Fair candidate ranking with spatial partitioning: Lessons from the SIOP ML competition

Ian Burke
iburke@axiomcp.com
Axiom Consulting Partners
Chicago, Illinois, USA

Robin Burke
robin.burke@colorado.edu
University of Colorado, Boulder
Boulder, Colorado, USA

Goran Kuljanin
g.kuljanin@depaul.edu
DePaul University
Chicago, Illinois, USA

ABSTRACT

The ranking and selection of candidates in hiring process is an important function in human resources systems and one that is increasingly become a site for the integration of new technologies. As such implementations have grown, concerns have emerged that unwanted biases may be codified in such systems, enshrining disadvantages for minoritized groups, and therefore algorithm fairness has become an urgent concern. The recent Society for Industrial and Organizational Psychology conference organized a competition in which researchers could explore fairness in candidate ranking for hiring decisions using a data set from a large retail chain. This paper describes our solution, detailing the data management (including feature engineering and missing data imputation), predictive modeling of candidate characteristics, and the multi-criteria spatial ranking algorithm that led to our successful entry.

1 INTRODUCTION

Automated recommendation and ranking systems are playing an increasing role in a variety of human resource functions. Some of these applications are designed to interface with job seekers: recommending open positions likely to be of interest to a particular job candidate [4, 6]. Other applications are targeted towards the employer or recruiter, and rank candidates for consideration; see the survey in [5].

In the United States and other countries, the processes involved in seeking, qualifying and ultimately hiring employees are subject to government regulation, particularly around equal opportunity for a variety of job seekers. It is therefore imperative that organizations designing and deploying systems in this area be cognizant of their fairness properties [16, 18].

In 2020, the Society for Industrial and Organizational Psychology (SIOP) conference organized its machine learning competition around hiring decisions, and it was natural in this context that fairness be a key outcome under consideration [7]. Indeed, fairness of selection tests and decisions and any resulting adverse impact on demographic groups of candidates is an active area of research in industrial and organizational psychology [3, 14, 17] as well as in machine learning. In this competition, the organizers focused on a form of group fairness: proportional equality in hiring outcomes between protected and unprotected groups, and the metric for the competition was designed in such a way that deviations from exact equality would be costly. However, protected group status was not a known variable for the candidates within the test data, so researchers could only estimate the fairness properties of their solutions.

Although the ultimate output of the system would be a binary hire/no-hire decision, we approached this task as a ranking challenge. Our approach was due in part to the structure of the task: since exactly half of the candidates had to be hired, we could think of the problem as one establishing a ranking with the "best" candidates at the top. In our specific solution described here, the parameters of candidate quality were determined by the competition guidelines. However, one can imagine those parameters being learned or being individually specified by recruiters or hiring managers, leading to a more personalized recommender system functionality. We discuss such future extensions of our work in Section 4.

As discussed below, the small amount of data and the associated prediction uncertainties meant that directly optimizing for expected score was not highly effective in practice. Our eventual solution CAndidate Ranking via Multi-model Aggregation (CARMA) incorporated both ranking and multi-dimensional selection, targeted specifically to enhancing protected group representation. This paper describes the competition itself and the characteristics of the data, our data management, preparation and modeling, leading to the algorithm for candidate ranking and eventual selection.

2 SIOP ML COMPETITION

2.1 Overview

The SIOP Machine Learning Competition is an annual contest that focuses on the design and development of the best performing algorithms for particular scenarios relevant to problems in organizational psychology. The 2020-2021 edition of the competition was organized jointly by Nick Koenig and Isaac Thompson of Modern Hire, a Wisconsin based provider of recruiting and hiring solutions, and hosted by EvalAI.[1] The task in this edition was to use a training data set of past hirees to build a system to identify potential hires from among a collection of new candidates, and to do so in a way that met a fairness criterion relative to a protected group.

The competition proceeded in two phases. In the initial development phase, teams could make predictions relative to a holdout set and make up to 100 submissions to the EvalAI platform for scoring. In the final test phase, teams had to make predictions against a different holdout set. Only 5 submissions were allowed and each team's highest scoring submission in this phase was used to determine the final rankings. The test phase was originally scheduled for March of 2020. Because the 2020 edition of the SIOP conference was canceled, the development phase of the competition was extended

RecSysHR '21, October 1, 2021, Amsterdam, The Netherlands

[1] https://eval.ai/web/challenges/challenge-page/527/overview

to February 15 of 2021, with the test phase complete on March 14. The full results of the competition are available on GitHub.[2].

