Google Rating Prediction in Alaska Report

Yunqi Zhang, Jason Wang, Rui Wang, Jiaqi Wu

December 2024

Abstract

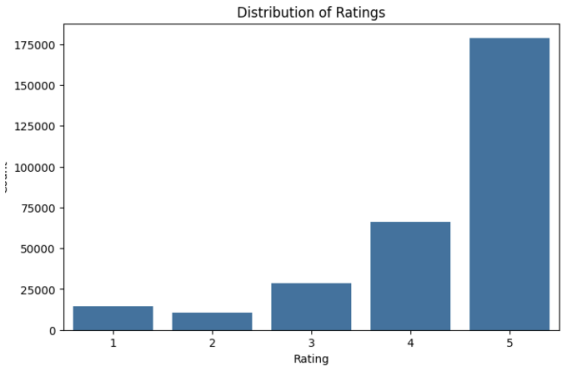
The Google reviews dataset contains a vast collection of review information from Google Maps. This project aims to predict Google ratings using machine learning models by analyzing reviews. After thorough data cleaning and feature engineering, the dataset comprising 521,515 entries serves as a robust foundation for predictive modeling. We explore multiple regression models and deep learning techniques to assess performance. Initial findings reveal that text-based features significantly impact the prediction accuracy, and incorporating user metadata improves model reliability. Comprehensive evaluation indicates that our best model achieves crucial significance, making it an effective tool for understanding and predicting user ratings on Google Maps.

1 Introduction

Review text and review rating are usually closely related. We will refer to the Google Local Review Dataset in Alaska for a series of explorations. The google reviews contains a series of review information and user ratings from google maps, which is a perfect dataset for training a rating prediction model. In order to exclude the impact of the geographic factors that potentially influence the accuracy of the result, we decided to choose google local Alaska review as our dataset. We will compare different models to explore the link between language and rating.

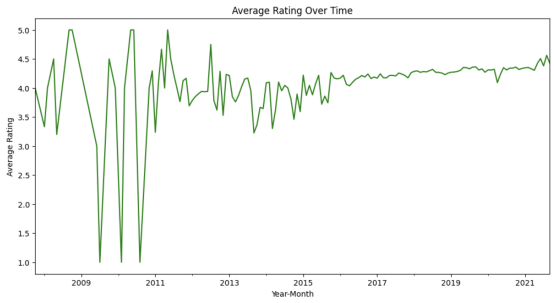
2 Data Exploration and Data Cleaning

It includes several critical columns, each with unique properties and potential contributions to the prediction model. The $user_id$ column provides a unique identifier for each user, facilitating the grouping of reviews and enabling the analysis of individual reviewing patterns, such as average ratings or review frequency. The name column offers user information, while the $gmap_id$ column, which has now been removed as part of data cleaning, previously served as a unique identifier for businesses. The time column, with no missing entries, records the timestamp of each review, enabling trend analysis over time. The rating column, the target variable for prediction, captures user-provided star ratings ranging from 1 to 5 and exhibits a skew toward higher values, with a mean of 4.28 and the majority clustered in the 4-5 range. This suggests that most reviews are positive. The text column contains user-generated reviews, which can be leveraged for natural language processing techniques to extract sentiment, tone, and keyword trends. However, 42.8% of entries in this column were missing and were removed during data cleaning to ensure meaningful textual analysis. Similarly, the pics and resp columns, which represent user-uploaded images and business replies to reviews, respectively, had significant missing values at 95.9% and 91.5%. These columns were also removed during data cleaning due to their limited contribution to the predictive model. After this cleaning process, the dataset was reduced to 298,250 rows, focusing on the most relevant and complete data for modeling and analysis.

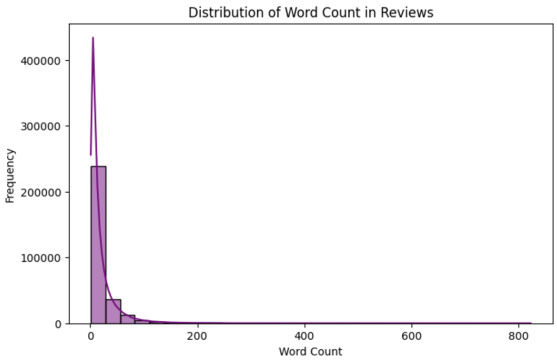


The distribution of ratings in the dataset reveals a mean rating of approximately 4.29, with a median of 5, and a strong skew toward positive reviews, as the majority of ratings, approximately 83.6%, fall within the 4-5 range.

