Assignment 2 - Google Restaurant Reviews

The dataset that we are analyzing contains reviews of restaurants from Google. Each of the reviews in the dataset contains a user ID for the person giving the review, a business ID for the business they are reviewing, a rating given by the user, a text review given by the user, and any images they may have attached along with the review. One thing to note is the tendency of the restaurant ratings to skew to the higher end. What is interesting about this dataset is that most of the reviewers give the restaurants they visited a high rating, as seen in the pie chart. The entire data is large with a distribution of 9782 businesses on 200,000 records. There are 169,380 users who provide ratings.

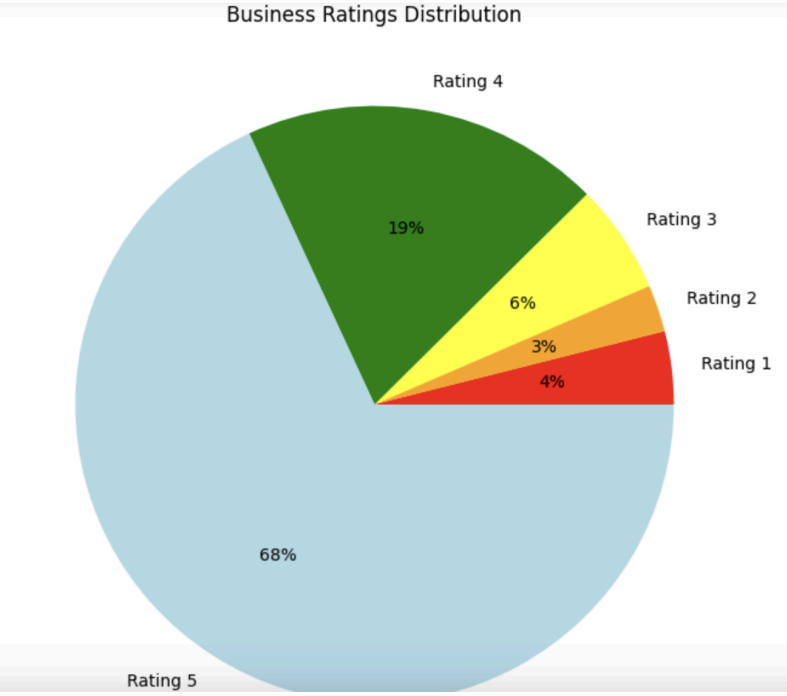

Figure 1.1 Distribution of Ratings

The above pie chart demonstrates that 68% of the ratings given by users were 5 stars. The observation that a majority of the ratings were high encouraged us to take into account not only the mean but also the mode as a hyperparameter for the baseline model.