2.2 Scoring

The data for the competition is described in more detail below. Key to the scoring of the competition however were three key binary-valued variables found in the training data, but omitted in the test data:

- **High Performer (*HP*):** This rating was assigned by managers to workers who were considered among the top employees in their particular positions.
- **Retained (*RT*):** The time window for the evaluation was 6 months. Employees still with the firm after this period were considered to be "retained".
- **Protected Group (*PG*):** For privacy purposes, the categories and values of personal and biographical data were obscured in the data. For the purpose of evaluating the fairness properties of the results, the data instead included a binary variable indicating whether a candidate was a member of a protected group.

All submissions were scored using the difference between their overall accuracy as a function of *HP* and *RT* (Equation 2 below) and their unfairness as measured by adverse impact ratio, a function of *PG* (Equation 1 below).[3]

Let C be the total set of candidates among which the algorithm is choosing. The competition required that this set be divided into two subsets of equal size, H (those hired) and $\neg H$, not hired. The score of a decision is determined by characteristics of the candidates in H. Let HP be the subset of C labeled by the supervisor as *high performing*; let RT be the subset of C labeled as *retained*. Finally, let PG be the subset of C that are members of the protected group. The intersection between each of these sets and the hired group H is denoted by the subscript: $HP_H = HP \cap H$.

One goal of the competition was to achieve fair representation of the protected group, the definition in this case, being proportional representation in the H. The *Adverse Impact Ratio* (AIR) can therefore be defined as

$$AIR = \frac{|PG_H|/|PG|}{|\neg PG_H|/|\neg PG|} \tag{1}$$

The organizers defined the accuracy A of the hiring recommendations in terms of the HP_H and RT_H sets, with special attention to their intersection $HP_H \cap RT_H$.

$$A = \frac{1}{4}\frac{|HP_H|}{|HP|} + \frac{1}{4}\frac{|RT_H|}{|RT|} + \frac{1}{2}\frac{|HP_H \cap RT_H|}{|HP \cap RT|} \tag{2}$$

The final score S for each submission was a simple combination of AIR and A, with deviations from proportional fairness being treated on par with accuracy loss.

$$S = 100(A - |1 - AIR|) \tag{3}$$

Note that experimenters were only returned their score when submitting a set of decisions against the test set. The components of the score, such as the AIR or the HP fraction among the hired predictions remained unknown, and thus it was not always easy to tell how different algorithm variants were performing on the key subtasks of the problem. Based on our own sampling of the data, it was clear that both the initial and final phase test data sets were fairly distinct from the training data, an additional challenge for building a generalizable model.

Because the test data was undisclosed, the practical upper bound of the score is unknown, but was estimated by the organizers to be around 80. The highest score in the development phase was 61.72. The winning score in the test phase was also in the low 60s: 62.53, achieved by a team from Bowling Green State University. Our solution scored 62.50, less than 0.05% lower. We estimate the difference between first and second place may have come down to a single candidate difference between the two solutions.

2.3 Data

The data set provided for the propose of the competition contains anonymized pre-employment assessment results for 48,602 entry-level Walmart retail workers. This data consists of a training set (n=44,102) as well as two smaller holdout sets (n=2,250 each). The training and holdout data sets both contain individual responses to assessment questions in four distinct categories. The key dependent variable of *HP* was provided only for a subset of the individuals (n=12,390). The employee retention (*RT*) and protected group (*PG*) variables were included for the full population.

Specific question text, data items, and personality scales associated with the candidate profiles were not divulged, only raw data and the associated category of information. They are listed here with the number of variables in each category.

