Systematic Improvement of Multinomial Naive Bayes Classifier for Sentiment Classification: Parameter Tuning, Enhancement Techniques, and Efficiency Optimization

Sijie Guo (50572482), Jaeyun Kim (44482235), Kaixiang Zhuang(73969468)

June 10th 2025

Introduction

Multinomial Naive Bayes (MNB) is a popular model for text classification due to its simplicity, speed, and interpretability. However, its standard form has limitations such as equal weighting of word frequencies and lack of context sensitivity. This research investigates how to systematically improve the performance of MNB for sentiment classification using the IMDb movie review dataset from Kaggle. We explore three complementary approaches: fine-tuning hyperparameters, incorporating enhancement techniques, and planning for efficiency optimizations. The goal is to enhance MNB's effectiveness while preserving its lightweight structure, demonstrating that classic models can still perform competitively when thoughtfully upgraded.

Resources: We train and evaluate our models on the IMDb 50K Movie Reviews dataset—a balanced benchmark consisting of 25,000 positive and 25,000 negative reviews commonly used for binary sentiment classification.

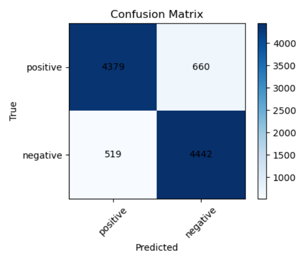
Dataset link: IMDb Dataset on Kaggle

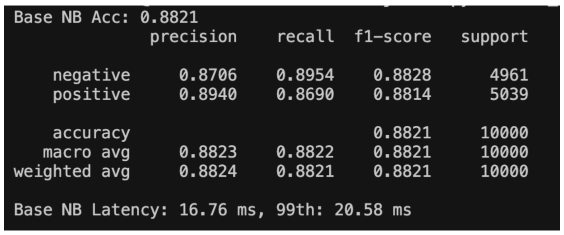
All code used in this project was written by us from scratch, including data processing, model training, evaluation, and enhancement implementations.

Methods and Implementations

Baseline Model: Multinomial Naive Bayes Classifier

The baseline system is a plain **Multinomial Naïve Bayes (MNB)** classifier, a probabilistic graphical model that assumes every word (and bigram) in a review is conditionally independent of the others given the sentiment label; we train it on an 80 / 20 split of the IMDB corpus using a vectoriser (1–2-gram, max_df=0.7), then estimate class-conditional multinomial probabilities with Laplace smoothing. On the held-out set the model attains 88.2% accuracy and a macro-/weighted-F1 of **0.882** (confusion-matrix: 4379 true-positives, 4442 true-negatives, 1179 total errors), but its simplicity brings three weaknesses: it over-weights very common yet sentiment-neutral tokens, lacks any mechanism to highlight the most discriminative features, and keeps static log-probability weights that cannot adapt once deployed.




```
Base NB Acc: 0.8821
              precision    recall  f1-score   support

    negative     0.8706    0.8954    0.8828      4961
    positive     0.8940    0.8690    0.8814      5039

    accuracy                         0.8821     10000
   macro avg     0.8823    0.8822    0.8821     10000
weighted avg     0.8824    0.8821    0.8821     10000

Base NB Latency: 16.76 ms, 99th: 20.58 ms
```

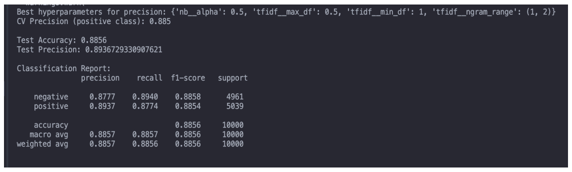
Goal: We explore three dimensions to improve the original Multinomial Naive Bayes model: Hyperparameters Tuning, Enhancement Techniques, and Efficiency Optimization.

