

Sentiment Analysis of Amazon Fine Food Reviews: A Comparative Study of Predictive Models

Helen Ton Chang, Nathan Chen, Cristina Ochoa, Shaojie Zhang, Harshdeep Singh

Motivation

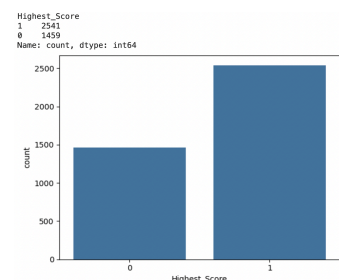
In today's society, e-commerce platforms like Amazon have revolutionized the way people shop. Unlike the traditional route of looking at a product in person and using it to test how good it is, most people nowadays tend to look at the reviews of a product. Through reading, they find what is and is not a good product. The most common rating metric being the number of stars out of 5, with 1 star meaning the product is really bad and a 5 star meaning the product is really good. However metrics like these fail to capture the sentiment of the consumers, especially in industries like food, where quality, taste, and personal preferences play a significant role in the ratings and purchase of the products. Additionally, automated and bot-generated ratings have made it harder to understand real customer questions. This project uses sentiment analysis on the Amazon Fine Food Reviews dataset to better understand customer opinions. By looking at the text of reviews, we aim to find useful patterns that star ratings alone don't show. Specifically, we focus on spotting a highest score (of 5) vs non-highest score sentiment in reviews to understand how customers feel about food products on Amazon.

Impact

This project can help both customers and businesses in important ways. Customers can use sentiment analysis to spot important keywords or themes in reviews, such as "fresh", "tasty", or "poor packaging," to quickly understand what others are saying about a product. These keywords help them focus on the aspects they care about the most, making it easier to decide whether a product meets their needs without reading through hundreds of reviews. For businesses, the analysis highlights the word or phrases customers mention most often that would result in the seller getting the highest rating, giving sellers insights into what drives positive or negative feedback. This allows them to improve their products by addressing common complaints or emphasizing features customers love. Products with consistently positive reviews also gain higher visibility on Amazon, which brings in more customers and increases sales.

Data Collection

For the data collection phase, we utilized the Amazon Fine Food Reviews dataset from Kaggle, which contains approximately 500,000 Amazon reviews spanning over a decade, up to October 2012. This dataset includes detailed information such as product and user details, numerical ratings (1–5), plain text reviews, timestamps, and reviews from other Amazon categories. For our purposes we only need to look at the ratings 'Score' and the customer reviews 'Text'. Its comprehensive nature and large size made it an ideal choice for our analysis, offering opportunities for diverse analytical approaches like sentiment analysis and consumer behavior studies. Because of its large size and our working environments' limitations, we imported 4000 rows for training, validation, and testing. Our preparation of the data and the exploratory analysis in this case was mostly done through text analysis. Firstly, we confirmed that there were no null entries. Then, we converted the 'review' column into being lower case, no punctuation and no digits involved. We were then able to delete the 'stopwords' and also did stemming to convert words into their base English form. Finally we produced our document term matrix (dtm2) with only words that appear in at least 1% of the reviews. We tested a couple parameters and found that this parameter of 1% did not result in much overfitting. Our models' features are these word stems and our dependent variable is predicting if ratings are Highest_Score or not (meaning a score of 5 or not) to be Highest_Score. Thus our problem becomes a binary classification task.

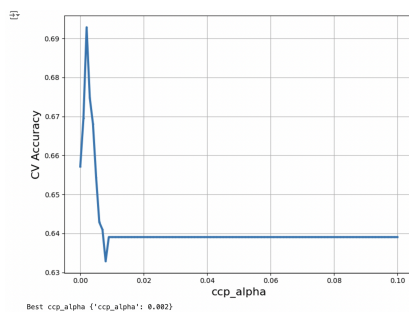


Models

1. Logistic Regression

We chose Logistic Regression because it is simple, efficient, and works well for binary classification tasks. It uses a linear approach to identify the relationship between features and the target outcome, making it quick to train and easy to interpret. For our dataset, Logistic Regression achieved an accuracy of 73.22%, meaning it correctly classified reviews as positive or non-positive 73% of the time. It had a true positive rate (TPR) of 79.49%, which shows it effectively recognized positive reviews, but its false positive rate (FPR) of 36.44% indicates it struggled to distinguish some negative reviews. It was thresholded at .6 for classification (determined through checking the training set accuracy), which improved the balance between precision and recall by prioritizing the accurate identification of positive reviews and thus predicting more 0's. Although Logistic Regression performed reasonably well, its linear nature limits its ability to capture complex patterns in the data, especially when dealing with interactions between features. Compared to other models like Random Forest, it is less capable of leveraging subtle and nonlinear relationships within the text data and has a high likelihood of overfitting. However, its simplicity and efficiency make it a solid baseline for understanding overall trends in sentiment classification. Logistic Regression provides valuable insights quickly, even if it lacks the depth and flexibility of more advanced models.

