Sofia Beyerlin

Maria DeCaro

Matt Maitland

Alex Szabo

Abstract

This study analyzes and explores the prediction of Yelp star ratings for reviews of Florida restaurants, leveraging review text, user information, and restaurant metadata. We explore the extent to which these features can forecast ratings, record which factors have the most weight when predicting ratings, and evaluate prediction accuracy across several model types. These findings demonstrate that large language models, especially in zero-shot settings, can rival or surpass traditional methods, highlighting the growing potential of NLP in real-world review analysis.

Background

The problem we aim to solve is predicting the # of stars a user will give a restaurant based on predictors like user's # of friends**,** cuisine type, and *review text*. A motivation to do this is because users tend to give inflated ratings, awarding high stars on average experiences. These findings will help businesses and platforms better understand behavioral patterns in fields where reviews heavily influence customer decisions.

Dataset

We initially planned to use the Yelp dataset filtered for New York City, given the city's high volume of tourism and the diversity of its restaurant scene. However, we quickly encountered our first challenge: Yelp's live database is partially closed and paywalled, limiting access to comprehensive data without a premium subscription.

In search of alternatives, we explored Zomato, a platform focused exclusively on restaurants that previously offered a public, free API. Unfortunately, after attempting to connect, we discovered that Zomato had discontinued API access within the past year, making it unusable for our project.

Finally, through additional research, we found Yelp's Academic Open Dataset, which is freely available for student and educational use. This version of the dataset included all essential data needed, including detailed business, user, and review data, and became the foundation for our project.

The original Yelp dataset totaled over 8 GB and was distributed across six large JSON files, each containing millions of entries. We began by extracting the dataset from a .tar archive, then loaded each file carefully in chunks to avoid memory issues.

Since the data was spread across separate files for businesses, reviews, users, check-ins, and tips, we performed multi-stage merges using keys like business_id and user_id to consolidate relevant information into one unified dataset. This process was technically challenging due to the size and structure of the files, but it was essential for building a clean, analysis-ready dataset for our Florida restaurant focus.

Milestones/Analysis

For the data cleaning milestone, we focused on preparing the Yelp dataset (filtered to Florida restaurants) for efficient and scalable machine learning. Given the size and complexity of the dataset, over 800,000 rows and high-cardinality text fields, we made several critical preprocessing decisions to ensure our models would perform well.

We began by removing rows with missing/null values, necessary to prevent downstream errors and reduce noise in our features. We then lemmatized the review text, standardizing different forms of the same word (e.g., “running,” “ran,” “runs”) to improve consistency across textual data and prepare it for potential NLP tasks like sentiment analysis or helpfulness prediction.

To resolve ambiguity from merging multiple tables (business, user, review), we renamed columns like name_x and stars_y to clearer labels like business_name and review_stars, improving readability and usability.

Before data cleaning: 869,925 rows of data and 40 columns

After data cleaning: 843,025 rows of data and 347 columns

We also engineered new features from user metadata. We created a binary column indicating whether a user had ever been an ‘elite’ user; similarly, the “friends” column originally listed each friend’s ID in a string, which we converted to a simple integer count to quantify user connectedness and reduce dimensionality.

One challenge involved the attributes and categories columns, which have multi-label text data with hundreds of possible values. Due to the large number of unique values, we determined that one-hot encoding would have dramatically inflated the feature space and introduced excessive sparsity, making the model inefficient and prone to overfitting. Instead, we applied TF-IDF embedding to both columns, which allowed us to extract meaningful numerical representations of business characteristics without overwhelming the model. This approach helped us retain semantic value while ensuring our pipeline remained computationally efficient.

Figure 1: Star Rating (Target) Distribution

EDA and Feature Selection

After creating TF-IDF features, we sought to eliminate redundancy: with over 340 columns and 800k rows, the dataset was still extremely large, warranting removal of both low-signal and highly-correlated features, which is useful even if planning to use models that are robust to collinearity. The first step was to create a ‘random’-valued column, train a Random Forest model on the numeric features, and remove any features with less importance than ‘random’.

The top 10 features are shown below:

Feature	Importance
average_stars	0.129671
random_feature	0.074733
user_review_count	0.063440
user_useful_given	0.052618
friends	0.046261
user_cool_given	0.037314
user_funny_given	0.036053
biz_stars	0.023400
rating_useful	0.022349
fans	0.018644

However, the random feature was the second-highest importance, so we decided to remove features that were <0.001 ‘importance’, i.e. less than .1% of predictive

power overall. This cutoff reduced the feature space to 116 numeric features, cutting ~200 TF-IDF metadata attributes like *'hawaiian', 'custom cakes',* or *'cinema'*. Finally, from the 116 remaining features, we derived the correlation matrix and removed any columns with >0.95 correlation with any other feature, trying to remove most redundant data while preserving useful variance. With these cutoffs, 91 features remained for use in modeling where dimensionality reduction was important (e.g. logistic regression).

