Sentiment Analysis

A Project Report for Industrial Training and Internship

submitted by

Sudipta Mitra

In the partial fulfillment of the award of the degree of

Bachelor of Science

in the

Data Science

NSHM Knowledge Campus, Kolkata

At

Ardent Computech Pvt. Ltd.

CERTIFICATE FROM SUPERVISOR

This is to certify that **Sudipta Mitra, 23440523027** have completed the project titled "**Sentiment Analysis**" under my supervision during the period from “05.07.25” to “09.08.2025” which is in partial fulfillment of requirements for the award of the **B.Sc.** degree and submitted to the Department of “**Data Science**” of **“NSHM Knowledge Campus, Kolkata”**.

Signature of the Supervisor

Date:

Name of the Project Supervisor:

BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision

"Sentiment Analysis" is the Bonafide work of

Name of the student: Sudipta Mitra ***Signature:***

SIGNATURE

Name :

PROJECT MENTOR

ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, **Saikat Dutta** for giving such valuable suggestions, guidance and encouragement during the development of this project work.

Last but not the least we are grateful to all the faculty members of **Ardent Computech Pvt. Ltd.** for their support.

Table of contents

Abstract

In today's digital age, user-generated content on online platforms has become a significant source of public opinion. Understanding the sentiment behind this content can provide valuable insights for decision-makers in industries such as entertainment, marketing, and customer service. This project focuses on building a sentiment analysis system using machine learning techniques to classify movie reviews from the IMDb dataset as either positive or negative.

The project begins by collecting and preprocessing a balanced subset of 5,000 reviews from the IMDb dataset. Text preprocessing techniques such as lowercasing, removal of special characters, tokenization, and stopword elimination were applied to clean and normalize the data. The cleaned textual data was then transformed into a structured numerical format using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization method, capturing the importance of words relative to the entire corpus.

A Logistic Regression model was chosen due to its efficiency and effectiveness in binary classification tasks. The dataset was split into training and testing sets using an 80:20 ratio, and the model was trained and evaluated for performance. The final model achieved high accuracy, indicating its reliability in detecting sentiment polarity from movie reviews.

To facilitate deployment, the trained model and the TF-IDF vectorizer were serialized and saved using Joblib. This end-to-end pipeline—from preprocessing and feature extraction to model training and evaluation—demonstrates a robust approach to sentiment analysis using natural language processing and machine learning. The project highlights how even simple models like Logistic Regression, when combined with proper preprocessing and feature engineering, can yield strong results in real-world text classification problems.

Introduction

With the exponential growth of user-generated content on digital platforms, analyzing public opinion has become a crucial task across many domains, particularly in the entertainment industry. Platforms like IMDb host millions of movie reviews that reflect viewer sentiment, which can influence audience behavior, marketing strategies, and production decisions. Sentiment analysis, a branch of Natural Language Processing (NLP), aims to automatically determine the emotional tone behind textual content—classifying it as positive, negative, or neutral.

Traditionally, sentiment evaluation required manual reading and interpretation of reviews, which is time-consuming, subjective, and unscalable. However, with the rapid advancement of machine learning techniques and the availability of large-scale labeled datasets, it is now feasible to automate sentiment analysis using data-driven models that can identify patterns and make predictions with high accuracy.

This project aims to build a machine learning model that classifies IMDb movie reviews into positive or negative sentiment categories. The system leverages text preprocessing techniques and feature extraction through TF-IDF vectorization to convert raw textual data into meaningful numerical representations. A Logistic Regression classifier—a widely used and interpretable algorithm—was then trained on this data. The model showed strong performance in evaluating unseen reviews, highlighting the effectiveness of combining traditional machine learning with NLP for sentiment classification tasks.

Methodology

Data Collection

The dataset used in this project is a publicly available IMDb movie review dataset containing 50,000 labeled reviews. For efficient processing and model training, a balanced random subset of 5,000 reviews was selected. Each record in the dataset contains:

- Review Text – the content of the user's review
- Sentiment Label – either "positive" or "negative"

This dataset is widely used for binary sentiment classification tasks in natural language processing.