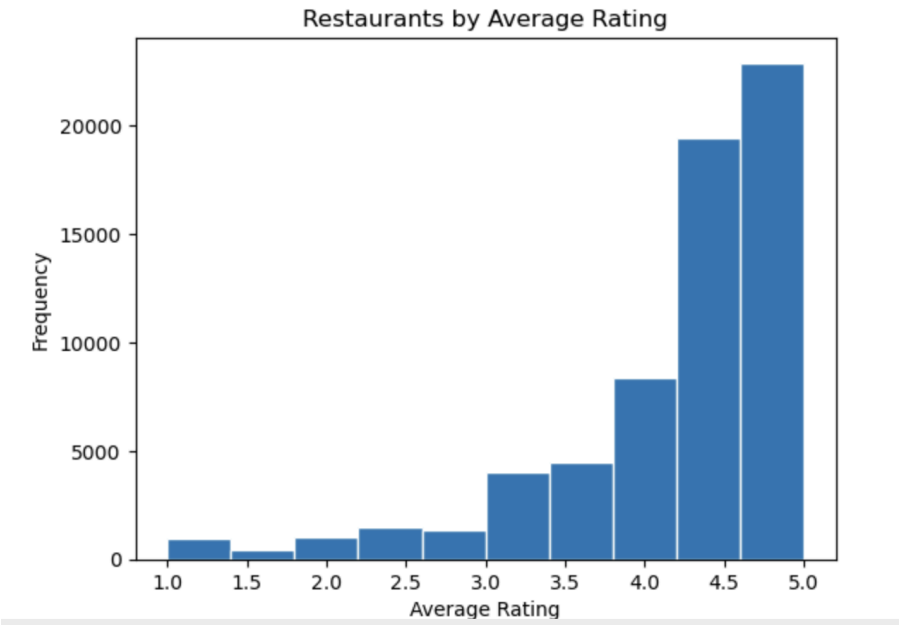

Figure 1.2 Histogram of Average Ratings by Restaurant

Most restaurants have a number of ratings that are high, so the data is skewed left.

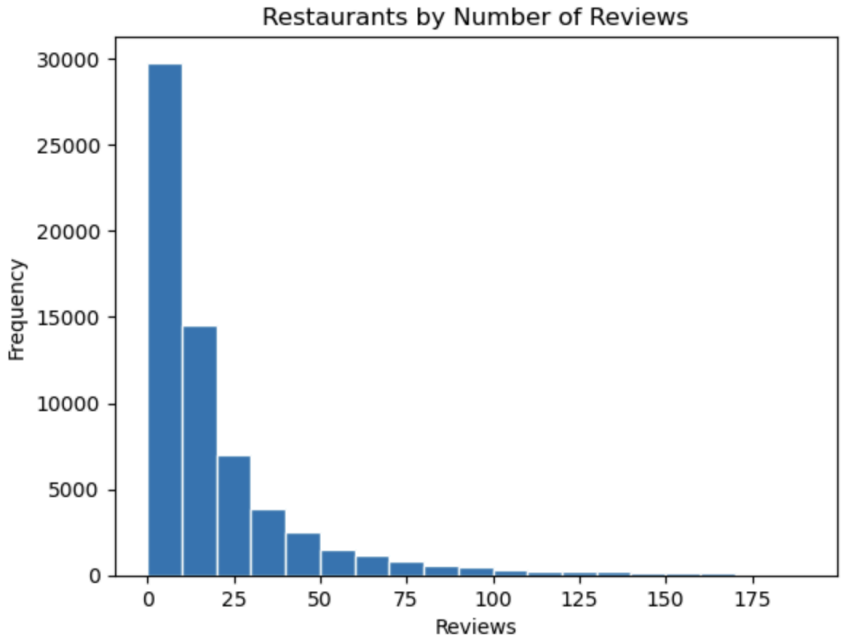

Figure 1.3 Histogram of Restaurants by Number of Reviews

Most users only submitted one review, so the histogram is skewed right.

