SentiMind: A Comprehensive Analysis of Sentiment Classification on Amazon Fine Food Reviews

WiDS 2025 Final Project Report

Prepared by:

24B0981 Divyaansh Narkhede

GitHub Repository Link:

https://github.com/divyaanshNarkhede/WiDS$_2$025$_S entiMind_F inal_P roject$

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background and Motivation

In the digital marketplace, customer reviews serve as a primary mechanism for trust and quality assurance. A single product on Amazon may receive thousands of reviews, generating a volume of data that is impossible to analyze manually. For businesses, this unstructured text contains vital insights: Are customers complaining about shipping times? Is the flavor profile of a new snack too salty?

Natural Language Processing (NLP) offers a solution by automating the extraction of sentiment from text. By classifying reviews into sentiments (Positive, Negative, Neutral), businesses can automatically flag critical issues or identify successful products.

1.2 Project Objectives

The primary objective of the *SentiMind* project was to build a machine learning pipeline capable of:

- **Data Ingestion:** Efficiently loading and cleaning raw text data from the Amazon Fine Food Reviews dataset.
- **Text Preprocessing:** Applying standard NLP techniques to normalize text, including stemming and stopword removal.
- **Feature Engineering:** Converting text into numerical representations using statistical (TF-IDF) and semantic (Word2Vec) methods.
- **Model Evaluation:** Training and comparing the performance of various supervised learning algorithms.

1.3 Report Structure

This report is organized as follows: Section 2 provides the theoretical background for the algorithms used. Section 3 details the Exploratory Data Analysis (EDA). Section 4 outlines the methodology for preprocessing and feature extraction. Section 5 presents the experimental results and error analysis. Finally, Section 6 concludes the study and suggests future improvements.

2 Theoretical Background

Before discussing the implementation, it is essential to define the mathematical and conceptual foundations of the techniques employed in this project.

2.1 Feature Extraction Techniques

2.1.1 Bag-of-Words (BoW)

The Bag-of-Words model is the simplest representation of text. It creates a vocabulary V of all unique tokens in the corpus. A document d is then represented as a vector $v \in \mathbb{R}^{|V|}$, where v_i is the count of term t_i in document d. While efficient, BoW ignores grammar and word order (the "bag" assumption) and results in highly sparse vectors.

2.1.2 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF improves upon BoW by weighing terms based on their importance. It is the product of two statistics:

- **Term Frequency (tf):** Measures how frequently a term occurs in a document.
- **Inverse Document Frequency (idf):** Measures how important a term is. While computing tf, all terms are considered equally important. However, certain terms, such as "is", "of", and "that", may appear a lot but have little importance.

Mathematically, the weight $w_{i,j}$ for a term i in document j is calculated as:

$$tf_{i,j} = \frac{\text{count}(t_i, d_j)}{\sum_k \text{count}(t_k, d_j)} \tag{1}$$

$$idf_i = \log\left(\frac{N}{|\{d \in D : t_i \in d\}|}\right) \tag{2}$$

$$w_{i,j} = tf_{i,j} \times idf_i \tag{3}$$

where N is the total number of documents. This technique was a core component of the feature extraction pipeline.

2.1.3 Word Embeddings (Word2Vec)

Unlike BoW and TF-IDF which produce sparse vectors, Word Embeddings represent words as dense vectors in a continuous vector space. The key idea is distributional semantics: words that appear in similar contexts have similar meanings. Word2Vec uses a shallow neural network to learn these embeddings. In this project, the Gensim library was utilized to train a Word2Vec model with a dimensionality of 100.

2.2 Machine Learning Classifiers

2.2.1 Logistic Regression

Despite its name, Logistic Regression is a linear classification algorithm. It models the probability that a given input x belongs to a class $y = 1$ using the sigmoid function σ:

$$P(y = 1|x) = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}} \tag{4}$$

It is highly effective for high-dimensional sparse data, such as text data vectorized by TF-IDF.

2.2.2 Support Vector Machine (SVM)

SVMs are powerful supervised learning models that look for a separating hyperplane that maximizes the *margin* between two classes. For non-linearly separable data, kernel functions can be used, though a Linear SVC was employed here, which is often sufficient for text classification due to the high dimensionality of the feature space.

2.2.3 Random Forest

Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the Random Forest is the class selected by the majority of trees. It uses a technique called *Bagging* (Bootstrap Aggregating) to reduce variance and prevent overfitting.