The dataset reflects a steady increase in review volume over time. Average ratings vary, with dips in January 2008 to 3.33 and May 2008 to 3.2 and spikes in April 2008 to 4.5 and June 2021 to 4.5, suggesting tem-

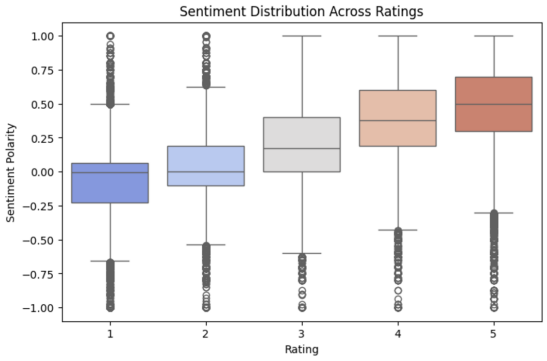


poral effects on user satisfaction influenced by seasonal factors or events. Recent years show a steady rise in ratings, with consistently high averages from mid-2018 to 2021, such as 4.35 in June 2020 and 4.56 in August 2021. These trends highlight the importance of temporal features in predictive models to capture cyclical patterns and evolving user sentiment.

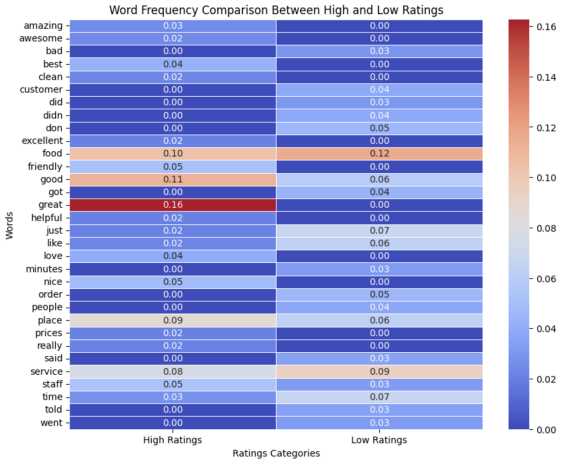


The word count statistics show that reviews vary in length, with an average of 20 words and a median of 10. Most reviews fall between 5 and 23 words, with the longest review containing 823 words. This variation suggests that word count could be a useful feature in predicting ratings, as longer reviews may correlate with more detailed user experiences, while shorter reviews might indicate straightforward or neutral feedback.

The sentiment statistics show that scores range from -1 to 1, where -1 represents very negative sentiment, 0 is neutral, and 1 is very positive. The average sentiment is 0.40, with most reviews leaning slightly positive. Half of the reviews have a sentiment between 0.16 and 0.65, and the variation in sentiment scores, standard deviation of 0.34, suggests a mix of emotions in



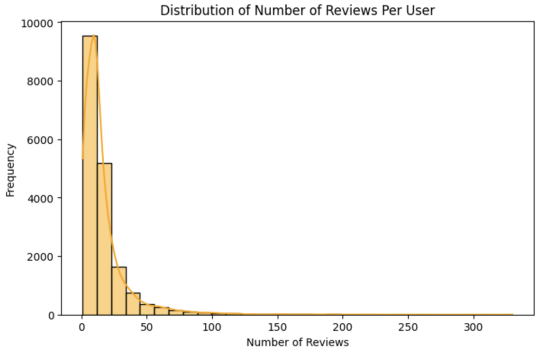
the reviews. While many reviews are favorable, some still express strong negative or neutral sentiments. This makes sentiment a useful feature for predicting user ratings, as it directly reflects the emotional tone of the text. Reviews with high positive sentiment are likely associated with higher ratings, while negative sentiments may align with lower ratings.

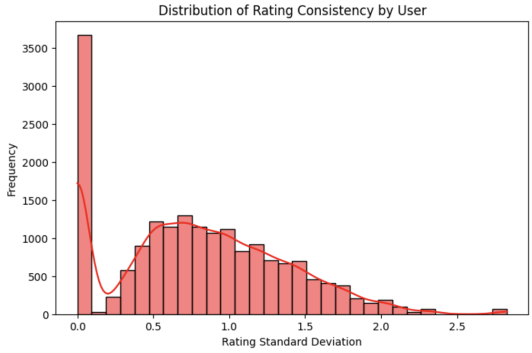


The word frequency analysis reveals distinct patterns in the language used in high and low-rated reviews. Words like great with 16.27%, good with 11.20%, friendly with 5.09%, and love with 3.93% dominate high ratings, reflecting positive experiences with service, cleanliness, and quality. In contrast, low-rated reviews frequently mention negative terms such as bad with 2.96%, down with 4.54%, and customer with 3.76%, highlighting dissatisfaction with service or operational issues.