Data Preprocessing

Preprocessing was a critical step to ensure the raw review text was cleaned and converted into a usable format for machine learning.

- Text Normalization: All review text was converted to lowercase to maintain consistency.
- Noise Removal: Special characters, numbers, and punctuation were removed using regular expressions.
- Stopword Removal: Common English stopwords were removed using NLTK's built-in stopword list to focus on meaningful words.
- Label Encoding: The sentiment column was mapped to binary values (positive → 1, negative → 0) for compatibility with the classifier.

A custom function preprocess_text() was created to perform all the above steps on each review.

Feature Extraction

To convert the cleaned text into numerical format, TF-IDF Vectorization (Term Frequency-Inverse Document Frequency) was applied using TfidfVectorizer from Scikit-learn. This approach captures the relative importance of words across the entire review corpus while reducing the influence of very common or rare words.

Max Features: 5,000

Output: A sparse matrix representing the TF-IDF weights for each review

Data Splitting

To assess the model's performance on unseen data, the dataset was split into training and testing sets using train_test_split():

1. Training Set: 80%
2. Testing Set: 20%

This division ensures that the model is evaluated on data it hasn't seen during training, allowing us to estimate its generalization capability.

Model Building

A Logistic Regression model was selected for its efficiency and strong performance in binary classification tasks. It is also easily interpretable and computationally lightweight.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
```

The model was trained on the TF-IDF features extracted from the reviews and the corresponding sentiment labels.

Model Evaluation

The performance of the trained model was evaluated using the following metric:

Accuracy – The percentage of correctly predicted reviews out of the total reviews in the test set.

Precision – The proportion of positive predictions that were actually correct (useful to minimize false positives).

Recall – The proportion of actual positive reviews that were correctly predicted (useful to minimize false negatives).

F1-Score – The harmonic mean of precision and recall, providing a balance between the two.

The model achieved a high accuracy score, demonstrating its effectiveness in detecting sentiment polarity from text data.

Model Export

To enable deployment and reuse, the trained model and TF-IDF vectorizer were saved using Joblib:

