Sentiment Analysis: Text Classification System

Group Members: Bernice Uwituze, Christian Mutabazi, Jean Chrisostome Dufitumukiza, Sifa Mwachoni

Github Repo : https://github.com/uwituzeb/IMDB_sentiment_analysis

1. INTRODUCTION

Sentiment Analysis is a natural language processing(NLP) task of analyzing large volumes of text to determine whether it expresses a positive sentiment, a negative sentiment or a neutral sentiment [1]. In this project, we explore and compare a traditional machine learning model (Logistic Regression) and a Deep Learning Model (LSTM) for sentiment analysis.

Dataset Overview:The dataset chosen was a **IMDB Movie Reviews dataset - IMDB Dataset of 50K Movie Reviews - Kaggle**. It contains 50,000 labelled reviews (25,000 training and 25,000 testing samples).

2. EXPLORATORY DATA ANALYSIS

We conducted statistical analysis to understand the dataset:

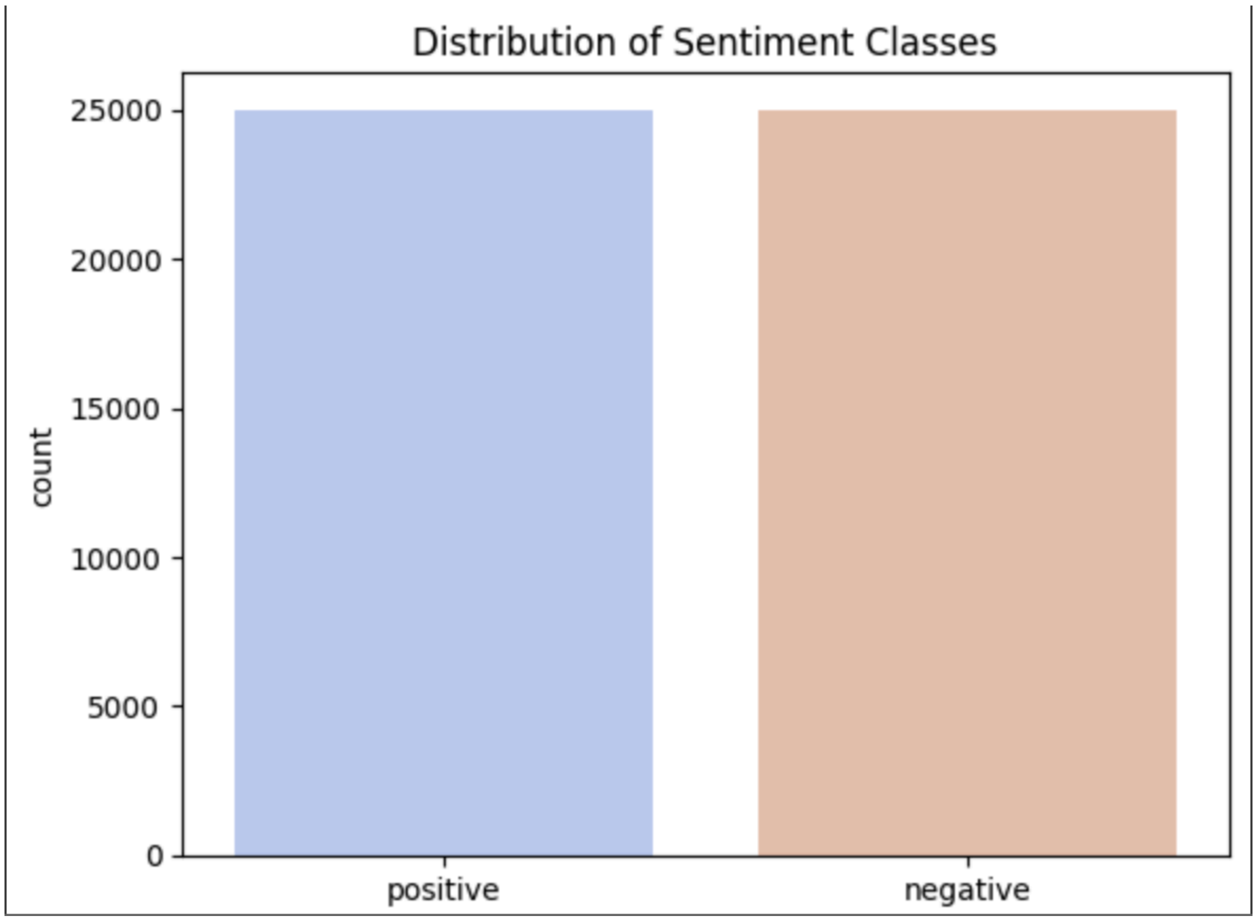
Key Findings:

- Class Distribution: Balanced dataset (50% positive, 50% negative)
- Text Length Analysis: Reviews vary in length (mean - 230 words)

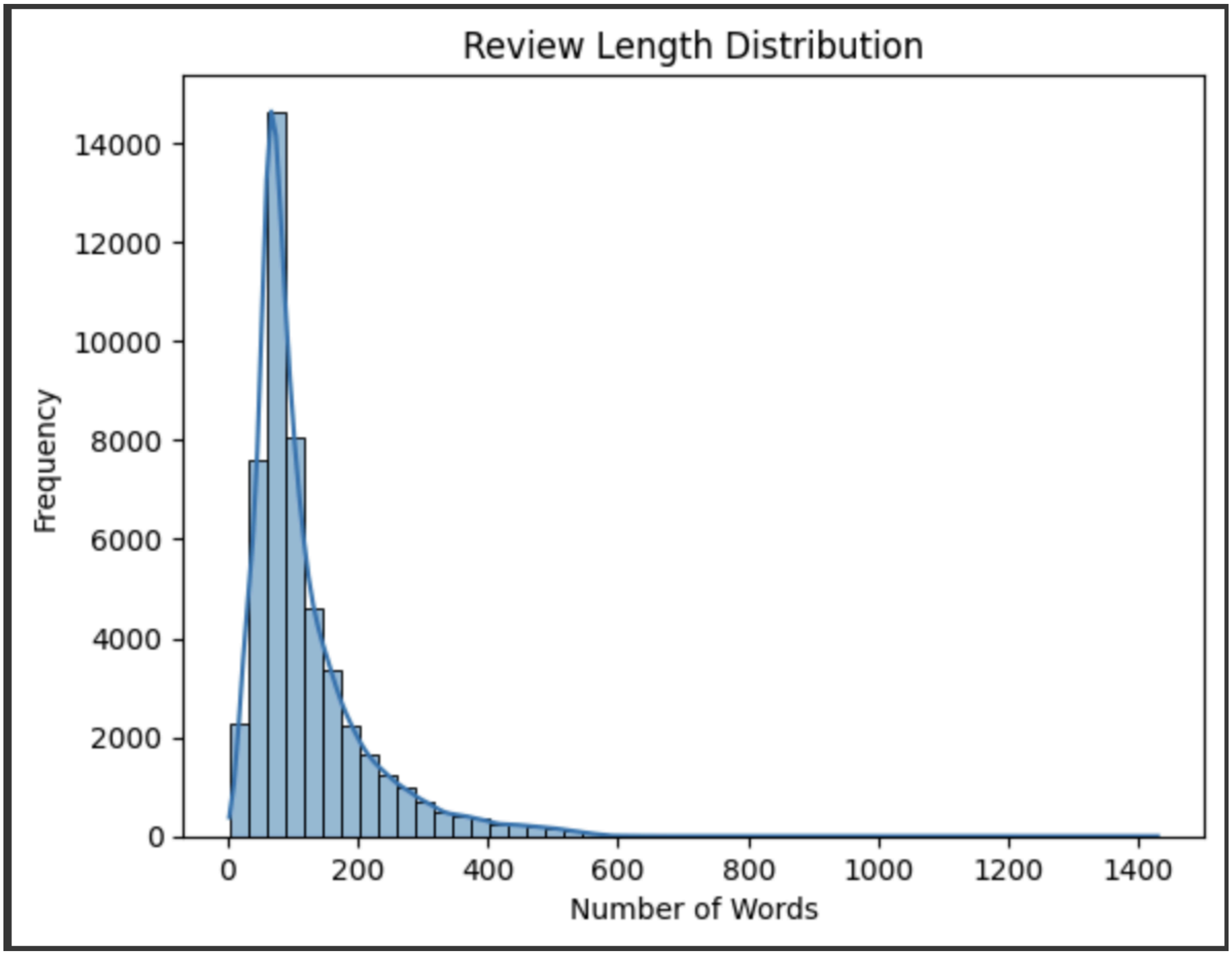
- Word Frequency: Common stopwords (‘the’ , ‘and’, ‘a’) dominate; sentiment-bearing words (‘great’, ‘awful’) are key.