2.2.4 Gradient Boosting

Gradient Boosting is another ensemble technique that builds the model in a stage-wise fashion. Unlike Random Forest which builds trees independently, Gradient Boosting builds trees sequentially, where each new tree attempts to correct the errors (residuals) of the combined ensemble of previous trees.

3 Exploratory Data Analysis (EDA)

3.1 Dataset Overview

The dataset consists of 50,000 reviews extracted from the Amazon Fine Food Reviews dataset. Each sample contains the raw text of the review and a numerical score (1-5 stars).

3.2 Target Variable Engineering

The raw scores were converted into three sentiment categories to formulate a classification problem:

- **Positive:** Scores 4 and 5
- **Negative:** Scores 1 and 2
- **Neutral:** Score 3

3.3 Class Imbalance Analysis

A critical finding during EDA was the severe class imbalance. As shown in Figure 1, the dataset is dominated by positive reviews.

- **Positive:** 38,418 samples (76.8%)
- **Negative:** 7,535 samples (15.1%)
- **Neutral:** 4,047 samples (8.1%)

This 77% baseline accuracy (the accuracy achieved by a model that simply predicts "Positive" for everything) sets a high bar for the models. It also highlights the difficulty of learning the minority classes (Neutral and Negative).

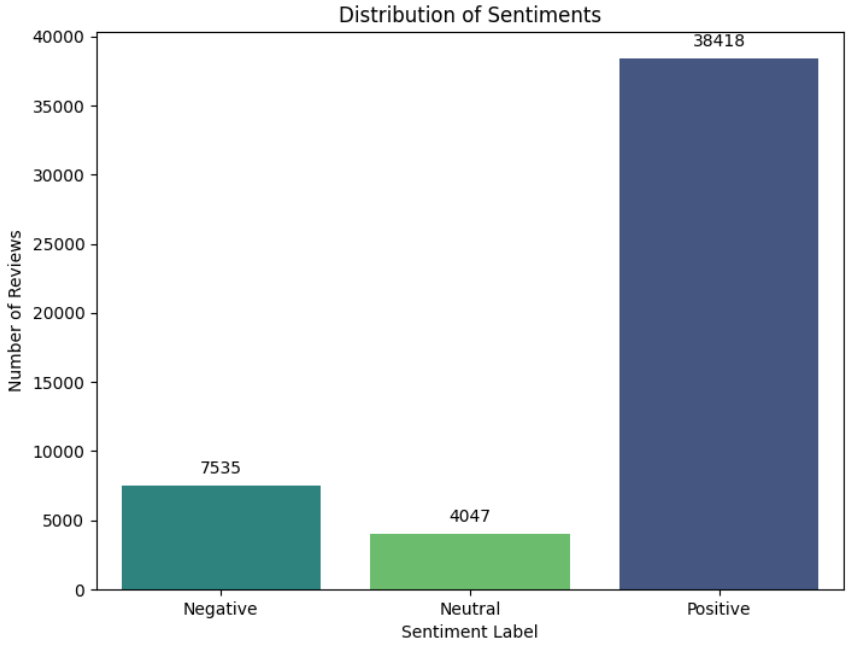


Figure 1: Distribution of Sentiments. The dominance of the Positive class presents a significant challenge for model training.

4 Methodology

4.1 Data Preprocessing Pipeline

To ensure the quality of the input features, a rigorous preprocessing pipeline was implemented in `preprocessing.py`. The steps included:

1. **Lowercasing:** All text was converted to lowercase to unify tokens (e.g., "Excellent" becomes "excellent").
2. **HTML Removal:** Reviews often contain HTML tags like `<br />`. These were stripped using regular expressions.
3. **URL Removal:** Hyperlinks were removed as they generally do not contribute to sentiment.
4. **Punctuation Stripping:** Special characters and punctuation were removed to focus purely on alphanumeric content.
5. **Stopword Removal:** The NLTK library was utilized to remove common English stopwords (e.g., "the", "a", "and") which add noise but little semantic value.
6. **Stemming:** The **Snowball Stemmer** was applied. Stemming reduces words to their root form (e.g., "tasted", "tasting", "tastes" $\rightarrow$ "tast"). This significantly reduces the size of the vocabulary and helps the model generalize better by grouping similar word forms.