1. Fine-tuning Hyperparameters

This section explores the effects of hyperparameter tuning on the performance of a Multinomial Naïve Bayes classifier for binary sentiment classification of IMDB movie reviews. We constructed a pipeline using TF-IDF vectorization and MultinomialNB, then performed grid search over several preprocessing and model parameters. Additionally, we investigated how modifying the classification threshold affects the trade-off between precision and recall. The results showed that tuning ngram_range, document frequency limits (min_df, max_df), and Laplace smoothing parameter alpha can significantly enhance model performance.

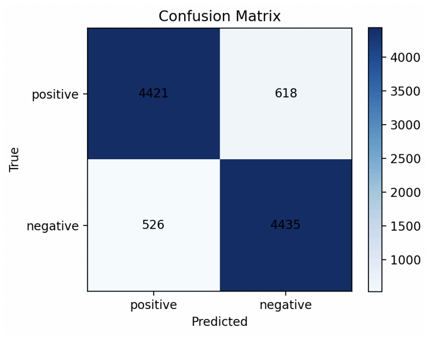
The best performance was yielded by:

1. Using bigrams (1,2) enhanced the model's ability to capture local context (e.g., "not good", "very bad"), which is critical for detecting nuanced sentiment.
2. A low max_df of 0.5 filtered out words appearing in more than half of the documents—terms too generic to be sentiment indicators.
3. The choice of alpha=0.5 maintains a balance between smoothing for robustness and preserving learned conditional probabilities.



```
Best hyperparameters for precision: {'nb__alpha': 0.5, 'tfidf__max_df': 0.5, 'tfidf__min_df': 1, 'tfidf__ngram_range': (1, 2)}
CV Precision (positive class): 0.885

Test Accuracy: 0.8856
Test Precision: 0.8936729330907621

Classification Report:
              precision    recall  f1-score   support

    negative     0.8777    0.8940    0.8858      4961
    positive     0.8937    0.8774    0.8854      5039

    accuracy                         0.8856     10000
   macro avg     0.8857    0.8857    0.8856     10000
weighted avg     0.8857    0.8856    0.8856     10000
```

These metrics confirm that the model maintains balanced performance across both classes, with nearly symmetric F1-scores and comparable precision-recall values.



The confusion matrix indicates that the model:

Correctly identified a large majority of both positive and negative reviews.

Misclassified 618 positive reviews as negative (false negatives).

Predicted 526 false positives, which were minimized through threshold tuning.

In conclusion, compared to the base model with default settings, the tuned classifier was able to have a slight improvement in performance. The best configuration achieved a test accuracy of 88.56% and positive-class precision of 89.37% at a decision threshold of 0.7, with a well-balanced F1 score. These results suggested that careful tuning and threshold control can substantially improve classification confidence in positive predictions without sacrificing overall performance.

2. **Incorporating Enhancement Techniques**

To improve the predictive performance of the **MNB baseline model**, we explored two enhancement techniques designed to strengthen the model's textual representation.

Enhancement 1: Adding TF-IDF Features

The baseline model used CountVectorizer to convert text into a bag-of-words representation. While simple and effective, it treats all word occurrences equally without considering their relative importance across documents. To address this, we replaced it with TfidfVectorizer, which adjusts term weights based on their frequency in individual reviews relative to the full dataset.

We applied the transformation:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(1, 2))
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

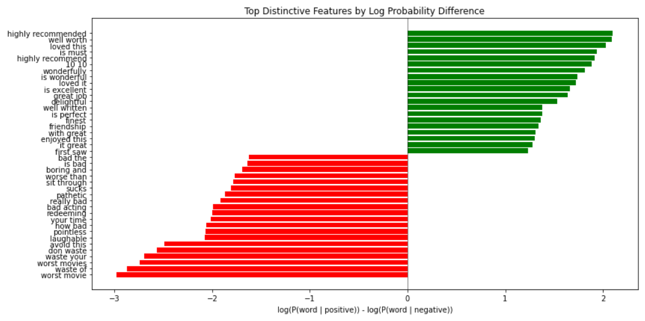
Result: Switching to TF-IDF increased the model's **accuracy to 0.8892** and **F1 score to 0.8867**, demonstrating improved sensitivity to meaningful n-grams without any additional model complexity.