Results:

Initial Models

Our first modeling step was exploring a variety of models trained on two sections of the data: one only included text data from the review column (vectorized and used to create bag of words, bi-gram, and tri-gram representations) as well as the full dataset (same text features as above, in addition to raw numerical and categorical features). First, we trained a one-versus-rest (OVR) logistic regression classifier, a multinomial logistic regression classifier, a linear regression, and a random forest on only the text data. The OVR classifier performed slightly better than the other two. Results are described:

	OVR	**Multi**	**Linear**	**RF**
Accuracy	0.6993	0.6885	-	0.6168
F1	0.7017	0.6958	-	0.6031
MAE	0.3470	0.3545	0.5302	0.5555
RMSE	0.6868	0.6856	0.6879	1.0415

Once we determined that OVR was the highest performer, we trained an OVR model on the entire dataset (text + numerical + categorical features), as well as tree-based LightGBM and XGBoost models. We see improvement over all original models, but LightGBM performed significantly better than the other two. The results:

	OVR	**LightGBM**	**XGBoost**
Accuracy	0.7106	0.7401	0.7228
F1	0.7130	0.7318	0.7102
MAE	0.3309	0.2997	0.3353
RMSE	0.6619	0.6377	0.7049

Given these results, it is apparent that the most salient aspect of our data are the text representations. We are pleased to achieve >70% accuracy and ~.33 MAE (indicating that our average error is about one-third of a star rating). Based on the improvements in our models once we include all the data, we are optimistic that we can yield further improvements using additional techniques, such as hyperparameter tuning.

Hyper-Parameter Tuning for LightGBM

Initial modeling revealed that LightGBM was more successful than the other model classes. In order to verify these results, we first re-ran the LightGBM in a 5-fold Cross Validation, which yielded almost identical results (Accuracy ~ 0.7433, F1 ~ 0.7359. This signaled that our dataset was large enough that we were not subject to significant biases due to our random sampling in train/test split.

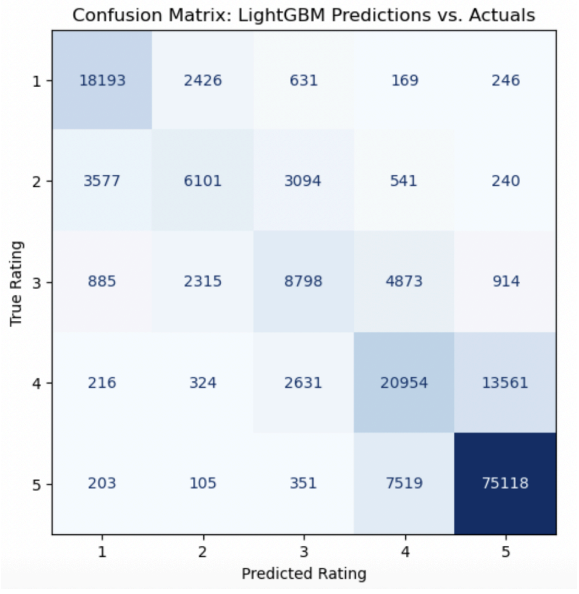
Once confident in our model class, our next task was to try to optimize the LightGBM. Due to the size of our dataset, and the fact that we include n-grams in the range [1,3], it would not be computationally reasonable to run a grid search to optimize. Rather, we ran a randomized search on ~10% of our original training data. Due to the same constraints, we were also not able to be exhaustive in our search space; instead, we limited our search to the hyperparameters most likely to move the needle—learning rate, number of leaves, feature fraction, and

bagging fraction—sampling only ten combinations via RandomizedSearchCV. Even on just a segment of our data, this strategy yielded a small lift: accuracy rose from 0.7401 to 0.7424, and our F1 improved similarly (from 0.7320 to 0.7350). It is important to note that the initial cross validated model was more accurate than even the tuned LightGBM, leading us to believe that if we had the compute required to run exhaustive searches across the entire hyperparameter space with multiple folds, we could achieve even better results.

Similarly, due to the success of XGBoost when initially tested, we suspect that an exhaustive hyperparameter search on this model class would likely surpass the LightGBM model. However, XGBoost is far more computationally expensive than the already tedious (for this task) LightGBM, so we were unable to optimize this model.

