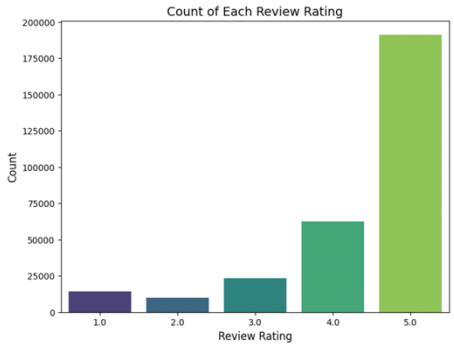
Predicting Vermont Business Ratings and Recommending Local Favorites: A Data-Driven Approach with XGBoost and SVD

1 INTRODUCTION

In the digital age, user-generated reviews play a critical role in shaping consumer decisions and business strategies. This project aims to leverage insights from reviews of Vermont businesses to address two key tasks: predicting the ratings for new businesses and developing a system to recommend businesses to users. These tasks are designed to bridge the gap between customer expectations and business performance while promoting stronger connections between local establishments and their patrons. To accomplish this, we analyzed a dataset containing review text, business metadata, and temporal information. The predictive task focuses on uncovering patterns in customer feedback to predict ratings for businesses, helping owners refine their services. The recommendation task aims to suggest businesses that align with user preferences by predicting ratings for businesses users haven't interacted with yet. This dual focus enables the creation of tools that benefit both businesses and consumers in the Vermont economic landscape.

2 DATASET

Initially, our dataset consisted of 852,203 reviews, but after data processing, it was refined to 301,311 reviews. In the refined dataset, there are 37,300 users, with each user averaging 8 reviews. The time of these reviews spans from 2007 to 2021. The trend in the count per rating as shown.

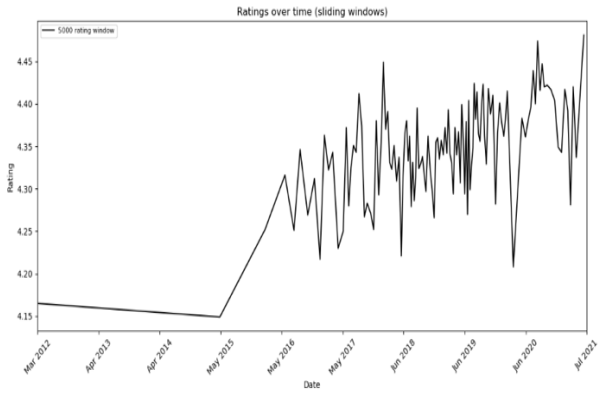


Clearly, our dataset is skewed toward higher reviews, but since it contains all business reviews from this time frame, this demonstrates that users often rate businesses in Vermont highly.

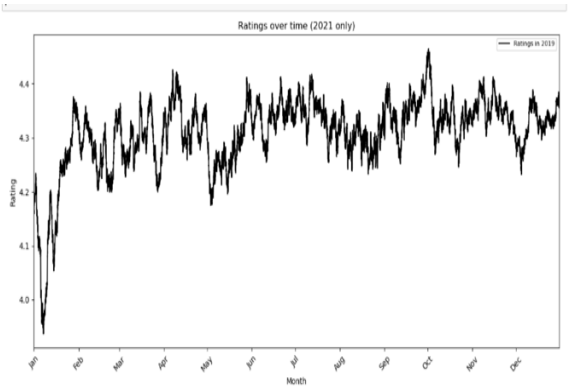
3 PREPROCESSING AND EXPLORATORY DATA ANALYSIS

Upon first glance of the data and metadata, we identified several features that we would not include, such as images and URLs. We removed entries that did not provide a text review or rating of the business. Since we planned to perform natural language processing on our dataset, ratings without accompanying text reviews were not useful to us. While these reviews might have improved the performance of our SVD model by providing more interactions, to accurately compare our models, we needed to ensure they all used the same data. Furthermore, we removed entries for users who left fewer than three reviews. By doing so, we effectively reduced the sparsity of the interaction matrix, mitigated the cold-start problem (as we focused on users with more interactions), and improved the reliability of our model.

