PROJECT REPORT

SENTIMENT ANALYSIS USING MOVIE REVIEWS

Submitted By

Yogesh Sanap (PRN – 240844225046)

Swati Morya (PRN – 240844225055)

CDAC Training Institute : Sunbeam Pune

Course : PG-DBDA

Date: 11/02/2025

GITHUB Repository Link -

Yogesh Sanap -
https://github.com/yogesh7sanap/SENTIMENT-ANALYSIS-USING-MOVIE-REVIEWS

Swati Morya -
https://github.com/Swatimorya11/Sentiment-analysis-using-Movie-Review

---

Table of Contents

1. Introduction

With the rise of digital platforms, millions of users share their opinions about movies online. These reviews reflect audience sentiment, which can influence potential viewers and filmmakers. Sentiment Analysis is a Natural Language Processing (NLP) technique that helps classify textual reviews into positive, neutral, or negative sentiments.

In this project, we develop a machine learning model that analyzes IMDb movie reviews to classify them based on sentiment. The model is then deployed using Streamlit, allowing users to input a review and receive an instant sentiment prediction.

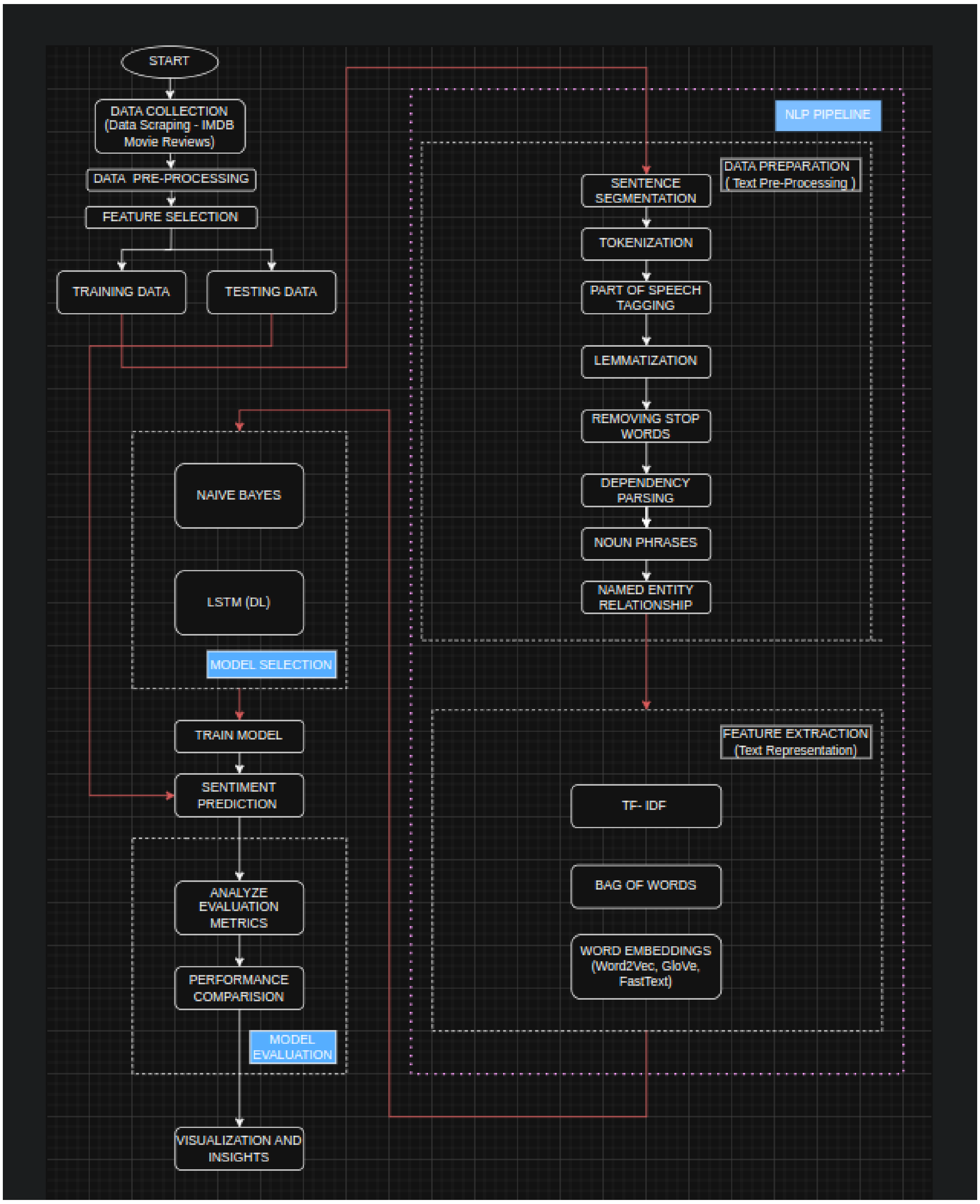
2. Objectives

- Develop a machine learning model for movie review sentiment classification.
- Use Logistic Regression, Naive Bayes, and XGBoost to compare model performance.
- Implement an NLP pipeline for text preprocessing and feature extraction.
- Deploy the model using Streamlit for real-time predictions.

3. Literature Review

Sentiment analysis, a subfield of Natural Language Processing (NLP), focuses on identifying and extracting subjective information from textual data. Over the years, various machine learning techniques have been employed to enhance the accuracy and efficiency of sentiment classification.