- **Situational Judgment (27):** Candidates are described an on-the-job situation and have to select an appropriate action. Each situational judgement item yielded three variables. The first variable indicates which of the actions the candidate was most likely to take. The second variable indicates the action the candidate was least likely to take, and the third item indicates the time the candidate took to answer the question. Items are coded 1 to 4 for both most and least likely and the time variable is in seconds.
- **Scenario Interpretation (18):** Candidates view a scenario, then have to respond with the number of times that each of the response options occurred during the scenario. There are 8 options per scenario followed by a variable indicating the time the candidate took to answer the question. Values in the items range between 0 and 9. Values for time items represent time taken in seconds.
- **Biodata / Work History (20):** Candidates read an item and select a single response. The number of response options for these items varies between 5 and 8.
- **Personality/Work Style (55):** These items have a bipartite response format. Anchors appear above and below the item and the respondent chooses a single response. The items are grouped and labeled by subscale. There are 13 of these subscales with varying numbers of items per subscale coded 1 to 4.

[2]https://github.com/izk8/2021_SIOP_Machine_Learning_Winners
[3]https://eval.ai/web/challenges/challenge-page/527/evaluation

3 OUR SOLUTION

The CARMA solution had a number of steps detailed here. First, missing data had to be imputed as a significant number of data elements across all feature categories were blank, approximately 4.5% of the total. Then, we used predictive modeling to derive predictors for the three key variables *HP*, *RT* and *PG*. Finally, we used a spatial partitioning technique to rank and select the candidates.

3.1 Feature Imputation

Missing elements (except for the three key dependent variables) in the training data were imputed using random forest imputation, customized to the type of item. We used a random forest implementation for missing data using the `missForest` [19] package from `R`. We imputed values by sets of feature variables. In particular, we imputed: (1) all of the original 18 situational judgment and 20 biodata items together as nominal factors, (2) the 55 personality items as ordinal factors, (3) the 16 scenario items as integer variables, and (4) the 11 situational judgment and scenario time variables as log-transformed numeric variables.

3.2 Feature Engineering

Our initial investigations using the raw data suggested that the key dependent variables were difficult to predict directly from this input. Using our knowledge of the different data types in the input, we developed a set of transformation methods to create more informative features from the initial data.

- **Situational Judgement**: Recall that these questions required the candidate to identify how they would be most likely and least likely to respond in a given situation. We understood that there was probably a "right" way to answer these questions, and so users would be similar in how they deviated from the general population. Therefore, we computed the popularity of each choice and created features for each user corresponding to the popularity (fraction of occurrences) of their answers. This created 2 floating point values (most/least) for each of the nine judgements.
  We used two strategies to combine these most/least scores on each judgement, averaging them to reflect the mean popularity of the candidate's answer and also multiplying them to amplify the effect of agreement or disagreement with the overall population. This created an additional 18 features.
  We found the time values to be very skewed with most individuals answering quickly and a few taking longer. We used a log transform of the time to create a more linear distribution of values, creating 9 more features for these judgements.
- **Scenario Interpretation**: As above, for these interpretation questions, we assumed there was a right answer and this was majority response. Since these were scalar responses, we calculated the absolute distance between the most common value and that supplied by the candidate. We also used a log transformation as above to process the time values.
- **Personality**: The 55 personality items were associated with 13 personality scales. We identified items whose values had a negative first principal component loading for the scale and we reversed those items scoring so that all associations were positive. Then, the responses for the items of a scale were averaged to create 13 personality scores.
- **Biodata/Work History**: The 20 biodata items were used in their raw form and converted to dummy features. We also created 5 categorical principal components using the `Gifi` [11] package from `R`.

The end result of the imputation and feature engineering stages was a new training data set containing 195 (including biodata dummy) features over the 44,102 candidates.

3.3 Predictive Modeling

Our team evaluated numerous predictive modeling strategies during the development phase of the competition before deciding on a particular solution for the test phase of the competition. To do our predictive modeling, we used machine learning packages from both *Python* [20] and *R* [15], while our ultimate solution relied on the *tidymodels* [10] package from *R*. We converged on our particular solution after considering five modeling decisions: (1) How to treat the three target variables? (2) What machine learning algorithms to apply to the data? (3) How to select optimal values for hyperparameters? (4) What features to select for the target variables? (5) How to evaluate the performance of our models before submissions? We review our decision evaluation processes and our ultimate decisions to these questions in this section.

