# Sentiment Analysis of Amazon Product Reviews

Cristian Aponte, Sandip Singh, Rhea Perumalil

April 30, 2025

## 1 Introduction

Online customer reviews have become crucial when it comes to understanding consumer purchasing behavior on E-commerce platforms such as Amazon, Yelp, and IMDB. However, these platforms receive millions of product reviews and manually processing and analyzing the textual data is not efficient at the moment. Furthermore, customer reviews often have specific language, slang, sarcasm, and mixed sentiment which makes classification of these reviews difficult. Businesses need a way to extract these insights from customer reviews to improve services.

In this project, we aim to use sentiment analysis on an Amazon Reviews Dataset to classify positive, neutral, and negative reviews using a combination of rule-based and machine learning techniques. We start of by using VADER as a baseline model and compare it several other supervised learning models such as Logistic Regression, Support Vector Machine, Random Forest, and XGBoost. Each of these models will categorize the data into positive, neutral, and negative reviews based on their sentiment.

Exploratory analysis on the data has shown that we are working with an imbalanced dataset and models may have a hard time specifically classifying neutral reviews not only because it is the minority class, but because of nuance and specific vocabulary. Our goal and contribution lies in building and evaluating these machine learning models and experimenting with data preprocessing, class-weighting strategies, and hyperparameter tuning to create an accurate model.

Ultimately, our goal is to create a sentiment classifier that can provide interpretable insights into sentiment and language patterns found in customer product reviews.

### Contribution

We contribute a data pipeline that will compare rule-based(VADER) and machine learning(SVM, Logistic Regression, Random Forest, and XGBoost) sentiment models, applying advanced text preprocessing, and resolving class imbalance using reweighting and SMOTE. We will then demonstrate which models perform best on large datasets and then we will analyze those models.

## 2 Methods

### 2.1 Data Cleaning and Preprocessing

We used the Amazon Reviews 2023 dataset from GitHub. The Reviews were sampled from three categories: Grocery, Clothing, and Electronics, with 500,000 reviews from each, totaling 1.5 Million rows.

**Data Cleaning Steps:**

First, we feature engineered a new variable called `full_text`, which combined the review title and main text of the review into a single column for analysis. We then removed duplicate entries and any reviews with null values, resulting in a cleaned dataset of approximately 640,000 reviews. Next, we converted all review text to lowercase strings and removed URLs, HTML tags, non-alphabetic characters, and punctuation. To improve generalization, we used the Natural Language Toolkit (NLTK) to remove common English stopwords such as “the” and “to.” We further standardized the text using spaCy for lemmatization, converting words to their base forms, for example “running” would become “run.”. Lastly we one-hot encoded the variable

`verified_purchase` to 1's and 0's. This complete preprocessing and data cleaning pipeline produced a final cleaned column, `full_clean_text`, which was used for text vectorization and modeling.