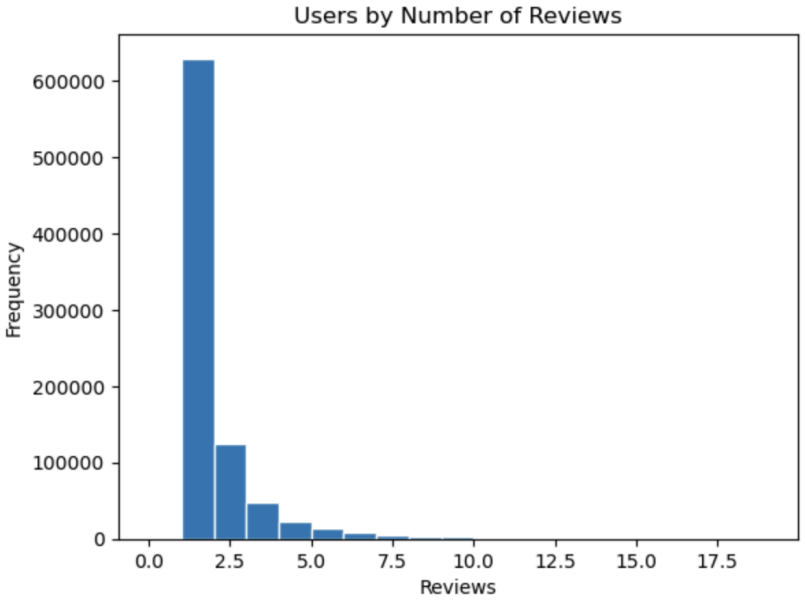

Figure 1.4 Distribution of Users by Number of Reviews

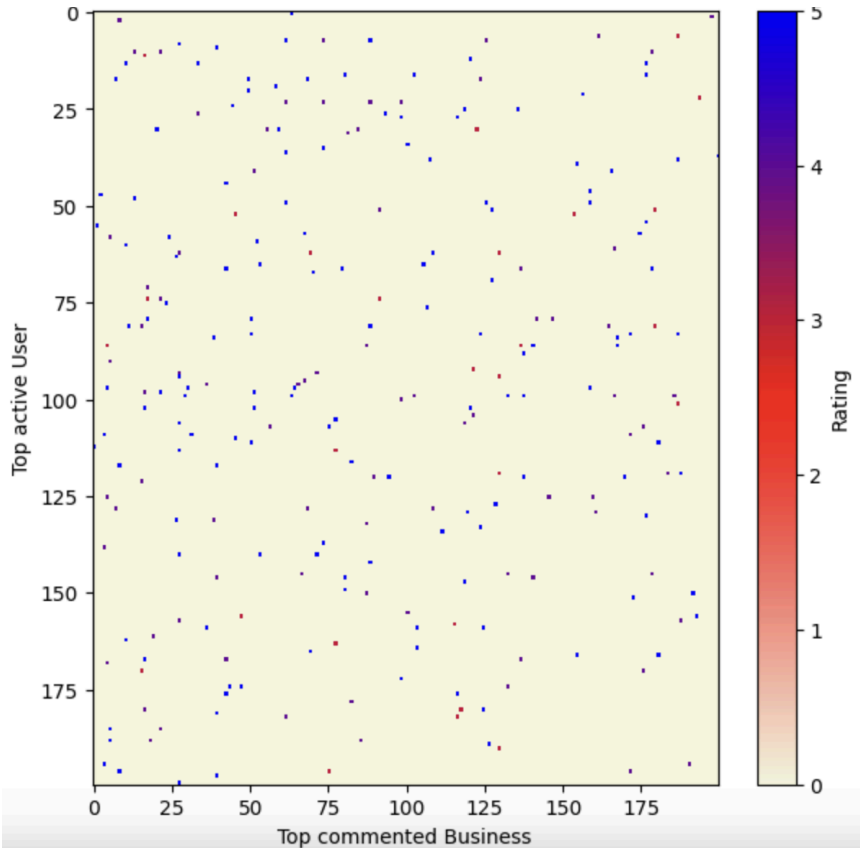

Figure 1.5 Heatmap of User-Business Ratings

The above heatmap is showing the rating distribution by the top 200 most active users on top 200 businesses in the database. There are a lot of businesses where the most active users have not given any comments. This shows that the data is diverse.

Our predictive task for this dataset was to be able to predict the rating a user gave a certain business based on any of the other features present in the dataset. Because the ratings only contain integers from 1 to 5, the way we evaluated our models was through a simple check of accuracy; whether or not the predicted rating of a given user/business pair was exactly equal to the actual rating provided, as a proportion. For one of the models, TF-IDF (Term Frequency - Inverse Document Frequency), processing of the review text was done. We removed the punctuation marks and made the reviews case insensitive. These two techniques were covered in the course and highly recommended. There is another technique called stemming, which lowers inflections in words to their root forms. As suggested in the class, we did not use this technique. We split the data into 85% training, 10% test, and 5% validation.

We used two baseline models to compare the models we were developing to. One was, given a review, take the business that the review is for, find the average rating of the business, and predict the rating of the review based on the average rating of the business. As the ratings were only integers, it also had to round before giving the rating. This gave an accuracy on the entire dataset of around 54.9%. The second baseline we used was given a dataset, predicting, for every review, the most common rating in the dataset. As we were doing exploratory analysis, we came across this interesting fact that the data is heavily biased towards the higher rating, with a supermajority of the ratings being a 5. This motivated us to use mode as the parameter. With this model, where every review is predicted to be with a rating equivalent to the mode, the accuracy was about 66.8%. As the second model was very simple and reasonably accurate, it provides a good baseline to compare against to evaluate our model.