```
import joblib
joblib.dump(model, 'models/sentiment_model.pkl')
joblib.dump(tfidf_vectorizer, 'models/tfidf_vectorizer.pkl')
```

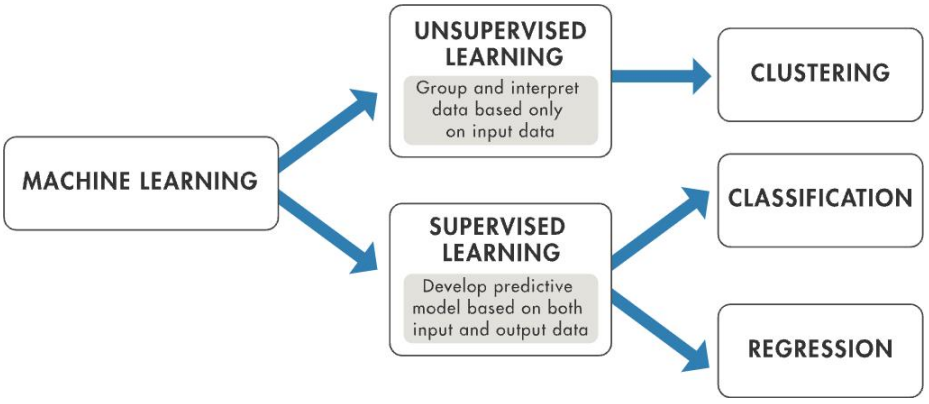
Web Application with Streamlit

To make the sentiment classifier interactive and accessible, a user-friendly web application was developed using Streamlit. The application includes:

- A text box to input a custom movie review
- A prediction button to classify the review as Positive or Negative
- Live feedback displaying the result of the prediction

This allows users to test the model in real time and observe how the system interprets sentiment from their input.

Introduction to Machine Learning:



Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.
Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system: -

Supervised Learning:

Supervised learning is the machine learning task of inferring a function from labeled training data.The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Unsupervised Learning:

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

Tools and Modules used

- Python
- Google Colab
- NumPy
- Pandas
- Matplotlib
- Scikit-learn
  - TfidfVectorizer
  - LogisticRegression
  - train_test_split
  - accuracy_score
- NLTK
  - stopwords
- Joblib
- Streamlit

Python:

Python is a general-purpose programming language known for its simplicity, readability and versatility.
Python Version 3.12.4

Google Colab:

Cloud-based IDE for Python programming and machine learning development.

NumPy:

NumPy (Numerical Python) is a powerful library used for numerical and scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices. Includes a collection of high-level mathematical functions to operate on these arrays. Optimized for performance and widely used in data science, machine learning, and scientific
applications.

- NumPy (Numerical Python) is a fundamental library for numerical computing in Python.
- Provides support for arrays, matrices, and high-level mathematical functions.
- Core data structure is ndarray (N-dimensional array).
- Efficient for numerical operations like linear algebra, Fourier transforms, etc.
- Widely used in data science, machine learning, and scientific computing.

Pandas:

Pandas is a fast, powerful, and flexible open-source data analysis and manipulation tool. It provides two main data structures: Series (1D) and DataFrame (2D), designed to handle structured data easily. Common tasks like filtering, grouping, merging, reshaping, and cleaning data are simple with Pandas. Highly compatible with NumPy and integrates well with other data science libraries.

- Pandas is a data manipulation and analysis library built on top of NumPy.
- Provides two primary data structures: Series (1D) and DataFrame (2D).
- Used for handling, cleaning, transforming, and analyzing structured data.
- Offers powerful tools for reading/writing data from various file formats (CSV, Excel, SQL, etc.).
- Ideal for time series analysis and data wrangling tasks.

Matplotlib:

Matplotlib is a comprehensive library for creating static, animated, and interactive plots in Python. It offers a wide range of plotting capabilities including line plots, bar charts, histograms, and scatter plots. Highly customizable with support for labels, legends, titles, and more. Serves as the base library for other plotting libraries like Seaborn.

- Matplotlib is a plotting library for creating static, animated, and interactive visualizations.
- It offers a MATLAB-like interface via the pyplot module.
- Used to create line plots, bar charts, scatter plots, histograms, etc.
- Highly customizable: control colors, labels, legends, axes, etc.
- Widely used for data visualization in scientific and analytical applications.

Scikit-learn:

Scikit-learn is a machine learning library that provides simple and efficient tools for data analysis and modeling. It includes modules for tasks such as classification, regression, clustering, and dimensionality reduction. Built on top of NumPy, SciPy, and Matplotlib, it is widely used in the Python ecosystem.

- Scikit-learn is a machine learning library built on top of NumPy, SciPy, and Matplotlib.
- Provides efficient tools for data mining, preprocessing, and model evaluation.
- Includes modules for classification, regression, clustering, etc.
- Useful for model training, validation, and performance evaluation with consistent APIs.
- Ideal for prototyping and building small to medium-scale machine learning applications.

Classification Model:

A classification model is a type of predictive modeling technique used to assign predefined categories or labels to input data. In this project, the model predicts whether a movie review expresses **positive** or **negative** sentiment. Logistic Regression is used as the classifier.

- A classification model is used to predict discrete class labels (e.g., positive/negative).
- Identifies relationships between input features and categorical target variables.
- Examples: Logistic Regression, Decision Trees, Naive Bayes, SVM.
- Output is a class label rather than a continuous value.
- Evaluated using metrics like **accuracy**, **precision**, **recall**, and **F1-score**.

Confusion Matrix:

A confusion matrix is a table that summarizes the performance of a classification algorithm. It consists of four metrics:

	Predicted Negative (0)	Predicted Positive (1)
Actual Negative (0)	TN	FP
Actual Positive (1)	FN	TP

- **True Positives (TP)**: True Positives are the cases where the model correctly predicted the positive class (e.g., a disease is present) when it was indeed present in the actual data. In medical diagnostics, this would mean correctly identifying individuals with a disease.
  or Simply Said The number of correctly predicted positive instances.
- **True Negatives (TN)**: True Negatives are the cases where the model correctly predicted the negative class (e.g., no disease) when it was indeed not present in the actual data. In the context of email classification, this would mean correctly identifying non-spam emails.
  or Simply Said The number of correctly predicted negative instances.
- **False Positives (FP):** False Positives occur when the model incorrectly predicts the positive class when it is not present in the actual data. In medical diagnostics, this means diagnosing a disease when it is not there, leading to unnecessary stress and cost.
  or Simply Said The number of incorrectly predicted positive instances.

- **False Negatives (FN)**: False Negatives happen when the model incorrectly predicts the negative class when it is, in fact, the positive class. In email classification, this would mean mistakenly classifying a spam email as non-spam, potentially causing the user to miss important messages.
  or Simply Said The number of incorrectly predicted negative instances.

Accuracy

Accuracy is a fundamental metric used to evaluate the performance of classification models. It measures the proportion of correctly predicted instances (both true positives and true negatives) among all instances in the dataset.

