Sentiment Analysis on Movie Reviews Using Machine Learning

Alex Micael Chen Hsieh
achenhs1@binghamton.edu
CS 435: Introduction to Data Mining
Binghamton University
Binghamton, New York, USA

1 Introduction

In the age of social media, user-generated content like reviews and comments hold immense value. On platforms such as Rotten Tomatoes and Letterboxd, non-professional reviews can have a demonstrable impact on audience choices and commercial success [6, 7]. Thus, understanding the sentiment behind this text can be crucial for businesses and filmmakers alike to grow effectively [4]. From market research to customer feedback analysis, there are many commercial applications for sentiment analysis. This project tackles the problem of sentiment analysis, a core task of Natural Language Processing (NLP), by classifying movie reviews as either positive or negative.

The main objective is to build and compare a number of machine learning models to determine which is most effective for this classification task. Using an IMDB Movie Review Dataset, this report details the project's process of data exploration, text preprocessing, model training, and a comparative analysis of the results. The models evaluated include traditional machine learning algorithms like Logistic Regression, Linear Support Vector Classifier (LinearSVC), and K-Nearest Neighbors (KNN), as well as a neural network approach using a Multi-Layer Perceptron (MLP).

2 Data Description and Exploration

The foundation of this project is the IMDB Movie Review Dataset, a widely used benchmark for binary sentiment classification.

2.1 Dataset Composition

The dataset contains a total of **50,000 movie reviews**, each labeled with its corresponding sentiment (positive, negative). A key characteristic of this dataset is its perfect balance, which is ideal for avoiding concerns of class imbalance skewing the results. In fact, the distribution is as follows:

- **Positive Reviews:** 25,000 entries
- **Negative Reviews:** 25,000 entries

This distribution ensures that a model achieving an accuracy of 50% is no better than random guessing, thus, providing a clear baseline for performance. The reviews not only vary greatly in length and style, but they contain nuanced human language, including sarcasm, idioms, and complex sentence structures, which pose significant challenges for models that rely on word statistics rather than true contextual understanding.

3 Methodology

To prepare the data for machine learning, a comprehensive pipeline was established. The dataset was split into a **75% training set** (37,500 reviews) and a **25% testing set** (12,500 reviews) to ensure performance was evaluated on unseen data.

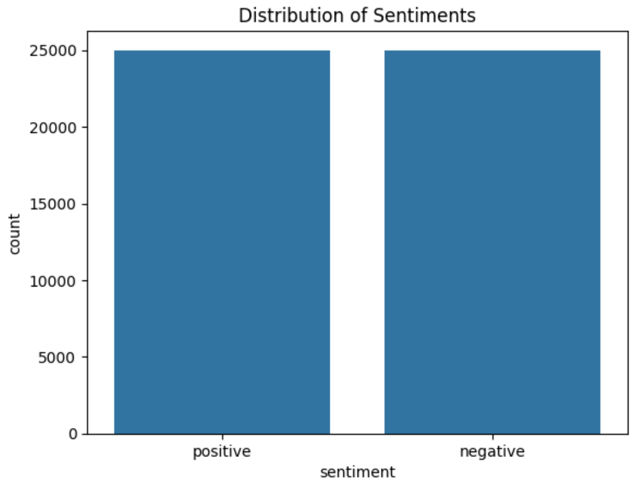

Figure 1: Distribution of Sentiments in the IMDB Dataset.

3.1 Data Pre-processing

It's important to note that raw text is unstructured and noisy, thus a multi-step cleaning process (illustrated in Table 1) was applied to normalize the text and reduce feature space dimensionality.

