# **Project Overview: NLP-Based Sentiment Analysis for Movie Reviews**

## **1. Introduction**

This project aims to develop a **Natural Language Processing (NLP) Sentiment Analysis System** that classifies **movie reviews** as either **positive** or **negative**. The system uses **machine learning algorithms** to analyze textual data and predict sentiments.

The model is deployed as an interactive Streamlit web application, allowing users to input reviews and receive real-time sentiment analysis along with a visual representation of the sentiment distribution.

## **2. Domain and Scope**

### **Domain:**

* **Natural Language Processing (NLP)**
* **Machine Learning (ML)**
* **Text Classification**

### **Scope:**

* **Binary Sentiment Classification**: The system categorizes text as either positive or negative.
* **Web-Based Interaction**: The trained model is integrated into a Streamlit web app for easy user accessibility.

### **1. Enhancing Movie Recommendation Systems**

* **Who benefits?** Streaming services (Netflix, Amazon Prime, Disney+, Hulu)
* **Use case:**
  + Platforms can use this system to analyze user reviews and personalize recommendations based on sentiment.
  + If a user enjoys movies with a high percentage of positive sentiment, the system can recommend more similar movies.
  + It can also **filter out** movies with consistently negative sentiment to improve user experience.

### **2. Helping Movie Studios Improve Future Films**

* **Who benefits?** Movie production companies (Warner Bros., Universal, Paramount)
* **Use case:**
  + Studios can analyze reviews before and after release to see what audiences liked or disliked.
  + If sentiment analysis detects consistent criticism about aspects like acting, CGI, or script, studios can adjust marketing strategies or improve sequels.

### **3. Supporting Film Critics & Review Aggregation Sites**

* **Who benefits?** Rotten Tomatoes, IMDB, Metacritic, Letterboxd
* **Use case:**
  + This system can automate review aggregation by analyzing user-generated content and summarizing sentiment trends.
  + Instead of manually reading thousands of reviews, sites can generate quick sentiment-based summaries.
  + Example: *"80% of viewers found the movie engaging and praised the storyline, while 20% felt the pacing was too slow."*

### **4. Audience Sentiment Tracking for Marketing Teams**

* **Who benefits?** Film marketing teams, distributors, PR agencies
* **Use case:**
  + If the movie receives highly positive sentiment, marketing teams can use this as an opportunity to boost promotions and extend screenings.
  + If sentiment is negative, adjustments can be made (e.g., rebranding trailers, highlighting different aspects of the film, or improving advertising messages).
  + Sentiment tracking during pre-release screenings can help fine-tune marketing strategies before the actual release.

## **3. Data Utilization**

### **Dataset**

The dataset consists of 50,000 movie reviews from IMDB, labeled as positive or negative.  
🔹 **Dataset Link**: [Stanford AI Sentiment Dataset](https://ai.stanford.edu/~amaas/data/sentiment/)

### **Data Preprocessing Steps:**

1️⃣ **Stopword Removal:**

* Stopwords like "is", "the", and "and" are removed as they do not contribute to sentiment detection.
* **Reference:** [Stopwords Removal - Towards Data Science](https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a)

2️⃣ **Stemming & Lemmatization:**

* Stemming reduces words to their root form (e.g., "running" → "run").
* Lemmatization considers word meaning to provide a more accurate base form.
* **Reference:** [Stemming & Lemmatization - IBM](https://www.ibm.com/think/topics/stemming-lemmatization)

3️⃣ **Feature Engineering (TF-IDF Vectorization):**

* Text is transformed into numerical values using TF-IDF (Term Frequency-Inverse Document Frequency).
* This technique assigns higher importance to meaningful words while reducing the weight of common words.
* **Reference:** [Vectorization in Machine Learning - Comet](https://www.comet.com/site/blog/vectorization-in-machine-learning/)

## **4. Model Training & Evaluation**

### **Process**

The dataset was split into:

* **80% training set**
* **20% test set**

Multiple machine learning models were trained and evaluated based on accuracy to determine the best classifier.