The first major predictive modeling question we considered involved evaluating whether we should treat this as a single multivariate prediction problem or as three separate univariate prediction problems. The three target variables (*HP*, *RT*, *PG*) were all coded as binary yes-no variables. Crossing the two levels for these three variables creates eight groups, and, essentially, turns this into a single multiclass prediction problem.[4]

For this multiclass prediction problem, we evaluated how successful various machine learning algorithms were at correctly predicting which candidates belonged to the eight groups (i.e., we estimated the probability of belonging to the eight groups for each candidate). We compared this approach to an approach where we treated each target variable independently from one another, where we predicted the two levels of each target variable independently of one another (i.e., we estimated the probability of belonging to the yes category for each target variable for each candidate). The multivariate prediction problem approach has the convenience of considering which of the eight groups is most appropriate for each candidate (i.e., the combined category with the highest probability). Despite this convenience, we did not find success with respect to our submissions during the development phase when considering this as a multivariate prediction problem. Thus, we focused on modeling each of the three target variables independently and combining the resulting probabilities per our ranking methodology.

Having characterized the problem as predicting the three variables separately, we needed to determine what machine learning algorithms to apply. We considered essentially two types of algorithms: (1) those that attempt to find interactions between features and (2) linear algorithms. Among non-linear methods, we tried random forests, gradient boosted machines, neural networks, bagged

[4]We also considered a sequential approach in which *PG* was predicted first, but this proved unreliable.

multivariate adaptive regression splines, support vector machines, among other algorithms. With respect to the linear models, we tried logistic regression, elastic nets, and linear discriminant models. When evaluating the various algorithms against each other, we relied on comparing them using standard hyper-parameters (due to time constraints of the competition) and via tuning (using multiple methods including grid search and simulated annealing) of hyper-parameters. Across our various evaluations, on the whole, we found elastic net models performed the best. In other words, the elastic net linear models were able to perform as well and often better than the machine learning algorithms attempting to capitalize on any interactions between features. We think one possible explanation for this result is that these particular features may not be sufficiently sensitive for interaction effects, given the limited amount of data available and the fact that so many values were imputed.

Finding optimal hyper-parameters for any given algorithm involved some trial and error. For instance, elastic net models come with two hyper-parameters that require tuning: (1) penalty and (2) mixture. We used simulated annealing [8, 9], a global search algorithm, to assist in finding optimal values for the two hyper-parameters. We did discover that only particular combinations of values for these two hyper-parameters resulted in optimal scoring performance, during our development phase data submissions. However, these optimal values of the test data were not the very best values found by the simulated annealing search process over the training data. Thus, as with other modeling decisions, the discrepancy between training set and test set performance meant a great deal of experimentation during the development phase.

We explored a number of options for additional feature selection and feature engineering once we had settled on the elastic net model. However, we did not find any benefits to removing feature variables based on relatively high inter-correlations between themselves. Furthermore, we did not find any benefits to manually constructing interactions between different sets of feature variables. This second result converges with our finding that the machine learning algorithms which attempt to find interactions among the features variables did not perform better than linear additive models. So, in the end, we had a total of 195 independent variables, and three different elastic net models trained on with *HP*, *RT*, and *PG*, respectively as the dependent variable.

To evaluate the performance of our models on the training data, we used 5-fold cross-validation and tracked for each of the target variables the following metrics: (1) sensitivity, (2) specificity, (3) positive predictive value, (4) negative predictive value, (5) the j-index, (6) overall accuracy, and (7) the area under the receiver-operator characteristic curve. For this particular problem, we found maximizing the j-index (i.e., the sum of sensitivity and specificity minus one) to be the most useful for model performance and best correlated with the results from our test data submissions.

The output of our elastic net models were in the range $[0, 1]$, which we interpreted as the inferred probability of each candidate belonging to the yes category for each of the three target variables. We refer to these outputs as $\hat{P}(HP)$, $\hat{P}(RT)$ and $\hat{P}(PG)$. We combined these three sets of probabilities to make our selection decisions as described in the next section.