(1) **HTML Tag Removal:** As the data is sourced from web pages, residual HTML tags (e.g., `<br />`) were removed to eliminate non-textual noise.
(2) **Noise Removal:** Punctuation, numbers, and special characters were removed to focus solely on the words carrying sentiment.
(3) **Lowercase Conversion:** All text was converted to lowercase. This is to prevent the model from treating words like "Great" and "great" as two distinct features.
(4) **Tokenization:** The cleaned string of text was segmented into a list of individual words, a.k.a. "tokens." This is a fundamental step of breaking unstructured text into manageable units.
(5) **Stop Word Removal:** Common English words like "the," "a," "in," and "is" that provide little semantic value for sentiment were filtered out. These words often add noise rather than signal.
(6) **Stemming:** Words were reduced to their root form. For example, "running" becomes "run". This process consolidates the vocabulary and allows the model to recognize related words as a single feature [5]. This project uses the popular Porter Stemmer algorithm.

(a) Positive Sentiment Word Cloud

(b) Negative Sentiment Word Cloud

Figure 2: Word clouds illustrating frequent terms in each class.

Table 1: Example of the Text Preprocessing Pipeline

Step	Example Text
Raw Text	"This movie was EXCELLENT! <br/> Great."
No HTML/Noise	"This movie was EXCELLENT great"
Lowercase	"this movie was excellent great"
No Stop Words	"movie excellent great"
Stemmed	"movi excel great"

3.2 Feature Engineering: TF-IDF

The preprocessed text was then converted into a numerical format using **Term Frequency-Inverse Document Frequency (TF-IDF)**. TF-IDF creates a vector for each document where each dimension represents a word, and the value is a weight indicating its importance [2]. This weighting scheme is composed of two parts:

- **Term Frequency (TF):** This measures how frequently a term appears in a document. It is based on the intuition that if a word appears often in a document, it is important to that document's meaning.
- **Inverse Document Frequency (IDF):** This measures how unique a term is across all documents in the corpus. The IDF of a rare word is high, whereas the IDF of a common word is low. This helps to undermine words that appear in many documents (like "movie") and give more importance to terms that are more specific to a particular review.

The final TF-IDF score is the product of these two values.

3.3 Data Modeling

Four different classification models were trained on the TF-IDF vectors:

- **Logistic Regression:** A linear model that predicts the probability of a review belonging to the "positive" class. It passes the weighted sum of the input features through a sigmoid function to produce a probabilistic output between 0 and 1.
- **LinearSVC:** A Support Vector Machine with a linear kernel. The goal of an SVM is to find the optimal hyperplane that best separates the two classes (positive and negative) in the high-dimensional feature space, maximizing the margin between them.
- **K-Neighbors Classifier (KNN):** A non-parametric, instance-based learning algorithm. It makes predictions for a new data point by finding the 'k' most similar data points in the training set and taking a majority vote of their classes. It is considered a "lazy learner" as it does no training until a prediction is requested [1].
- **Multi-Layer Perceptron (MLP):** A classic feedforward artificial neural network. It consists of an input layer (representing the TF-IDF features), one or more hidden layers with non-linear activation functions (like ReLU or tanh), and an output layer that makes the final classification [3].

4 Results and Discussion

The models were evaluated on the 25% test set. The results in Table 2 and Figure 3 show a clear performance hierarchy.

Table 2: Model Performance Summary

Model	Accuracy
Logistic Regression	**0.8852**
LinearSVC	0.8804
MLP (100, 50, relu)	0.8698
MLP (100, relu)	0.8690
MLP (100, tanh)	0.8661
KNeighborsClassifier	0.7605

4.1 Analysis of Findings

The results are highly informative. **Logistic Regression and LinearSVC models were the top performers**. Their high accuracy confirms that linear models are very effective for high-dimensional, sparse datasets like TF-IDF-vectorized text. For many text classification problems, the relationship between features and the target is approximately linear, and these models are highly efficient at finding the decision boundary.