Visualizations:

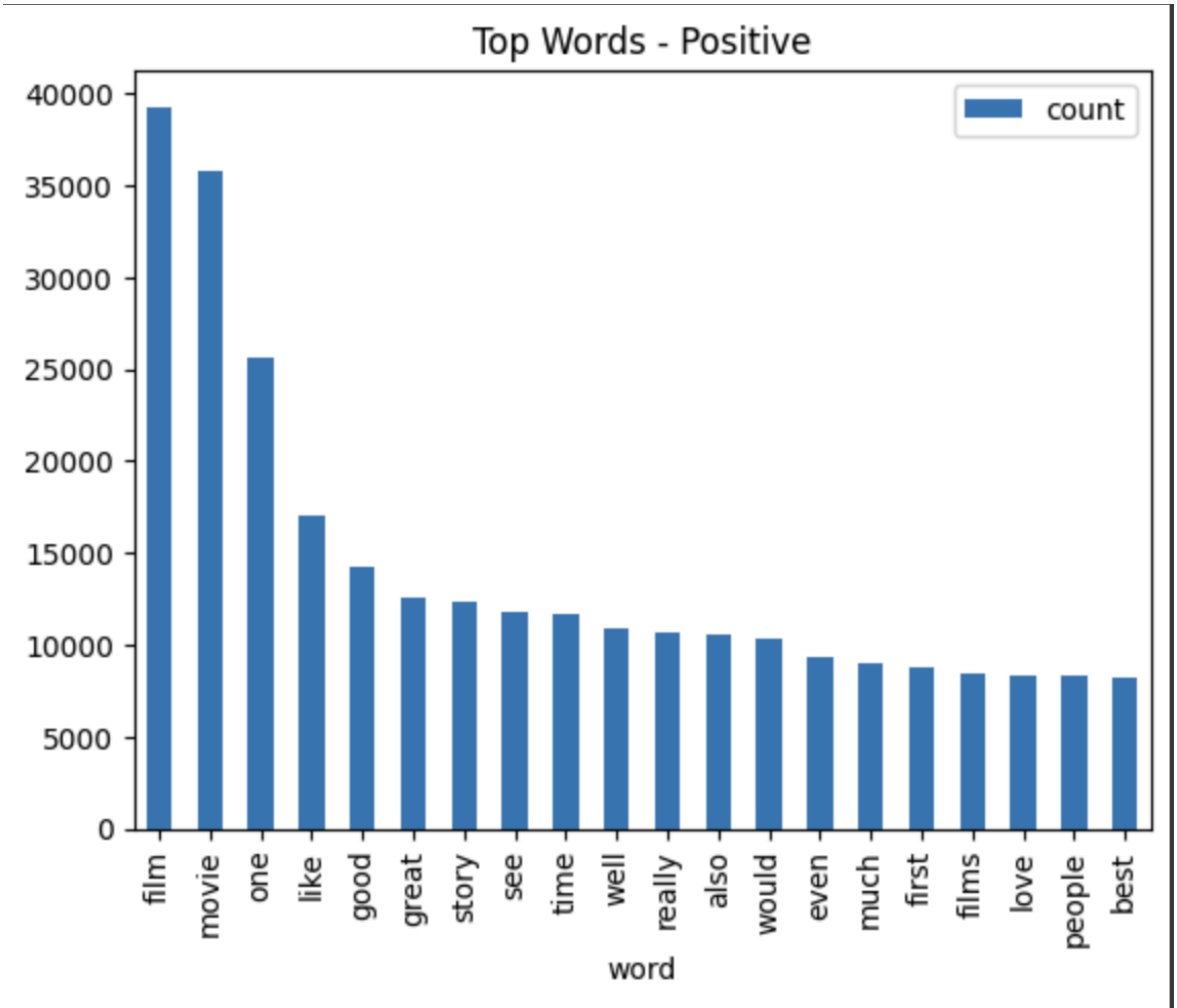
1. **Class Distribution Bar Plot**

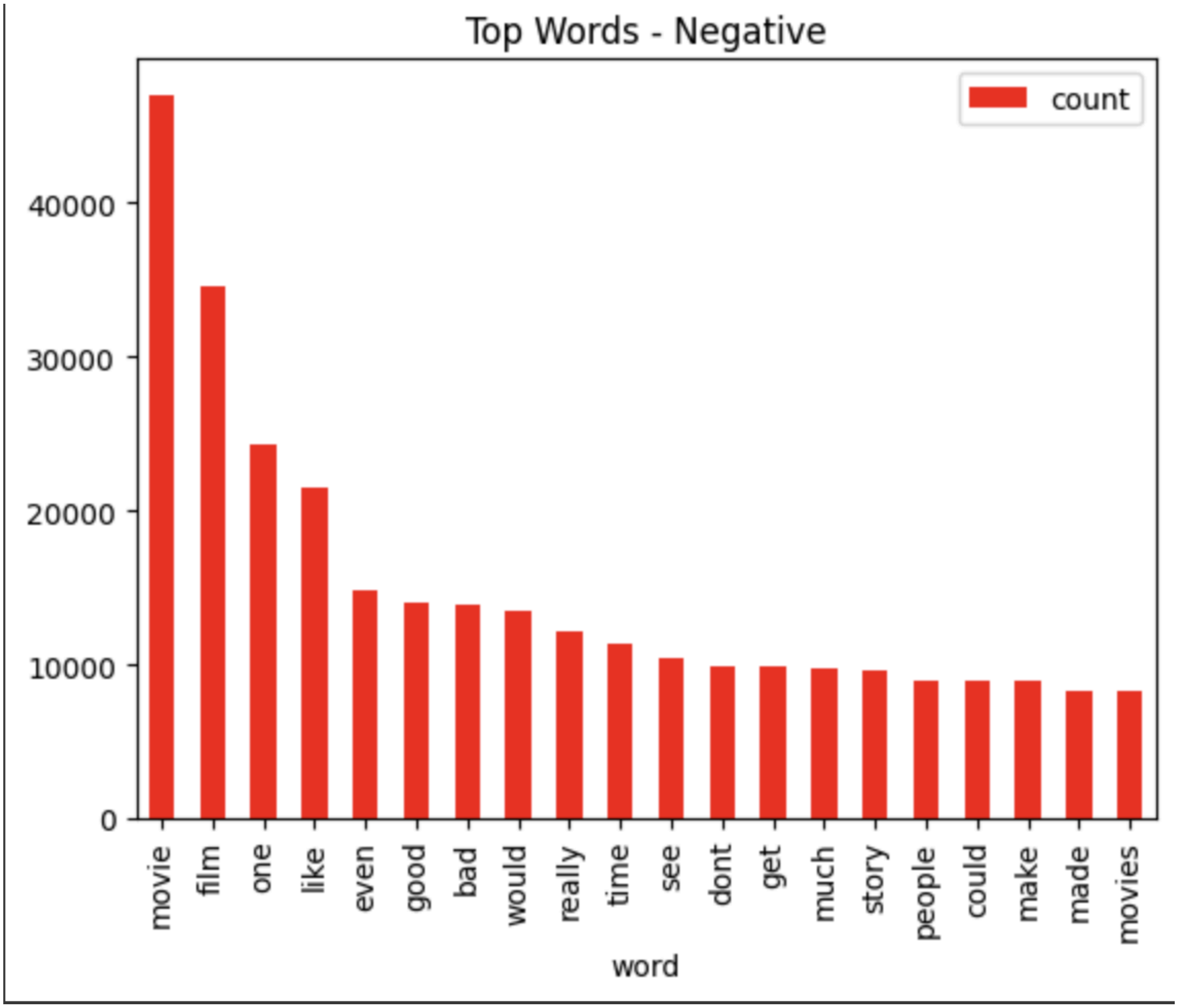


2. Review Length Histogram

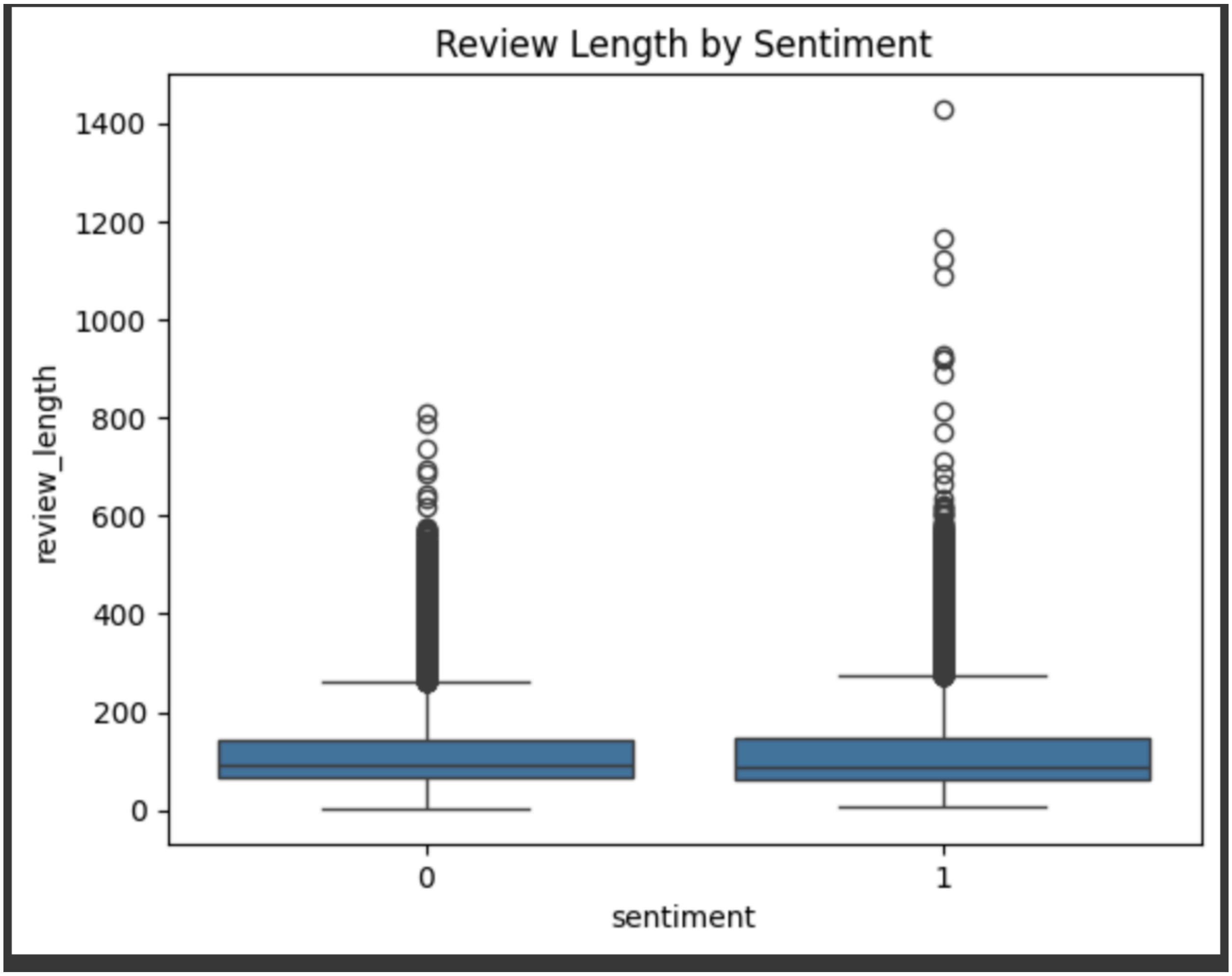


3. Word Cloud for Positive and Negative Reviews


Top Words - Positive
count
40000
35000
30000
25000
20000
15000
10000
5000
0
film
movie
one
like
good
great
story
see
time
well
really
also
would
even
much
first
films
love
people
best
word



4. Review Length Sentiment



Insights:

- Shorter reviews tend to be more polarized
- Neutral words dominate, requiring stop word removal.

3. **DATA PREPROCESSING AND FEATURE ENGINEERING**

Steps

- Handling Missing Values : None Found
- Text Cleaning:
    - Lowercasing

    - Removing HTML tags, punctuation and special characters.
    - Stopword removal (NLTK's English stopwords)
- Tokenization: Using nltk.word_tokenize()
- Embeddings:
    - TF-IDF (Logistic Regression)
    - Word2Vec (LSTM)

Justification:

- TF-IDF works well with traditional ML models by weighting important words.
- Word2Vec captures semantic relationships which improves deep learning performance.