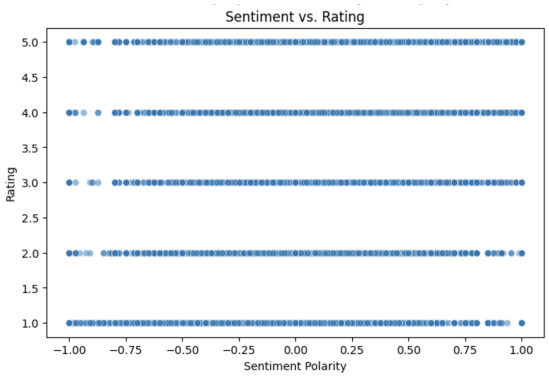
The user review count statistics show that the dataset contains 18,251 unique users, with an average of approximately 16 reviews per user. The number of reviews varies widely, with a standard deviation of 19.50.

The user rating consistency analysis shows an average consistency score of 0.79, with a standard deviation of 0.57, indicating moderate variation in individual user rating patterns. The minimum value is 0, representing users with uniform ratings, while the maximum value is 2.83, reflecting significant variability in ratings by cer-

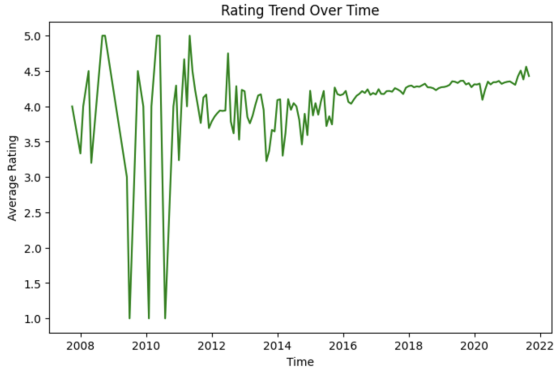




tain users.



The correlation analysis between rating and sentiment reveals a moderate positive correlation of 0.47. This indicates that as sentiment scores increase, user ratings tend to be higher, suggesting that sentiment effectively captures the emotional tone of reviews and aligns with user satisfaction.



Over time, average ratings have generally improved, showing that users have become more satisfied as the platform grew and stabilized.

3 Predictive Task: Rating Prediction Based on Review Text

We used several regression models to perform the prediction: count-vectorizer and un-regularized linear regression, Linear Regression with Lasso/Ridge Regularization; L2-Norm Regularized Logistic Regression, L1-Norm Regularized Logistic Regression. To evaluate a linear regression model's performance and validity in predicting review ratings based on text, we will use the Root Mean Squared Error (RMSE) and R^2 score as primary metrics. For Logistic Regression models, we use test/train AUC, which indicate the performance of the logistic regression model on the training and test datasets, respectively, in terms of its ability to discriminate between the two classes (ratings of 1 and 5), and, model complexity, as primary metrics.

To transform the review text into features and prepare the data for both logistic regression and linear regression tasks, the dataset is first filtered appropriately. For logistic regression, the dataset is limited to reviews with ratings of 1 (negative) and 5 (positive) to focus on binary classification. For linear regression, the entire range of ratings is used as the target variable. The data is split into training (80%) and test (20%) sets using `train_test_split`, with a fixed random state for reproducibility. Text data is processed using `CountVectorizer`, which converts review text into numerical feature vectors. For both tasks, unigrams (single words) and bigrams (two consecutive words) are extracted to capture contextual information, with a minimum document frequency (`min_df=10`) applied to remove infrequent terms, reducing noise and sparsity. For logistic regression, the `fit_transform` method is applied to the training data to create a sparse matrix, followed by transforming the test data using the same vectorizer. For linear regression, a similar process is followed, but the sparse matrices for both training and test sets are converted into dense NumPy arrays (`toarray`) for compatibility with the regression models. This process results in feature matrices where each column represents an n-gram and each value indicates the frequency of that n-gram in the text. These numerical representations allow both models to learn relationships between the textual patterns and the target variable, whether binary ratings for logistic regression or continuous ratings for linear regression.

