Sentiment Classification of Product Reviews using Machine Learning and Text Mining Techniques

Argha Das, Ariana Haque Ami, Ibraheem Ibn Anwar, Jishnu Kumar Bachhar
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
{argha.das, ariana.haque.ami, ibraheem.ibn.anwar, jishnu.kumar.bachhar}@g.bracu.ac.bd

***Abstract*—This work presents a sentiment analysis pipeline built and evaluated on a large Amazon product reviews dataset. We focused on clear preprocessing steps, simple but effective feature engineering (TF-IDF n-grams combined with lexical statistics), and a comparison of four classifiers: Decision Tree, Logistic Regression, Support Vector Machine, and AdaBoost. Hyperparameter tuning is added as an option, which uses cross-validated GridSearchCV. Our experiments report accuracy, precision, recall, F1-score, and confusion matrices, and discuss the trade-off between prediction quality and training time. The final notebook and generated artifacts (data splits, feature matrices, and vectorizers) are organized for easy reproducibility. The results show that Logistic Regression and linear SVM are strong baselines for sparse text features, while more complex models can require much longer training without clear gains.**

***Index Terms*—Sentiment analysis, TF-IDF, GridSearchCV, Logistic Regression, SVM, text classification.**

I. Introduction

Sentiment analysis aims to classify text according to its polarity (for example, positive or negative). In this project we build a complete and reusable workflow that cleans raw review text, extracts TF-IDF and lexical features, and trains several supervised classifiers. Our main goal is to find a good balance between predictive performance and computational cost on a large dataset of 400k reviews.

II. Related Work

Text classification is a well-studied problem. Classic approaches include TF-IDF features combined with linear models [1], and kernel-based methods such as SVMs [2]. Several works on movie and product reviews compare traditional classifiers such as Naive Bayes, Logistic Regression, Decision Trees, KNN and SVM, and often find that SVM or Logistic Regression give the best accuracy on binary sentiment tasks [4].

Other studies focus specifically on e-commerce data. For example, some work on Amazon and Flipkart product reviews applies standard preprocessing, TF-IDF or CountVectorizer features, and evaluates multiple machine learning models. These studies usually report that SVM or Logistic Regression provide strong performance and are practical for large review datasets [6].

More recent research explores deep learning and transformer-based models. Hybrid neural architectures, fuzzy systems, and BERT-style transformers have achieved very high accuracy (often above 90%) on sentiment benchmarks [5], while review articles and surveys highlight a general trend: deep learning and transformer models outperform classic methods but require more computation and more complex implementation [6]. Our project positions itself on the classic side of this spectrum, using TF-IDF features with relatively simple classifiers to keep the pipeline easy to understand and efficient to run.

III. Dataset

The dataset consists of 400,000 Amazon product reviews (200,000 positive and 200,000 negative), sourced from Kaggle [3]. The word-count distribution is: mean $\approx$ 78 words, median $\approx$ 70, 75th percentile $\approx$ 108. We use a default train/validation split of 80/20 with stratification on the label.

IV. Preprocessing

We first clean the raw review text to remove noise and make the data more consistent. All text is converted to lowercase, and we remove URLs, HTML tags, and non-alphanumeric characters, keeping only letters, digits, spaces, and apostrophes. Extra whitespace is collapsed so that tokens are clearly separated. These steps are implemented in the `clean_text()` function using regular expressions. After cleaning, we compute simple statistics for each review, including character count, word count, and a token list obtained by splitting the cleaned text. These values are stored as additional columns in the DataFrame and are later used to build lexical features. Finally, we create a stratified 80/20 train/validation split on the label column and save the resulting CSV files so that the same split can be reused in later experiments.

V. Feature Engineering

We combine TF-IDF representations with simple lexical statistics. We apply `TfidfVectorizer(ngram_range=(1,2), min_df=2, max_df=0.95, stop_words='english')` to the cleaned text, which generates unigram and bigram features while filtering out very rare and very common terms. In parallel, we derive a small set of numeric lexical features for each review, including character count, word count, average word length,

and counts of exclamation marks and question marks. We then horizontally stack the TF-IDF matrix and the lexical feature matrix to form a single high-dimensional sparse representation. The resulting train and validation matrices are stored as CSR arrays and can be loaded directly during the modeling stage.