All of the features in the dataset will be used in our models, although not all of them at the same time. Some of the models will use solely the user_id and business_id pairings in addition to the ratings while trying to predict, while others will attempt to incorporate data such as the review text as well to try to predict. In order to use the review text, it will need to be processed based on the needs of each individual model that uses it. We will not be using the images in our predictions, as that is fairly outside the realm of our understanding at this moment in time.

We experimented with another model: logistic regression using TF-IDF. Two hyperparameters were selected. The first hyperparameter is the length of the most frequent words, with various values tested. The final code runs with two specific values: 750 and 500. The second hyperparameter is the regularization constant for logistic regression. In the code, we experimented with the values five and seven, but offline, we tested various other values. This way, we explored four combinations of the two hyperparameters.

We used a validation test to select the regularization constant and found the best accuracy to be 73.1%. This accuracy exceeds both the baseline model accuracy and that of the alternative model we tried. The optimal accuracy is achieved when the word vector is 750, and the regularization constant is 7. At the end of the notebook, there is a summary plot illustrating all the accuracies.

One of the models we tried using to predict rating was a Singular Value Decomposition (SVD), implemented using the Surprise library. The reason we decided to try SVD was because it is normally good at finding patterns between users and items. It was also fairly simple to implement while providing

good performance. To optimize the SVD model, we tried tuning the hyperparameters, such as the number of latent factors, the number of iterations, learning rate, and regularization term, and evaluating based on the same accuracy we used for everything else. As the model returned decimal values for its predictions, we also had to round our predictions to the nearest whole number in order to assess our predictions. While scalability wasn't an issue this time around, it might have become an issue with a much larger, or an exceptionally large dataset. Techniques such as model-based matrix factorization inherently guard against overfitting by capturing only significant patterns in the data. One of the biggest strengths of SVD is its simplicity and ease of use. We didn't really have to clean the data in any way in order to use it, and it ran pretty quickly as well. However, this was also one of its biggest problems. Because it only considered the User ID, Business ID, and the rating given by the review, it was not very robust compared to other models we could have used, such as TFIDF. This resulted in it attaining a lower accuracy compared to other models, and even a lower accuracy with the hyperparameters we found compared to the baseline models.

Another model we tried to use was Alternating Least Squares (ALS) matrix factorization, implemented using the PySpark library. A lot of the decision to use ALS was due to our unfamiliarity with it, so it was something additional to practice. It is also known to be fairly scalable to large datasets and able to capture user-item interactions well too. Similarly to the SVD, ALS is also tuned using hyperparameters such as number of latent factors and number of iterations. ALS is well-suited for scalability, particularly when dealing with sparse matrices. It naturally mitigates overfitting by regularizing latent factors during the optimization process. However, there were a couple times where the model seemed to be overfitting, likely due to the regularization term chosen. Initially, we tried using the ALS functions within the implicit library, but ultimately, were not able to get them to run for some unknown reasons. Ultimately, the PySpark libraries worked for us, but at exceptionally low accuracies. In addition to that, the PySpark code took incredibly long to run, making it seem like a poor choice to try to get working in the long run. That is to say, while the model was fit for scalability, the ways we found to implement it were not.

The dataset we are using comes from a paper that was previously written about recommender systems. In fact, very recently by our professor and others. However, their project seems to have been very different from ours. While ours is using the data, aside from the image data, available to try to predict ratings, their work done on the project was mostly focused on restaurant recommendations. Also, the main differentiating point of the application compared to work done in the past was that they used images in their personalized showcases, rather than simple text recommendations that have been done in the past. That is to say, images are a main focus of the work previously done on this dataset, while it is not relevant to how we are using the dataset.

