Amazon Product Review Classification and Clustering

Ahmed Elmi
COSC74 Final Project
Yujun Yan
03/06/2025

Introduction:

The goal of this project was to analyze Amazon product reviews using machine learning techniques to classify and cluster reviews effectively. The tasks involved binary classification, where reviews were categorized as either "good" or "bad" based on different rating cutoffs, multiclass classification to predict ratings on a five-star scale, and clustering to group reviews by product categories. While this was a technical challenge, it was also a transformative learning experience that deepened my understanding of how real-world machine learning problems are approached.

One of the biggest challenges I faced was time constraints. Ideally, thorough hyperparameter tuning could have maximized model performance, but given the scope of the project, I had to make strategic trade-offs. Instead of exhaustively fine-tuning every model, I focused on ensuring the data was properly preprocessed, as clean input often has the most significant impact on model accuracy. Once the data was structured effectively, I selected models that balanced interpretability and performance. This approach allowed me to implement different classification and clustering techniques while staying within the project timeline.

Through this process, I gained a deeper appreciation for the **practical aspects** of machine learning—particularly the importance of **data quality, model selection, and performance evaluation**.

Dataset and Preprocessing

The dataset consisted of Amazon product reviews, including metadata such as review text, ratings (1-5), verification status, and timestamps. The primary feature was reviewText, while the target variable for classification was overall (rating). Additional fields like verified, vote, and category were considered but not the main focus.

Preprocessing Steps:

- **Missing Values**: Removed reviews with empty text fields.
- **Text Cleaning**: Lowercased text, removed stopwords, punctuation, and extra spaces.
- **Feature Extraction**: Applied TF-IDF to convert text into numerical form.
- **Data Splitting**: Divided into training and validation sets for model evaluation.
- With the data structured, the next step was implementing classification and clustering models.

Approach and Model Selection.

In addition to textual features, **metadata features** such as verified (whether a review was verified) and vote_bins (categorizing the number of helpful votes) were included. These numerical features were scaled appropriately—**StandardScaler** was used for Logistic Regression and SVM, while **MinMaxScaler** was used for Complement Naïve Bayes to keep features non-negative.

Once the data was preprocessed, it was split into **training and validation sets** using an **80-20 split** with stratification to ensure an even label distribution.

For model selection, three different machine learning algorithms were chosen:

- **Logistic Regression** was selected as a baseline model due to its simplicity, strong performance in text classification, and interpretability.
- **Complement Naïve Bayes (ComplementNB)** was specifically optimized for handling imbalanced text datasets, making it a strong contender for this task.
- **Linear SVC (Support Vector Classifier)** was used with **CalibratedClassifierCV** to generate probability estimates, as SVMs typically do not provide probability scores by default.

To ensure optimal performance, **hyperparameter tuning** was performed for each model using **GridSearchCV with 5-fold cross-validation**. The primary objective was to maximize the **macro F1 score**, which accounts for imbalanced classes by averaging the F1 scores of both positive and negative reviews.

Task 1: Binary classification

Hyperparameter Tuning and Model Selection

For both **cutoff 1** and **cutoff 2**, **3 and 4,** models were optimized using **GridSearchCV with 5-fold cross-validation**, selecting hyperparameters that maximized **macro F1 score**.

Logistic Regression (Final Model)

- Tuned: **C (Regularization)** {0.1, 1, 10}, **Solver:** 'liblinear', **Class weight:** 'balanced'
- **Best params (both cutoffs): C=1, balanced class weight**
- **Reason:** Achieved the highest **macro F1 score** and balanced precision-recall.

Complement Naïve Bayes (Runner-Up)

- Tuned: **Alpha (Smoothing)** {0.1, 0.5, 1.0}, **Norm:** {True, False}
- **Best params: Alpha = 0.5 (Cutoff 1), Alpha = 1.0 (Cutoff 2), norm=True**
- **Reason:** Strong performance but slightly lower **ROC AUC** than Logistic Regression.

Linear SVC (SVM with Calibration)

- Tuned: **C (Regularization)** {0.01, 0.1, 1, 10}, **Max Iterations:** 1000
- **Best params (both cutoffs): C=0.1, max_iter=1000**
- **Reason:** High **ROC AUC** but lower **macro F1 score**, meaning weaker balance across classes.