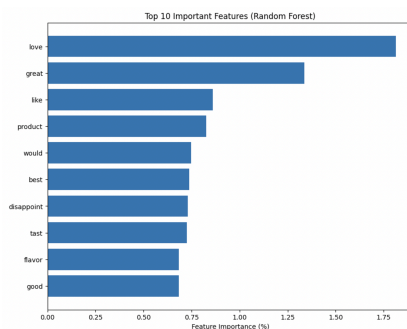
2. CART



We selected CART as a starting point towards our analysis mainly because of its simplicity and interpretability. The training of this model required understanding what the best ccp_alpha parameter was, which for our dataset turned out to be 0.002, and by tuning our hyperparameter using cross validation to optimize for tree depth and minimum samples per split. In order to achieve this we used the DecisionTree Classifier and the GridSearchCV, ultimately reaching a 79 node decision pruned tree with insights on our data which is very interpretable and easy to understand. When put in context, it is interesting to see the key features (words) in which the splits were made, starting with 'love'. In this case, the simplicity of the model might have been

the cost of the lower accuracy out of all the ones we used, at only 0.6611.

3. Random Forest



We chose Random Forest for its robustness and ability to handle complex relationships in text data. By fine-tuning hyperparameters like $n_estimators$ and max_depth using GridSearchCV with 3-fold cross-validation, the model achieved strong results: an accuracy of 76.00% and an impressive TPR of 92.31%. However, its FPR of 49.15% indicates a tendency to over-predict positive reviews. The model's feature importance analysis highlighted "love" as the top feature, frequently appearing in the highest positive-score-of-5 reviews. This aligns with expectations, as such words strongly signal positivity. While Random Forest outperformed simpler models, its higher FPR suggests room for improvement in balancing

precision and recall.

4. Gradient Boosting

We chose this model because in general Gradient Boosting Classifier performs well with sparse, high-dimensional data like our document-term matrix (dtm2), especially after the feature engineering we

performed. The GradientBoostingClassifier is trained on the word frequencies from our document term matrix (DTM2) to build trees to sequentially improve the predictions; working iteratively to minimize classification errors by learning from residuals. The final model combines the learned trees to output probabilities for whether a review is positive or not. Cross-validation was performed using GridSearchCV with a 3-fold cross-validation strategy to identify the best hyperparameters. 3 folds was chosen to reduce the computational load for better runtime. The accuracy is 0.743.

5. Blending

OLS Regression Results						
Dep. Variable:	Highest_Score	R-squared (uncentered):	0.721			
Model:	OLS	Adj. R-squared (uncentered):	0.720			
Method:	Least Squares	F-statistic:	643.6			
Date:	Mon, 09 Dec 2024	Prob (F-statistic):	2.78e-274			
Time:	19:08:00	Log-Likelihood:	-567.52			
No. Observations:	1800	AIC:	1143.			
Df Residuals:	996	BIC:	1163.			
Df Model:	4					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
y_pred_lg	0.2293	0.044	5.172	0.000	0.142	0.316
y_pred_cart	0.1774	0.031	5.648	0.000	0.116	0.239
y_pred_rf	0.3118	0.050	6.216	0.000	0.213	0.410
y_pred_gbc	0.1160	0.047	2.471	0.014	0.024	0.208
Omnibus:	48.990	Durbin-Watson:	1.758			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	55.033			
Skew:	-0.568	Prob(JB):	1.12e-12			
Kurtosis:	3.179	Cond. No.	7.36			

smf.logit, we were actually able to better the results on our validation set given that our base model was already the logistic regression model. Every model being a feature had a p-value less than 0.015 indicating every model was significant in our blending model. By doing this we were able to enhance the predictive accuracy, reducing a bit of the bias and variance. However, some of the limitations of this model would be to overfit without proper validation and losing a bit of interpretability.

Model Comparison

	Accuracy	TPR	FPR
Model			
Baseline	0.606667	1.000000	1.000000
Logistic Regression	0.732222	0.794872	0.364407
CART with CV	0.661111	0.739927	0.460452
Random Forest with CV	0.760000	0.923077	0.491525
Gradient Boosting with CV	0.743333	0.879121	0.466102
Blending	0.768889	0.904762	0.440678

Model Accuracy, TPR, and FPR were all computed using the test set. These metrics allow us to evaluate how well the models generalize to unseen data, focusing on both their predictive power and their ability to minimize critical classification errors. Mistakenly classifying FP’s vs FN’s is not life-threatening for example scores of 4 is still a positive review close to 5. Accuracy is a good metric to know the predictability of our model and we also consider FPR and TPR to account for the slight class imbalance of our data set.