Due to similar scores across the best models thus far, we are confident that we are nearing the limits of the selected approach.

Figure 2: Confusion Matrix of LightGBM Output



Neural Networks

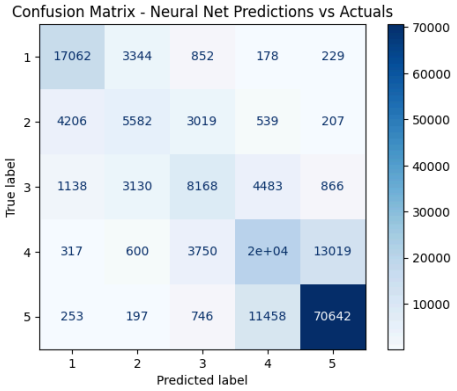
We also trained 4 different neural networks on the normalized text + numeric data to compare with previous tree-based and linear models. We tested 4 model structures, trained over 20 epochs: 1 *'Base'* model with hidden layer sizes 128 and 64 and ReLU activation, which is fairly basic & standard; 1 *'Deeper'* model with hidden layers 256, 128, and 64, plus a 20% Dropout layer and Sigmoid activation; 1 *'Wide-Shallow'* model with a single hidden layer of 512 and ReLU activation; and finally a model that closely resembled the neural network from a Stanford paper[1] predicting Toronto restaurant reviews, with one hidden layer of 100, plus Sigmoid activation of the hidden layer and Softmax for the output. The results from each model are printed below:

	Stanford 3 Layer	**Base**	**Deeper**	**Wide-Shallow**
Accuracy	0.7120	0.6963	0.7097	0.6985
F1 Score	0.7072	0.6927	0.7090	0.6959
MSE	0.4282	0.4635	0.4222	0.4610

The Stanford 3-layer model had the best accuracy, but a lower F1 and MSE than the Deeper model; additionally, none of these neural nets had a better accuracy or MSE than the tree-based LightGBM. We expected that the neural networks would be able to pick up on more signals within the high-volume dataset, but in reality the tree-based models appear both simpler and more robust to the data used.

Figure 3: Confusion Matrix of NN Output

[1]https://cs229.stanford.edu/proj2017/final-reports/5244334.pdf

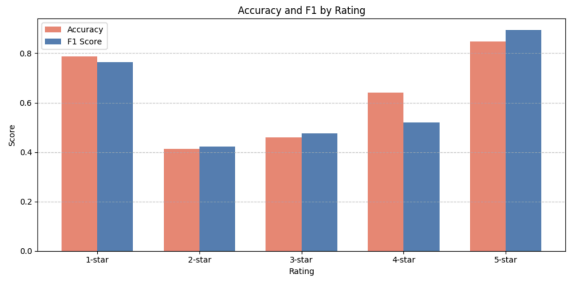


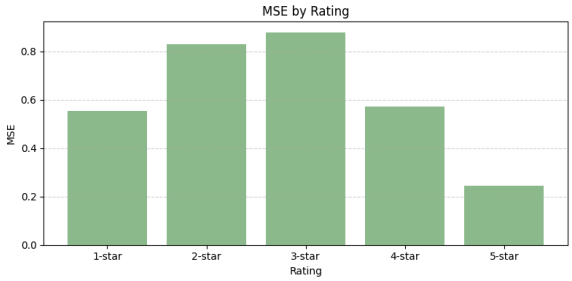
The confusion matrix shows that bad-neutral (2-3) ratings were more often misclassified downwards than upwards, while 4-star ratings were almost always predicted as 5-stars, which hurt the overall accuracy.

It also made sense to analyze accuracy, F1 and MSE by star rating to better understand the best (Deeper) neural net's predictive power – if this were a real model used by Yelp, it could inform the model's use cases and drawbacks. These metrics are shown here, plus visualizations:

	Accuracy	**F1**	**MSE**
1-star	0.787	0.765	0.555
2-star	0.412	0.423	0.830
3-star	0.459	0.476	0.879
4-star	0.640	0.521	0.573
5-star	0.848	0.895	0.244

Figs 4-5: Accuracy, F1, & MSE by Rating





The figures show that the neural net performs best on extreme ratings (1 and 5-star), with accuracies of 78.7% and 84.8% respectively. Middle ratings (2 and 3-star) have the lowest accuracies at 41.2% and 45.9%, with F1 and MSE following the same trends. Overall, this suggests that our model has a tendency to predict the extreme ratings more accurately, a common pattern in sentiment analysis models where neutral sentiments are harder to distinguish. Therefore, we concluded that our models likely over-rely on the non-text data like average user rating or overall restaurant rating for the middling (2-4) star predictions, since sentiment was harder to parse from the highly-important textual data. This may also stem in part from the class imbalances: 5-star was the most common rating, at ~47%, and users are more likely to review a restaurant if they have a great or terrible experience (users' *selection bias*).

