Improving Jobseeker-Employer Match Models at Indeed Through Process, Visualization, and Exploration

Benjamin Link, Eric Lawrence, Rosemarie Scott, Aaron Pigeon, and Jon Witte
Indeed, San Francisco
{benjaminl, elawrence, rosemarie, aaron, jwitte}@indeed.com

Introduction and Model Overview

At Indeed, we have spent over six years developing machine learning models to predict the fit between employers and jobseekers. These models substantially contribute to Indeed's core mission to help people find jobs, and we strive to ensure that predictions are both accurate and useful to jobseekers. In this talk we walk our listeners through a real world model error and describe how to identify, fix, and prevent bugs in models through rigorous prototyping and release processes, tools to help us interpret and visualize our model predictions, and by using human labels to confirm online results.

Fundamentally, our applicant quality models take a job description and a resume as input, and predict whether or not the candidate will be a good match. We try to solve this problem in a supervised way by using jobseeker outcomes as labels. The feature vectors consist of job-specific text features, resume-specific text features, and engineered interaction features. We serve >100M predictions per day at low latency to drive functionality in a number of Indeed products (see appendix 1).

Model Development and Release

Indeed uses a conservative approach to model builds and releases [1]. We use a consistent and repeatable model build tool that performs hyperparameter optimization, and generates performance metrics and diagnostic plots according to standard practices [2, 3, 4]. For applicant quality models in particular, we also evaluate models on multiple holdout sets to ensure performance does not degrade over time or across different data distributions.

These reports, as well as project-specific analyses, are easily recorded by the data scientist in a model build ticket. Once the prototyping phase is nearly complete, a peer data scientist will review the model build and provide feedback. This process is similar to code review for software engineering, but is focused on data sources, features, training process, and the production use case. See appendix 2 for an example template.

Models are then soft-released to production: predictions are logged for a small share of traffic but do not influence the user-facing behavior of the product. Production model predictions are compared to actual outcomes and offline performance. Finally, the new model is enabled in the product and A/B tested using standard testing methodology.

Debugging

While a careful model release process will prevent many bugs, there may be unexpected errors. As an illustrative example, we describe the tools and methods used to detect and fix various forms of "job overmatching". Since most jobs have multiple applicants, our datasets typically contain far fewer distinct jobs than distinct resumes. Models can potentially over-index on job-specific features and may lead to predictions that, while correct in a technical sense, are not useful or personalized to the specific jobseeker or employer.

Some of the tools and processes that we have used to debug this problem in various contexts include:

- Creating effortless access to individual production data samples, predictions, and explanatory visualizations (see appendix 3).
- Hand labeling many application pairs, especially with specific feature combinations to get more fine-grained data
- Designing specific metrics and diagnostic plots to detect discovered problems, and adding them to our standard workflow to prevent regressions.
- Soliciting feedback from users about the quality of model predictions, and using these to assess whether jobseekers find the predictions useful, not just accurate.
- Tracking model degradation across train, validation, test, and holdout sets in model build reports.