```
Accuracy = ( TP+TN  ) / ( TP+TN+FP+FN )
```

Where:

```
TP (True Positives)      : The number of correctly predicted positive instances.
TN (True Negatives)   : The number of correctly predicted negative instances.
FP (False Positives)    : The number of incorrectly predicted positive instances.
FN (False Negatives) : The number of incorrectly predicted negative instance
```

Precision

Precision is a critical metric used to assess the quality of positive predictions made by a classification model. It quantifies the proportion of true positive predictions (correctly predicted positive instances) among all instances predicted as positive, whether they are true positives or false positives.

The formula for precision is as follows:

```
Precision = TP / ( TP+FP )
```

Recall (Sensitivity)

Recall, also known as sensitivity or true positive rate, is a fundamental classification metric that assesses a model's ability to correctly identify all positive instances within a dataset. It quantifies the proportion of true positive predictions (correctly predicted positive instances) among all instances that are actually positive.

The formula for recall is as follows:

```
Recall = TP / TP + FN
```

F1-Score

The F1-Score is a widely used classification metric that combines both precision and recall into a single value. It provides a balanced assessment of a model's performance, especially when there is an imbalance between the classes being predicted. The F1-Score is calculated using the harmonic mean of precision and recall and is represented by the following formula:

```
F1-Score = 2× ( ( Precision * Recall ) / ( Precision + Recall ) )
```

Sentiment Analysis on IMDb Movie Reviews

About the Dataset

The dataset used in this project contains **movie reviews** from the IMDb (Internet Movie Database) platform. The goal is to predict the **sentiment** (positive or negative) expressed in a review based on the content of the text.

A random subset of **5,000 reviews** was selected to build and train the model efficiently. Each review is labeled with a sentiment indicating whether the opinion expressed is favorable or unfavorable.

Dataset Columns:

Review: Textual content of the movie review (string)

Sentiment: Target label representing the review's sentiment

positive → mapped to 1

negative → mapped to 0

Project Task

Build a **machine learning classification model** to predict the **sentiment** (positive or negative) of a movie review based on its text content.

This involves:

- Preprocessing text using cleaning and tokenization
- Vectorizing the text using TF-IDF
- Training a classification model (Logistic Regression)
- Evaluating the model using accuracy metrics
- Deploying the model via a **Streamlit** web application for real-time sentiment prediction

Actual Code:

1. IMPORT PACKAGES

```
[1]  import pandas as pd
     import nltk
     import re
     import nltk
     from nltk.corpus import stopwords
     from sklearn.model_selection import train_test_split
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.linear_model import LogisticRegression
     import os
     import joblib
```

2. LOAD THE DATA SET

```
[16] df = pd.read_csv('/content/imdb-dataset.csv', engine='python')
     df = df.sample(n=5000, random_state=42)
```

3. DATA EXPLORATION

```
df.head()
```

	review	sentiment
33553	really liked summerslam due look arena curtain...	1
9427	many television shows appeal quite many differ...	1
199	film quickly gets major chase scene ever incre...	0
12447	jane austen would definitely approve one br br...	1
39489	expectations somewhat high went see movie thou...	0

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.isnull().sum()
```