LLM-Based Models

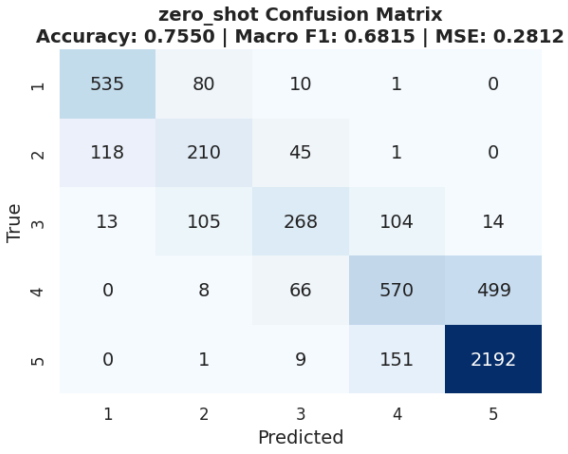
To explore the predictive capabilities of large language models (LLMs) in classifying Yelp star ratings from free-text reviews, we implemented three approaches using OpenAI's gpt-4o-mini model: zero-shot prompting, few-shot prompting, and fine-tuning. Each method was evaluated on a 5,000-review subset selected to balance class distribution while staying within the project's time and cost constraints.

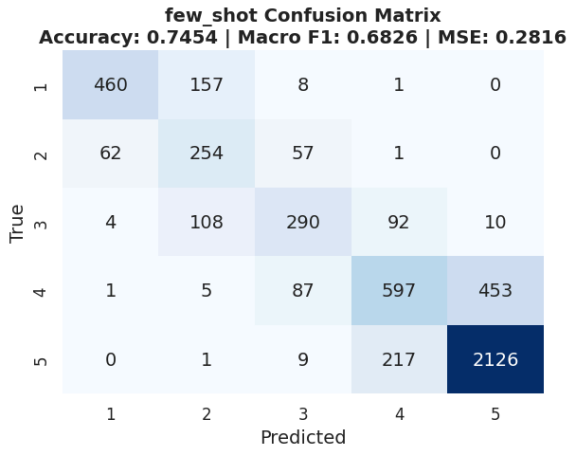
In the zero-shot approach, the model was prompted with a brief instruction to assign a

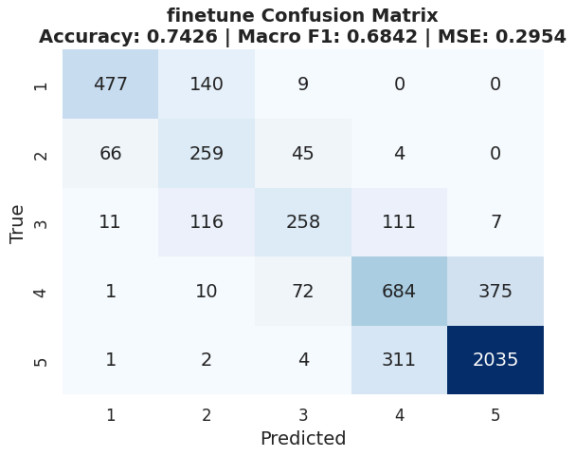
star rating (1–5) based solely on the review text. The few-shot version expanded on this by adding five labeled examples (one for each star level) to guide the model. Finally, for fine-tuning, we trained gpt-4o-mini on 500 labeled reviews and evaluated its custom predictions on the same 5,000-review test set.

We designed our preprocessing pipeline to preserve review-specific signals like punctuation, repeated characters, and capitalization, which are often critical for capturing tone, sarcasm, or emphasis in user-generated content. After implementing this cleaner, more natural approach, the zero-shot model emerged as the best-performing method, achieving 75.50% accuracy, a macro F1 score of 0.6815, and mean squared error (MSE) of 0.2812. The few-shot model followed closely with 74.54% accuracy and nearly identical F1 and MSE scores. The fine-tuned model ranked third with 74.26% accuracy.

