Predicting Yelp Ratings from Review Sentiment & Popularity Using Bayesian Regression

COMPSCI 179: Graphical Models - Final Project Report

By Rohan Mistry and Cole Thompson
June 2025

1. INTRODUCTION

Online reviews offer a vast amount of data about consumer experiences, and platforms like Yelp use aggregated star ratings as an indicator of business quality. However, we weren't sure whether star ratings alone truly capture all the nuance and variability of written feedback. In this project, we explored whether a restaurant's average Yelp rating can be effectively predicted using two features: the *average sentiment score* derived from its written reviews and the *log-transformed review count*, to capture both popularity and the consistency of ratings over many reviews. We implemented a *Bayesian linear regression model* that estimates the strength and uncertainty of these relationships, and evaluated how well it predicts star ratings and interprets underlying trends.

2. RESOURCES AND IMPLEMENTATION

a. DATA SOURCES

We used Yelp's Open Dataset [1], specifically `yelp_academic_dataset_business.json` (business metadata, specifically the stars, review count, and categories features) and `yelp_academic_dataset_review.json` (written reviews and their associated star ratings). From this, we then analyzed all the frequencies and statistics from the raw data to accurately tune various filters which we applied so that filtered businesses would only include California restaurants (`state == "CA"` and `"Restaurants"` in `categories`) with 50 to 1,000 reviews (`50 < review_count < 1000`). This ended up returning 684 restaurants and 157,916 reviews which seemed like a more than reasonable enough dataset size for us to complete our project. From this, we processed the raw Yelp JSON files in a script we made (`filter_city_data.py`), where we calculate the VADER sentiment scores using NLTK's `sentimentIntensityAnalyzer` [2] for each review, aggregate to compute `avg_sentiment_score`, and compute `log_review_count`. The resulting dataset was exported to (`ca_restaurants_bayesian_dataset.csv`), containing columns `business_id`, `name`, `avg_rating`, `avg_sentiment_score`, and `log_review_count`.

b. TOOLS & LIBRARIES

- Data & Preprocessing: `pandas` [3], `numpy` [4], `nltk` [5], `json` [6]
- Modeling: `pyro` [7], `torch` [8]
- Visualization: `matplotlib` [9], `seaborn` [10]

c. CODE

All Bayesian modeling and evaluation code lives in our Jupyter notebook (`project_code.ipynb`). We essentially just adapted the model structure and inference workflow directly from Pyro's Bayesian Regression example [11], which is why our `BayesianYelpRegression` class and our optimizer's [12] code look nearly identical. We did opt to replace their `AutoDiagonalNormal` guide with `AutoNormal` [13] and changed the σ prior to a `HalfNormal` distribution. The train/test split, evaluation code, and custom visualization plots (although very heavily inspired from the Pyro documentation) were written by us.

3. EVALUATION AND FINDINGS

a. DATA PREPARATION & TRAIN-TEST SPLIT

The dataset was loaded from the CSV file into a `pandas DataFrame` and we extracted the columns `avg_sentiment_score` and `log_review_count` as the features along with `avg_rating` as the target. These were then converted into tensors `X_sentiment`, `X_log_reviews`, and `y_all` to be used as direct input for our Pyro model. We randomly split the dataset: 80% for training and 20% for testing - predictions on test data were made using posterior mean estimates of the parameters.

b. MODEL SPECIFICATION

We defined our Bayesian linear regression model using the following formal definition:

$$r_d \sim \mathcal{N}\big(\alpha + \beta_s \cdot s_d + \beta_r \cdot \log(1 + \text{review\_count}_d), \sigma^2\big)$$

where r_d is the average rating of the restaurant d, s_d is the average compound sentiment score of its reviews, and $\log(1 + \text{review\_count}_d)$ is its log-transformed review count. We assigned *prior distributions* to each parameter: global bias (intercept) α , drawn from $\text{Normal}(0, 10)$; regression coefficients β_s and β_r, both drawn from $\text{Normal}(0, 1)$; and *observation noise* standard deviation σ, drawn from $\text{HalfNormal}(1)$. We used a `HalfNormal` prior for σ rather than Uniform to ensure that the noise was larger than zero while allowing for heavier concentrations near zero. We believe that the rating noise is likely small but can grow if the data requires it. The model was implemented in Pyro

using a `PyroModule[torch.nn.Linear]` layer [7] with the priors placed directly on the weight and bias parameters.

c. INFERENCE METHOD & TRAINING

The model was fitted using *stochastic variational inference (SVI)* with Pyro's built-in `AutoNormal` guide [12] for simplicity. We then trained the model with the `Adam` optimizer (with learning rate $\eta = 0.01$ which gave us stable *evidence lower bound (ELBO) convergence*) and the built-in loss `TRACE_ELBO` [14] over 5,000 steps. The stable loss value decrease from over 5,000 to below 80 indicated successful convergence of the variational parameters. Our input matrix of shape `[train_size, 2]` and `y_train` was constructed by combining the two feature tensors.