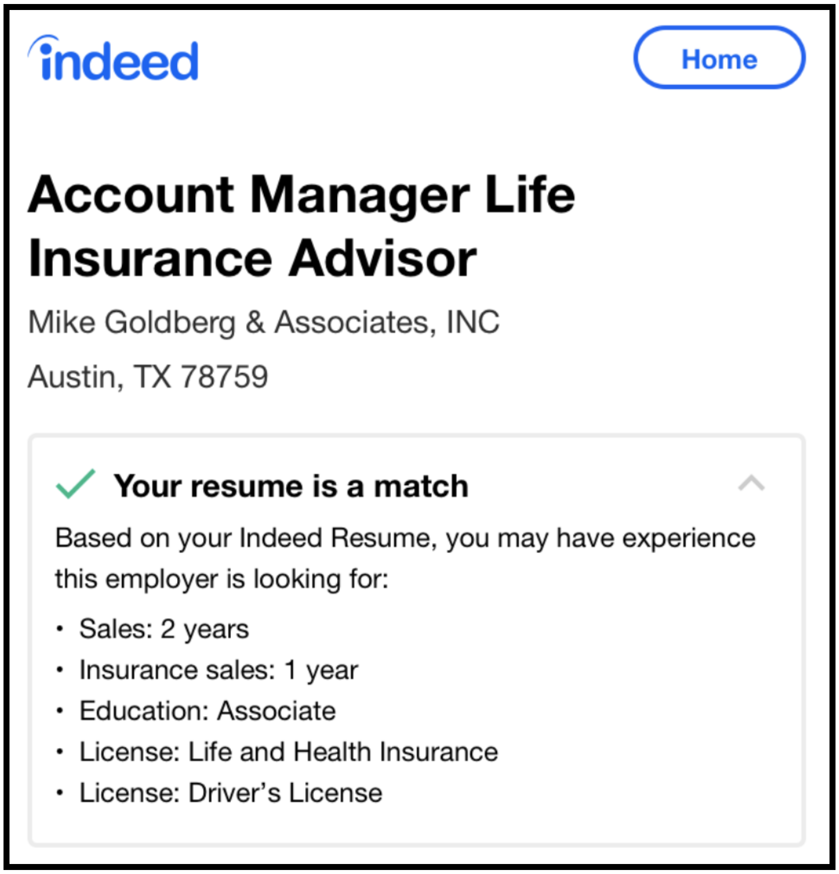
References

[1] Benjamin Link. 2017. From Data to Deployment: Full Stack Data Science. https://engineering.indeedblog.com/talks/data-to-deployment/
[2] Christopher M. Bishop. 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg.
[3] Andrew P. Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recogn. 30, 7 (July 1997), 1145-1159. DOI=http://dx.doi.org/10.1016/S0031-3203(96)00142-2
[4] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in Machine learning systems. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15), C. Cortes, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 2. MIT Press, Cambridge, MA, USA, 2503-2511.

Appendix 1

Resume Match

When we predict that a jobseeker is a match for a job, we show "Your resume is a match" and relevant job requirements on the job page. The model evaluation is run in real-time with the page request as job and resume data can change quickly over time.



Appendix 2

Model review template

Model prototyping results are tracked in a ticket in the company-wide issue tracker. Once a model is ready for review, data scientists insert a template and fill out relevant aspects of their model build (how data was collected, what it is predicting, performance metrics and plots, and so on). A peer data scientist will then review the model for production readiness, and provide suggestions and feedback.

[redacted] added a comment - 2 minutes ago

Stage 0 - Before Project Starts

Please review this graph to get familiar with model building workflow

Please review this Lucid chart to get familiar with collaboration process with engineering team, if model implementation requires engineering support

Stage 1 - Data Review + Code Review

Context

What should this model enable us to do (highlighting, filtering, sorting, etc.)

Are there any similar, related, or prior models or tools? What does this model do differently?

What products / interfaces / workflows will use this model

How will stakeholders evaluate this model? What is the impact of incorrect predictions?

What areas of machine learning does this model involve?

Response variable

Response variable and how it was labeled or collected

How reliable do you think the labels are? How easy is it to acquire more labels?

What the model outputs (predictions) represent and how they should be scaled or thresholded

What metrics are relevant to assessing the model?

Data sampling

What queries and filters were used and over what time range the data was sampled

What is the source of the data? How was it collected? Is there a team maintaining this data source? Are there known/documented inaccuracies or limitations? Was there anything unusual or unintuitive about the data? (Please update the data documentation if you find anything.)

Any manual up or down sampling that you did outside of Moody (e.g. downsampling advertisers with a ton of associated applies)

Will the resampled data (including up/down-sampling) match the distributions in production?

What are the dimensions of training dataset? (how many features and observations)

Appendix 3

Data Viewer and Model Explainer

Internal webapps allow immediate access to individual instances of production job and resume pairs. These webapps can be configured to simply show text for hand labeling, or can provide model predictions, feature importances, and perturbative model explanations.