Final Decision: for my calculations, **Logistic Regression performed the best and I used it to pass the baseline scores due to its strong F1 score, interpretability, and computational efficiency**. **I did not do a lot of tuning as my models were already performing above the baseline score, but it could be possible that with more tuning that some of these models might perform better than others.**

Results for Cutoff 1 (Positive if rating > 1)

Model	Accuracy	ROC AUC	Macro F1 Score
Logistic Regression	0.83	0.815	0.775
ComplementNB	0.847	0.79	0.775
Linear SVC	0.870	0.90	0.774

At **cutoff 1**, where **only ratings of 1 were classified as negative (0)** and all other ratings (2, 3, 4, 5) were positive (1), the classification task was relatively easier since most reviews tend to be positive.

Among the models tested, **Logistic Regression performed the best overall**, achieving a **macro F1 score of 0.775** with the optimal hyperparameter setting of C=1. It maintained a strong balance between **precision and recall**.

ComplementNB followed closely with **similar performance**, benefiting from its suitability for text classification tasks.

Linear SVC had the highest accuracy (0.87) and **ROC AUC score (0.91)**, meaning it was able to distinguish between positive and negative reviews well. However, its **macro F1 score (0.77)** was slightly lower, indicating potential challenges in balancing performance across both classes.

Overall, **Logistic Regression was chosen as the best model for this cutoff due to its stability and strong recall, and besides it was able to easily pass the baseline score**

Results for Cutoff 2 (Positive if rating > 2)

Model	Accuracy	ROC AUC	Macro F1 Score
Logistic Regression	0.809	0.805	0.81
ComplementNB	0.796	0.793	0.804
Linear SVC	0.891	0.802	0.801

At **cutoff 2**, where **ratings of 1 and 2 were considered negative (0)** and all higher ratings (3, 4, 5) were positive (1), the classification task became more balanced compared to cutoff 1.

Logistic Regression **again achieved the highest macro F1 score (0.81), accuracy (0.81), and ROC AUC (0.81)**, solidifying its position as the best-performing model for this cutoff. Its ability to generalize well across different classification tasks made it a strong choice.

ComplementNB remained competitive but showed a slight drop in both accuracy and F1 score, likely due to its reliance on word frequency-based probabilities, which may have been affected by increased class overlap at this cutoff.

Linear SVC performed nearly identically to Logistic Regression but had slightly lower stability in generalization, as seen in its cross-validation scores.

Given the consistently **high F1 score and AUC-ROC**, **Logistic Regression was again chosen as the best model for cutoff 2**

And I performed the same models and got different results for the cutt off 3 and 4 but logistic regression always came on top.

Results for Cutoff 3 (Positive if rating > 3)

Model	Accuracy	ROC AUC	Macro F1 Score
Logistic Regression	0.82	0.82	0.814
ComplementNB	0.812	0.81	0.805
Linear SVC	0.82	0.90	0.81

Results for Cutoff 4

Model	Accuracy	ROC AUC	Macro F1 Score
Logistic Regression	0.817	0.796	0.748
ComplementNB	0.824	0.764	0.741
Linear SVC	0.85	0.88	0.727

Task 2: Multiclass classification

Approach and Key Differences

Multiclass classification introduced new complexities compared to binary classification, as the task required predicting the exact rating (1-5) rather than simply distinguishing between positive and negative reviews. This shift brought several challenges, including handling an expanded target variable, where the model needed to classify reviews into five distinct categories instead of two. Additionally, class imbalances had to be addressed, as extreme ratings (1 and 5) appeared more frequently than middle ratings (2, 3, and 4), making classification harder for less common classes.

To account for these challenges, adjustments were made in feature engineering. Unlike in binary classification, where a single TF-IDF transformation was applied to the entire review, multiclass classification benefited from separate TF-IDF vectorization for summary and reviewText to capture nuanced information from both fields. Categorical features such as category and vote were ordinally encoded to preserve rank-like relationships, ensuring that their numerical representations aligned with meaningful progressions. Additionally, boolean features like verified were retained as numerical inputs to provide contextual signals for review credibility.

