female Proportion: 0.0625
Bottom 3 Male Proportion: 0.0625
Bottom 3 Female Proportion: 0.2500
```

```
Flipped Gender Ranks (#groups = 2)
Top 3 Male Proportion: 0.2500
Top 3 Female Proportion: 0.1250
Middle 2 Male Proportion: 0.1250
Middle 2 Female Proportion: 0.1250
Bottom 3 Male Proportion: 0.1250
Bottom 3 Female Proportion: 0.2500
```

```
Combined Gender Ranks (#groups = 4)
Top 3 Male Proportion: 0.2188
Top 3 Female Proportion: 0.1562
Middle 2 Male Proportion: 0.1875
Middle 2 Female Proportion: 0.0938
Bottom 3 Male Proportion: 0.0938
Bottom 3 Female Proportion: 0.2500
```


7.4.2 Finance Officer

```
In [206]: print("Original Gender Ranks (#participants = %d)" %len(fo_ranks_o))
group_fairness(fo_ranks_o,fo_gender[0],False)

print("Flipped Gender Ranks (#participants = %d)" %len(fo_ranks_f))
group_fairness(fo_ranks_f,fo_gender[0],False)

print("Combined Gender Ranks (#participants = %d)" %len(fo_ranks_c))
group_fairness(fo_ranks_c,fo_gender[0],False)
```

```
Original Gender Ranks (#participants = 22)
Top 3 Male Proportion: 0.2500
Top 3 Female Proportion: 0.1250
Middle 2 Male Proportion: 0.1364
Middle 2 Female Proportion: 0.1136
Bottom 3 Male Proportion: 0.1136
Bottom 3 Female Proportion: 0.2614
```

```
Flipped Gender Ranks (#participants = 16)
Top 3 Male Proportion: 0.2734
Top 3 Female Proportion: 0.1016
Middle 2 Male Proportion: 0.1250
Middle 2 Female Proportion: 0.1250
Bottom 3 Male Proportion: 0.1016
Bottom 3 Female Proportion: 0.2734
```

```
Combined Gender Ranks (#participants = 38)
Top 3 Male Proportion: 0.2599
Top 3 Female Proportion: 0.1151
Middle 2 Male Proportion: 0.1316
Middle 2 Female Proportion: 0.1184
Bottom 3 Male Proportion: 0.1086
Bottom 3 Female Proportion: 0.2664
```

```
In [207]: with open("CV Json Research Data/Job 2 - Finance Officer/group_rankings_origin
al.csv", encoding='utf-8-sig') as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ranks=np.asfarray(ranks,float)

print("Original Gender Ranks (#groups = %d)" %len(ranks))
group_fairness(ranks,fo_gender[0],True)

with open("CV Json Research Data/Job 2 - Finance Officer/group_rankings_flippe
d.csv", encoding='utf-8-sig') as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ranks=np.asfarray(ranks,float)

print("Flipped Gender Ranks (#groups = %d)" %len(ranks))
group_fairness(ranks,fo_gender[0],True)

with open("CV Json Research Data/Job 2 - Finance Officer/group_rankings_combin
ed.csv", encoding='utf-8-sig') as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ranks=np.asfarray(ranks,float)

print("Combined Gender Ranks (#groups = %d)" %len(ranks))
group_fairness(ranks,fo_gender[0],True)
```

```
Original Gender Ranks (#groups = 2)
Top 3 Male Proportion: 0.3125
Top 3 Female Proportion: 0.1250
Middle 2 Male Proportion: 0.1250
Middle 2 Female Proportion: 0.0625
Bottom 3 Male Proportion: 0.0625
Bottom 3 Female Proportion: 0.3125
```

```
Flipped Gender Ranks (#groups = 2)
Top 3 Male Proportion: 0.1875
Top 3 Female Proportion: 0.1875
Middle 2 Male Proportion: 0.1875
Middle 2 Female Proportion: 0.0625
Bottom 3 Male Proportion: 0.1250
Bottom 3 Female Proportion: 0.2500
```

```
Combined Gender Ranks (#groups = 4)
Top 3 Male Proportion: 0.2500
Top 3 Female Proportion: 0.1562
Middle 2 Male Proportion: 0.1562
Middle 2 Female Proportion: 0.0625
Bottom 3 Male Proportion: 0.0938
Bottom 3 Female Proportion: 0.2812
```

7.4.3 Recruitment Officer