4.2 Experimental Setup

- **Data Split:** The data was split into 80% training (40,000 samples) and 20% testing (10,000 samples). Crucially, `stratify=y` was used during the split to ensure the distribution of classes in the test set mirrored the training set.
- **Vectorization Parameters:** For both BoW and TF-IDF, the vocabulary was limited to the top 5,000 features. Unigrams and bigrams (ngram_range=(1,2)) were also included to capture local context phrases like "not good".
- **Hardware:** The experiments were conducted on a standard CPU environment, utilizing the `scikit-learn` library for modeling and `gensim` for Word2Vec training.

5 Results and Discussion

5.1 Evaluation Metrics Definition

To evaluate the models, the following metrics were used:

- **Accuracy:** The ratio of correctly predicted observations to total observations.
- **Precision:** The ratio of correctly predicted positive observations to the total predicted positives. High precision relates to a low false positive rate.
- **Recall (Sensitivity):** The ratio of correctly predicted positive observations to the all observations in actual class.
- **F1 Score:** The weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

5.2 Model Performance Summary

Four models were trained using TF-IDF features. Table 1 summarizes the results.

Model	Accuracy	F1 Score (Weighted)	F1 Score (Macro)
Logistic Regression	**85.33%**	**0.830**	0.613
SVM (Linear SVC)	84.95%	0.832	**0.620**
Random Forest	83.01%	0.789	0.544
Gradient Boosting	81.11%	0.761	0.482

Table 1: Performance comparison of all models on the test set.

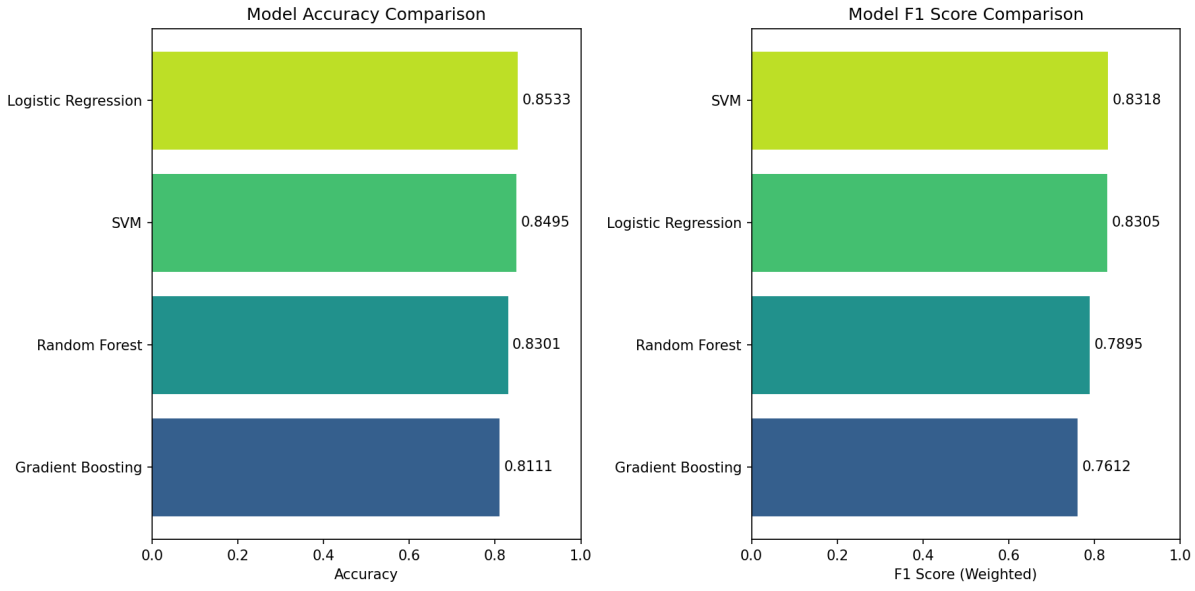


Figure 2: Visual comparison of Accuracy and Weighted F1 Score across models.

