Predicting Job Application Success with Two-Stage, Bayesian Modeling of Features Extracted from Candidate-Role Pairs

Jon Krohn* Gabe Rives-Corbett* Ed Donner*

Abstract

We describe a two-stage model for identifying the best-suited candidates for a given role that (1) learns hiring manager preferences for the role; (2) can be updated frequently, with low latency; and (3) scales to a very large number of roles. In the first stage, dozens of candidate-role features are modeled in a regression across all applications and roles. The feature weights are subsequently fed as priors into n individual Bayesian models representing n roles. Cross-validated results indicate this approach improves classification accuracy, with the area under the curve of the receiver operating characteristic improving from 71.8% to 77.0%.

Key Words: high-dimensional data, Bayesian, web application, natural language processing, structured features, unstructured features

1. Introduction

We have developed a digital platform for matching software engineers with personalized suggestions for full-time roles in the technology sector. Via a web application hosted at `untapt.com`, candidates provide us with job search-relevant questionnaire responses (e.g., geographical location) and descriptions of their work experience, level of expertise with relevant skills, and educational background.

Our objective was to build a statistical model that identifies the best-suited engineers, from a pool containing tens of thousands of candidates, for roles posted by our clients. For our use case, we had three key criteria. First, the model must learn the preferences of the hiring manager for a given role. Second, it must be possible to update the model frequently (i.e., at least daily) to incorporate recent manager decisions. Third, the model needs to be scalable beyond 10^4 roles.

Our data consisted of all 10^4 applications made to several hundred software engineering roles via our platform since its launch in March 2015. We defined cases as positive (y = 1) if the applicant was invited to interview and negative (y = 0) otherwise. At the time of analysis, we had 10^6 instances of candidate-role pairs where there was a skills match and therefore an opportunity to predict the suitability of the candidate for the particular role. These 10^6 predictions, computed as probability p_i, can then be made available downstream within the platform for operational purposes (e.g., automated application handling), engagement with candidates (e.g., personalized role-specific recommendations and rankings), and engagement with clients (e.g., providing estimates of the availability of well-qualified candidates given the particular attributes of a role).

Job-specific feature weights (posterior means) could theoretically be estimated with PyMC3 (Salvatier et al., 2016) or Stan (Carpenter et al., in press) via NUTS sampling (Hoffman & Gelman, 2014). In practice, however, such a hierarchical model with dozens of features and 10^2 roles is computationally expensive. With 10^3 roles, it becomes intractable. To overcome these limitations, we have developed a methodology that separates modeling into two stages.