## 2.2 Sentiment Labeling, Tokenization and Feature Engineering

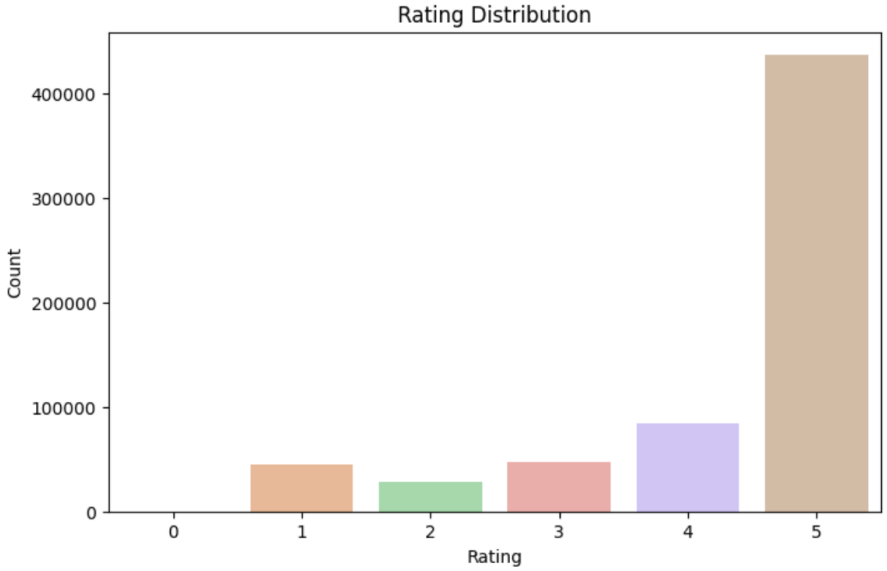
A crucial step to take before beginning with our classification models is to map numerical rating to sentiment labels. Since our ratings range from 1 to 5, we categorized reviews with 1 or 2 star reviews as negative, 3 star reviews as neutral, and 4 or 5 star reviews as positive. This labeling helped us turn the 1-5 star rating scale into clear sentiment categories that our models could learn from.

Next, we converted the review text into numerical features using TF-IDF (Term Frequency Inverse Document Frequency). This method works by highlighting the most important words in each review. It does this by reducing the influence of words that show up in many reviews, and highlightings those that are more unique or meaningful. We limited it to 8,000 features in order to keep the model efficient and to avoid high dimensionality.

Some additional parameters were used to fine-tune the feature space. This includes setting ngram_range=(1, 2) to include both unigrams and bigrams, which helps the model capture short phrases like "not good" or "works well" that might carry sentiment. Additionally, to reduce noise, we used min_df=3, which ignores rare words that appear in fewer than three reviews. We also used max_df=0.9 to filter out words that are too common and appear in more than 90% of documents, such as "the" or "this." These filtering steps help ensure that the model focuses on informative and distinctive terms while discarding those that are too rare or too generic to be useful.

## 2.3 Class Imbalance Handling

A challenge we encountered in our project was the significant imbalance between sentiment classes. As seen below, the majority of reviews were labeled as positive, with only very few negative and neutral ones:




To address class imbalance, we first applied class_weight='balanced' in our models, which adjusts class importance based on frequency. We also experimented with manually increasing the weight of the neutral class to encourage better performance on underrepresented labels.

In addition to weighting strategies, we explored two data-level balancing techniques. We used SMOTE (Synthetic Minority Oversampling Technique) to simulate examples for minority classes.

We also applied stratified downsampling, which reduces the size of the majority class while preserving equal representation across all classes. This ensured a balanced training set and gave underrepresented classes more influence during model training.

## 2.4 Baseline: VADER

In order to have something to compare our Machine Learning models to, we utilized VADER. VADER is a pre-trained model that is specifically designed to analyze the sentiment of certain texts. It achieves a sentiment score by analyzing certain words, ranks importance based off capitalization and frequency, then outputs a positive neutral or negative. Our goal is to outperform VADER in our trained models, so having VADER as a baseline gives us an idea of how good, at minimal, our model should be. We compared VADER predictions to rating-based sentiment labels and show how well the model performs here.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| negative | 0.56 | 0.37 | 0.45 | 73172 |
| neutral | 0.13 | 0.09 | 0.11 | 47828 |
| positive | 0.87 | 0.94 | 0.90 | 520782 |
| accuracy | | | 0.81 | 641782 |
| macro avg | 0.52 | 0.47 | 0.49 | 641782 |
| weighted avg | 0.78 | 0.81 | 0.79 | 641782 |

VADER performs well on positive reviews (F1=0.90) but poorly on neutral (F1=0.11). It is shown that VADER fails to identify sarcasm and in addition to this it is hard to identify whether something is "between" a positive and negative review (i.e. neutral review)

## 2.5 Model Training

We trained and tested:

- Logistic Regression (class-weighted)
- Support Vector Machine (SVM) with grid search tuning
- Random Forest
- XGBoost with grid search and tuned hyperparameters

TF-IDF features were used for all models. Stratified train-test split (80/20) ensured class proportions were maintained.