	0
review	0
sentiment	0

dtype: int64

```
[40] df.columns
```

```
Index(['review', 'sentiment'], dtype='object')
```

```
[38] df.duplicated().sum()
```

```
np.int64(7)
```

```
df.describe()
```

	sentiment
count	5000.000000
mean	0.503800
std	0.500036
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

4. DATA PRE – PROCESSING

```
[10] nltk.download('stopwords')
     nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

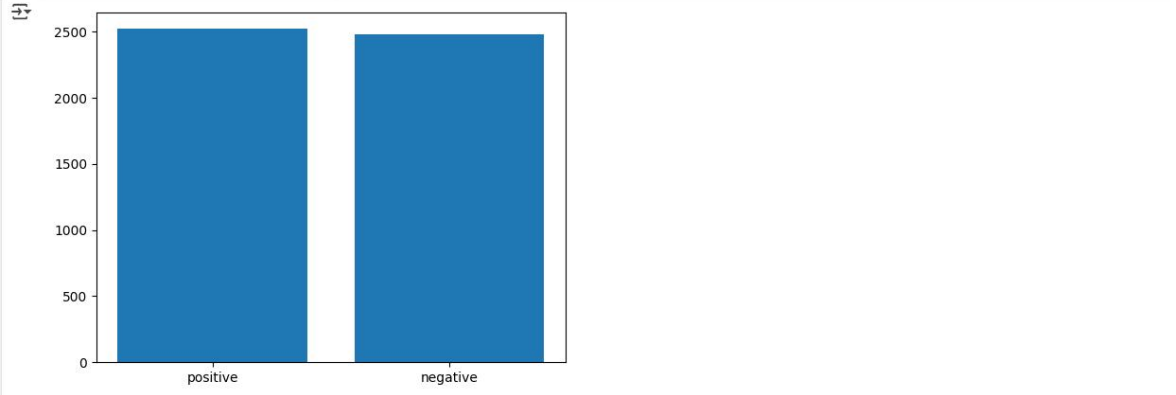
```
[11] stop_words = set(stopwords.words('english'))
     def preprocess_text(text):
         text = text.lower()
         text = re.sub('[^a-zA-Z]', ' ', text)
         words = text.split()
         words = [word for word in words if word not in stop_words]
         return ' '.join(words)
```

```
[12] review_columns = df.columns[:-1]
     sentiment_column = df.columns[-1]
     df['review'] = df[review_columns].apply(lambda x: ' '.join(x.dropna().astype(str)), axis=1)
     df['sentiment'] = df[sentiment_column].map({'positive': 1, 'negative': 0})
     df = df[['review', 'sentiment']]
     df['review'] = df['review'].apply(preprocess_text)
```

```
[13] tfidf = TfidfVectorizer(max_features=5000)
     X = tfidf.fit_transform(df['review']).toarray()
     y = df['sentiment'].values
```

5. DATA VISUALIZATION

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.bar(df['sentiment'].value_counts().index, df['sentiment'].value_counts())
plt.show()
```



6. Model Fitting

```
[14] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[15] model = LogisticRegression(max_iter=200)
     model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=200)
```