```
In [208]: print("Original Gender Ranks (#participants = %d)" %len(ro_ranks_o))
group_fairness(ro_ranks_o,ro_gender[0],False)

print("Flipped Gender Ranks (#participants = %d)" %len(ro_ranks_f))
group_fairness(ro_ranks_f,ro_gender[0],False)

print("Combined Gender Ranks (#participants = %d)" %len(ro_ranks_c))
group_fairness(ro_ranks_c,ro_gender[0],False)
```

Original Gender Ranks (#participants = 20)

Top 3 Male Proportion: 0.1625

Top 3 Female Proportion: 0.2125

Middle 2 Male Proportion: 0.1187

Middle 2 Female Proportion: 0.1313

Bottom 3 Male Proportion: 0.2188

Bottom 3 Female Proportion: 0.1562

Flipped Gender Ranks (#participants = 17)

Top 3 Male Proportion: 0.1618

Top 3 Female Proportion: 0.2132

Middle 2 Male Proportion: 0.1618

Middle 2 Female Proportion: 0.0882

Bottom 3 Male Proportion: 0.1765

Bottom 3 Female Proportion: 0.1985

Combined Gender Ranks (#participants = 37)

Top 3 Male Proportion: 0.1622

Top 3 Female Proportion: 0.2128

Middle 2 Male Proportion: 0.1385

Middle 2 Female Proportion: 0.1115

Bottom 3 Male Proportion: 0.1993

Bottom 3 Female Proportion: 0.1757

```
In [209]: with open("CV Json Research Data/Job 3 - Recruitment Officer/group_rankings_original.csv", encoding='utf-8-sig') as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ranks=np.asfarray(ranks,float)

print("Original Gender Ranks (#groups = %d)" %len(ranks))
group_fairness(ranks,ro_gender[0],True)

with open("CV Json Research Data/Job 3 - Recruitment Officer/group_rankings_flipped.csv", encoding='utf-8-sig') as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ranks=np.asfarray(ranks,float)

print("Flipped Gender Ranks (#groups = %d)" %len(ranks))
group_fairness(ranks,ro_gender[0],True)

with open("CV Json Research Data/Job 3 - Recruitment Officer/group_rankings_combined.csv", encoding='utf-8-sig') as rankings_file:
    rankings = csv.reader(rankings_file)
    ranks = np.array(list(rankings))
    ranks=np.asfarray(ranks,float)

print("Combined Gender Ranks (#groups = %d)" %len(ranks))
group_fairness(ranks,ro_gender[0],True)
```

```
Original Gender Ranks (#groups = 2)
Top 3 Male Proportion: 0.1875
Top 3 Female Proportion: 0.1875
Middle 2 Male Proportion: 0.1250
Middle 2 Female Proportion: 0.1250
Bottom 3 Male Proportion: 0.1875
Bottom 3 Female Proportion: 0.1875
```

```
Flipped Gender Ranks (#groups = 2)
Top 3 Male Proportion: 0.1875
Top 3 Female Proportion: 0.1875
Middle 2 Male Proportion: 0.1875
Middle 2 Female Proportion: 0.0625
Bottom 3 Male Proportion: 0.1250
Bottom 3 Female Proportion: 0.2500
```

```
Combined Gender Ranks (#groups = 4)
Top 3 Male Proportion: 0.1875
Top 3 Female Proportion: 0.1875
Middle 2 Male Proportion: 0.1562
Middle 2 Female Proportion: 0.0938
Bottom 3 Male Proportion: 0.1562
Bottom 3 Female Proportion: 0.2188
```

7.5 Permutation Test

The basic idea of the permutation test is to calculate the number of ways you can select your top candidates, and see how many times a certain number of males and females are selected. This method assumes that each candidate has the same likelihood of being selected.

```
In [316]: combs = list(itertools.combinations(['M','M','M','M','F','F','F','F'],3))

total_counter = [0,0,0,0]
for comb in combs:
    male_count = 0

    for resume in comb:
        if resume == 'M':
            male_count += 1

    if male_count == 3:
        total_counter[0] += 1
    elif male_count == 2:
        total_counter[1] += 1
    elif male_count == 1:
        total_counter[2] += 1
    else:
        total_counter[3] += 1

print([x / len(combs) for x in total_counter])

[0.07142857142857142, 0.42857142857142855, 0.42857142857142855, 0.07142857142857142]
```

This first section of code calculates the likelihoods of selecting 3 males, 2 males, 1 male and 0 males in your top-3 candidates. For example, there are only 4 permutations have 3 males in your top-3 out of 56 combinations, therefore the likelihood is $4/56 = 0.07$.