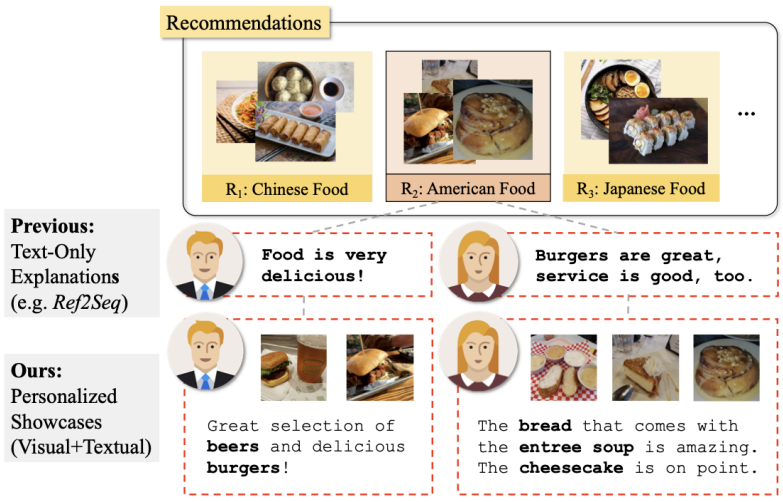

Figure 4.1 Sample of Previous Work Done on this Dataset

There has been a lot of work done on similar datasets in the past. For instance, this paper was done on a Yelp review dataset. In this dataset, we can see a similar trend going on in terms of review ratings; the ratings are highly skewed to the right.

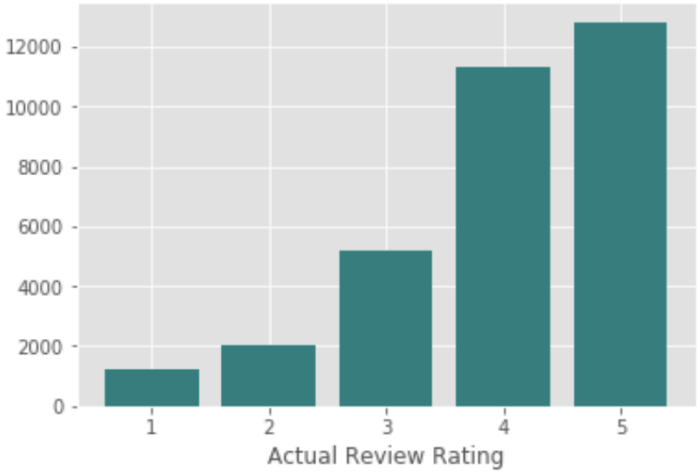

Figure 4.2 Yelp Review Dataset Review Ratings

In this paper, the author primarily focuses on text-based analysis, and learning what words are correlated in certain review ratings with machine learning. He vectorizes the review text with a TFIDF vectorizer, and uses methods such as SVMs and Random Forests to try to predict the ratings on the data. In addition, sentiment analysis was performed on the text to determine whether or not the review would be positive or negative in terms of rating.

	Positive Prediction	Negative Prediction
CountVectorizer	'gem', 'disappoint', 'heaven', 'phenomenal', 'pleasantly', 'olives', 'kinds', 'perfection', 'nut', 'efficient', 'affordable', 'caramel', 'generous', 'suggestion', 'perfectly', 'thankfully', 'die', 'perfect', 'creative', 'awesome'	'mediocre', 'horrible', 'worst', 'meh', 'soggy', 'disappointing', 'uncomfortable', 'gross', 'overpriced', 'sick', 'lacked', 'worse', 'bland', 'awful', 'sad', 'memorable', 'tough', 'ok', 'alright', 'hoping'
TfidfVectorizer	'delicious', 'amazing', 'perfect', 'great', 'awesome', 'love', 'favorite', 'definitely', 'excellent', 'loved', 'perfectly', 'best', 'perfection', 'disappoint', 'fantastic', 'yummy', 'friendly', 'recommend', 'bomb', 'glad'	'ok', 'mediocre', 'bland', 'okay', 'horrible', 'dry', 'wasn', 'disappointing', 'worst', 'average', 'meh', 'unfortunately', 'overpriced', 'disappointed', 'terrible', 'bad', 'maybe', 'sad', 'awful', 'soggy'

Figure 4.3 Sentiment Analysis on Yelp Review Dataset

Ultimately, our Logistic Regression model with the TF-IDF vectorized review text into the ratings' predictions performed the best. The Logistic Regression models performed slightly better than the mode baseline model, and significantly outperformed the models that did not integrate the review text at all. The SVD model even underperformed the baseline model, which could show that not all models are better at predicting than a very simple model.