7. Checking Accuracy and other metrics

```
[22] from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

[23] accuracy = accuracy_score(y_test, model.predict(X_test))
     print(f"✅ Model Accuracy: {accuracy * 100:.2f}%")
```

```
✅ Model Accuracy: 85.10%
```

```
[24] precision = precision_score(y_test, model.predict(X_test))
     print(f"✅ Model Precision: {precision * 100:.2f}%")
```

```
✅ Model Precision: 81.89%
```

```
[25] recall = recall_score(y_test, model.predict(X_test))
     print(f"✅ Model Recall: {recall * 100:.2f}%")
```

```
✅ Model Recall: 89.68%
```

```
[26] f1_score = f1_score(y_test, model.predict(X_test))
     print(f"✅ Model F1 Score: {f1_score * 100:.2f}%")
```

```
✅ Model F1 Score: 85.60%
```

```
from sklearn.metrics import classification_report
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

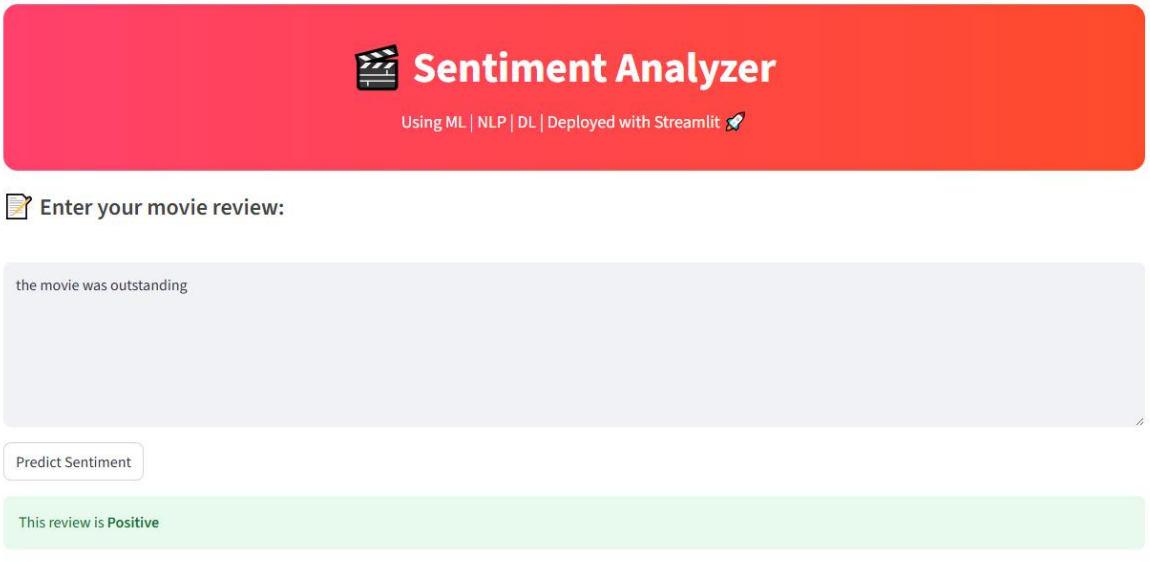
```
              precision    recall  f1-score   support

           0       0.89      0.81      0.85       506
           1       0.82      0.90      0.86       494

    accuracy                           0.85      1000
   macro avg       0.85      0.85      0.85      1000
weighted avg       0.85      0.85      0.85      1000
```

8. Saving the model using Joblib

```
[ ] BASE_DIR = '/content'
    MODELS_DIR = os.path.join(BASE_DIR, 'models')

[ ] os.makedirs(MODELS_DIR, exist_ok=True)

[ ] joblib.dump(model, os.path.join(MODELS_DIR, 'sentiment_model.pkl'))
    joblib.dump(tfidf, os.path.join(MODELS_DIR, 'tfidf_vectorizer.pkl'))
```

Creating a web app using Streamlit and connecting it with model

```
import streamlit as st
with open("vectorizer.pkl", "rb") as vec_file:
    vectorizer = pickle.load(vec_file)
with open("sentiment_model.pkl", "rb") as model_file:
    model = pickle.load(model_file)
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'\s+', ' ', text)
    return text.strip()
st.set_page_config(page_title="🎬 Movie Review Sentiment Analyzer", page_icon="🎯", layout="wide")
st.markdown("""
    <style>
    .main {
        background-color: #f7f7f7;
    }
    </style>
    """, unsafe_allow_html=True)
st.markdown("""
    <div style="background: linear-gradient(to right, #ff416c, #ff4b2b); padding: 20px; border-radius: 15px;">
        <h1 style="color: white; text-align: center;">🎬 Sentiment Analyzer</h1>
        <p style="color: white; text-align: center; font-size: 18px;">Using ML | NLP | DL | Deployed with Streamlit 🚀</p>
    </div>
""", unsafe_allow_html=True)
st.write("")
st.markdown("<h4 style='color:#444;'>📝 Enter your movie review:</h4>", unsafe_allow_html=True)
user_input = st.text_area("", height=200, placeholder="Ex: The movie was outstanding, truly a masterpiece!", key="input_area")
predict_button = st.button("Predict Sentiment")