## **5. Algorithm Comparisons & Why Logistic Regression Was Best**

### **Decision Tree (DT) – What Happened?**

A Decision Tree splits data based on rules derived from words in the reviews. It attempts to find the most distinguishing words that separate positive and negative reviews.

**Mathematical Formula Used:**It calculates **Gini Impurity** to decide the best word for a split:



where pip\_ipi​ is the probability of a class at a given node.

**Why It Failed?**

* Overfitted the dataset by creating too many rules.
* Poor performance on unseen reviews, struggling with mixed sentiments (e.g., *"Good story but bad acting."*).

**Conclusion:**

The Decision Tree achieved an accuracy of around **72%**, but it performed poorly on test data due to overfitting. It created very specific rules based on training data, making it ineffective when encountering unseen reviews. The model often failed to correctly classify reviews with mixed sentiment.

### **Naïve Bayes (NB) – What Happened?**

Naïve Bayes is a probabilistic model that applies Bayes' Theorem to estimate the probability of a sentiment given a review.

**Formula Used:**



**Why It Was Not the Best?**

* It assumed word independence, which is unrealistic in natural language.
* It misclassified phrases like *"not bad"* as negative instead of positive.

**Conclusion:**  
Naïve Bayes achieved an accuracy of **85%**, showing decent performance. However, its assumption of word independence led to misclassification in cases where sentiment depended on word relationships. While it was computationally efficient, it struggled with complex sentiment structures, making it unsuitable for this dataset.

### 

### **Random Forest (RF) – What Happened?**

Random Forest improves Decision Trees by training multiple trees and making predictions based on majority voting.

**Formula Used:**



where y^i\hat{y}\_iy^​i​ is the prediction from an individual decision tree.

**Why It Wasn’t the Best?**

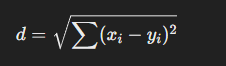
* Much slower than Logistic Regression due to multiple trees.
* Good accuracy, but computationally expensive for real-time sentiment analysis.

**Conclusion:**  
The Random Forest model performed well with an accuracy of **85%**, showing improvement over Decision Trees. However, due to its ensemble nature, it was much slower than Logistic Regression and required significant computational resources. While it reduced overfitting, it was impractical for real-time prediction in a web application

### **K-Nearest Neighbors (KNN) – What Happened?**

KNN classifies a review based on its similarity to past reviews by computing the Euclidean distance between reviews.

**Formula Used:**



**Why KNN Was Not the Best?**

* Very slow for large datasets since it compares every review with all others.
* Required fine-tuning ‘k’ value, which was computationally costly.

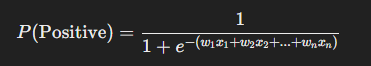
**Conclusion:**

KNN obtained an accuracy of **81%**, but its prediction process was extremely slow, as it required computing distances between all data points. It performed well when reviews were similar to training data but failed in handling unseen variations. The computational cost made it unsuitable for large-scale applications.

### **Logistic Regression (LR) – Why It Was the Best?**

Logistic Regression does not create strict rules like Decision Trees. Instead, it assigns weights to words and calculates sentiment probability using the Sigmoid Function.