*untapt, 225 W 35th St, New York, NY 10001

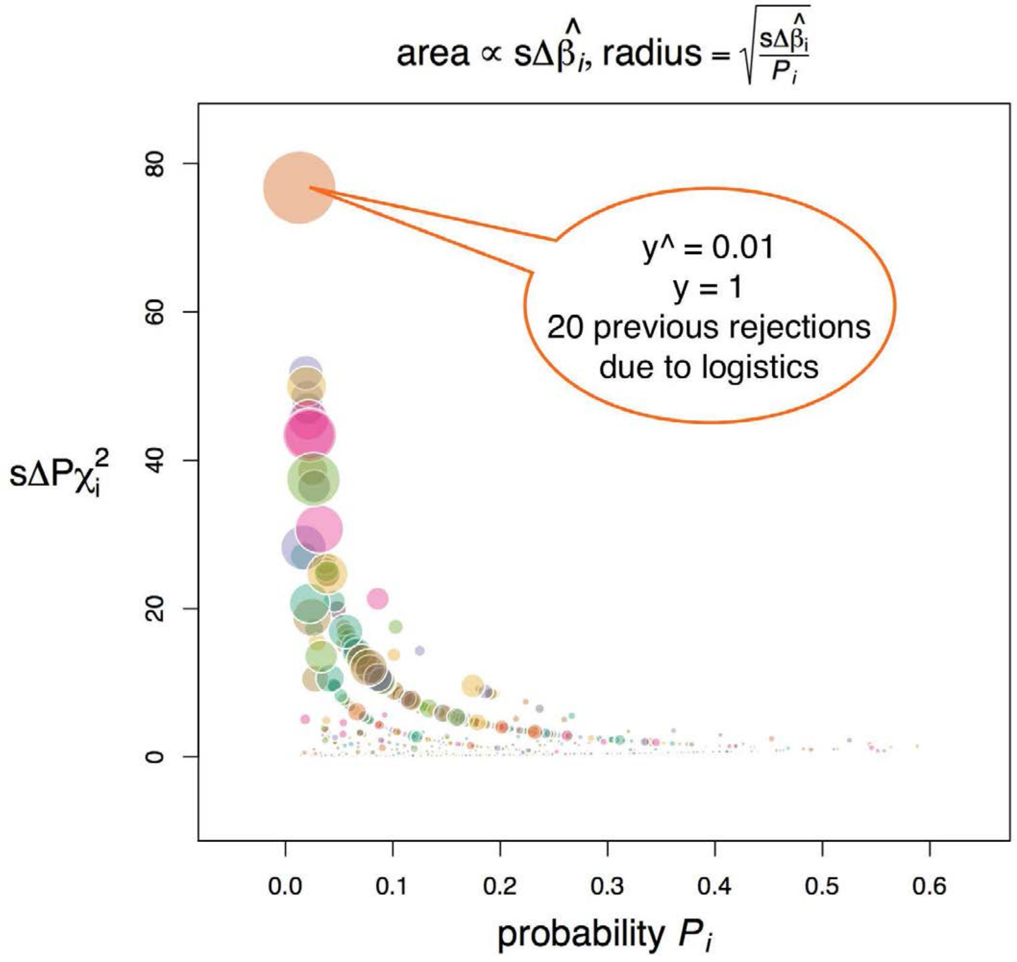


Figure 1: Residual analysis.

2. Method

2.1 General Model

$$ln\Big(\frac{p_i}{1-p_i}\Big) = \beta_0 + \beta_1 x_{1,i} + ... + \beta_m x_{m,i}$$

We began by fitting an Elastic Net-regularized logistic regression model across all applications and roles. The features ($x_{1,i}$ through $x_{m,i}$) we engineered from each 10^6 candidate-role pair i included structured application data (e.g., whether the candidate's years of relevant technical experience approximate the requirements of the job description); unstructured application data (e.g., abstractions of the natural language used within a candidate's work experience entries); clusters of related technical skills; and within-platform behavior.

While developing this general model, residual analysis as illustrated in Figure 1 played a key role in identifying over-leveraged cases and adjusting our feature generation accordingly. In this example, the model predicts a low probability (.01) of invitation to interview because the candidate had been rejected on 20 previous occasions, but the application was nevertheless successful. An effective, leverage-attenuating resolution was to cap the feature.

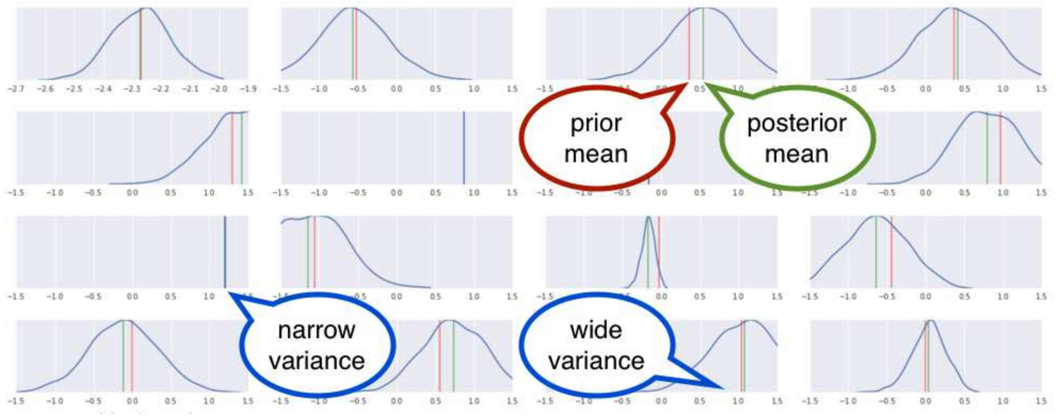


Figure 2: Prior means for sixteen features as assessed by the first-stage general model in §2.1 and the posterior distributions learned for a single hiring manager via the Bayesian model described in §2.2.

2.2 Job-Specific Model

To learn hiring manager preferences for n individual roles, we passed the general feature weights β_1 through β_m from §2.1 as prior means to n individual Bayesian models. We sampled with the No U-Turn Markov chain Monte Carlo approach (Hoffman & Gelman, 2014). Figure 2 depicts this across 16 features for a single role. The red vertical line segments indicate priors while the green vertical line segments represent the mean of the sampled posteriors, which reflect role-specific preferences. We leveraged domain-specific knowledge to assign narrow prior variance to some features and wider variance to others.