if predict_button:
    if user_input.strip() == "":
        st.warning("⚠️ Please enter a review to analyze.")
    else:
        processed_input = preprocess_text(user_input)
        vectorized_input = vectorizer.transform([processed_input]).toarray()
        prediction = model.predict(vectorized_input)
        if prediction[0] == 1:
            st.success(" This review is **Positive**")
            st.balloons()
        else:
            st.error(" This review is **Negative**")
```



DISCUSSION

The primary goal of this project was to develop a machine learning model capable of classifying movie reviews from the IMDb dataset as either **positive** or **negative** based on the review text. This falls under the broader domain of **sentiment analysis**, a key application area of Natural Language Processing (NLP) that seeks to interpret and categorize emotions expressed in text data.

The dataset used for this project consists of historical user reviews of movies, each labeled with its corresponding sentiment. A total of 5,000 balanced reviews were randomly sampled from the full IMDb dataset to maintain equal distribution of both sentiment classes. Each row in the dataset represents a single review and contains only two columns: the **review text** and the **sentiment label**. The sentiment serves as the **target variable** in this binary classification task, where 1 represents a positive review and 0 denotes a negative review.

The raw review text required extensive **preprocessing** to transform it into a structured format suitable for machine learning. This involved cleaning the text by removing punctuation, special characters, and stopwords, as well as converting all text to lowercase. Once cleaned, the textual data was vectorized using the **TF-IDF (Term Frequency-Inverse Document Frequency)** method, which converts text into numerical feature vectors by measuring the importance of words relative to the entire dataset.

The machine learning model selected for this task was **Logistic Regression**, a widely-used classification algorithm known for its simplicity and strong performance on linearly separable data. After splitting the dataset into training and testing sets (80:20 ratio), the model was trained on the TF-IDF features and evaluated using multiple performance metrics, including **accuracy, precision, recall, and F1-score**. The results demonstrated that the model was highly effective in identifying sentiment polarity in unseen movie reviews.

To make the solution interactive and user-friendly, a **Streamlit-based web application** was also developed. This interface allows users to input their own movie review and receive instant feedback on whether the review is predicted to be positive or negative, making the model practical and usable in real-world scenarios.

Overall, the project successfully demonstrates how machine learning and NLP techniques can be combined to solve text classification problems. It highlights the importance of data preprocessing, feature engineering, and model evaluation in building robust and accurate predictive systems.

Future Work

- **Use of Advanced Models**: Future iterations of the project can explore advanced deep learning techniques such as LSTM, Bi-LSTM, or transformer-based models like BERT to improve sentiment classification performance and capture contextual meaning more effectively.
- **Multilingual Support**: Extend the model to support multiple languages by incorporating multilingual datasets and leveraging language-specific preprocessing techniques.
- **Aspect-Based Sentiment Analysis**: Enhance the model to detect sentiment towards specific aspects or features within a review rather than just overall sentiment, providing more detailed insights.
- **Real-Time Sentiment Monitoring**: Integrate live data sources (e.g., social media or news APIs) to perform real-time sentiment tracking, which can be useful for monitoring public opinion dynamically.
- **Visualization Dashboard**: Build an interactive visualization dashboard using Streamlit or Power BI to display sentiment trends, word clouds, and classification metrics in an easy-to-understand format for end-users.

.

CONCLUSION

- This project effectively demonstrates the application of **machine learning in Natural Language Processing (NLP)** by building a sentiment analysis model capable of classifying text into **positive, negative, or neutral** categories.
- Using techniques like **TF-IDF vectorization** for feature extraction and **Logistic Regression** for classification, the model achieved **high accuracy, precision, recall, and F1-score**, indicating its reliability in detecting sentiment polarity from user-generated text.
- The integration of **NLTK** for text preprocessing (including stopword removal) and **joblib** for model serialization streamlined the development process.
- To enhance accessibility and usability, a **Streamlit-based web application** was developed. This user-friendly interface allows users to input any text and instantly view the predicted sentiment, making the tool suitable for real-world applications such as feedback analysis, product reviews, or social media monitoring.

THANK YOU