3.4 Ranking Methodology

Extensive study of the data sets and the predictive capacity of our algorithms led to the conclusion that, while optimizing for our probability of the *HP* feature was effective at yielding high scores for some components of the scoring function, simply sorting candidates by this probability yielded a low score on the *AIR* part of the function. In other words, the $\hat{P}(HP)$ feature was skewed towards the unprotected group, and, as noted above, the nature of the scoring function meant that a high score required a very precise balance between the protected and unprotected groups.

The competition required entrants to divide the test set in half and recommend placement for exactly half (1125) of the candidates. Our solution to this aspect of the problem was as follows:

(1) Partition the candidate set C into two halves C_1 and C_2, sorting by the $\hat{P}(HP)$ variable and keeping the top scorers in C_1.
(2) Construct a 3-dimensional space of the C_1 partition using the predictions $\hat{P}(HP)$, $\hat{P}(RT)$, and $\hat{P}(PG)$, and from the corner of the space closest to origin, identify an axis-aligned rectangular partition P_1 of C_1 containing exactly k candidates.
(3) Construct a 3-dimensional space of the C_2 using the predictions $\hat{P}(HP)$, $\hat{P}(RT)$, and $\hat{P}(PG)$, and from the corner of the space farthest from the origin, identify an axis-aligned rectangular partition P_2 of C_2 also containing exactly k candidates.
(4) From C_1, drop P_1 and add P_2, returning this set of candidates as the desired hires.

This process has several useful characteristics. First, it removes from the high performer set only a small number of candidates. This was appropriate because we knew that the candidate set based on *HP* prediction was fairly good, lacking only a small number of protected candidates to achieve balance. The lower corner of the C_1 space is the one with the candidates least likely to score well (lower $\hat{P}(HP)$ and least likely to cause us to lose protected group individuals (lower $\hat{P}(PG)$. The individuals added in from P_2 would be close to C_1 in terms of $\hat{P}(HP)$, and they would have higher likelihood relative to the other variables.

The choice of a rectangular boundary has implications for procedural fairness and also the explainability of the algorithm. All users within a particular range of (predicted) scores are treated the same by the algorithm. This would not be true if we had ranked the list by expected score or if we had chosen a single candidate prototype and sorted by distance from it.

A sketch of the spatial search process is given in Figure 1. The algorithm is choosing candidates to drop based on two dimensions, x and y. The user defines the shape, in this case expressing a 2:1 preference for y axis over x. Thus, the y height for the shape is smaller: the algorithm will drop candidates with higher scores in x before it will drop candidates with similar y scores. Assume that the user also specifies 10 candidates to be dropped. The small version of the shape contains only 2 candidates, too few; the larger-scaled shape in grey encloses too many candidates. The binary search yields the black bounding shape enclosing exactly 10 candidates.

As a practical matter, the data will not always be distributed in such a way that the minimum value in all dimensions is zero. To preserve the preference relation encoded in the shape parameter,

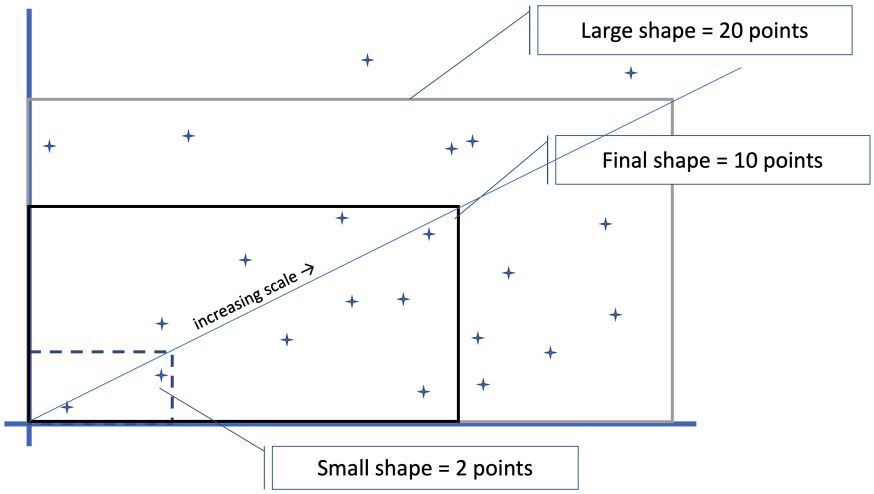


Figure 1: Schematic of spatial search. Desired shape emphasizes y axis over x axis by a ratio of 2:1. Query seeks 10 points, thus the solid black boundary is found after rejecting larger and smaller scaled versions. Diagonal line indicates the scale factor associated with each shape.

therefore, it may be necessary to normalize or shift the data so that the origin is a reasonable bound.

