Report on Sentiment Analysis of IMDB Reviews Using Naive Bayes and Logistic Regression

Advanced Visual Text Analytics

Merve Pakcan Tufenk

Introduction

With the rapid growth of user-generated content on the internet, understanding public opinion has become more important than ever. Among various forms of online feedback, movie reviews serve as a rich source of data, offering valuable insights into audience preferences and reactions. Sentiment analysis, a subfield of natural language processing (NLP), focuses on automatically detecting and classifying emotions or opinions expressed in text. In this context, the IMDB movie review dataset is a widely used benchmark for evaluating the effectiveness of sentiment classification models.

This report presents a comparative study of **three machine learning models** Logistic Regression, Naive Bayes, and a Neural Network applied to the task of classifying IMDB reviews as either positive or negative. While these models are commonly used in text classification, their performance can vary significantly depending on how they represent and learn from textual data. The objective is to assess their accuracy, generalization ability, and robustness when applied to real-world sentiment analysis tasks.

Dataset Description

This study uses the IMDB Large Movie Review Dataset, which consists of 50,000 labeled movie reviews sourced from Kaggle. Each review is annotated with a binary sentiment label (positive or negative) and the dataset is evenly balanced, containing 25,000 positive and 25,000 negative samples.

The dataset is organized into two columns:

- **review**: the raw text of the movie review
- **sentiment**: the associated sentiment label (positive or negative)

For model evaluation, the data is either split evenly into 25,000 training and 25,000 testing instances or divided using an 80/20 train-test split depending on preprocessing requirements. During the cleaning phase, one malformed record was identified and removed to maintain data integrity.

Originally introduced by Andrew L. Maas et al. in the 2011 ACL paper *"Learning Word Vectors for Sentiment Analysis"*, this dataset has become a widely adopted benchmark in sentiment classification research due to its size, balance, and clear sentiment polarity.

Preprocessing

Two separate preprocessing pipelines were designed to match the needs of traditional models and neural networks.

For **Logistic Regression** and **Naive Bayes**, the text was first converted to lowercase, and all HTML tags, punctuation marks, and links were removed. Tokenization was performed using NLTK's TweetTokenizer, followed by the removal of common stopwords and word stemming with the PorterStemmer. These steps helped simplify the input and reduce irrelevant content. Additionally, for the Naive Bayes model, a word–sentiment frequency table was created to count how often each word appeared with a given label, which supports its probability-based structure.

For the **Neural Network**, preprocessing focused on preserving raw structure. Text was tokenized using Keras' tokenizer and padded to a fixed length of 200. No stopword removal or stemming was applied, as the model uses an embedding layer (vocab size = 10,000, embedding dim = 16) to learn patterns directly from the sequence of words. This allowed the model to capture semantic context and word order, which are critical in deep learning.

The differences in preprocessing reflect the nature of each model: while traditional models need clean, compact input features, neural networks benefit from richer, unaltered text sequences.

Model Development and Evaluation

Logistic Regression

A linear classification algorithm that models the probability of a binary outcome based on input features.The Logistic Regression model was initially implemented using Scikit-learn with default parameters. It achieved a modest accuracy of **62.08%**, demonstrating a strong **bias toward positive predictions**, as shown by a low recall score of **0.30** for the negative class. The model frequently mislabeled mixed or ambiguous reviews as positive.

To improve performance, **GridSearchCV** was applied to optimize the regularization parameter C and the solver method (liblinear). After tuning, the model's accuracy **improved to 88.0%**, and class recall became balanced **(0.87 for negative, 0.90 for positive).** This highlights how proper hyperparameter tuning can transform the performance of linear models.

Naive Bayes

A probabilistic classifier based on Bayes' Theorem, assuming independence between features (words).The Naive Bayes classifier was implemented both manually (using word-label frequency dictionaries) and with Scikit-learn's MultinomialNB. After testing a range of smoothing (alpha) values through grid search, **the alpha of 1.0 yielded the best result, with accuracy of 85.0%.** The model showed **balanced recall and precision (~0.85)** across both sentiment classes and performed well on clearly polarized reviews. However, it tended to misclassify **neutral or mixed-tone reviews** as negative, due to its assumption of word independence and reliance on simple word occurrence statistics.