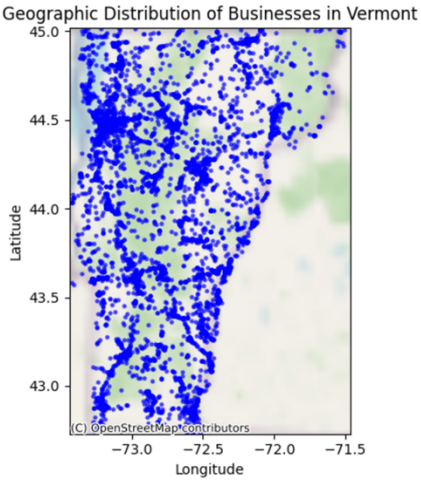
We wanted to examine features that we believed might impact the rating of a business, such as the time of the review, text features, and the location of the businesses. We performed an exploratory analysis of the impact of temporal features on the average rating. Using a moving average method with a window size of 1,000, we visualized trends over time in our dataset. Through experimentation with different window sizes, we determined that averaging over 1,000 points provided the best visualization of these trends.



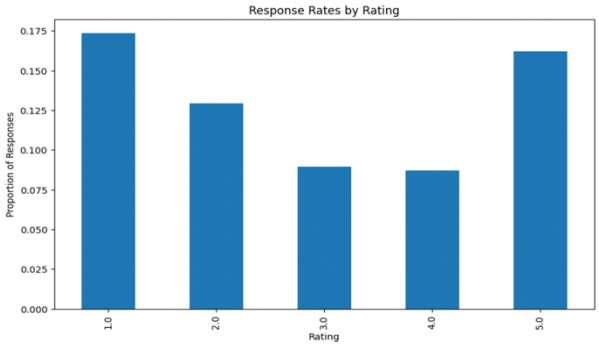
Within our chart of moving averages (across our entire dataset), we observed a significant drop in the average rating of business during the pandemic. We also wanted to focus on month-by-month trends to determine if there was a seasonal effect on ratings. To examine these trends, we analyzed a year-long subset of our dataset.



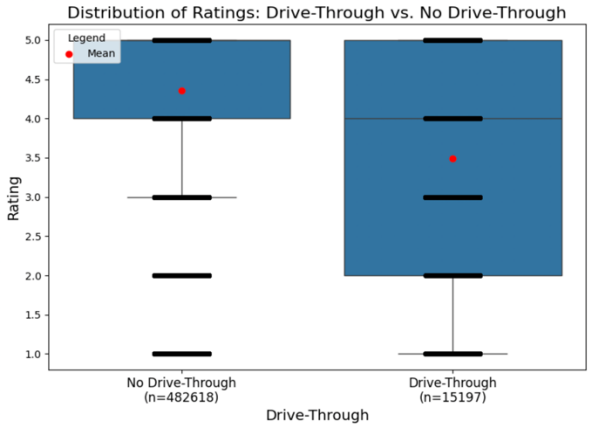
We noticed that there wasn't a clear trend between different seasons, however, the time of the review definitely influenced how the business was rated. We examined the geographical distribution of businesses in Vermont to identify any inconsistencies but found nothing unusual. We overlaid the geographical coordinates of the businesses on a map of Vermont and confirmed that our dataset provided adequate coverage of the state. As expected, there were clusters of businesses around larger cities.



We also conducted an exploratory analysis of the "resp" column, which represented a business's response to a user review. The average ratings of businesses that responded to reviews were slightly higher than those that did not. However, we also observed that businesses tended to respond only to extreme reviews—typically ratings of 1 or 5.



Ultimately, because it failed the assumption of independence and posed a risk of potential bias in our model, we chose not to include responses as a feature for training. Additionally, we explored the features "category" and "MISC" to determine their value for our predictive model. After experimenting with "category," we concluded that this feature had no significant impact on our model's predictive ability. Regarding "MISC," certain sub features of this category were found to have a significant impact on business ratings. However, due to inconsistencies in this column and the number of businesses lacking data for this feature, we decided not to include it in our models.



4 PREDICTIVE TASKS

Before training our models, we randomly split our data into a training and testing set using the scikit-learn train test split method. We kept eighty percent of our data for training and used the other twenty for testing. We further divided the train set to create a validation set as a means of evaluating performance prior to exposing our model to the test set.

4.1 ESTABLISHING BASELINE MODELS

For our rating predictions, we implemented simple baseline models for benchmarking. The first predicted the global average rating, yielding an MSE of 1.13749. We improved this by predicting each business's average rating, reducing the MSE to 0.97110. These benchmarks set a foundation for evaluating future models.