The implementation of steps 2 and 3 of the algorithm outlined above can be understood as a type of search over spatial partitions. The experimenter identifies priorities to be associated with the different dimensions in each of the P_1 and P_2 partitions, and the precise number of candidates to swap. If too few candidates are swapped, there is not enough "space" to improve the protected group distribution. If too many candidates are swapped, other aspects of the score will suffer.

A tool in performing this work is the search algorithm described in Algorithm 1. This algorithm efficiently identifies a boundary containing k points in a multi-dimensional space such that the boundary is proportional to the input shape. The input shape defines the priorities to be placed on each dimension of the space and the search determines how to scale the input shape to select the right number of candidates.[5]

Our implementation of this search in Python enables easy experimentation with different tradeoffs between k and the shapes prioritizing different dimensions of the data. The computational complexity arises in the implementation of the *encloses* function, which is essentially a multi-dimensional range query, but this can be optimized with a spatial data structure such as a k-d tree.

Figure 2 shows a two-dimensional projection of the candidate dropping phase of the operation using the SIOP data. Only the $\hat{P}(RT)$ and $\hat{P}(PG)$ axes are shown. The left figure shows the full data set with all the candidates and the selected candidates shown in red. The right figure shows just the items in the inset box where the boundary between selected and non-selected candidates can be more clearly seen.

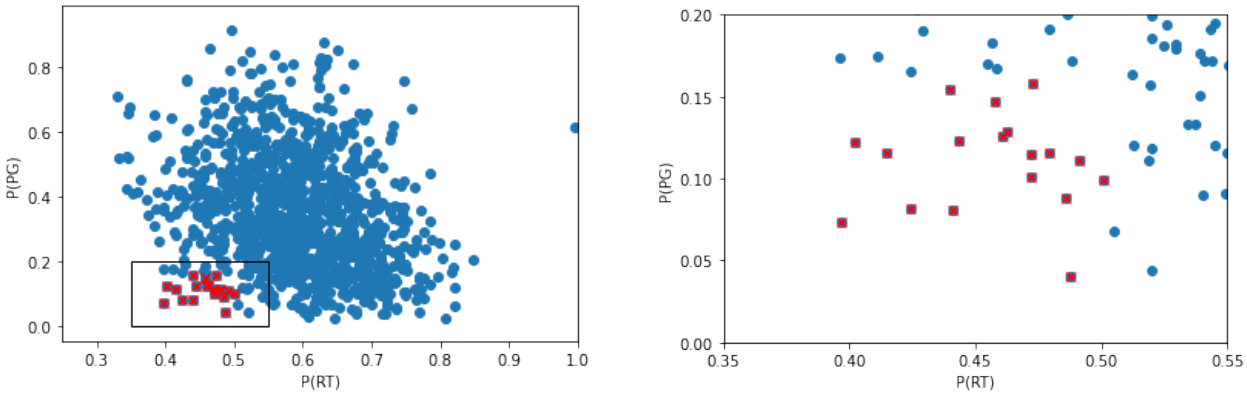
[5]A repository for our implementation of this algorithm can be found at https://github.com/that-recsys-lab/ratchet-search. Note that this code differs from that submitted to the SIOP competition in that this code is generalized to any number of dimensions and allows for automatic, rather than manual, detection of segment boundaries.