Among other downstream uses across the platform mentioned in §1, p_i for each candidate-role pair i determines which applications to forward to hiring managers, who get real-time feedback on their candidates and their own in-platform behavior. An example panel for providing this feedback is provided in Figure 3.

3. Results

Using $k = 5$ cross-validation, the Bayesian model described in §2.2 demonstrably improves classification accuracy over the general model in §2.1, as assessed, for example, by the area under the curve of the receiver operator characteristic and root-mean-square error in Table 3. Figure 4 illustrates this improvement on a role-by-role basis. Each of the points represents a job and the diagonal demarcates equal performance before and after the Bayesian model in §2.2. All but two roles with more than 30 hiring manager decisions exhibit improved classification accuracy.

Model	ROCAUC	RMSE
Stage 1	.718	.316
All Stage 2s	.770	.309

4. Discusssion

With the approach outlined here, we are able to predict role-by-role hiring preferences. By breaking modeling into multiple phases, achieve this while updating our model inexpen-

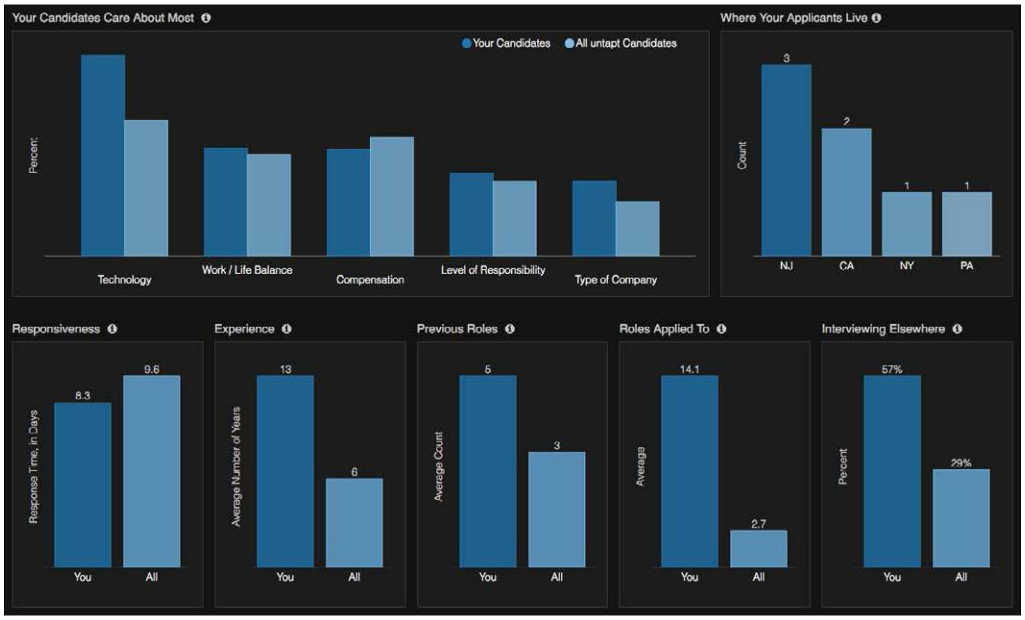


Figure 3: Real-time applicant and hiring manager behavioral metrics, as rendered for hiring managers while using the platform.

sively, and with a low latency. Further, the approach trivially scales to a very large number of roles with a linear increase in computational complexity, with each role-specific model fit on a dedicated core.

The next step for us is the incorporation of deep-learning derived features into our model, such as those we provide for user interaction at `deep-orange.untapt.com`, with examples provided in Figure 5.

5. Acknowledgements

Special thanks to David K.A. Mordecai (New York University) for guidance and feedback on our research, particularly with respect to feature development and refinement via residual analysis.

REFERENCES

Carpenter, B., Gelman, A., Hoffman, M. et al. (in press), “Stan: A probabilistic programming language,” *Journal of Statistical Software*.

Hoffman, M., and Gelman, A. (2014), “The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo,” *Journal of Machine Learning Research*, 15, 1351–1381.

Salvatier, J., Wiecki, T. V., and Fonnesbeck, C. (2016), “Probabilistic programming in Python using PyMC3,” *PeerJ Computer Science*, 2, e55.