```
In [211]: combs = list(itertools.combinations(['M','M','M','M','F','F','F','F'],4))

total_counter = [0,0,0,0,0]
for comb in combs:
    male_count = 0

    for resume in comb:
        if resume == 'M':
            male_count += 1

    if male_count == 4:
        total_counter[0] += 1
    elif male_count == 3:
        total_counter[1] += 1
    elif male_count == 2:
        total_counter[2] += 1
    elif male_count == 1:
        total_counter[3] += 1
    else:
        total_counter[4] +=1

print([x / len(combs) for x in total_counter])
```

```
[0.014285714285714285, 0.22857142857142856, 0.5142857142857142, 0.22857142857142856, 0.014285714285714285]
```

We then expanded the idea of the permutation test to all of the data points, where we look at the permutations of the top-3/top-4 permutations. This makes for a good approximation for the p-value of the data but there is a caveat where it makes the assumption that each datapoint is independent but this is not true because each panelist is reviewing the same set of resumes.

```
In [212]: def perm_distri(num_parts,num_top,probs):
    combs = itertools.product(range(num_top+1),repeat=num_parts)
    distri = [0] * (num_parts*num_top+1)
    j = 0

    for i in combs:
        temp_prob = 1
        total = 0
        for j in i:
            temp_prob *= probs[j]
            total += j
        distri[total] += temp_prob

    return distri
```

```
In [213]: def visualise_perm_distri(num_parts,num_top,probs):
           combs = itertools.product(range(num_top+1),repeat=num_parts)
           distri = []
           j = 0

           for i in combs:
               temp_prob = 1
               total = 0
               for j in i:
                   total += j
               distri.append(total)

           return distri
```

7.5.1 Top 3 P-values

```
In [218]: top3 = perm_distri(9,3, [0.0714, 0.4286, 0.4286, 0.0714])
print('Top 3')
print('Data Analyst')
print('Top Females p-value: %.4f'%top3[12])
print('Top Males p-value: %.4f'%top3[15])
print('Bottom Females p-value: %.4f'%top3[11])
print('Bottom Males p-value: %.4f'%top3[16])
print()
print('Finance Officer')
print('Top Females p-value: %.4f'%top3[11])
print('Top Males p-value: %.4f'%top3[16])
print('Bottom Females p-value: %.4f'%top3[18])
print('Bottom Males p-value: %.4f'%top3[9])
print()
print('Recruitment Officer')
print('Top Females p-value: %.4f'%top3[13])
print('Top Males p-value: %.4f'%top3[14])
print('Bottom Females p-value: %.4f'%top3[14])
print('Bottom Males p-value: %.4f'%top3[13])
print()
```

Top 3

Data Analyst

Top Females p-value: 0.1438

Top Males p-value: 0.1438

Bottom Females p-value: 0.0954

Bottom Males p-value: 0.0954

Finance Officer

Top Females p-value: 0.0954

Top Males p-value: 0.0954

Bottom Females p-value: 0.0224

Bottom Males p-value: 0.0224

Recruitment Officer

Top Females p-value: 0.1764

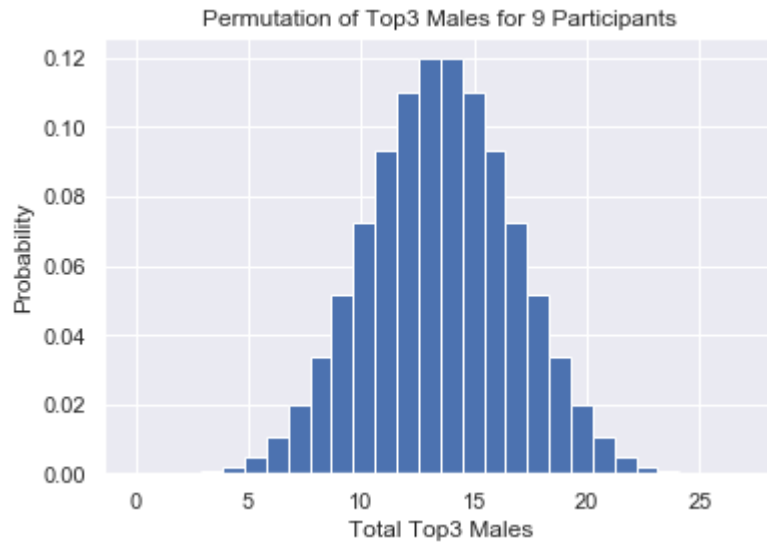
Top Males p-value: 0.1764

Bottom Females p-value: 0.1764

Bottom Males p-value: 0.1764