Algorithm 1: Binary spatial search. The goal is to find a uniform scale transform of shape that encloses exactly k points . The scaleToFit function scales S such that all points in C are enclosed.

input : shape S, set of points C, k points to return
output: boundary = $S \times scale$
$S \leftarrow$ scaleToFit(S, C);
scale $\leftarrow$ 1;
i $\leftarrow$ 1;
repeat
  i $\leftarrow$ i + 1;
  if *encloses(boundary, C)* = k **then**
    return boundary
  else if *encloses(boundary, C)* > k **then**
    scale $\leftarrow$ scale - $1/2^i$;
  else
    scale $\leftarrow$ scale + $1/2^i$
  boundary $\leftarrow$ S * scale;
until *stopping condition*;

4 DISCUSSION

Much of the recommender systems research in hiring has focused on recommendations made to job seekers, in line with the RecSys Challenge from 2016 and 2017 [1, 2]. The challenge of providing recommendations on the employer side of the hiring process (also known as e-recruitment) has received less attention. According to the survey from Freire and de Castro [5], the major research areas on the e-recruitment side have been around the extraction of features to support candidate matching (from natural language, i.e. resumes and job descriptions, and from social media sites). Multi-objective ranking from candidate screening tests, which was our problem here, has seen less attention. In addition, although fairness is obviously a key (and often legally mandated) characteristic in



(a) All candidates. **(b) Close-up of lower left box.**

Figure 2: Spatial search in the SIOP data.

hiring decisions, there has been little published research examining fairness-aware HR processes from the employer side, no doubt due to the sensitivity of the associated data.

Characteristically for machine learning competitions, the simplified and somewhat artificial nature of the problem leads to solutions that are, to some extent, highly tailored to the competition objective and not necessarily generalizable beyond it. However, there are several directions that we see for future development and generalization of CARMA.

First, our solution is flexible with respect to the dimensions of the model prediction space. The competition required that we predict the "high performer", "retained" and "protected group" features, and the scoring methodology dictated the importance of high performer as the first stage ranking function. However, any number of such features deemed important by an employer could be employed. In addition, user- or company-specific learning-to-rank could be employed to take the place of the simple scoring procedure employed here.

Second, the multi-stage aspect of the system is specifically targeted towards achieving protected group balance in a setting in which the protected characteristic is unknown in the prediction setting but known in the training data. More typical for the algorithmic fairness setting would be one in which the protected feature(s) are known. That simplifies the spatial partitioning task, as the exact number of candidate "swaps" can be determined and there is no uncertainty about the fairness properties of the resulting result set. However, it is also possible that legal or internal guidelines would prevent a candidate ranking algorithm from having access to protected group data when making decisions, and in that case, this feature would need to be modeled as it was here.

A key element of the CARMA solution is the shape parameter passed to Algorithm 1, which represents the user's relative preferences over the different dimensions of the candidates. This is a hard constraint, as is the number of candidates sought within the region. However, in a real situation, it might be desirable for the user to explore the space of tradeoffs in a more supported manner. For example, the system could return a family of results based on slight variations of the original shape specification. This would be particularly important if the dimensionality of the space were higher.

It may also be the case that the dimensions and specific shape of the data may render the origin-based approach implemented in our spatial search ineffective. For example, if data forms a "shell" at some distance from the origin, it will not be possible to provide an intuitive rectangular cut of candidates. Analysis of the distribution of the data across the various candidate dimensions is necessary to ensure the assumptions of this methodology hold. Extensions of the technique to more complex shapes are also possible.

The multidimensionality of the SIOP competition scoring function was an important aspect of the competition. It is necessary to manage the trade off between "high performer" and "retained" because the very best employees are likely to move on from what in this case are entry-level positions, and there is value in having workforce stability. In this light, the value of the CARMA approach, especially in the candidate swapping procedure, is that it assembles a portfolio of individuals with characteristics in a desirable range rather than assuming a prototype ideal worker against which everyone is compared. This contrasts to a matching methodology, which assumes a single candidate will be hired, based on best fit to a job description, for example.

One interesting challenge for a methodology like ours (and for any result deriving from a competition like the SIOP one) is how the solutions would fare when the dynamic nature of workforce management must be considered. One could imagine a situation in which only a percentage of the candidates is seen in a given time period and the decision maker has to make a local online decision without knowing who will show up in the next interval. It is unclear what are reasonable fairness properties to expect in this context: should every slate be balanced relative to the *AIR* criterion or is it sufficient that the total set of offers be balanced in hindsight? These are questions that practical deployments of fairness-aware systems will require.