The **Multi-Layer Perceptron (MLP) models also performed well**, trailing just slightly behind. This indicates that while they can capture more complex, non-linear relationships, the additional

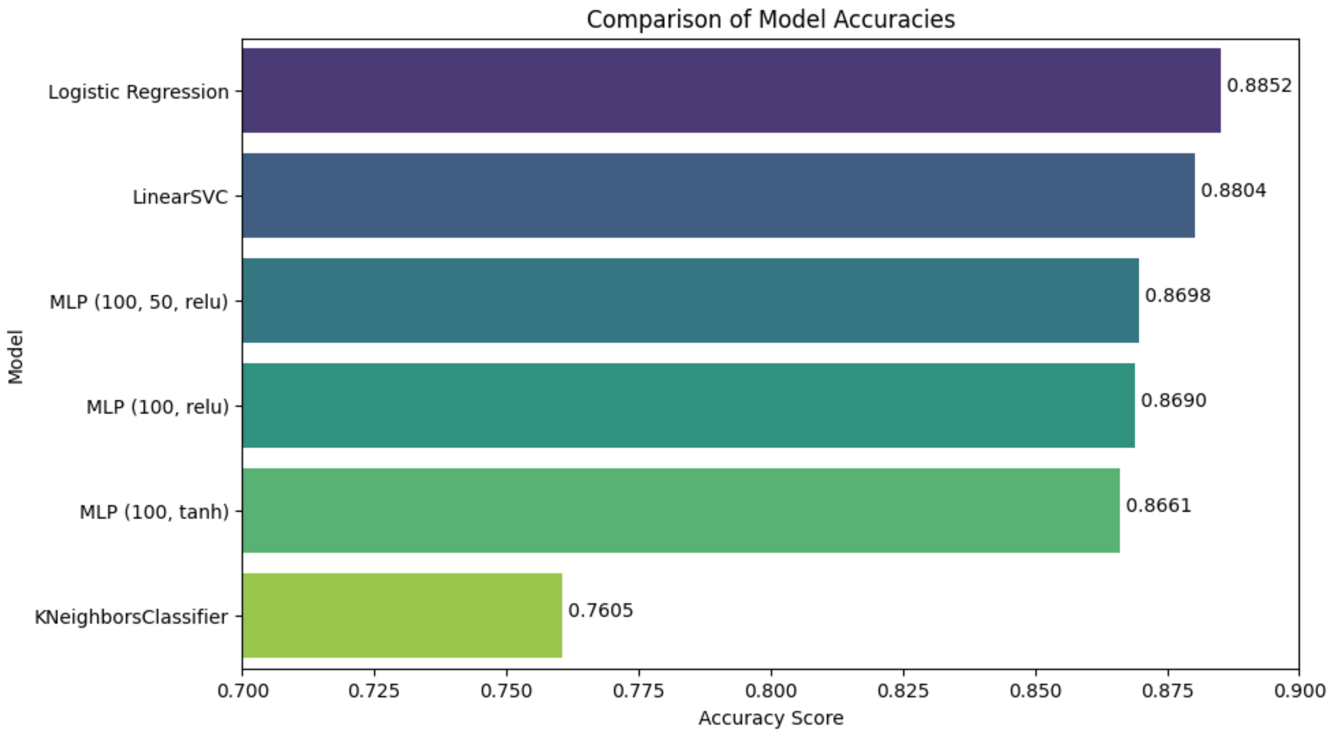

Figure 3: Comparison of Overall Model Accuracies Across All Trained Models. Linear models (LinearSVC, Logistic Regression) demonstrate the highest performance, while the distance-based KNN model lags significantly.

complexity did not yield a significant advantage for this specific classification task.

Lastly, I'd like to note the poor performance of the **K-Nearest Neighbors (KNN) classifier**. This outcome is expected due to the "curse of dimensionality" [1]. In a space with 5,000 dimensions, the concept of Euclidean distance becomes less meaningful, as data points tend to be sparse and equidistant from one another. This makes it difficult for KNN to identify a truly local neighborhood for making predictions.