Enhancement 2: Keyword Boosting

Building on the MNB + TF-IDF model, we introduced **keyword boosting** to guide the model's attention toward sentiment-relevant features. We computed the log-probability difference of each feature between the positive and negative classes using the trained MNB model:

$$\log P(word \mid positive) - \log P(word \mid negative)$$

We selected the top 20 distinctive positive and negative words after removing overlaps and stopwords.

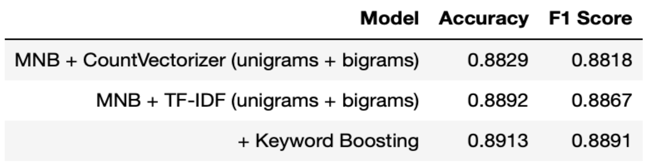


Boosting Method

To emphasize these keywords, we increased their TF-IDF values by multiplying their corresponding feature columns in the matrix by a boost factor (e.g., 2.0):

```
X_train_boosted[:, trusted_indices] *= boost

X_test_boosted[:, trusted_indices] *= boost
```

Result: This boosted the model's sensitivity to critical features without altering its structure. With keyword boosting, the model achieved **accuracy of 0.8913** and **F1 score of 0.8891**, improving upon the previous TF-IDF model. The following figure is the **Summary of Performance Gains.**

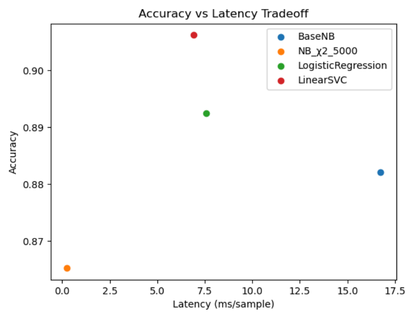


Model	Accuracy	F1 Score
MNB + CountVectorizer (unigrams + bigrams)	0.8829	0.8818
MNB + TF-IDF (unigrams + bigrams)	0.8892	0.8867
+ Keyword Boosting	0.8913	0.8891

These enhancement techniques helped the model better capture the contextual importance of words and emphasized discriminative features. Both improvements led to **modest but consistent performance gains**, while requiring only minimal changes to the overall model pipeline—making them practical and effective upgrades to a classic probabilistic approach.

3. Efficiency Optimization

Reducing latency boosts responsiveness, throughput, and cost-efficiency, so we start by sanitising IMDb reviews, splitting 80/20 (random_state 42), and vectorising with TF-IDF (max_df 0.7, min_df 1, 1–2 g), counts, or hashing; a vanilla MultinomialNB(α 1.0) gives 4379 TP, 4442 TN, and 16.8 ms mean latency (P99≈20.6 ms) on 1000 test calls. Offline χ^2 selection (k=5000) rebuilds the TF-IDF with a 10 % vocabulary, shrinking vectors 10× and cutting latency to 0.23 ms (P99≈0.39 ms) for only a 1.7-point accuracy drop (0.882→0.865)—a 70× speed-up and the best sub-ms trade-off.



The scatter plot reveals a clear trade-off: χ^2-pruned NB achieves sub-millisecond inference at slight accuracy cost, while Linear SVC and Logistic Regression gain the best accuracy but run an order of magnitude slower, leaving vanilla NB both slower than the other models, but is slightly more accurate than the χ^2-pruned NB.

Conclusion

Starting from a basic Multinomial Naïve Bayes sentiment classifier on the IMDb reviews, we successively refined the model through three targeted approaches: **hyperparameter tuning** to improve predictive accuracy, **enhancement techniques** such as TF-IDF feature engineering and keyword boosting to improve feature quality, and **efficiency optimization** using χ^2-based vocabulary pruning to reduce dimensionality and runtime. These incremental changes demonstrate that lightweight tweaks to a classic model can deliver competitive performance while remaining efficient enough for real-time, large-scale applications. Through this research, we show that even simple models like MNB can be made significantly more effective and practical with thoughtful, low-cost modifications.

References

- IMDb Dataset: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
- scikit-learn documentation