```
In [219]: plt.hist(visualise_perm_distri(9,3, [0.0714, 0.4286, 0.4286, 0.0714]),bins=28,
density=True)
plt.xlabel('Total Top3 Males')
plt.ylabel('Probability')
plt.title('Permutation of Top3 Males for 9 Participants')
```

Out[219]: Text(0.5, 1.0, 'Permutation of Top3 Males for 9 Participants')



7.5.2 Top 4 P-values

```
In [220]: top3 = perm_distri(9,4, [0.0143, 0.2286, 0.5143, 0.2286, 0.0143])
print('Top 4')
print('Data Analyst')
print('Top Females p-value: %.4f'%top3[16])
print('Top Males p-value: %.4f'%top3[20])
print('Bottom Females p-value: %.4f'%top3[20])
print('Bottom Males p-value: %.4f'%top3[16])
print()
print('Finance Officer')
print('Top Females p-value: %.4f'%top3[15])
print('Top Males p-value: %.4f'%top3[21])
print('Bottom Females p-value: %.4f'%top3[21])
print('Bottom Males p-value: %.4f'%top3[15])
print()
print('Recruitment Officer')
print('Top Females p-value: %.4f'%top3[18])
print('Top Males p-value: %.4f'%top3[18])
print('Bottom Females p-value: %.4f'%top3[18])
print('Bottom Males p-value: %.4f'%top3[18])
print()
```

Top 4

Data Analyst

Top Females p-value: 0.1195

Top Males p-value: 0.1195

Bottom Females p-value: 0.1195

Bottom Males p-value: 0.1195

Finance Officer

Top Females p-value: 0.0737

Top Males p-value: 0.0737

Bottom Females p-value: 0.0737

Bottom Males p-value: 0.0737

Recruitment Officer

Top Females p-value: 0.1756

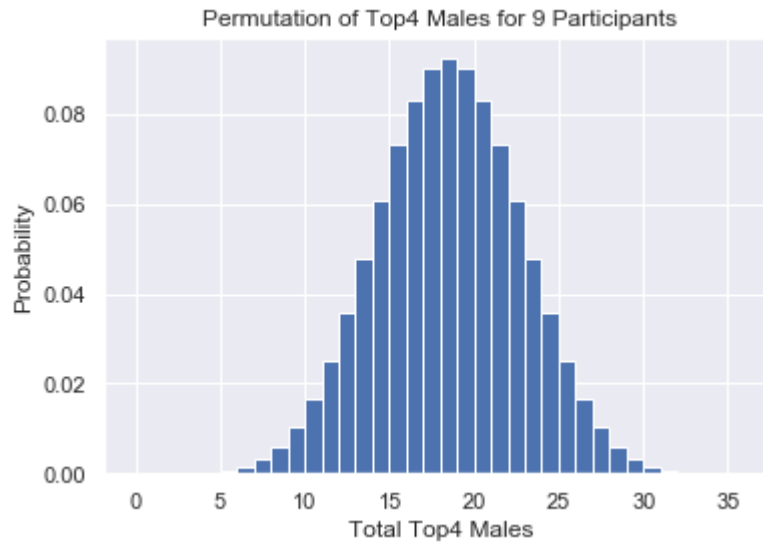
Top Males p-value: 0.1756

Bottom Females p-value: 0.1756

Bottom Males p-value: 0.1756

```
In [221]: plt.hist(visualise_perm_distri(9,4, [0.0143, 0.2286, 0.5143, 0.2286, 0.0143]),  
bins=36,density=True)  
plt.xlabel('Total Top4 Males')  
plt.ylabel('Probability')  
plt.title('Permutation of Top4 Males for 9 Participants')
```

```
Out[221]: Text(0.5, 1.0, 'Permutation of Top4 Males for 9 Participants')
```



References

[1] Ailon, Nir, Moses Charikar, and Alantha Newman. "Aggregating inconsistent information: ranking and clustering." *Journal of the ACM (JACM)* 55.5 (2008): 1-27.