VI. Modeling

We train four supervised classifiers: Decision Tree, Logistic Regression, Support Vector Machine (SVM), and AdaBoost. For each model, we provide an option to run hyperparameter tuning using GridSearchCV; simple boolean flags in the notebook allow us to switch tuning on or off per model. When tuning is enabled, we use conservative settings such as three-fold cross-validation to keep runtime manageable. For the default SVM configuration, we rely on `LinearSVC`, a fast linear SVM that works well with high-dimensional sparse TF-IDF features. Kernel SVMs are more expensive to train and require more careful parameter search.

VII. Experimental Setup

All experiments use the same 80/20 stratified train/validation split so that the positive and negative classes remain balanced in both sets. To compare models, we report standard classification metrics: accuracy, precision, recall, weighted F1-score, and the confusion matrix computed on the validation set. The entire pipeline is run on a local machine; kernel SVMs with exhaustive grid search can become slow on the full 400k-sample dataset, so we prefer LinearSVC or smaller tuning subsets when training time is a concern. For quick experiments, we recommend subsampling the data to between 10k and 50k reviews, and then retraining the final chosen model on the full dataset with the best-found hyperparameters.

VIII. Results

TABLE I
Model Performance on Validation Set

Model	Accuracy	Weighted F1 Score
Decision Tree	0.7706	0.77
Logistic Regression	0.8960	0.90
SVM (LinearSVC)	0.8640	0.86
AdaBoost	0.7279	0.73

The class-wise precision and recall are generally balanced for the positive and negative classes, especially for Logistic Regression and SVM. Confusion matrices show that these two models misclassify fewer samples compared to Decision Tree and AdaBoost.

IX. Discussion

Overall, the experiments show that Logistic Regression is the best performer on this dataset, reaching about 89.6% accuracy and a weighted F1-score of around 0.90 while remaining relatively fast to train on the full feature set. The linear SVM model also performs strongly, with about 86.4% accuracy and a weighted F1-score of roughly 0.86, and it is a good choice when a margin-based classifier is preferred. In contrast, the Decision Tree and AdaBoost models achieve noticeably lower accuracy, around 77% and 73% respectively, which suggests that they are less well suited to the high-dimensional sparse feature space produced by TF-IDF plus lexical features. More complex variants of SVM (for example, RBF kernels) or deeper ensemble methods might improve performance on smaller subsets of data, but they would likely require much longer training times and more extensive hyperparameter search. In practice, it is therefore sensible to tune models on a small stratified subset of the data (for example, 3k–10k samples) and then retrain the chosen configuration on the full training set.

X. Conclusion

This project presents a simple and reproducible sentiment classification pipeline that is suitable for both coursework and early-stage research. The codebase provides switches to enable or disable hyperparameter tuning and to subsample the dataset, which helps to control runtime during experimentation. Overall, the results support the use of TF-IDF features with linear models as a strong and practical baseline for large-scale text classification.

Acknowledgements

We acknowledge the contributions of the project team:

- Argha Das — Model training, hyperparameter tuning, and evaluation
- Ariana Haque Ami — Prediction pipeline
- Ibraheem Ibn Anwar — Data cleaning and preprocessing
- Jishnu Kumar Bachhar — Feature engineering

References

[1] B. Pang and L. Lee, "Opinion mining and sentiment analysis," Foundations and Trends in Information Retrieval, 2008.

[2] T. Joachims, "Text categorization with Support Vector Machines: Learning with many relevant features," 1998.

[3] Bittlingmayer, "Amazon reviews for sentiment analysis," Kaggle, 2015. [Online]. Available: https://www.kaggle.com/datasets/bittlingmayer/amazonreviews

[4] G. Vinodhini and R. M. Chandrasekaran, "Sentiment analysis and opinion mining: A survey," International Journal of Advanced Research in Computer Science and Software Engineering, 2012.

[5] Y. Liu, J. Lu and J. Yang, "Sentiment analysis for e-commerce product reviews by deep learning model of BERT-BiGRU-Softmax," 2020.

[6] A. Daza et al., "Sentiment analysis on e-commerce product reviews: A comprehensive view of machine learning and deep learning algorithms," 2024.