4 Model Description

4.1 Linear Regression

4.1.1 Linear Regression as Baselines

For this section, we implemented 3 models: count vectorizer and unregularized linear regression, Linear Regression with Lasso/Ridge regularization. As mentioned above, RMSE quantifies the average prediction error in the same units as the target variable (ratings), while R^2 measures how well the model explains the variance in the data. Lower RMSE and higher R^2 indicate better model performance.

We start with a count vectorizer and unregularized linear regression model as a relevant baseline. `CountVectorizer` with n-grams (bi-grams) and a minimum document frequency filter (`min_df=10`). This preprocessing reduced noise and controlled the feature space's dimensionality. Logistic Regression's regularization effectively addressed the high-dimensional, sparse nature of text data. The Count Vectorizer converts the words in the input text into tokens, creating a "bag of words" or vocabulary. These tokens serve as features for modeling. It then transforms the list of documents (sentences or reviews) into a matrix, where each row represents a document, and each column shows how often a word from the vocabulary appears in that document.

To include single words and two-word phrases as features, we use `ngram_range=(1, 2)`. The parameter `min_df=10` ensures that only words or phrases appearing at least 10 times in the entire dataset are included. Using this setup, we train a linear regression model with 80% of the data for training and 20% for testing. After fitting the regression model with the training data, we got an R^2 score of -0.24 and RMSE of 1.19, which indicated that the model is not yet a good fit to explain the data.

After several attempts, we find it hard to make a more accurate prediction based on the current model. Then we deep dive into the model's important features for more insight (see in Figure below).

Out[17]:

	Coefficient	Feature_Name	Coefficient_Magnitude
8163	4.144546	mooses	4.144546
9866	-3.992146	poisoning	3.992146
8164	-3.824276	mooses tooth	3.824276
11882	-3.517178	supposed	3.517178
7718	-3.302333	low carb	3.302333
8470	3.294776	negative reviews	3.294776
2558	3.256417	carb	3.256417
11883	2.985089	supposed to	2.985089
12237	2.951029	the absolute	2.951029
11906	2.798051	surrounded by	2.798051

Obviously, most of these words are neutral words, which cannot correspond well to their positive and negative coefficients.

In order to improve the model's performance, we tried regularization techniques. Here, lasso and ridge regularization are used with varying levels of strength, controlled by the alpha value. A higher alpha means stronger regularization, which adds more penalty to complex models, leading to simpler models with reduced complexity. It improves the accuracy a little bit, but the model still cannot meet our expectations.

4.1.2 Lasso/Ridge Regularized Linear Regression as Baselines

Firstly, the lasso regularization is applied.

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|=\mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|.$$

In the context of the dataset, the Lasso regression model predicts review ratings based on the review text, transformed into n-gram features. By applying L1 regularization, it reduces the impact of less significant n-grams, setting some coefficients to zero, effectively selecting the most influential textual patterns. The alpha parameter controls this regularization, balancing the model's ability to minimize RMSE in both training and test sets while avoiding overfitting and enhancing interpretability. The complexity of the model is measured using various methods and in this case the L1 norm (the sum of the absolute values of the coefficients) is used as an indicator and displayed alongside the results (see Figure below).

Out[18]:

	Alpha	Training RMSE	Model Complexity - Coef Norm1	Model Complexity - Coef Sum	Test RMSE
0	0.0001	0.750221	481.920199	481.920199	0.856181
1	0.0010	0.877507	39.116763	39.116763	0.871062
2	0.0100	0.980581	4.055629	4.055629	0.955758
3	0.1000	1.077239	0.139842	0.139842	1.047398

Based on the results, we will take the Alpha with the best RMSE test to dive deep into the features (see Figure Below).

Out[22]:

	Coefficient	Feature_Name	Coefficient_Magnitude
15316	-1.378388	worst	1.378388
9866	-1.362892	poisoning	1.362892
13974	-1.265639	unprofessional	1.265639
10647	1.256906	said she	1.256906
13924	-1.186987	ugh	1.186987

This begins to make sense, words like "worst", "unprofessional" clearly indicate the negative attitude of the rating.

We continued to try Ridge Regression after we got

this result.

