Amazon Review Sentiment Analysis

Using Weka Machine Learning Tool

Author: Eniola Talabi

Course: Data Mining / Text Analytics

Date: February 2026

Project: Sentiment Analysis with Machine Learning

Executive Summary

This project analyzes Amazon product reviews to automatically classify them as positive or negative using machine learning. Using a dataset of 716 cleaned reviews, three classification algorithms were tested: Naive Bayes, J48 Decision Tree, and Random Forest. All three achieved 69.83% accuracy. The analysis revealed challenges with imbalanced datasets and demonstrated a complete data science workflow from data collection through model evaluation.

1. Introduction

1.1 Background

Sentiment analysis is a critical application of text mining that automatically determines the emotional tone of text. For e-commerce platforms like Amazon, understanding customer sentiment helps businesses identify product issues quickly, improve customer service, make data-driven product decisions, and monitor brand reputation.

1.2 Project Objectives

The primary objectives of this project were to: (1) Collect and preprocess Amazon product reviews, (2) Apply text mining techniques to extract features, (3) Build machine learning classification models, (4) Evaluate model performance, and (5) Analyze results and identify limitations.

1.3 Tools and Technologies

Python 3.x for data preprocessing, Pandas for data manipulation, Regular Expressions for text pattern matching, Weka 3.8.6 for machine learning classification, and GitHub for version control.

2. Methodology

2.1 Dataset

Source: Amazon Fine Food Reviews (Kaggle)
- Original dataset: 568,454 reviews
- Reviews selected for analysis: 1,500
- Final cleaned dataset: 716 reviews

Sentiment Distribution:
- Positive reviews (4-5 stars): 500 (69.8%)
- Negative reviews (1-2 stars): 216 (30.2%)
- Neutral reviews (3 stars): Removed for cleaner binary classification

2.2 Data Preprocessing Pipeline

The data preprocessing consisted of five steps:

Step 1: Data Loading
Loaded raw CSV file containing Amazon reviews and extracted relevant columns (review text and star rating).

Step 2: Text Cleaning
Removed HTML tags, special characters, punctuation, extra whitespace, and converted all text to lowercase.

Step 3: Sentiment Label Creation
Star ratings were converted to binary sentiment labels: Ratings 4-5 became "Positive", Ratings 1-2 became "Negative", and Rating 3 (neutral) was removed.

Step 4: Data Filtering
Removed reviews with less than 10 characters and empty/null text entries, resulting in 716 valid reviews.

Step 5: Data Export
Saved cleaned dataset as CSV file ready for Weka import.

2.3 Feature Extraction

In Weka, the StringToWordVector filter converted raw text into numerical features. This process:
- Tokenized text into individual words
- Extracted the top 1000 most frequent words
- Created word frequency features for each review
- Applied IDF (Inverse Document Frequency) transformation

This transformed the text data into a format suitable for machine learning algorithms.

2.4 Classification Algorithms

Three machine learning algorithms were tested:

1. Naive Bayes
A probabilistic classifier based on Bayes' theorem. Assumes independence between features and calculates the probability that a review belongs to each class.

2. J48 Decision Tree
A decision tree algorithm that creates a tree-like model of decisions. It splits the dataset based on feature values to classify reviews.

3. Random Forest
An ensemble method that builds multiple decision trees and combines their predictions for more robust classification.

2.5 Evaluation Method

All models were evaluated using 10-fold cross-validation. This method splits the data into 10 parts, trains on 9 parts and tests on 1 part, repeating this process 10 times. This provides a reliable estimate of model performance and reduces overfitting.

3. Results

3.1 Classification Performance

Algorithm	Accuracy	Correctly Classified	ncorrectly Classified
Naive Bayes	69.83%	500/716	216/716
J48 Decision Tree	69.83%	500/716	216/716
Random Forest	69.83%	500/716	216/716

All three algorithms achieved identical 69.83% accuracy. This uniform performance across different algorithm types indicates a systematic pattern in the data rather than algorithm-specific behavior.

3.2 Confusion Matrix Analysis

The confusion matrix revealed that all three classifiers exhibited the same prediction pattern:

- All 500 positive reviews were correctly classified as positive
- All 216 negative reviews were incorrectly classified as positive
- Zero reviews were predicted as negative

This indicates that all models learned to predict only the positive class, essentially ignoring the negative class entirely.