The Blending Model achieved the highest accuracy at 0.7689. It was the most successful because it combined the strength of all the other models-Logistic Regression, CART, Random Forest, and Gradient Boosting-through an OLS regression approach that assigned optimal weights to each model’s predictions. By doing so it reduced the bias and variance associated with individual models, creating a more balanced and reliable classifier. Additionally, it had a strong performance which was demonstrated by it having the second highest TPR behind Random Forest and second lowest FPR behind Logistic Regression.

Random Forest with Cross-Validation had the highest TPR at 0.9231(92.31%) indicating that it was the best at correctly identifying reviews with highest scores (rating = 5). Its ensemble nature, where multiple decision trees vote on the prediction, allows it to effectively capture patterns in the text data, even when individual trees make errors. Feature importance analysis further validated this model’s strength, with words like “love” strongly correlating with positive reviews. However, this strong focus on identifying positive reviews came at the cost of an FPR of 49.15%, which was the highest among all models. This means that it frequently misclassified non-positive

reviews as positive, likely due to its bias toward the majority class of positive reviews. While this model demonstrated effectiveness in capturing sentiment, its high FPR limits its applicability in situations where false positives must be minimized.

Gradient Boosting showed a more balanced performance with an accuracy of 74.3% slightly outperforming Random Forest in overall classification. By iteratively improving predictions through learning from residual errors, this model effectively captured detailed patterns in the high-dimensional document-term matrix. However, its FPR of 46.61% revealed challenges in distinguishing non-positive reviews, similar to Random Forest. The iterative nature of this model made it computationally intensive, particularly during hyperparameter tuning, which poses practical limitations in large-scale applications.

Logistic Regression served as a strong baseline model, achieving an accuracy of 73.22% with the lowest FPR at 36.44%. This makes it a reliable choice in applications where minimizing false positives is crucial. Its linear nature however limited its ability to capture complex relationships in the data, resulting in a TPR of 79.49%-lower than ensemble models like Random Forest. Despite its simplicity, Logistic Regression proved to be efficient and interpretable, making it a valuable starting point for sentiment analysis tasks.

The CART model, while interpretable and computationally efficient, demonstrated the weakest performance among the models, with an accuracy of 66.11% and the lowest TPR. Its simplicity limited its ability to generalize well to unseen data. This outcome highlights the trade-off between interpretability and predictive power, particularly for single-tree models.

Overall the Blending model was the best overall performer, leveraging the complementary strengths of individual models. Random Forest performed well in identifying positive reviews but struggled with false positives, while Gradient Boosting offered a balanced approach with slightly higher computational demands. Logistic Regression provided a reliable and interpretable baseline, and CART highlighted the value of simplicity at the cost of predictive accuracy. Future improvements, such as addressing class imbalance through techniques like SMOTE or incorporating advanced natural language processing methods, could further enhance the performance of these models.

Conclusion/Future improvements

We can argue that we had success with our analysis in terms of our initial goal. On the test set, our best model, Blending, achieved a 16% increase from the baseline. The impact of our report lies in the high accuracy of sentiment analysis through different models, but with a shared purpose. By understanding how important sentiment analysis is for something so simple as a review, it makes us think of how this could be taken further and explained in an industry like online sales in which product producers are given feedback mostly through reviews, and consumers are basing their buying decisions on the same reviews, meaning the convergence of these towards a positive tone would be highly beneficial. If we could take this further, it would be interesting to see the connection to business decisions and profit made after changes are made. In terms of sentiment analysis it would also be very interesting to see the amount of 'bot' reviews and analyze how that affects our findings. Another improvement that we could have explored would be to convert the reviews into sentence embeddings with something like Tensorflows Universal Sentence Encoder into high dimensional vectors, then use this as the training set as opposed to our Document Term Matrix. This way it may capture contextual or more nuanced meanings than our current method.

In addition to our current analysis, an area for future improvement could be incorporating customer behavior into the model. By analyzing user behavior patterns, such as their historical average rating, rating variance, and review frequency, we could provide the model with valuable context to better interpret sentiment. For instance, if a user

consistently gives 4-star reviews regardless of the product, their ratings may carry less weight compared to someone with more varied scores.

Currently, I've started implementing this approach by extracting features like the user's historical average score, review frequency, and rating variance. I've also converted these features into descriptive text (e.g., "This user has given an average rating of 4.5 with low variance") and combined them with the actual review content. This combined input is then used in a fine-tuned BERT model for sentiment classification. This method enables the model to not only analyze the review text but also incorporate behavioral patterns for deeper and more accurate sentiment predictions. If this approach continues to show promise, it could address limitations in purely text-based analysis and significantly enhance the model's performance.

Appendix

<https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews> We found our dataset on Kaggle.