Neural Network

A machine learning model that uses embedding and dense layers to extract semantic meaning from text.A feedforward neural network was developed using **Keras Sequential**, comprising an Embedding layer, a GlobalAveragePooling1D layer, and two Dense layers. A maximum vocabulary size of **10,000**, sequence length of **200**, and embedding dimension of **16** were used. The model was trained with the Adam optimizer and early stopping based on validation loss to prevent overfitting. The model achieved **88.0% test accuracy**, with **balanced precision and recall (0.88)**, matching the tuned Logistic Regression. Importantly, it performed **better on ambiguous reviews**, correctly capturing subtle sentiment cues.

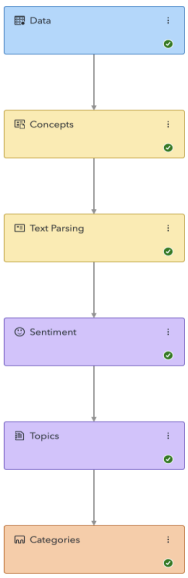
Additional improvements being tried:

Bidirectional LSTM: Increased model complexity but did **not improve accuracy** over the simpler architecture.

Learning rate adjustment: Using Adam(learning_rate=0.0005) had **no increase in accuracy** compared to the last optimizer settings. These results suggest that, in this case, a simple, well-regularized model can be as effective as deeper architectures.

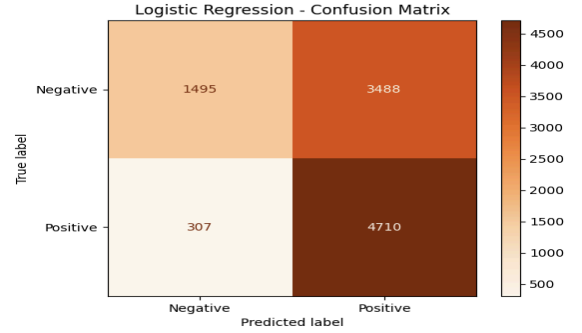
SAS Text Analytics

The project also leveraged **SAS Visual Text Analytics.** A prebuilt template was used to run a complete pipeline from text parsing and concept extraction to sentiment analysis and unsupervised topic modeling. SAS automatically assigned **positive, negative, or neutral** sentiment labels to all **50,000** reviews. In total, **64,477** unique terms were extracted and grouped into dominant discussion themes. These results helped validate and enrich the findings obtained before.



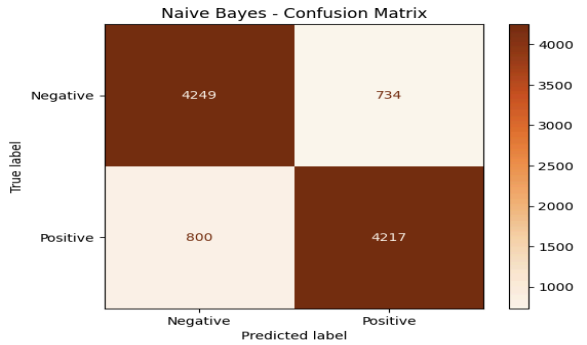
Results and Evaluation

All three models demonstrated distinct strengths. Logistic Regression initially underperformed with 62% accuracy but improved to 88% after hyperparameter tuning. Naive Bayes achieved 85% accuracy and was effective on clearly polarized reviews, though it struggled with mixed sentiment. The Neural Network consistently delivered the most balanced performance, reaching 88% accuracy and successfully handling both simple and more complex reviews. Its ability to learn patterns directly from the text using embeddings without needing extensive preprocessing made it the most adaptable and capable model in this project. These results are clearly reflected in the confusion matrices and classification reports.



Logistic Regression – Classification Report

	precision	recall	f1-score	support
Negative	0.83	0.30	0.44	4983
Positive	0.57	0.94	0.71	5017
accuracy			0.62	10000
macro avg	0.70	0.62	0.58	10000
weighted avg	0.70	0.62	0.58	10000



Naive Bayes – Classification Report

	precision	recall	f1-score	support
Negative	0.84	0.85	0.85	4983
Positive	0.85	0.84	0.85	5017
accuracy			0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000

	precision	recall	f1-score	support
0	0.87	0.89	0.88	4940
1	0.89	0.87	0.88	5060
accuracy			0.88	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000