As mentioned above, various models were tested to predict the rating. Following is the test accuracy for different models:

Baseline Model	0.66830
SVD Model	0.50484
LogisticRegression_vector750_c7	0.73135
LogisticRegression_vector500_c5	0.72755
Alternating least square	0.61200

Clearly, as demonstrated by the table, logistic regression with the fine tuning of its hyperparameters is the winner. The model proves the importance of incorporating textual data in this predictive modeling. The TF-IDF vectorization effectively captures the significance of words in the review text, balancing their frequency in a specific document against their frequency across all documents.

For the baseline model one interesting finding was better accuracy for using the mode rating of the training set for unmatched business in the test set. We observed test accuracy of 66.8% with mode vs just 19.37% with mean. The mode, being the most frequently occurring rating, offers a better heuristic for prediction than the mean, especially in a dataset where ratings might not be normally distributed.

The optimal accuracy is attained when employing logistic regression with a TF-IDF vector based on the most frequent words. To generate this TF-IDF vector, the 'review_text' undergoes preprocessing involving the removal of punctuation and conversion to lowercase.

Below is the tuple of test accuracy , word vector length, regularization constant tried:
(0.71695, 300, 6)
(0.71785, 310, 6)
(0.7193, 320, 6)
(0.71945, 330, 6)
(0.72145, 360, 7)
(0.72315, 390, 7)
(0.72315, 400, 5)
(0.72615, 450, 7)
(0.72755, 500, 5)
(0.72815, 550, 5)
(0.7296, 600, 7)
(0.73035, 650, 7)
(0.73035, 700, 7)
(0.73135, 750, 7)

The highest accuracy achieved was 73.13%. This result was associated with a vector length of 750 for TF-IDF and a regularization constant of 7. The failure of other models can be attributed to not having the 'review_text' feature.

The higher performance of the Logistic Regression model with TF-IDF vectorization demonstrates how processing of text data can yield more accurate predictions than simpler models or those that don't leverage textual information. This insight is particularly valuable in fields like sentiment analysis, customer feedback analysis, and other applications where understanding language is crucial.

Our results highlight the importance of feature selection and model tuning. The variations in performance across different configurations of the Logistic Regression model illustrate that parameters such as vector length and regularization constants positively impact Logistic Regression model effectiveness.

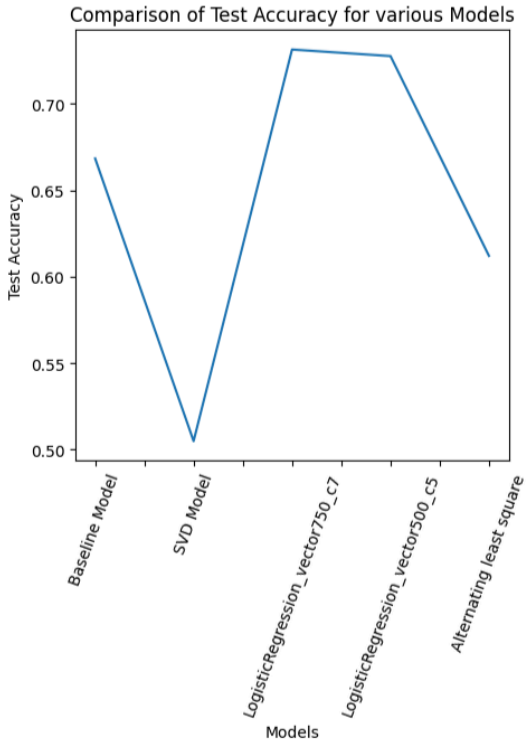

Figure 5.1 Line Graph Comparing Performance of Various Models