### 2.5.1 Logistic Regression

We trained a multinomial logistic regression model using the TF-IDF vectorization as input features. To address the class imbalance, our model used SMOTE (Synthetic Minority Over-sampling Technique), which essentially generates new samples for the underrepresented class in the training set. Additionally, we used grid-search over the regularization parameter C, identifying C=0.1 as the optimal value to generalize well and reduce overfitting. Overall, this model performed well for the positive class, but still exhibited weaker recall on the neutral class, which suggests more fine tuning or resampling is needed to improve performance scores across all the sentiment classes.

### 2.5.2 Support Vector Machine (SVM)

Initially, all we did was train the SVM model using the TF-IDF vectorization and this model performed poorly. In order to address the poor performance and optimize the importance, it was essential to establish where it was going wrong. A significant issue was the neutral class, so in efforts to combat this, we began with a class-weighted SVM model to address this. Then, we performed a GridSearchCV procedure to tune hyperparameters such as Penalty, Loss Function, and Class Weight, optimizing based on highest F1 Score. Once we identified the best parameters, we performed cross-validation to evaluate the model. To attempt

to tackle class imbalance more, we experimented with SMOTE to use synthetic samples, but our model decreased in all metrics. In efforts to try something else, we decided to find the most effective configuration of weights, by looping through a range of class weight values (ranging from 1 to 15), particularly increasing the weight on the poorly performed neutral class. This iterative approach allowed us to identify the pattern that yielded the best balance of precision and recall across all classes (Positive: 1, Negative: 2, Neutral: 4).

#### 2.5.3 Random Forest

We trained and evaluated a Random Forest classifier using two feature representations. The initial model with TF-IDF vectorization performed poorly in classifying underrepresented sentiment labels due to high dimensionality and class imbalance. To improve this, we used sentence embeddings from a pre-trained transformer model, which provided informative, context-aware input features.

To further address class imbalance, we applied stratified downsampling to ensure equal representation of each sentiment class. Although this led to a reduction in overall accuracy, it significantly improved the model's ability to detect all three sentiment categories, especially improving recall for the neutral and negative classes.

#### 2.5.4 XGBoost

We trained and evaluated an XGBoost model due to its strong track record in data classification and its ability to handle imbalanced classes through both built-in and manual weighting options. After running our model with TF-IDF vectorization, we needed to improve the F1 score so in order to do this we introduced multiple different ideas. We first performed a hyperparameter search using GridSearchCV, tuning key parameters such as learning rate, maximum tree depth, number of estimators, and subsampling ratios. This allowed us to identify the best combination that maximized averaged F1 score, ensuring better performance across all classes. Additionally, in order to account for the poor performance on the neutral class, we experimented with manually reweighting the training samples using a class-specific sample weight array, which gave more influence to underrepresented classes during training (Negative and Neutral). This two-step approach helped improve generalization and reduce bias toward the best performing positive class.

## 3 Experiments

To evaluate the effectiveness of our sentiment classification models, we trained and tested multiple classifiers using the same TF-IDF features and stratified 80/20 train-test split. Each model was evaluated based on accuracy, macro-averaged precision, recall, and F1-score to account for class imbalance across the three sentiment classes: positive, neutral, and negative.

### 3.1 Performance Comparison

Table 1 summarizes the results across all models tested. Macro-averaged metrics were used to balance the contribution of all three classes, especially given the dominance of positive sentiment in the data due to class imbalance.