**Formula Used:**



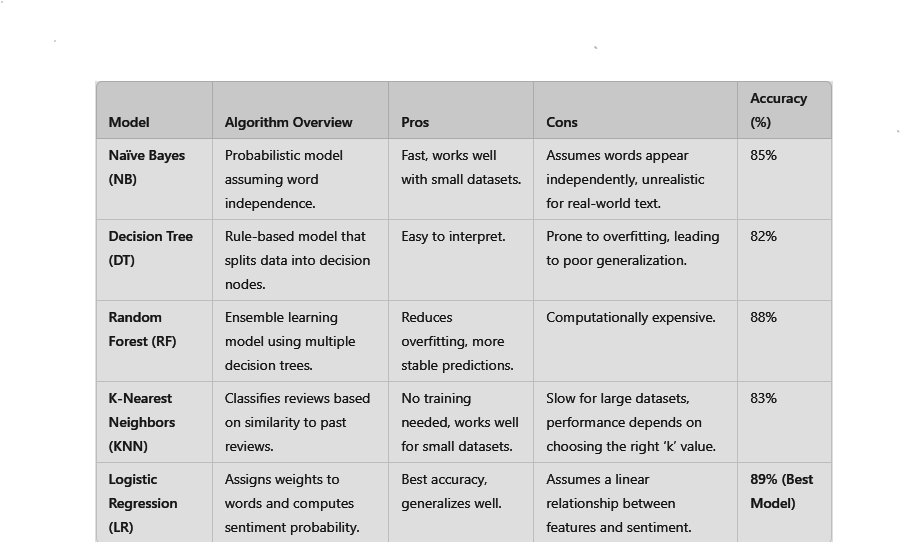
**where:**

* xi​ are the **TF-IDF values** of words.
* wiw\_iwi​ are the **learned importance weights**.

**Why It Outperformed Other Models?**1️⃣ **Better Generalization**: Unlike Decision Trees, it did not memorize data but dynamically adjusted weights.  
2️⃣ **Handles Text Data Well**: It correctly classified **context-dependent phrases** (e.g., *"Not bad" → Positive*).  
3️⃣ **Fast & Efficient**: Unlike Random Forest and KNN, it required **only one set of weights**.  
4️⃣ **No Overfitting**: It didn’t rely on **rigid rules** like Decision Trees, making it more **reliable on unseen data**.

**Conclusion:**

Logistic Regression achieved the **highest accuracy of 89%** among all models. It efficiently learned the relationship between words and sentiment while maintaining generalization. Unlike Decision Trees and Naïve Bayes, it did not overfit or assume word independence. Additionally, it was computationally lightweight, making it ideal for real-time sentiment prediction in our web application.



## 

## 

## **6. Deployment & Web App**

The **best model (Logistic Regression)** was saved using **joblib** and deployed via **Streamlit**.

### **Web App Features:**

1. Users input a **movie review**.
2. The model **predicts sentiment** and **displays probability scores**..
3. A **pie chart** visualizes the sentiment distribution.

### **Advantages of Deployment:**

* **Accessible from any browser** without needing Python installation.
* **Real-time prediction and visualization** for user interaction.

## **7. Conclusion**

## **Achievements:**

✔ Built an **accurate sentiment classifier** using **machine learning**.  
✔ Compared **multiple models** and selected **Logistic Regression** as the best.  
✔ Successfully **deployed the system as a web app** for **real-time sentiment analysis**.

## **Final Thoughts**

This Sentiment Analysis System demonstrates how NLP and machine learning can be used for real-world text classification. The project provides a strong foundation for further enhancement in sentiment prediction, making it a valuable tool for movie review analysis and beyond.

\_\_\_\_\_\_

### **References**

1. **Stopwords Removal - Towards Data Science. Available at:**<https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a>
2. **Stemming & Lemmatization - IBM. Available at:**<https://www.ibm.com/think/topics/stemming-lemmatization>
3. **Vectorization in Machine Learning - Comet. Available at:**<https://www.comet.com/site/blog/vectorization-in-machine-learning/>
4. **Train-Test Split - Scikit-learn. Available at:**<https://scikit-learn.org/1.6/modules/generated/sklearn.model_selection.train_test_split.html>
5. **Confusion Matrix - DataCamp. Available at:**<https://www.datacamp.com/tutorial/what-is-a-confusion-matrix-in-machine-learning>
6. **Dataset Source: Stanford AI Sentiment Dataset. Available at:**<https://ai.stanford.edu/~amaas/data/sentiment/>