4.2 Analysis of the Best Model

The top model, Logistic Regression, was analyzed further with a confusion matrix (Figure 4). The matrix shows the model correctly classified 5373 negative reviews (True Negatives) and 5692 positive reviews (True Positives). The number of misclassifications is relatively low and balanced. The model incorrectly classified 784 negative reviews as positive (False Positives) and 651 positive reviews as negative (False Negatives). This showcases a slight tendency to predict "positive," but the performance is still robust and does not show significant bias towards any class.

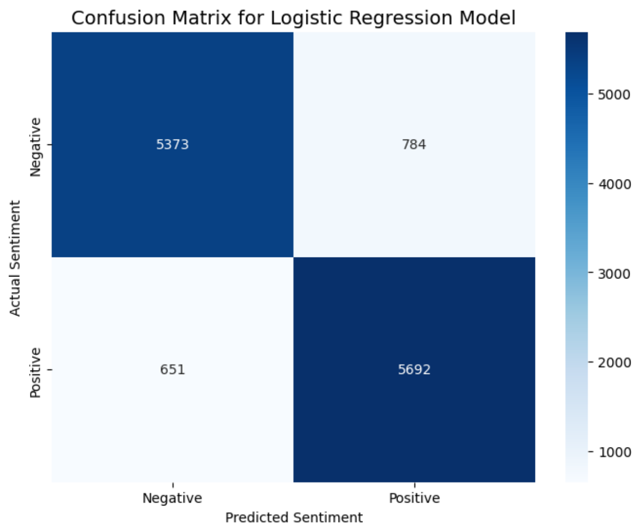

Figure 4: Confusion Matrix for the Logistic Regression Model.

5 Conclusion

This project demonstrated a workflow for building a sentiment analysis classifier. Through careful preprocessing and the application of TF-IDF, raw text was transformed into a suitable format for machine learning.

The comparative analysis revealed that the **Logistic Regression model was the most effective classifier**, achieving the highest accuracy of 88.52%. The findings underscore that for text classification with TF-IDF, linear models remain powerful and reliable choices. However, the "bag-of-words" nature of TF-IDF means that word order, grammar, and context are lost, making it impossible to understand complex nuances like sarcasm.

Future work could explore more advanced techniques to overcome these limitations. Feature representations like Word2Vec or GloVe, which capture semantic relationships between words, could be used. Furthermore, employing more complex deep learning architectures such as Recurrent Neural Networks (RNNs) or Transformers (e.g., BERT), which are designed to process sequential data, could lead to a significant improvement in performance by properly incorporating word order and context.

References

[1] GeeksforGeeks. 2025. K-nearest neighbors and curse of dimensionality. https://www.geeksforgeeks.org/machine-learning/k-nearest-neighbors-and-curse-of-dimensionality/

[2] GeeksforGeeks. 2025. Understanding TF-IDF (term frequency-inverse document frequency). https://www.geeksforgeeks.org/machine-learning/understanding-tf-idf-term-frequency-inverse-document-frequency/

[3] Seja Jaiswal. 2025. Multilayer Perceptrons in Machine Learning: A Comprehensive Guide. https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning

[4] Sangkil Moon, Paul K. Bergey, and Dawn Iacobucci. 2010. Dynamic Effects among Movie Ratings, Movie Revenues, and Viewer Satisfaction. *Journal of Marketing* 74, 1 (2010), 108–121. http://www.jstor.org/stable/20619083

[5] Jacob Murel and Eda Kavlakoglu. 2023. What is Stemming? https://www.ibm.com/think/topics/stemming

[6] Jacob R. Pentheny. 2015. *The Influence of Movie Reviews on Consumers.* Ph. D. Dissertation. University of New Hampshire. https://scholars.unh.edu/honors/265/

[7] Cole Skuse. 2024. Film critics and the rise of audience reviews. https://the-tartan.org/2024/04/15/film-critics-and-the-rise-of-audience-reviews/