Predicting Job Security via Bayesian Neural Networks

Morgan McCarty; Thomas McBride

Introduction and Literature Review

Problem Statement

Since the end of the tech boom in the early 2020s with the dawn of COVID-19, job security has reduced substantially. During the pandemic, companies hired an exorbitant number of tech workers to capitalize on the surge in profitability achieved when the populace switched to conducting their lives and business ventures online. In addition to surging consumer activity, companies saw the opportunity to hire past their office capacity given the option of work-from-home policies [1].

The rapid economic growth of tech companies was followed by a sharp decline. The lingering effects of the pandemic drastically increased inflation and interest rates. To mitigate losses, tech companies conducted massive layoffs that are still occurring today. Technologically experienced individuals are no longer guaranteed to find positions in the technical sector, and even when they obtain a job, the increasing economic instability has done little to reduce the chances of a layoff.

This volatile era in the job market for software developers is especially troublesome for those new to the profession, such as recent college graduates or self-starters with little-to-no work experience. Those who chose to study computer science for its bountiful job outlook near the pandemic are now entering a far different job market from what they originally planned it to be.

In this project, we propose a tool that, when provided with characteristics of a company - its operational stage, location, funds raised, industry, etc. - provides a percentage corresponding to the level of job security that the company is predicted to have. We implement a Bayesian Neural Network that provides a conditional distribution of our weight parameters given training data. This allows us to monitor the uncertainty of weight parameters and, additionally, helps to mitigate overfitting by representing weights as distributions representing uncertainty.

Past Work

Traditional Methods for Making Economic Predictions

Machine learning has long been a method used by economists to perform research. As we approach a relatively more sophisticated supervised learning technique, there are more fundamental machine learning techniques to consider. We found no relevant literature while researching machine learning methods to predict job security. However, we were able to find literature discussing techniques used to make general economic predictions. Bayesian Neural Networks have the potential to provide more clarity than conventional Multilayer Perceptions since they provide us with the amount of uncertainty associated with parameter estimations and predictions.

Lasso Regression

Lasso regression is a variant of linear regression created by adding a penalty term obtained by taking the L1 norm of the weight vector. This is done to mitigate the “blowing up” of coefficients that can lead to overfitting. In economics, Lasso regression is often used to make optimal predictions using a small subset of highly covariate big data [2]. This is useful in macroeconomic forecasting, where the goal is to make estimations of the future of the national

economy [2]. Another specific application of Lasso regression was used to predict community mental health center employees' turnover, which is a similar supervised learning problem [3].

Random Forest Regression

Random Forest Regression is a supervised machine learning technique that utilizes multiple random decision trees, averages their results, and makes a singular prediction for a target value [4]. Random Forest Regressors can be very accurate for data with non-linear relationships but are also susceptible to overfitting, especially without hyperparameter tuning. One advantage of Random Forest Regression is the interpretability of the results. Random Forest Regressors can be used in similar supervised learning problems to the one we pose for this project [3].

Methodology and Data Collection

As mentioned previously, in this experiment, we implement a Bayesian Neural Network to predict the proportion of a company's workforce that has been laid off, using features from the "layoffs.fyi" data set.

Data Preparation

For feature selection, we remove the "company" feature that contains the companies' names since the companies' names should be unrelated to our objective. We also decided to encode the "date" feature as an integer representing the number of quarters past January 2020. This was done to reduce the number of features after encoding so that we could train our models within a reasonable time frame and make the feature more temporally expressive. As for the "stage" feature, we decided to chunk stages into less descriptive groups in an additional attempt to reduce the feature map after encoding. The "stage" feature was a feature we predicted to be

highly informative for layoff prediction, so we decided to leave the feature as it was. Finally, we removed all observations with missing data since the data set was large and sparse. After encoding, we had a feature map of 69 columns derived from the original feature map of 9 columns. 2 of these columns represented our target feature, either as a percentage or a real number value, and 67 columns represented the observed features.

Multilayer Perceptron

To compare our Bayesian Neural Network to a baseline metric, we implement a simple, 2-hidden-layer, Multilayer Perceptron in R. This way, we can compare the final weight parameters to identify similarities and differences in predictions and performance.

Architecture and Training

The Multilayer perception is implemented manually in base R. We create the network with 2 hidden layers with ReLU activation. The output layer has a sigmoid activation function to support our domain of 0 to 1 needed for predicting our target. Even though sigmoid activation is typically used for classification, as this is not a classification problem, we use the Mean Squared Error loss function as our objective. The weights of our model are trained using gradient descent with an eta of 0.05.