4. MODEL IMPLEMENTATION

Model 1 : Logistic Regression

Feature Extraction

- Text Vectorization - Converted raw text to TF-IDF features and selected top 5000 most words with the most frequency
- Train Test Split - 80% training data, 20% test data

Training: For training a **Logistic Regression (Binary Classifier) algorithm** was used. It had the following default settings:

- Penalty - L2 regularization.
- Max iterations - 100
- Solver - *lbfgs* (preferred for small to medium datasets.)

Justification

- TF-IDF - Preferred over Bag of Words as it downweights frequent but less informative words.
- Logistic Regression - Simple and efficient for binary test classification.

Model 2 : LSTM (Deep Learning)

Architecture - The LSTM model had the following layers:

- Embedding layer - Converts tokenized words into dense vectors.
- LSTM layer - 2 stacked LSTM layers with tanh activation.
- Regularization - Dropout applied after each LSTM layer to avoid overfitting.
- Output Layer - Dense layer with sigmoid activation for binary classification

Hyperparameters:

- Sequence Length - Padded to 150-200 tokens based on ED
- Batch Size - Fixed at 64 for stable gradient updates.
- Learning Rate - 0.0001 with Adam optimizer
- Early Stopping -Monitored validation loss with a patience of 3

Justification:

- Vocabulary Size - Limited to 5000 for optimal performance as higher sizes reduced the accuracy.
- LSTM Units - The deeper networks e.g 64, 32 outperformed the smaller ones, 32,16 which suggested that complex sentiment patterns require higher capacity.

- Dropout - 0.2 showed better performance than 0.4, showing that moderate regularization was sufficient.

Results

Best Model - Model 1 (Accuracy - 89%, F1 Score - 0.89)

Model 3 comes in second because it is lighter and still captures a lot of the information without being computational heavy.

5. EXPERIMENTATION AND RESULTS

Experiment Table 1: LSTM (Deep Learning Model) - Hyperparameter Tuning

Model Number	Vocabulary size	Sequence length	Embedding dimensions	LSTM units	Dropout rate	Batch Size	Learning Rate	Accuracy	Precision	Recall	F1-Score
1	5000	200	64	64,32	0.2	64	0.0001	0.89	0.89	0.89	0.89
2	10,000	200	64	64,32	0.2	64	0.0001	0.54	0.57	0.54	0.48
3	5,000	200	64	32,16	0.4	64	0.0001	0.88	0.88	0.88	0.88
4	10,000	150	100	32,16	0.4	64	0.0001	0.87	0.87	0.87	0.87
5	7,500	150	100	32,16	0.4	64	0.0001	0.85	0.85	0.85	0.85

Experiment Table 2: Logistic Regression Model (Traditional ML Model)

Model Number	Embedding	Accuracy	Recall	Precision	F1-Score

1	TF-IDF	0.89	0.89	0.89	0.89
2	TF-IDF	0.89	0.89	0.89	0.89

6. CONCLUSION AND FUTURE WORK

Key Findings

- The LSTM model slightly outperformed the Logistic Regression model. The takeaway from this was that **deep learning better captures complex sentiment in text.**
- Logistic regression offered faster training as compared to the deep learning model.
- LSTM showed better precision with proper hypertuning.

Future Work

- Adopt transformer architectures for maximum accuracy and fast training.

7. TEAM CONTRIBUTIONS

Team Member	Task
Bernice Uwituze	Preprocessing, README, dataset research and analysis, visualizations
Jean Chrisostome	Preprocessing, visualizations, logistic regression model
Sifa Mwachoni	LSTM architecture design, model training and validation, report writing
Christian Mutabazi	Hyperparameter tuning, experiment tracking, model optimization

8. REFERENCES

- [1]IBM, “Sentiment Analysis,” Ibm.com, Aug. 24, 2023.

  https://www.ibm.com/think/topics/sentiment-analysis
- [2] “IMDB Dataset of 50K Movie Reviews,” www.kaggle.com.

  https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
- [3] GeeksforGeeks, “Bagofwords vs TFIDF,” GeeksforGeeks, Dec. 10, 2024.

  https://www.geeksforgeeks.org/bag-of-words-vs-tf-idf/