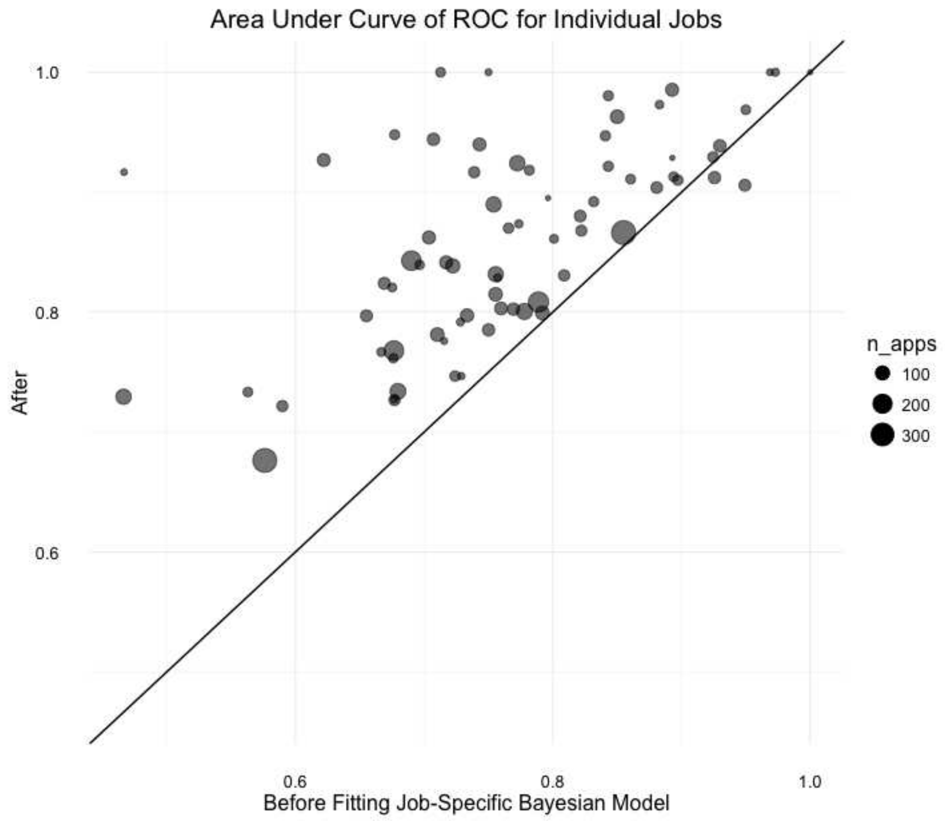
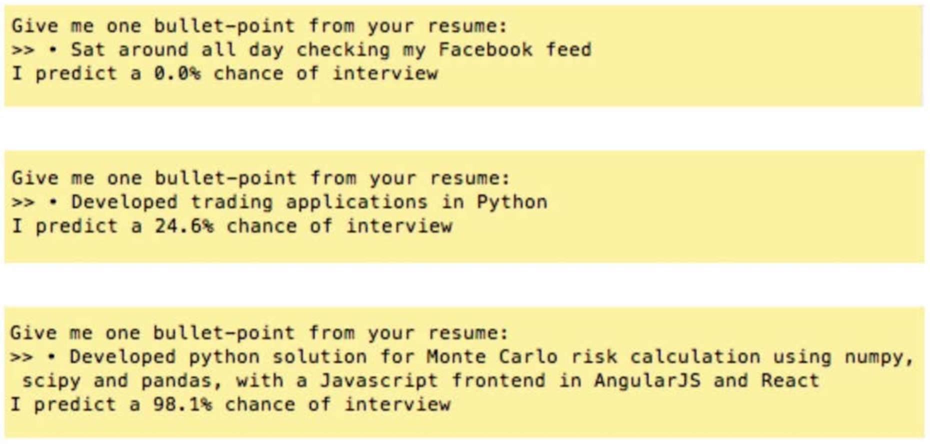


Figure 4: Percentage of the area under the curve of the receiver operator characteristic for all jobs with more than 30 decisions shown before and after the role-specific modeling in §2.2.



```
Give me one bullet-point from your resume:
>> • Sat around all day checking my Facebook feed
I predict a 0.0% chance of interview
```

```
Give me one bullet-point from your resume:
>> • Developed trading applications in Python
I predict a 24.6% chance of interview
```

```
Give me one bullet-point from your resume:
>> • Developed python solution for Monte Carlo risk calculation using numpy,
 scipy and pandas, with a Javascript frontend in AngularJS and React
I predict a 98.1% chance of interview
```

Figure 5: User inputs and outputs from the interactive resume bullet-point assessment tool "Deep Orange".