Multiple models were explored to find the best-performing approach. Logistic Regression served as the baseline model due to its robustness in handling high-dimensional text data. Multinomial Naïve Bayes was considered due to its optimization for text classification, leveraging word frequency distributions to predict ratings effectively. A Random Forest classifier was also tested as a non-linear alternative to capture complex relationships between features, though it came at a higher computational cost.

Similar to binary classification, hyperparameter tuning was performed using GridSearchCV with 5-fold cross-validation to optimize model performance. Key parameters included the regularization strength **(C = {0.1, 1.0, 5.0}) for Logistic Regression, the smoothing parameter (alpha = {0.5, 1.0})** for Multinomial Naïve Bayes, and the number of trees (n_estimators = 50) with varying depth constraints (max_depth = None vs. 10) for Random Forest.

Given the multiclass nature of the task, evaluation metrics were adjusted accordingly. **Macro F1 Score** was prioritized to ensure fair assessment across all classes, preventing dominant ratings (such as 5-star reviews) from disproportionately influencing performance. ROC AUC scores were computed for each class using a **one-vs-rest approach**, providing insight into overall model separability. Confusion matrices were also analyzed to identify common misclassification patterns, helping to refine model selection and tuning strategies.

Model	**Accuracy**	**ROC AUC**	**Macro F1 Score**
Logistic Regression	0.6038	0.8819	0.6023
ComplementNB	0.5985	0.8825	0.6006
Linear SVC	0.5469	0.8383	0.5456

Logistic Regression outperformed the other models, achieving the highest F1 score and AUC, making it the best choice for final deployment. MultinomialNB was a close second, performing well due to its natural ability to model word distributions. Random Forest struggled with high-dimensional text data, leading to lower performance.

Task 3: Clustering Analysis: Discovering Patterns in Reviews

Unlike classification, where the goal was to assign predefined labels, clustering aimed to identify natural groupings within the data without predefined categories. K-Means was chosen as the clustering algorithm to group similar reviews based on textual and metadata features.

Approach and Key Differences from Classification

1. Feature Representation for Unsupervised Learning
    - Textual data (reviewText and summary) was transformed using CountVectorizer instead of TF-IDF to preserve word frequency information.
    - Categorical features (verified, vote, category) were ordinally encoded to ensure numerical consistency for clustering.
2. Choosing the Number of Clusters
    - The optimal number of clusters is not predefined in unsupervised learning.
    - K-Means clustering was tested with 2 to 10 clusters, and performance was evaluated using Silhouette Score (measuring how well points fit into clusters) and Adjusted Rand Index (ARI) (comparing clusters to product categories).
3. Evaluation Metrics
    - Silhouette Score: Measures how well-separated clusters are (higher is better).
    - Rand Index Score (ARI): Evaluates similarity between clusters and the true product categories (higher suggests meaningful grouping).

Results and Optimal Cluster Selection

Number of Clusters	Silhouette Score	Rand Index Score
2	0.6741	0.0028
3	0.4592	0.0112
4	0.4354	0.0122

Conclusion

This project explored binary classification, multiclass classification, and clustering of Amazon product reviews. Logistic Regression consistently outperformed other models across tasks, achieving the highest macro F1 scores and ROC AUC, while Multinomial Naïve Bayes performed well in text-based classification. Random Forest struggled with high-dimensional data.

For binary classification, predicting positive vs. negative reviews was relatively straightforward, with Logistic Regression maintaining strong recall. Multiclass classification proved more challenging due to class imbalances, particularly in mid-range ratings (2, 3, 4). Clustering using K-Means showed that smaller cluster numbers (2-4) provided the most coherent groupings, but weak alignment with product categories indicated room for improvement.

One key limitation was the limited scope of hyperparameter tuning due to time constraints, which may have prevented some models from reaching optimal performance. Class imbalance posed another challenge, particularly in multiclass classification, where mid-range ratings (2, 3, and 4) were harder to classify accurately. Additionally, while K-Means clustering provided some meaningful groupings, it struggled to align with product categories, suggesting that alternative clustering methods, such as hierarchical clustering or topic modeling, could yield better results.