Our initial focus was recommending new businesses to users using two models: latent factor models and factorization machines. Both rely on a user-item interaction matrix, but latent factor models focus solely on user-item interactions, while factorization machines incorporate additional features.

4.2 SVD RECOMMENDER

We implemented a latent factor model using the SVD algorithm from the Surprise library, leveraging its effectiveness in detecting user preferences without additional features. It uses a matrix to account for user and item biases while capturing preferences, as outlined in the equation below:

$$f(u,i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$$

Training on our dataset yielded an MSE of 0.90450 on the test set.

Using GridSearch, we optimized our model's hyperparameters, including the number of factors, iterations of stochastic gradient descent, and regularization terms: GridSearch attempts a variety of hyperparameter combinations and cross validating these combinations to achieve the best results. Although we should have included more hyperparameter tuning, we were limited by processing power as even just our hyperparameter combinations GridSearch took a significant amount of time to run. After tuning its hyperparameters, we tested our model and got an MSE of 0.895954.

Even without any additional features about the data, a latent factor model can discover a latent dimension corresponding to the different features. However, latent factor models struggle when users have not interacted with many businesses which is why we decided to remove users who reviewed less than three businesses to help combat this issue.

Latent factor models can identify latent dimensions without additional features but struggle with users who have few interactions. Hence, why users with fewer than three reviews were removed.

4.3 IMPLEMENTING FACTORIZATION MACHINES

Based on our EDA, we knew features like temporal data could impact what rating a user would give a business, so we implemented a factorization machine. Unlike latent factor models, factorization machines incorporate additional features alongside user-business interactions, addressing sparse user data. This helps in situations where a user hasn't reviewed many businesses, as solely observing their

interactions may not tell us much about their preferences. Using the fastFM and lightFM libraries, we found the models underperformed on the test set, likely due to implementation issues. While our approach didn't succeed, factorization machines should theoretically match or surpass latent factor models. A neural network-based factorization machine could have been a promising direction, but time constraints limited our efforts.

4.4 LATENT FACTOR EXTRACTION

We also tried to extract the latent factors from our SVD model and use them along with other features in a regression model. We extracted the user latent factors and item latent factors from our best performing SVD model and combined them with other features such as time, location, average ratings, and number of reviews. We then trained a regression model on this feature matrix but found that it performed worse than the SVD model. However, this approach did perform better than both of our baselines. We didn't find much research on this technique, but it could be interesting to explore further how combining extracted features from multiple models can impact performance.

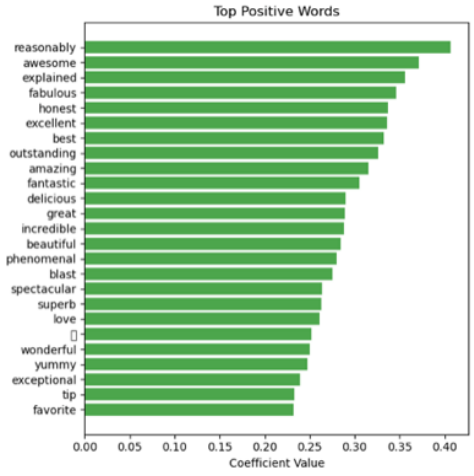
4.5 FROM PREDICTIONS TO RECOMMENDATIONS

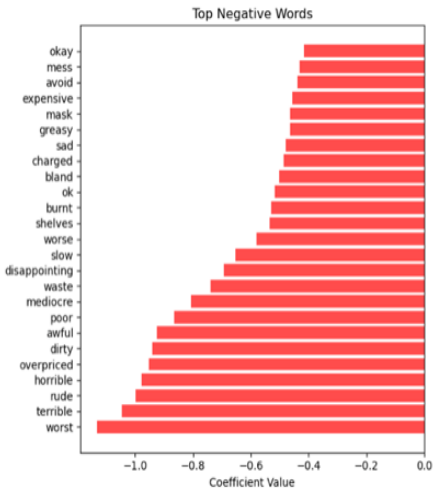
After determining that our best text independent model was the surprise latent factor model implementation, we still needed to incorporate it into recommending new businesses to a user. We created a function that took a user's id as input, and then found all the businesses in our dataset that the user had not rated. We then used our SVD model to predict the rating that the user would give to that business. After making these predictions, we just needed to sort these businesses by rating and return the nth highest rated businesses. To make this more practical we added a location parameter to the function so that the user can filter to only businesses within a certain range of their given location.