Bayesian Neural Network

Our Bayesian Neural Network is implemented in Python using the PyMC library. The BNN has two hidden layers with ReLU activation and weights with a standardized normal prior distribution. These weights, along with the training data, create our “mu”, which represents the predicted average layoff rate. Our “phi” parameter is given a Gamma prior with fixed alpha and beta, which allows us to shape our Beta likelihood. We use the Beta likelihood to model our

expected target value distribution. We choose the Beta likelihood for its support across the (0,1) domain. We drew 10000 samples from the model after tuning for 1000 samples.

Weights

$$w_1 \sim N(0,\ 1),\ dim\ =\ 67\ \times\ 8;\ \ w_2 \sim\ N(0, 1),\ dim\ =\ 8\ \times\ 1$$

Hidden layer; output layer; predicted average

$$h\ =\ \sigma(Xw_1),\ \ o\ =\ h\ \cdot\ w_2,\ \ \ \mu\ =\ \sigma(o)$$

Standard deviation prior

$$\sigma \sim HalfNormal(1)$$

Data likelihood

$$y\ \ \sim TruncatedNormal(\mu,\ \sigma; 0, 1)$$

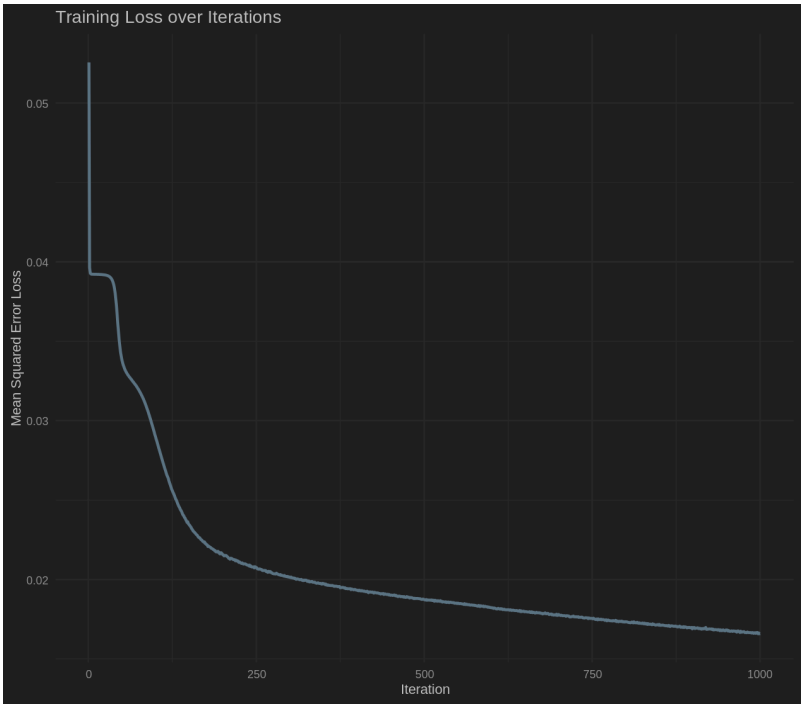
Results and Interpretation

Multilayer Perceptron

The Multilayer Perceptron performs moderately well, with a R^2 score of ~0.6 across 1000 gradient descent iterations (single sample iterations). When looking at sample predictions for the target, there are a few predictions that are close to the target value.

	Predicted	Actual
1	0.096	0.03
2	0.164	0.14
3	0.183	0.28
4	0.189	0.20
5	0.164	0.05
6	0.127	0.14
7	0.134	0.14
8	0.320	0.15
9	0.228	0.40
10	0.280	0.37