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\text{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2.$$

Ridge regression applies L2 regularization, which penalizes the square of the coefficients, shrinking them towards zero but not eliminating them entirely. This helps manage the high-dimensional feature space created by the n-gram representation of text, balancing the trade-off between model complexity and overfitting. The regularization strength is controlled by the alpha parameter, with smaller values allowing more complex models and larger values enforcing stronger regularization. By minimizing the RMSE on training and test data, Ridge regression ensures the model captures significant n-grams while maintaining generalization, making it effective for predicting ratings from textual data. Performance as below:

Out[23]:

	Alpha	Training RMSE	Model Complexity - Coef Norm2	Model Complexity - Coef Sum	Test RMSE
0	0.0001	0.620827	51.545854	4583.029412	1.188618
1	0.0010	0.620825	51.530191	4581.037861	1.188329
2	0.0100	0.620832	51.021582	4552.451123	1.185537
3	0.1000	0.621121	47.679754	4326.976643	1.161977

The performance of Ridge Regression appears suboptimal, as its test RMSE shows no significant improvement. To further analyze this, we examine the feature coefficients derived from the Ridge Regression model (see Figure Below).

Out[24]:

	Coefficient	Feature_Name	Coefficient_Magnitude
8470	3.051692	negative reviews	3.051692
9866	-2.828586	poisoning	2.828586
12237	2.599575	the absolute	2.599575
8843	2.290167	of july	2.290167
3980	-2.205408	ended up	2.205408
13120	-2.159511	they refused	2.159511
13435	-2.129830	to contact	2.129830
15376	2.074362	wrong the	2.074362
14509	2.032258	was literally	2.032258
5306	-1.980936	going down	1.980936

Unexpectedly, neutral words in these features are assigned strong positive/negative coefficients, resulting in poor performance similar to the previous baseline.

4.2 Logistic Regression

Considering the limitations of the rating system itself, especially the ambiguous attitude between rating of 2 to 4, we will take a more extreme approach. We will treat the problem as a binary classification of the two extreme classes: 1 and 5 rating reviews, by creating a subset of the main dataset, followed by the same 80–20 split for the train-test sets and vectorization of the review texts into features. In this way, we can use Logistic Regression Models.

In SKLearn's logistic regression, the regularization strength is controlled by the C parameter. A smaller C value means stronger regularization, imposing greater penalties on complex models.

To evaluate the model's classification performance, the area under the ROC curve (AUC) is used as a metric and analyzed across different regularization strengths.

First, we'll implement L2-regularized logistic regression models and track their AUC values in a simple DataFrame. The complexity of each model is measured by summing the magnitudes of its coefficients (see Figure Below).

Out[28]:

	C	Train AUC	Model Complexity - Coef Sum	Test AUC
0	0.001	0.917593	41.290877	0.913769
1	0.010	0.967260	187.567697	0.961990
2	0.100	0.989563	754.892569	0.974043
3	1.000	0.997718	2306.366188	0.970842
4	10.000	0.999494	5347.487915	0.962702
5	100.000	0.999818	10234.329456	0.954565

The model with C=0.01 seems to be the most desirable. This is because while the complexity is 2nd-lowest, it can achieve 97% of the train and test AUC of the most complex model. Here is the further analysis of feature coefficients when applying C = 0.01:

Out[30]:

	Coefficient	Feature_Name	Coefficient_Magnitude
4483	1.249256	great	1.249256
6999	-0.821376	not	0.821376
1589	0.724780	best	0.724780
6225	0.696577	love	0.696577
6964	-0.647048	no	0.647048
4345	0.609341	good	0.609341
1298	0.602756	awesome	0.602756
405	0.587002	amazing	0.587002
8552	-0.569034	rude	0.569034
4058	0.545198	friendly	0.545198

This time the results are very clear! All these words show positive and negative attitudes corresponding to their coefficients.

Similarly, we can build multiple L1-regularized logistic regression models and evaluate their performance using an AUC table. L1 regularization penalizes the absolute values of coefficients, encouraging sparsity by reducing some coefficients to exactly zero. This results in a model that selects the most relevant features (n-grams) for predicting sentiment, effectively acting as a feature selection mechanism.