There are potential issues with this system we should acknowledge. Some might argue against using the MSE metric to rate model performance since MSE treats all errors the same. Since we are using the high predictive ratings to recommend new businesses, it has been argued that it is more important to accurately predict these ratings than low ratings, as our system wouldn't recommend low rated businesses to the user. Additionally, because of the nature of our dataset and model, our recommendation may not perform as well for users that have rated very few businesses.

4.6 TEXT-BASED RATING PREDICTIONS USING REGRESSION AND XGBOOST

Aside from making accurate business recommendations to a user, we were also interested in how we could use the text of a review to more accurately make predictions. Naturally, the text of a review tells us a lot about how a user feels about a particular business. We started by performing basic sentiment analysis to see what features are associated with positive and negative ratings. To prepare the reviews for our model, we removed punctuation and converted every word to lowercase. We experimented with stemming and removing stop words but saw that doing so actually decreased the performance of our model. We then reduced our dictionary to only include the first thousand words. We experimented with different dictionary sizes but ultimately found that only using the 1000 most common words resulted in the best performance.





$$\text{rating} = \theta_0 + \sum_{w \in \mathcal{D}} \text{count}(w) \cdot \theta_w.$$

We were able to construct a simple regression model like the one outlined in the equation above and found that it was able to better predict ratings on the test set than the latent factor model and had a MSE of 0.6859. This demonstrates how powerful text data is for predicting someone's opinions. We also extracted the coefficients for our model and mapped them to the words, so we could see which words were making the most significant contribution to the rating. We noticed that words such as “awesome”, “honest”, and “fantastic” have a significant positive impact while “mess”, “avoid”, and “expensive” have a significant negative impact. This follows what we would assume a rating would be based on the words it contains. We also experimented with using n-grams to capture basic syntax. The bag of words model is limited as it may struggle to capture interactions between words by separating phrases like “not good” into “not” and “good” which clearly alters the meaning of the phrase. After implementing a bigram however, we noticed that it did not substantially improve our model performance and was not worth the increase in dictionary size. We decided to proceed with the bag-of-words representation for the review text, but wanted to see how we could make improvements past the simple linear model and incorporate additional features.

The features that we thought would be interesting to incorporate into our predictions based on our EDA were temporal data, average ratings for the businesses, number of reviews for the businesses, and location data. We extracted all these features for each business and combined it with the text features from our bag-of-words extraction. We started by feeding these features into a ridge regression model which resulted in an MSE of 0.6349580. Despite the improved results, we still wanted to experiment with other regression-based models that could better capture the complexity of our feature matrix.

We knew we had to implement a regression model because of our goal to make predictions based on features. We explored the use of XGBoost, a powerful library commonly used for datasets with complex structures such as ours. XGBoost Regression is a form of regression that uses gradient boosting to improve prediction performance iteratively. The algorithm begins with a baseline prediction for the target variable and calculates the residuals. It then fits a series of decision trees to the residuals, with each tree learning to correct the errors of the previous trees. The model combines the predictions of all the trees, optimizing them to minimize the residuals and produce the final prediction. This approach leverages gradient boosting to create a highly accurate and efficient regression model.

This model is an industry for regression, “Take the challenges hosted by the ma- chine learning competition site Kaggle for example. Among the 29 challenge winning solutions published at Kaggle’s blog during 2015, 17 solutions used XGBoost.” (Chen 3). After training it on our train set and performing GridSearch to optimize parameters, we found that it performed better than every model that we have tried thus far with a MSE of 0.5737450 on the test set. This was a significant improvement over the baseline model, SVD model, and linear regression model using only the bag-of-words features.

Ultimately, our XGBoost model, leveraging gradient boosting and optimized feature integration, demonstrated significant

improvements over collaborative filtering and regression approaches, solidifying its utility for predictive tasks in this domain.

5 LITERATURE

With our project focusing on predicting and recommending new Vermont businesses to users utilizing data from Google Local Reviews 2021, this section compares our dataset with similar datasets in past literature and evaluates state-of-the-art methods for solving similar problems. We researched similar datasets to examine literature relating to our task and found the Amazon Reviews 2023 dataset, which similarly contains user reviews for Amazon products. A key difference between the two, however, is that our dataset includes geospatial coordinates, which allow us to examine how users interact with businesses in specific areas.