d. EVALUATION METRICS

After training, we ended up drawing 1,000 samples from the approximate posterior using Pyro's `Predictive` class [15]. For each model parameter, we computed its *posterior mean* and 95% *credible interval* (2.5th and 97.5th percentiles) to quantify both the strength and uncertainty of each parameter's effect. The metrics ultimately ended up suggesting that the sentiment coefficient β_s had a strong, positive effect (indicating that higher sentiment scores generates higher ratings) with low uncertainty, while the popularity coefficient β_r had a very weak, negative effect (indicating that ratings are actually slightly lower for more popular restaurants) with consistency. Additionally, the observation noise σ results affirm the model's confidence because its predicted ratings (rounded in half-star increments to match Yelp's rating scale) were typically within +-0.25 stars of the actual rating.

e. VISUALIZATIONS

To better understand the behavior and predictive performance of our model, we produced three key visualizations: (I) a regression curve with uncertainty, (II) a posterior predictive check, and (III) a comparison of sentiment effects for different popularities.

(I) Regression Line with 95% Credible Interval

To visualize how sentiment alone affects predicted ratings, we generated a posterior predictive curve over a range of sentiment scores while holding the log-transformed review count fixed at its median value. For each grid point, we drew 1,000 samples from the model's predictive distribution and calculated a 95% credible interval. We plotted this curve and uncertainty band on a scatter plot of restaurants. The plot revealed a strong, positive correlation between average sentiment score and predicted Yelp rating, which is consistent with the large posterior weight we observed for the sentiment feature. The

narrow width of the credible interval also reflects the model's strong confidence in this relationship. See Figure (I).

(II) Posterior Predictive Check on Test Ratings

To assess the quality of the model's predictions on unseen data, we performed a *posterior predictive check* on the test set. We generated 1,000 samples from the predictive distribution for each test example and averaged them to obtain expected ratings `y_pred_mean`. These were then rounded to the nearest half-star to match Yelp's rating scale. We plotted side-by-side (or rather, overlapping) histograms of the predicted vs. actual test ratings. The comparison shows that the distributions aligned very closely, suggesting that the model captured the structure of the data and generalized well beyond the training set. Notably, the predicted ratings covered the same shape and variance as the observed ratings, with no significant overestimation or underestimation biases. See Figure (II).

(III) Density of Sentiment Effect at Low vs. High Popularity

We also explored whether the impact of sentiment on predicted ratings varied depending on a restaurant's popularity. To do this, we evaluated the combined effect of sentiment and popularity using posterior draws from the model, by calculating $\text{Effect} = \beta_s + \beta_r \cdot \log(1 + \text{review count})$. We compared this effect for the least popular and most popular, plotting histograms of the two resulting posterior distributions. The charts show that sentiment had a slightly stronger influence for low-popularity restaurants (which doesn't necessarily mean a whole lot). This reflects the negative posterior mean of the popularity coefficient found earlier but because the two curves lie almost atop each other - peaking at about 2.92 for low-popularity and 2.83 for high-popularity - this indicates that popularity has virtually no impact on the sentiment effect. In other words, restaurants with fewer reviews get a slightly bigger rating lift from the same sentiment boost, but this difference is very minor. See Figure (III).

4. CONCLUSION

This project demonstrates that a Bayesian linear regression model can effectively predict average Yelp ratings from textual sentiment and (kind of) popularity. Our results showed that sentiment is a strong and reliable predictor of rating, while popularity has a much smaller effect. The model achieved accurate predictions with low uncertainty, and posterior predictive checks confirmed strong generalization to unseen data. Overall, this approach demonstrates how simple probabilistic models can yield super interpretable insights and robust performance in real-world text-driven datasets.

