Final Report: Sentiment Classification

Introduction

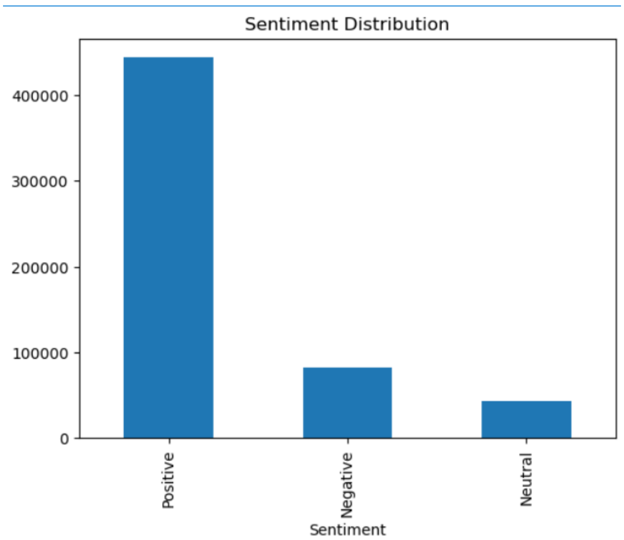
This project aimed to develop a sentiment analysis system to classify Amazon reviews. The primary goal was to build a machine learning model that could classify reviews into sentiment categories: Positive, Neutral, or Negative.

Sentiment analysis is valuable for businesses as it enables optimal resource allocation, addressing complaints (negative sentiments) and strengthening areas where positive sentiments thrive.

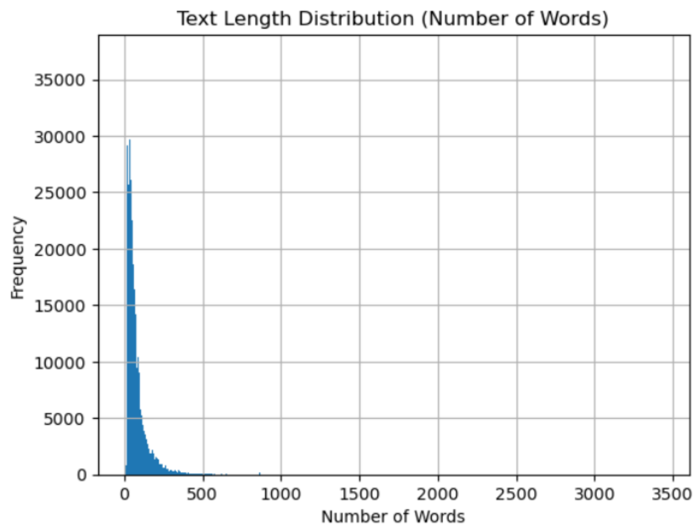
Data Preprocessing

The initial dataset consisted of 568,454 entries with the following features: Id, ProductId, UserId, ProfileName, HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary, and Text.

Features deemed unnecessary for this project (ProfileName, Id, Summary, HelpfulnessNumerator, HelpfulnessDenominator) were removed. The Score column (rated 1-5) was mapped into three categories: **Negative**, **Neutral**, and **Positive**. However, the dataset showed strong class imbalance, with **Positive** being the dominant category.



The focus was placed on the Text column, which served as the explanatory variable. Outliers with a text length exceeding 5,000 words were checked and retained.



Text Cleaning

The Text column underwent the following preprocessing steps:

1. Convert to lowercase.
2. Remove HTML tags.
3. Remove special characters and numbers.
4. Remove extra spaces.
5. Remove stopwords.
6. Handle repeated characters.
7. Handle negations.
8. Perform lemmatization.

Stopword removal was an iterative process to ensure that opinion-bearing words dominated the resulting vocabulary. The transformation was visualized using word clouds:

Before Cleaning:

After Cleaning:

Finally, an **N-Gram analysis** was performed to identify common bi-grams and tri-grams.

Modeling Approach and Evaluation

Round 1: Oversampling

The dataset was transformed using **TF-IDF vectorization** (max_features=5000). To address class imbalance, **SMOTE (Synthetic Minority Oversampling Technique)** was applied. The following models were tested:

- Logistic Regression
- Support Vector Machine (SVM)
- Naive Bayes
- LightGBM

- Random Forest

The results showed that **Random Forest** achieved the best F1-scores across all classes, followed by Logistic Regression and SVM. However, all models struggled with the **Neutral** and **Negative** classes, performing well only with the **Positive** class.

Logistic Regression Results (with Resampled Data):

	precision	recall	f1-score	support
Negative	0.60	0.71	0.65	16402
Neutral	0.26	0.61	0.36	8528
Positive	0.96	0.80	0.87	88756
accuracy			0.77	113686
macro avg	0.60	0.70	0.63	113686
weighted avg	0.85	0.77	0.80	113686

SVM Results (with Resampled Data):

	precision	recall	f1-score	support
Negative	0.59	0.71	0.64	16402
Neutral	0.26	0.59	0.36	8528
Positive	0.96	0.80	0.87	88756
accuracy			0.77	113686
macro avg	0.60	0.70	0.62	113686
weighted avg	0.85	0.77	0.80	113686

Naive Bayes Results (with Resampled Data):

	precision	recall	f1-score	support
Negative	0.53	0.67	0.59	16402
Neutral	0.21	0.60	0.31	8528
Positive	0.96	0.75	0.84	88756
accuracy			0.72	113686
macro avg	0.57	0.67	0.58	113686
weighted avg	0.84	0.72	0.76	113686