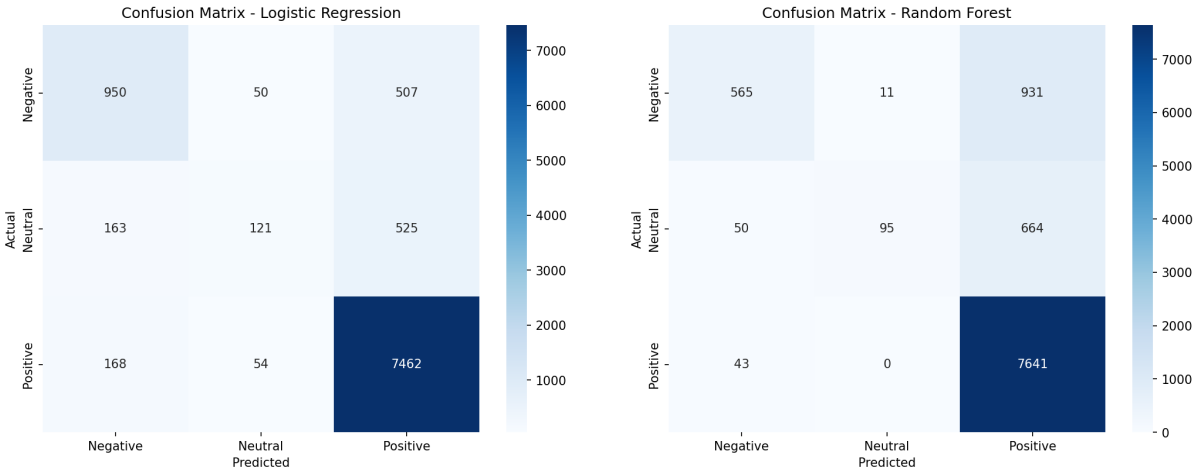
5.3 Analysis of Best Performing Model: Logistic Regression

Logistic Regression emerged as the most accurate model with an accuracy of 85.33%.

Looking at the confusion matrix in Figure 3a:

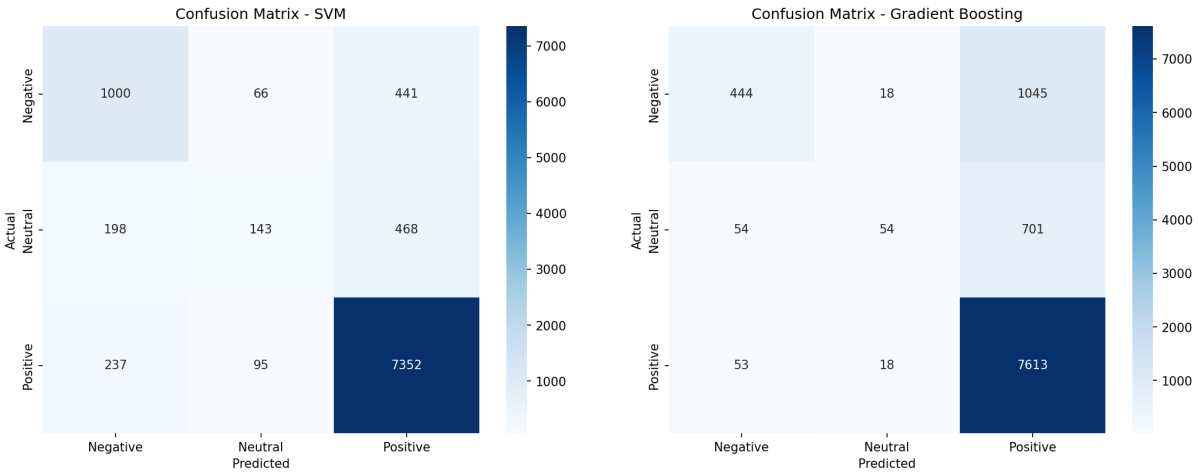
- **Positive Class:** The model excelled here, correctly classifying 7,462 out of 7,684 positive reviews.
- **Negative Class:** It correctly identified 950 negative reviews but missed 557 (misclassified as Neutral or Positive).
- **Neutral Class:** This was the failure point. The model only correctly identified 121 neutral reviews, while misclassifying 163 as Negative and 525 as Positive.

This behavior indicates that the model is biased towards the majority class (Positive), which is a direct consequence of the class imbalance discussed in Section 3.3.



(a) Logistic Regression (b) Random Forest

Figure 3: Confusion Matrices comparing Linear vs. Tree-based models.



(a) Support Vector Machine (SVM)

(b) Gradient Boosting

Figure 4: Confusion Matrices for SVM and Gradient Boosting models.

5.4 Why Linear Models Outperformed Tree Ensembles

A significant finding of this study is the superiority of linear models (Logistic Regression, SVM) over complex ensemble methods (Random Forest, Gradient Boosting).