Figs 6-8: LLM Model Comparison: Confusion Matrix, Accuracy, F1, & MSE







Conclusion

To summarize, we started with a raw dataset of Yelp reviews for restaurants in the Florida area, then engineered and cleaned the data for use in training models to predict users' star ratings of these restaurants from textual review data as well as various restaurant- and user-specific features. We first trained basic, non-tuned instances of linear regression, One vs Rest (OVR) Logistic Regression, and tree-based LightGBM to evaluate baseline performance; and in the following weeks used more complex neural networks with varying architectures, as well as large language model (LLM)-based models from OpenAI to arrive at the most accurate predictions possible. Ultimately our highest-performing models were

LLM-based; the top 4 models are shown from most- to least-accurate below:

	LLM Zero-Shot	LLM Few-Shot	LLM Fine-Tuned	LightGBM (Tuned)
Accuracy	0.7550	0.7454	0.7426	0.7424
F1	0.6815	0.6826	0.6842	0.7350
MSE	0.2812	0.2816	0.2954	0.3072

Among LLM-based models, zero-shot LLMs remained the best, reaching 75.5% accuracy and the lowest MSE of 0.2812. Interestingly, despite slightly lower accuracy scores, the fine-tuned LLM had the best F1 score among the LLMs, indicating more balanced performance across classes. However, LightGBM, while narrowly trailing in accuracy (74.2%), achieved the highest F1 score (0.7350), suggesting it was better at capturing minority classes—critical in an imbalanced dataset. These findings highlight a tradeoff: while LLMs may better generalize overall sentiment, gradient boosting models like LightGBM may be more reliable in consistently identifying less common star ratings. For platforms like Yelp, this means combining LLMs for robust sentiment interpretation with structured models like LightGBM for class-level calibration could lead to the most trustworthy star prediction system.

We note, though, that the LLM-based models were trained *only on text data*, no numeric features, and even the lowest-performing LLM approach (fine-tuned) exceeded LightGBM. This suggests that LLMs, even with minimal supervision, are strong baselines for review classification when working with text alone. Their ability to interpret context, tone, and subtle language cues gives them a distinct advantage in cases where structured features are limited or unavailable. With more representative few-shot examples or additional fine-tuning, LLMs may consistently outperform traditional models in similar tasks.

Hyper-parameter tuning the best-performing initial model (LightGBM) using Grid Search gave only 0.002 lift in accuracy vs. the base model, with nearly zero lift in F1 score; tuning the model parameters did not materially change the model's predictive, and was still a robust and powerful predictor from the ground up. With more compute, we would hope to achieve more notable gains.

It was also apparent that the models did not have equal prediction strength across star rating levels. Predictions for the neural net had the highest accuracy for 1- and 5-star reviews (~79 and ~84% vs ~70% overall) and the lowest MSE (0.55 and 0.22 vs. ~0.4 overall), which was an interesting result stemming from the fact that neutral sentiment is not as easy to distinguish as positive or negative sentiment. Additionally, 5-star ratings were by far the most common rating, at about 47%, followed by 4-star at 21% and 1-star at 12%, which indicated a class imbalance that may have shifted the predictions unequally towards 5-stars.

Overall, we are proud of the progress and results we achieved. Our models demonstrated strong predictive performance, especially given the complexity and imbalance of real-world review data. The success of modern language models—particularly in zero-shot and few-shot setups—shows how far general-purpose LLMs have come in understanding nuanced sentiment and applying it to structured outputs like star ratings. At the same time, traditional models like LightGBM were strong, offering more consistent performance across all rating classes. This balance between cutting-edge NLP and older structured methods reinforces that hybrid modeling approaches may be the most effective path forward.

Ultimately, our work not only achieved solid technical outcomes but also offered meaningful insight into sentiment-based star prediction—an area with real applications for platforms like Yelp, and one that presents a rich challenge for future machine learning research.

Contributions

Sofia: Cleaned and imputed the data, performed feature engineering, implemented random forest and xgboost classifiers, wrote the abstract and background of the report.

Maria: Investigated data sources and ruled out the Yelp API, saving and processing the academic dataset, performed initial EDA and data cleaning, implemented and evaluated three LLM-based models (zero-shot, few-shot, fine-tuning), created visualizations, helped assist with tuning on LightGBM model, and wrote the Data Cleaning and LLM sections of the report.

Matt: Vectorized text data, implemented various initial models (OVR/Multinomial Logreg, LinReg, LightGBM, ran CV on base LightGBM, ran randomized search on LightGBM for hyperparameter tuning, attempted tuning XGBoost

Alex: Conducted exploratory data analysis (EDA), feature selection with Random Forest, trained and evaluated neural networks, created visualizations and analysis of prediction discrepancies by star rating, wrote Neural Network, EDA, & Conclusion sections.

Addendum

1. Project Folder including slides, all code, and video

References

1. https://cs229.stanford.edu/proj2017/final-reports/5244334.pdf