LightGBM Results:

	precision	recall	f1-score	support
Negative	0.52	0.51	0.52	16402
Neutral	0.24	0.43	0.31	8528
Positive	0.89	0.83	0.86	88756
accuracy			0.75	113686
macro avg	0.55	0.59	0.56	113686
weighted avg	0.79	0.75	0.77	113686

Random Forest Results (with Subsampled Data):

	precision	recall	f1-score	support
Negative	0.68	0.71	0.69	16402
Neutral	0.48	0.54	0.51	8528
Positive	0.93	0.91	0.92	88756
accuracy			0.85	113686
macro avg	0.70	0.72	0.71	113686
weighted avg	0.86	0.85	0.85	113686

Hyperparameter tuning was attempted on Logistic Regression, but no significant improvements were observed. Hyperparameter tuning for Random Forest was not feasible due to high computational costs caused by the inflated dataset (over 1 million rows post-SMOTE) and high-dimensional TF-IDF features.

Round 2: Undersampling

To reduce computational demands, the following changes were made:

- Removed outliers (Text > 5000 words).
- Reduced max_features in TF-IDF to **2000**.
- Replaced SMOTE with **undersampling** to balance the classes.

Class distributions post-undersampling:

- Positive: 33,843
- Neutral: 33,843
- Negative: 33,843

With the reduced dataset, Random Forest continued to outperform other models, particularly in recall and F1-scores for the Negative and Positive classes. Surprisingly, XGBoost underperformed, likely due to the smaller dataset, as XGBoost typically thrives with larger datasets and fine-tuned hyperparameters.

Random Forest Results:

	precision	recall	f1-score	support
Negative	0.55	0.77	0.64	16246
Neutral	0.30	0.68	0.41	8461
Positive	0.96	0.78	0.86	88238
accuracy			0.77	112945
macro avg	0.60	0.74	0.64	112945
weighted avg	0.85	0.77	0.79	112945

XGBoost Results:

	precision	recall	f1-score	support
Negative	0.47	0.55	0.51	16246
Neutral	0.19	0.52	0.28	8461
Positive	0.91	0.73	0.81	88238
accuracy			0.69	112945
macro avg	0.53	0.60	0.53	112945
weighted avg	0.80	0.69	0.73	112945

Logistic Regression Results (with Resampled Data):

	precision	recall	f1-score	support
Negative	0.54	0.67	0.60	16246
Neutral	0.22	0.59	0.31	8461
Positive	0.96	0.76	0.84	88238
accuracy			0.73	112945
macro avg	0.57	0.67	0.59	112945
weighted avg	0.84	0.73	0.77	112945

LightGBM Results:

	precision	recall	f1-score	support
Negative	0.48	0.57	0.52	16246
Neutral	0.19	0.53	0.28	8461
Positive	0.92	0.73	0.82	88238
accuracy			0.69	112945
macro avg	0.53	0.61	0.54	112945
weighted avg	0.80	0.69	0.73	112945

Round 3: Removing Neutral Reviews

The **Neutral** class was removed, focusing on the **Positive** and **Negative** classes to improve classification accuracy. This decision was based on the business value of accurately identifying actionable sentiments (Positive and Negative).

After removing the Neutral class, Logistic Regression and Random Forest were retrained. While recall for the Negative class improved slightly for Logistic Regression, precision remained low (~0.5). Random Forest exhibited a similar trend, with slight improvements in recall for the Positive class but no significant breakthroughs for the Negative class.

Logistic Regression Results (after GridSearch):

	precision	recall	f1-score	support
Negative	0.53	0.85	0.65	16246
Positive	0.97	0.86	0.91	88238
accuracy			0.86	104484
macro avg	0.75	0.86	0.78	104484
weighted avg	0.90	0.86	0.87	104484

Random Forest Results Binary (after GridSearch):

	precision	recall	f1-score	support
Negative	0.50	0.71	0.59	16246
Positive	0.94	0.87	0.90	88238
accuracy			0.84	104484
macro avg	0.72	0.79	0.75	104484
weighted avg	0.87	0.84	0.85	104484

Challenges and Limitations

Several challenges impacted the performance of this project:

1. **Computational Constraints:**
   High-dimensional TF-IDF features and large datasets inflated by SMOTE significantly increased the computational load, limiting hyperparameter tuning efforts for Random Forest and other models.

2. **Imbalanced Data:**
   Despite undersampling and oversampling techniques, models struggled with the Negative class, which inherently had less training data and more ambiguity in the text.

3. **Text Preprocessing:**
   Excessive stopword removal might have stripped context necessary for classifying Neutral and Negative reviews effectively. Balancing between removing noise and retaining context is a challenge in NLP projects.

Potential Improvements:

- Experiment with advanced text embeddings (e.g., **BERT**, **Word2Vec**) instead of TF-IDF for richer feature representation.
- Use smaller N-Grams or include character-level features to capture subtle variations in text.
- Implement ensemble models combining Logistic Regression and Random Forest for better handling of imbalanced classes.
- Use dimensionality reduction techniques (e.g., PCA or TruncatedSVD) to optimize computational efficiency.

Conclusion and Next Steps

This project successfully demonstrated the challenges of sentiment analysis on imbalanced datasets. The models consistently struggled with the Negative class, highlighting the importance of context preservation and the limitations of TF-IDF in this scenario.

While Random Forest emerged as the best-performing model overall, future iterations could explore transformer-based models like **BERT** for improved performance. Additionally, deploying the classifier in a **Streamlit app** or another interactive platform would enhance its usability and showcase its application.

Next steps include:

1. Exploring advanced text embeddings and deep learning approaches.
2. Refining preprocessing techniques for better handling of context and noise.
3. Incorporating these learnings into future projects.