1. **High Dimensionality:** The TF-IDF feature space has 5,000 dimensions. In such high-dimensional spaces, data points tend to be sparse and pushed to the edges of the hypercube. Linear models are exceptionally good at finding a separating hyperplane in these sparse, high-dimensional spaces.

2. **Linear Separability:** Text classification problems are often linearly separable. The presence of specific words (e.g., "delicious", "terrible") often correlates linearly with sentiment.

3. **Tree Fragmentation:** Decision trees split the data orthogonally. To approximate a diagonal decision boundary (which a linear model finds easily), a decision tree requires many deep splits. With limited data per leaf, trees can overfit or fail to capture the general trend in sparse data.

5.5 The Challenge of the Neutral Class

Across all models, performance on the Neutral class was poor. In the Gradient Boosting model, for example, the recall for the Neutral class was nearly zero. Neutral reviews are inherently difficult because they lack strong "polarity" words. A review stating "The tea was okay, but arrived late" contains both positive ("okay") and negative ("late") signals. Bag-of-Words and TF-IDF models, which ignore word order, struggle to parse

this nuance, often aggregating the counts and assigning the review to the class with the strongest individual keywords.

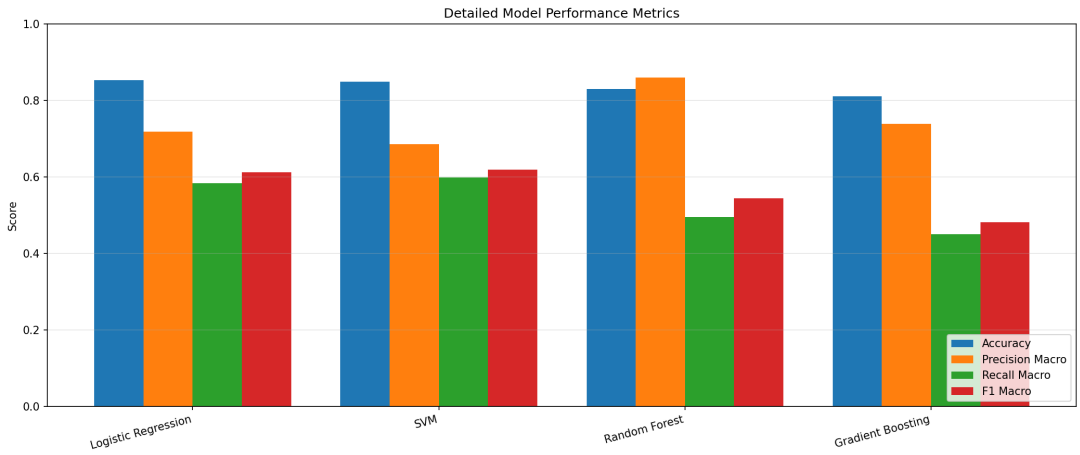


Figure 5: Detailed metrics breakdown. Note the significant drop in Macro Recall compared to Accuracy, highlighting the struggle with minority classes.

6 Conclusion and Future Work

6.1 Conclusion

The *SentiMind* project successfully demonstrated the application of NLP techniques to real-world e-commerce data. It was established that:

1. **Preprocessing is Critical:** Cleaning noise and stemming words effectively reduced the feature space.

2. **Linear Models are Efficient:** Logistic Regression provided the best balance of speed and accuracy (85.33%), outperforming computationally expensive ensemble methods.

3. **Imbalance is the Main Obstacle:** The overwhelming number of positive reviews biased the models, leading to poor performance on neutral and negative sentiments.

6.2 Future Scope

To further improve the system, the following enhancements are proposed:

- **Handling Class Imbalance:** Techniques such as SMOTE (Synthetic Minority Over-sampling Technique) or assigning higher class weights to the loss functions of the classifiers could force the models to pay more attention to Negative and Neutral reviews.

- **Deep Learning (BERT):** While computationally expensive, Transformer-based models like BERT (Bidirectional Encoder Representations from Transformers) can capture context and nuance that TF-IDF misses. The code base already contains the scaffolding for BERT extraction, and enabling this would be the logical next step.
- **Hyperparameter Tuning:** Basic Grid Search was performed. Expanding the search space for regularization parameters (C in Logistic Regression/SVM) could yield marginal improvements in F1 scores.