Sample predictions from MLP

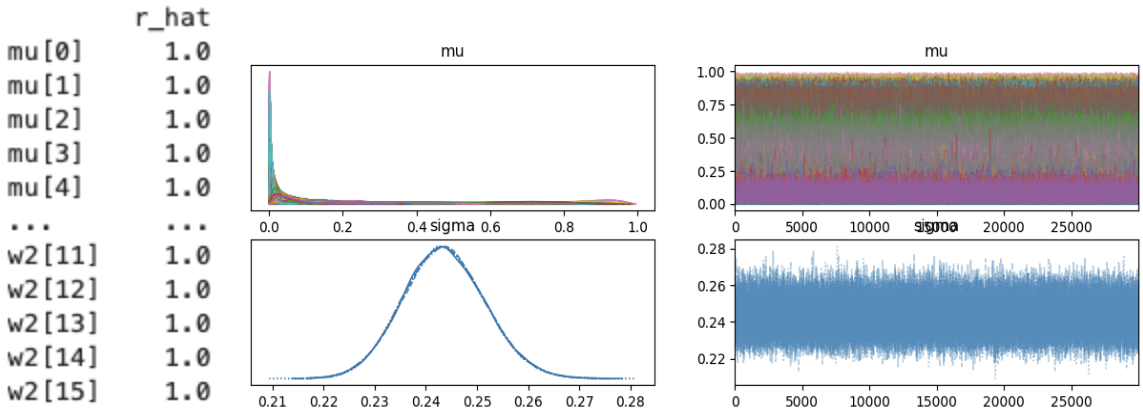


Loss across training intervals for MLP

As for the final model weights, there are far too many features to conduct a meaningful analysis, given our resources and time frame for the experiment. However, given our results, we deduce that there is some significant correlation between the explanatory variables and the target. We also note that the training curve of the model follows a very strong downward exponential curve, which is expected of a well-trained model. With more training iterations and a much larger dataset, it may be possible to improve the model significantly. Given the small size of our dataset, however, not much improvement beyond this is possible.

Bayesian Neural Network

The Bayesian Neural Network is unable to reliably predict the target, with a R^2 score of ~0. However, what differentiates the Bayesian Neural Network from the Multilayer Perception was its expressiveness in terms of describing uncertainty for posterior predictions. When observing posterior predictions, it is shown that the model is unsure of its predictions. Below is the summary of the Bayesian Neural Network.



Chains seem to be mixing well, with high, but incorrect confidence in mu and sigma

We note that the model appears to be mixing well as the draws for each weight are consistent (and consistently different from other weights), additionally, they are not significantly noisy. The sigma draws are also mixing well, as there is little extreme variation throughout all of our draws.

Conclusions and Future Work

By conducting this experiment, we were able to explore a relatively new, interdisciplinary technique in Bayesian Neural Networks and compare it to a more traditional technique in Multilayer Perceptrons. While we were unable to predict layoff percentages for companies reliably, we were able to conclude that layoff data is noisy and volatile. While conducting our research, we learned of the uncertain nature of company hiring and firing patterns, especially during the COVID-19 pandemic. Our results, even using tried and true prediction techniques, match what the research indicates.

Our choice of data was likely not fit for our use case due to the prevalence of missing values, which could indicate noisy and inaccurate data. After cleaning our data, we lost around ⅔ of our original number of observations, which was a significant amount of data lost. For

complicated supervised learning tasks such as this one, more observations would have been useful, especially for the Bayesian Neural Network.

The tried and true Multilayer Perceptron has proven to be an easy and effective technique for supervised learning, generating better results than the Bayesian model, with significantly less training time.

In the future, more analysis and experimentation with different priors is necessary for the success of our BNN. Bayesian Neural Networks can be highly biased/inaccurate when the priors and likelihoods used for the data and parameters are inaccurate to their actual behavior. While the truncated normal distribution matches our domain theory, perhaps the use of a different likelihood would have been more appropriate for modeling layoff trends. Our choice of priors and likelihoods for the parameters of our BNN was likely suboptimal, which could lead to bias in our posterior predictions. We continue to hold the belief that Bayesian Modeling can be used for this supervised learning task.

References

[1]K. Bindley and J. Pisani, “Tech Jobs Have Dried Up—and Aren’t Coming Back Soon,” WSJ, Sep. 19, 2024. https://www.wsj.com/tech/tech-jobs-artificial-intelligence-cce22393?mod=Searchresults_pos10&page=1 (accessed Mar. 29, 2025).

[2]S. Memon, “Machine Learning for Economists: An Introduction,” *The Pakistan Development Review*, vol. 60, no. 2, pp. 201–211, 2021, doi: https://doi.org/10.2307/27262008.

[4]S. Fukui et al., “Applying Machine Learning to Human Resources Data: Predicting Job Turnover among Community Mental Health Center Employees,” The journal of mental health policy and economics, vol. 26, no. 2, p. 63, Jun. 2023, Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC10424701/

[4]S. Mullainathan and J. Spiess, “Machine Learning: An Applied Econometric Approach,” The Journal of Economic Perspectives, vol. 31, no. 2, pp. 87–106, 2017, Available: https://www.jstor.org/stable/44235000