Table 1: Model Performance Comparison (Macro-Averaged Scores)

| **Model** | **Accuracy** | **Macro Precision** | **Macro Recall** | **Macro F1** |
|---|---|---|---|---|
| VADER | 0.81 | 0.52 | 0.47 | 0.49 |
| Logistic Regression | 0.81 | 0.59 | 0.66 | 0.62 |
| SVM (Manual Weights) | 0.87 | 0.67 | 0.62 | 0.64 |
| Random Forest | 0.86 | 0.66 | 0.51 | 0.54 |
| Stratified Downsample | 0.53 | 0.52 | 0.53 | 0.53 |
| XGBoost (Tuned) | 0.87 | 0.69 | 0.55 | 0.59 |

As shown, the manually weighted SVM and the tuned XGBoost model performed the best in terms of macro F1. These models achieved high accuracy while maintaining strong performance across all sentiment classes. As shown in the table, models that incorporated manual class balancing (e.g., SVM, XGBoost) outperformed those without adjustment, especially in detecting neutral sentiment.

### 3.2 Observations

Among the models evaluated, SVM with manually tuned class weights achieved the highest overall performance with the best overall macro-averaged F1 score and accuracy. While XGBoost achieved similar accuracy, its inability to detect neutral sentiments led to a decrease in the macro-average F1 score. The Logistic Regression Model showed a high and strong performance on the positive class, but like many of the other models, it underperformed in the neutral class despite using SMOTE. The Random Forest Model portrayed similar behavior with high accuracy but pretty low F1 scores due to being less effective with the minority class. VADER also performed competitively, achieving high accuracy, but exhibited issues similar to many of the other models.

### 3.3 Insights

The main problem across all of the models was the class imbalance, specifically the imbalance of the neutral reviews. However, after training multiple Machine Learning models, we noticed that neutral reviews are difficult to detect, in general. This may be because there are no significant words that specifically go to "Neutral" and, in our case, "Neutral" ratings are only 3 stars, making it hard to identify.

VADER and Logistic Regression predicted positive sentiments well due to the large presence of 4 and 5 star reviews in the dataset. While Logistic Regression showed high precision for negative and positive classes after TF-IDF transformation, the recall failed to increase.

Support Vector Machines (SVM), when paired with specific class weights and hyper-tuning using grid search, delivered the best macro F1 scores, suggesting that it was our most robust model.

Random Forest gave us a much better balance across sentiment classes after using sentence embeddings and stratified downsampling. Although the overall accuracy was lower compared to the TF-IDF model, it was able to pick up on neutral and negative reviews more effectively. This trade-off was important because it showed the model could recognize all types of sentiment instead of just favoring positive ones.

XGBoost achieved the highest precision and recall for positive and negative reviews but kept doing poorly in neutral detection despite attempting hyper-parameter tuning and manually balancing the classes.

Ultimately, this indicates that tree-based models, while powerful, may require additional tuning to better capture specific sentiment. Overall, no single model dominated all metrics, highlighting the difficulty of multiclass sentiment tasks and the need for different strategy when working with imbalanced and noisy datasets.

## 4 Conclusion

We built an extensive sentiment analysis pipeline using Amazon Reviews and explored multiple models, all aiming to classify sentiment in Amazon product reviews. After tuning the models and doing an in-depth evaluation, we concluded that SVM with class reweighting outperformed all of our other approaches in terms of balance and generalizability. However, we observed that neutral sentiment remains difficult to capture. Even our best performing model struggled significantly to classify neutral reviews. This is likely due to class ambiguity and overlapping language with other classes. Though it might be difficult due to the vagueness of neutral reviews, future work can help improve classification in this are. This would involve fine-tuning BERT or using ensemble models to boost minority class prediction.

# References

[1] Amazon Reviews 2023 Dataset. https://amazon-reviews-2023.github.io/

[2] I. Chakraborty, R. Pandey, and V. Bansal, "A comparative analysis of machine learning techniques for sentiment analysis," in *Proc. Int. Conf. Advances Comput., Commun. Informat. (ICACCI)*, 2018, pp. 2373–2378. https://ieeexplore.ieee.org/document/8376299