Appendix A: FIGURES

Figure (I): Regression Line with 95% Credible Interval

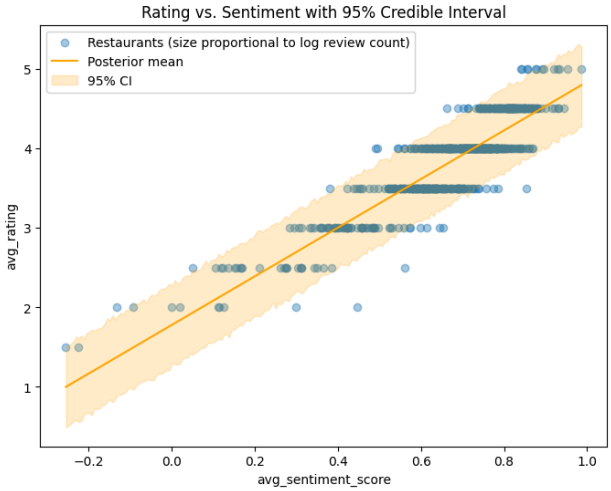


Figure (II): Posterior Predictive Check on Test Ratings

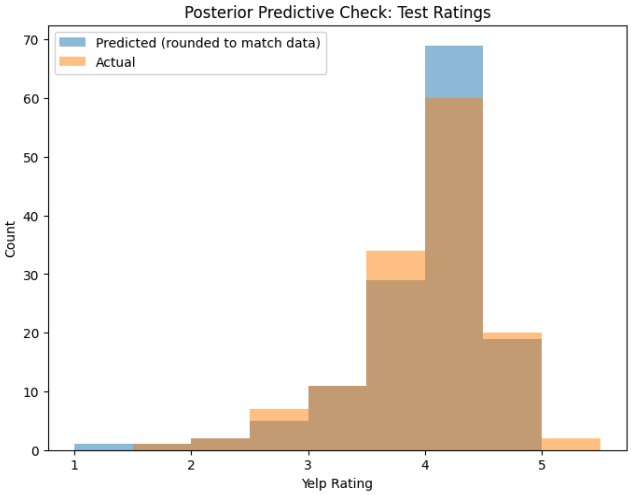
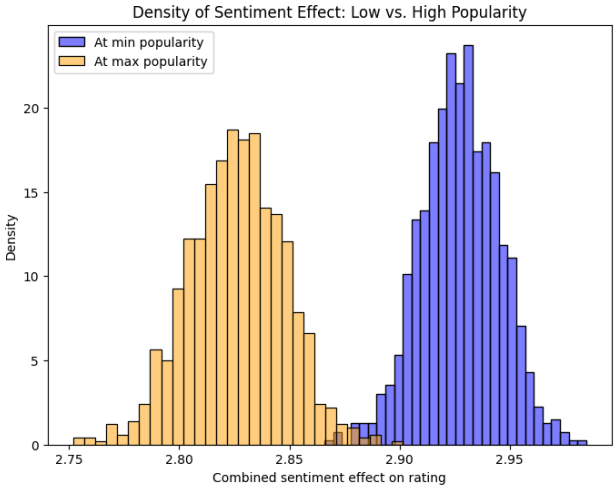


Figure (III): Density of Sentiment Effect at Low vs. High Popularity



Appendix B: REFERENCES

[1] Yelp Inc., "Yelp Open Dataset," Yelp Data Resources. [Online]. Available: https://business.yelp.com/data/resources/open-dataset/

[2] C. J. Hutto and E. E. Gilbert, "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text," in *Proc. 8th Int. Conf. on Weblogs and Social Media (ICWSM)*, Ann Arbor, MI, 2014. [Online]. Available: https://www.nltk.org/api/nltk.sentiment.vader.html

[3] The pandas development team, *pandas-dev/pandas: Pandas (v2.2.3)*, Zenodo, 2024. [Online]. Available: https://pandas.pydata.org/

[4] C. R. Harris, K. J. Millman, S. J. van der Walt, et al., "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://numpy.org

[5] S. Bird, E. Loper, and E. Klein, *Natural Language Processing with Python*. O'Reilly Media, 2009. [Online]. Available: https://www.nltk.org/

[6] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoc, "Foundations of JSON schema," in *Proc. 25th Int. Conf. on World Wide Web (WWW)*, 2016, pp. 263–273. [Online]. Available: https://www.json.org/json-en.html

[7] E. Bingham, J. P. Chen, M. Jankowiak, et al., "Pyro: Deep Universal Probabilistic Programming," *Journal of Machine Learning Research*, vol. 20, no. 28, pp. 1–6, 2018. [Online]. Available: https://pyro.ai/

[8] A. Paszke, S. Gross, F. Massa, et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, pp. 8024–8035, 2019. [Online]. Available: https://pytorch.org/

[9] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. [Online]. Available: https://matplotlib.org/

[10] M. L. Waskom, "seaborn: Statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. DOI: 10.21105/joss.03021. [Online]. Available: https://seaborn.pydata.org

[11] Pyro Development Team, "Bayesian Regression Example," Pyro Examples. [Online]. Available: https://pyro.ai/examples/bayesian_regression.html

[12] Pyro Development Team, “SVI: Stochastic Variational Inference,” Pyro Examples. [Online]. Available: https://pyro.ai/examples/svi_part_i.html

[13] Pyro Documentation, “AutoNormal Guide,” [Online]. Available: https://docs.pyro.ai/en/1.5.1/infer.autoguide.html#autonormal

[14] Pyro Documentation, “Trace_ELBO Class Reference,” [Online]. Available: https://docs.pyro.ai/en/dev/_modules/pyro/infer/trace_elbo.html#Trace_ELBO

[15] Pyro Documentation, “Predictive Class Reference,” [Online]. Available: https://docs.pyro.ai/en/dev/_modules/pyro/infer/predictive.html

[16] Pyro Documentation, “Distribution Classes,” [Online]. Available: https://docs.pyro.ai/en/stable/distributions.html