A final limitation to note on the question of fairness is the problem of representation bias [12] in the data supplied for the competition. Only individuals actually hired by Walmart are represented in the data supplied for the competition. Any system working from this data would have no opportunity to learn from false negatives: individuals who would have made great employees but were not hired. Their personality traits, judgements and other characteristics are not represented in the data and cannot be modeled, increasing the chance that similar individuals will excluded in the future. The competition therefore could be said to embody what O'Neil [13] calls a "weapon of math destruction", a positive feedback loop of reinforced bias. It is an open question whether the effort to achieve protected group balance in hiring recommendations is sufficient to break this loop, and this fact should give us pause about the ultimate fairness of any associated solution.

REFERENCES

[1] Fabian Abel, András Benczúr, Daniel Kohlsdorf, Martha Larson, and Róbert Pálovics. 2016. Recsys challenge 2016: Job recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 425–426.

[2] Fabian Abel, Yashar Deldjoo, Mehdi Elahi, and Daniel Kohlsdorf. 2017. Recsys challenge 2017: Offline and online evaluation. In *Proceedings of the eleventh acm conference on recommender systems*. 372–373.

[3] Wilfried De Corte, Filip Lievens, and Paul R Sackett. 2007. Combining predictors to achieve optimal trade-offs between selection quality and adverse impact. *Journal of Applied Psychology* 92, 5 (2007), 1380.

[4] Mamadou Diaby, Emmanuel Viennet, and Tristan Launay. 2014. Exploration of methodologies to improve job recommender systems on social networks. *Social Network Analysis and Mining* 4, 1 (2014), 227.

[5] Mauricio Noris Freire and Leandro Nunes de Castro. 2020. e-Recruitment recommender systems: a systematic review. *Knowledge and Information Systems* 63 (2020), 1–20.

[6] Francisco Gutiérrez, Sven Charleer, Robin De Croon, Nyi Nyi Htun, Gerd Goetschalckx, and Katrien Verbert. 2019. Explaining and exploring job recommendations: a user-driven approach for interacting with knowledge-based job recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*. ACM, New York, 60–68.

[7] Nick Koenig and Isaac Thompson. 2021. The 2020-2021 SIOP Machine Learning Competition. Presented at the 36th annual Society for Industrial and Organizational Psychology conference, New Orleans, LA.

[8] Max Kuhn and Kjell Johnson. 2019. *Feature engineering and selection: A practical approach for predictive models*. CRC Press. http://www.feat.engineering/

[9] Max Kuhn and Julia Silge. 2021. *Tidy Modeling with R*. https://www.tmwr.org/

[10] Max Kuhn and Hadley Wickham. 2020. *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles*. https://www.tidymodels.org

[11] Patrick Mair, Jan De Leeuw, and P Groenen. 2019. *Gifi: Multivariate Analysis with Optimal Scaling*. R package version 0.3.9.

[12] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–35.

[13] Cathy O'Neil. 2016. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Crown.

[14] Robert E Ployhart and Brian C Holtz. 2008. The diversity–validity dilemma: Strategies for reducing racioethnic and sex subgroup differences and adverse impact in selection. *Personnel Psychology* 61, 1 (2008), 153–172.

[15] R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

[16] Manish Raghavan, Solon Barocas, Jon Kleinberg, and Karen Levy. 2020. Mitigating bias in algorithmic hiring: evaluating claims and practices. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (Barcelona, Spain) *(FAT* '20)*. Association for Computing Machinery, New York, NY, USA, 469–481. https://doi.org/10.1145/3351095.3372828

[17] Ann Marie Ryan, Robert E Ployhart, and Lisa A Friedel. 1998. Using personality testing to reduce adverse impact: A cautionary note. *Journal of Applied Psychology* 83, 2 (1998), 298.

[18] Javier Sánchez-Monedero, Lina Dencik, and Lilian Edwards. 2020. What does it mean to 'solve' the problem of discrimination in hiring? social, technical and legal perspectives from the UK on automated hiring systems. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (Barcelona, Spain) *(FAT* '20)*. Association for Computing Machinery, New York, NY, USA, 458–468. https://doi.org/10.1145/3351095.3372849

[19] Daniel J. Stekhoven. 2013. *missForest: Nonparametric Missing Value Imputation using Random Forest*. R package version 1.4.

[20] Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.