The Amazon Reviews 2023 dataset includes 570 million reviews and 48 million items across 33 categories. A 2023 study by McAuley, Hou, Li, He, Yan, and Chen introduces BLAIR, a series of pre-trained sentence embedding models to enhance recommendations. The BLAIR framework links user-generated text with item metadata, allowing the model to make predictions based on a user's product search. While our models did not consider user input in recommendations, exploring how to use review metadata to recommend businesses based on a user's search could be an interesting direction to pursue.

Both our recommender system and BLAIR utilize user-item interactions for personalized recommendations. BLAIR uses text embedding techniques to align user reviews with item metadata, which helps bridge language and items. To adopt a similar approach, we would need to analyze user reviews alongside business metadata for search-based recommendations.

While our SVD model achieved a best MSE of 0.895954 after hyperparameter tuning, further improvement could come from reducing reliance on latent factors. Given our focus on Vermont business reviews, we could explore BLAIR techniques, such as generating text embeddings that align with metadata and experimenting with contrastive learning.

Focusing more on our task of predicting ratings, the study by Viard and Fournier-S'niehotta (2018), Movie Rating Prediction Using Content-Based and Link Stream Features, explores how XGBoost can be applied to recommender systems. By combining content-based features and modeling user interactions through link streams, which represent dynamic relationships over time, the study aligns with our project's goal of predicting ratings and recommending items using review metadata and temporal features.

The study utilized the MovieLens 20M dataset, which included release years, movie genres, and two types of interactions: (1) users rating movies at specific times and (2) users assigning textual tags to movies at specific times. Features were engineered to incorporate content-based information and temporal dynamics, with link streams detecting patterns in user interactions. These features were integrated to predict user ratings using XGBoost, a state-of-the-art machine learning tool. Similarly, we incorporated features like review text, number of reviews, and average ratings in our model. However, our XGBoost implementation lacks dynamic temporal modeling, as it does not utilize sequential time interactions.

Both our project and the study demonstrate that XGBoost regressors significantly improve performance over traditional collaborative filtering models. These regressors effectively handle complex and diverse data by combining multiple types of features. The study achieved a minimum RMSE of 0.7652, outperforming other models in recent literature. Our XGBoost regressor achieved an MSE of 0.5737450 on the test set, outperforming the other models we investigated.

The sparsity of user reviews in our dataset (after filtering out users with fewer than three reviews) impacts the performance of our recommendation models, particularly collaborative filtering methods like SVD. The movie ratings study addressed sparse data by dynamically modeling

temporal interactions through link streams, enabling the system to predict ratings based on evolving patterns. Incorporating temporal sequences or encoding time-sensitive patterns into our XGBoost model could help mitigate the limitations caused by sparse data.

6 RESULTS

To conclude, our study developed recommender systems such as SVD and factorization machines and created an XGBoost regression model with feature engineering to predict ratings for Vermont businesses. The optimized SVD model (MSE: 0.895954) demonstrated the value of user-business interactions for recommendations without requiring additional features, though it faced challenges with dataset sparsity. Additionally, factorization machines were limited by library constraints. Future work could incorporate text embeddings, as demonstrated in the successful BLAIR framework. For rating predictions, XGBoost outperformed alternative models, achieving the best test set MSE of 0.5737450 by leveraging Word2Vec embeddings, temporal features, and optimized hyperparameters such as maximum depth and learning rate. Compared to ridge regression, XGBoost more effectively handled non-linear feature interactions and prevented overfitting through regularization. Our approach to predicting ratings for Vermont businesses offers a framework for similar datasets and emphasizes the importance of integrating textual, temporal, and geospatial features in predictive modeling tasks.

REFERENCES

An Yan, Zhankui He, Jiacheng Li, Tianyang Zhang, and Julian McAuley. 2022. Personalized Showcases: Generating Multi-Modal Explanations for Recommendations. arXiv preprint arXiv:2207.00422.

Chen, Tianqi, and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 785–94. Crossref, https://doi.org/10.1145/2939672.2939785.

Jiacheng Li, Jingbo Shang, and Julian McAuley. 2022. UCTopic: Unsupervised Contrastive Learning for Phrase Representations and Topic Mining. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).

McAuley, J. (2022). Personalized Machine Learning. Cambridge University Press.

Tiphaine Viard and Raphaël Fournier-S'niehotta. 2018. Movie Rating Prediction Using Content-Based and Link Stream Features. arXiv preprint arXiv:1805.02893.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2023. Bridging Language and Items for Retrieval and Recommendation. arXiv preprint arXiv:2305.10403.