This time we'll choose C = 0.1, as it's of the third lowest complexity, but it achieves 95% of the train and

Out[32]:

	C	Train AUC	Model Complexity - Coef Sum	Test AUC
0	0.001	0.651283	0.109635	0.665348
1	0.010	0.877230	7.663545	0.881453
2	0.100	0.961820	96.158238	0.957857
3	1.000	0.994152	902.051931	0.968263
4	10.000	0.999463	4245.945487	0.956822
5	100.000	0.999876	10278.518655	0.942418

test AUC of the most complex model. Further analysis of features are shown below:

Out[33]:

	Coefficient	Feature_Name	Coefficient_Magnitude
8552	-2.930599	rude	2.930599
5133	-2.513893	horrible	2.513893
4483	2.338434	great	2.338434
1298	2.333975	awesome	2.333975
12368	-2.242010	worst	2.242010
7968	-2.173417	poor	2.173417
405	2.133674	amazing	2.133674
3394	2.105657	excellent	2.105657
9755	-2.079116	terrible	2.079116
1589	2.056812	best	2.056812

The best-performing L1-regularized model shows significantly lower complexity compared to the selected L2-regularized model. Despite this simplicity, the top 10 most significant features in the L1 model have clearer positive or negative meanings, making it easier to interpret and predict 1-star or 5-star ratings. Unlike the L2 model, the L1 model does not include any obviously ambiguous features.

5 Related literature

The analysis of customer reviews has been pivotal in understanding consumer behavior, optimizing business strategies, and building predictive models. Extensive studies have explored sentiment analysis and text mining to uncover how user-generated content reflects customer satisfaction and shapes business outcomes. Foundational research, such as Pang et al.[2], demonstrated the potential of sentiment classification using textual data, providing a framework for the analysis of reviews on platforms such as Yelp, Amazon, and Google Maps. More recent work, such as Yan et al.[3], has expanded this understanding by introducing multimodal approaches, integrating textual, visual, and structured metadata to generate personalized explanations for recommendations. This multifaceted approach addresses the complexity of user preferences by providing clearer insights into customer experiences and satisfaction. State-of-the-art methodologies for review analysis often involve Natural Language Processing (NLP) and machine learning techniques. For instance, the Bag-of-Words (BOW) and n-gram approaches remain foundational for text vectorization, transforming textual data into numerical representations. Studies frequently employ CountVectorizer and other text processing tools to extract features such as unigrams and bigrams, which capture both individual words and contextual phrases. These techniques are particularly useful for tasks such as sentiment prediction, as evidenced by their application in the linear and logistic regression models of this study. Beyond traditional methods, Yan et al.[3] demonstrated the essence of leveraging pre-trained NLP models such as BERT and multimodal embeddings to provide context-aware recommendations, incorporating both textual and visual elements to improve interpretability and relevance. Recent research has also shown improvement through domain-specific adaptations. For example, Li et al.[1] demonstrated fine-tuning pre-trained language models on domain-specific datasets to capture contextual nuances, while Yan et al.[3] emphasized the benefits of generating personalized showcases that combine structured metadata, such as price levels and business categories, with unstructured data like customer reviews. These approaches provide deeper insight into the dynamics of user satisfaction by tailoring the methods to specific domains. The integration of text-based feature extraction, multi-modal embeddings, and domain-specific fine-tuning mirrors trends in modern review analysis while tailoring methods to the unique characteristics of datasets such as those from Alaska local reviews. The study's approach builds on prior work by offering a robust framework for analyzing consumer-business dynamics, contributing to the growing body of research that emphasizes context-aware and multi-modal strategies for review analysis.

6 Conclusion

Above all, this assignment successfully predicts ratings based on customer review data using sentiment analysis and machine learning techniques. As we realized most of the words in reviews are neutral words, which should have slight impacts on the model, we tried Lasso and Ridge Linear Regression to see if we are able to improve the accuracy a little bit. Based on our results, Lasso regularization performs better than Ridge regularization due to its ability to avoid overfitting and enhance interpretability. This may be because when customers give ratings, the boundaries between the words and attitudes corresponding to ratings 2, 3, and 4 are not clear enough, at least they do not show the extreme differentiation like that of 1 and 5. As a result, we also tried Logistic regression to evaluate the model's classification performance. We built L1 and L2 regularized models re-

spectively. The L1-regularized model shows significantly lower complexity compared to the L2-regularized model, and the L1 model does not have any ambiguous features as the L2 model does, which makes it the best model to predict 1-star to 5-star ratings.

References

[1] Jiacheng Li, Jingbo Shang, and Julian McAuley. Uctopic: Unsupervised contrastive learning for phrase representations and topic mining. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

[2] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, 2002.

[3] An Yan, Zhankui He, Jiacheng Li, Tianyang Zhang, and Julian McAuley. Personalized showcases: Generating multi-modal explanations for recommendations. *arXiv preprint arXiv:2207.00422*, 2022.