4. Discussion

4.1 Why 69.83% Accuracy?

The 69.83% accuracy directly corresponds to the proportion of positive reviews in the dataset (500/716 = 69.8%). This is not a coincidence. The classifiers learned that predicting "positive" for every review maximizes accuracy given the class imbalance. This is a well-known issue in machine learning called the **majority class bias problem**.

4.2 Root Cause: Class Imbalance

The dataset contained 500 positive reviews (69.8%) versus 216 negative reviews (30.2%). This 70/30 split created an imbalanced dataset. Machine learning algorithms optimize for overall accuracy, so when faced with imbalanced data, they often learn to predict the majority class exclusively because it minimizes error.

4.3 Real-World Implications

While 69.83% accuracy sounds reasonable, this model would be useless in practice because it cannot identify negative reviews at all. A business using this model would miss every customer complaint, defeating the entire purpose of sentiment analysis. This demonstrates why accuracy alone is not sufficient for evaluating classification models, especially with imbalanced datasets.

4.4 Potential Solutions

To improve this analysis, the following approaches could be implemented:

1. Balance the Dataset
Collect equal numbers of positive and negative reviews (500/500) or use undersampling/oversampling techniques.

2. Use SMOTE
Synthetic Minority Over-sampling Technique creates synthetic examples of the minority class to balance the dataset.

3. Adjust Class Weights
Many algorithms support class weights that penalize misclassification of the minority class more heavily.

4. Use Different Metrics
Instead of accuracy, use F1-score, precision, recall, or ROC-AUC which better handle imbalanced

data.

5. Try Advanced Algorithms
Deep learning models or ensemble methods specifically designed for imbalanced data might perform better.

4.5 What This Project Successfully Demonstrated

Despite the class imbalance limitation, this project successfully accomplished:

- **Data Collection:** Obtained and processed a real-world dataset of 716 Amazon reviews
- **Data Preprocessing:** Implemented a complete text cleaning and preprocessing pipeline
- **Feature Engineering:** Converted raw text into numerical features suitable for ML
- **Model Implementation:** Successfully trained and evaluated three different ML algorithms
- **Critical Analysis:** Identified the class imbalance problem and proposed solutions
- **Professional Documentation:** Created reproducible code with version control

These skills represent the core competencies of a data scientist and demonstrate understanding of the complete machine learning workflow.

5. Conclusion

This project successfully demonstrated a complete sentiment analysis workflow using Amazon product reviews and Weka machine learning tool. Three classification algorithms (Naive Bayes, J48 Decision Tree, and Random Forest) were implemented and evaluated, all achieving 69.83% accuracy.

The uniform performance across algorithms revealed a fundamental issue with the dataset: class imbalance (69.8% positive, 30.2% negative). All classifiers learned to predict only the majority class (positive), which maximized accuracy but resulted in zero negative predictions. This demonstrates an important lesson in machine learning: accuracy alone can be misleading, especially with imbalanced datasets.

The project achieved its educational objectives by:
- Implementing a complete data science pipeline from collection to evaluation
- Gaining hands-on experience with Weka and Python for text mining
- Understanding real-world challenges in sentiment analysis
- Learning to critically analyze model performance beyond simple accuracy metrics

Future work could address the class imbalance through dataset balancing techniques, weighted algorithms, or alternative evaluation metrics. The skills and insights gained from this project provide a strong foundation for more advanced text mining and machine learning applications.

6. Technical Details

6.1 Code Repository

All project code and data files are available at:
GitHub: github.com/eniolatalabi/sentiment-analysis-weka

Repository contents:
- prepare_data.py: Python preprocessing script
- amazon_sentiment_clean.csv: Cleaned dataset (716 reviews)
- naivebayes_results.txt: Naive Bayes classification results
- j48_results.txt: J48 Decision Tree results
- randomforest_results.txt: Random Forest results
- README.md: Project documentation

6.2 System Requirements

To reproduce this analysis:
- Python 3.x
- Pandas library (pip install pandas)
- Weka 3.8.6 (weka.sourceforge.io)
- Amazon Fine Food Reviews dataset (Kaggle)

Metric	Value
Total Reviews Analyzed	716
Positive Reviews	500 (69.8%)
Negative Reviews	216 (30.2%)
Algorithms Tested	3
Best Accuracy	69.83%
Dataset Source	Amazon (Kaggle)