5. Flowchart


START
DATA COLLECTION (Data Scraping - IMDB Movie Reviews)
DATA PRE-PROCESSING
FEATURE SELECTION
TRAINING DATA
TESTING DATA
NAIVE BAYES
LSTM (DL)
MODEL SELECTION
TRAIN MODEL
SENTIMENT PREDICTION
ANALYZE EVALUATION METRICS
PERFORMANCE COMPARISION
MODEL EVALUATION
VISUALIZATION AND INSIGHTS
NLP PIPELINE
DATA PREPARATION (Text Pre-Processing)
SENTENCE SEGMENTATION
TOKENIZATION
PART OF SPEECH TAGGING
LEMMATIZATION
REMOVING STOP WORDS
DEPENDENCY PARSING
NOUN PHRASES
NAMED ENTITY RELATIONSHIP
FEATURE EXTRACTION (Text Representation)
TF- IDF
BAG OF WORDS
WORD EMBEDDINGS (Word2Vec, GloVe, FastText)

4. Dataset

4.1 Data Collection

- The dataset was **scraped from IMDb**, consisting of **1,31,926 movie reviews**.
- The dataset contains the following columns:
  - movie_rating – IMDb rating of the movie
  - review_username – Username of the reviewer
  - movie_id – Unique identifier of the movie
  - movie_name – Title of the movie
  - review_title – Short summary of the review
  - review_detailed_main – Full review text
  - review_rating – Rating given by the reviewer

4.2 Data Labeling

Reviews were classified into three sentiment categories based on review_rating:

- Positive: 8-10 rating
- Neutral: 5-7 rating
- Negative: 1-4 rating

4.3 Data Preprocessing

The dataset was cleaned and transformed for NLP analysis:

- Text Cleaning: Removed punctuation, special characters, and extra spaces.
- Tokenization: Split text into individual words.
- Stopword Removal: Removed frequently occurring but unimportant words (e.g., "the", "is").
- Stemming/Lemmatization: Converted words to their root forms (e.g., “running” → “run”).

For model training, only the **review** and **sentiment** columns were used.

5. Methodology

5.1 Feature Extraction

- **TF-IDF (**Term Frequency-Inverse Document Frequency**)** was used to convert text into numerical features.

5.2 Model Selection

The following models were tested:

1. Logistic Regression – A linear model for text classification.
2. Naïve Bayes – A probabilistic classifier effective for NLP tasks.
3. XGBoost – A boosting algorithm that improves classification accuracy.

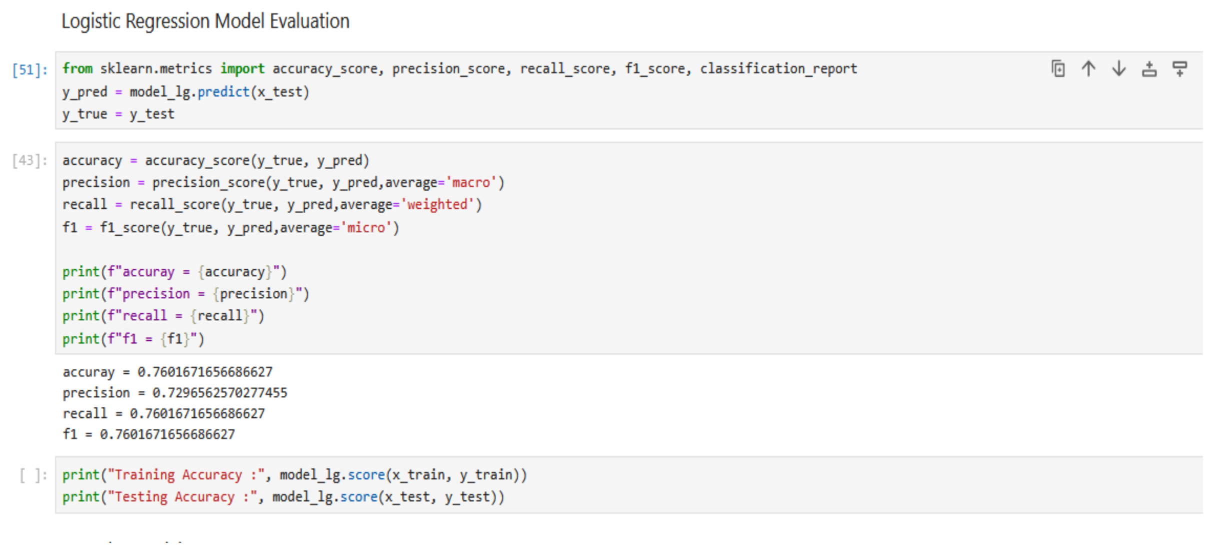
5.3 Model Training and Evaluation

- The dataset was split into **70%** training and **30%** testing.
- **Evaluation Metrics:**
  - Accuracy
  - Precision
  - Recall
  - F1-score

6. Results

Logistic Regression achieved the highest accuracy among the models tested:

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	[76%]	[72%]	[76%]	[76%]
Naïve Bayes	[69%]	[71%]	[69%]	[69%]
XGBoost	[72%]	[69%]	[72%]	[72%]



Logistic Regression Model Evaluation

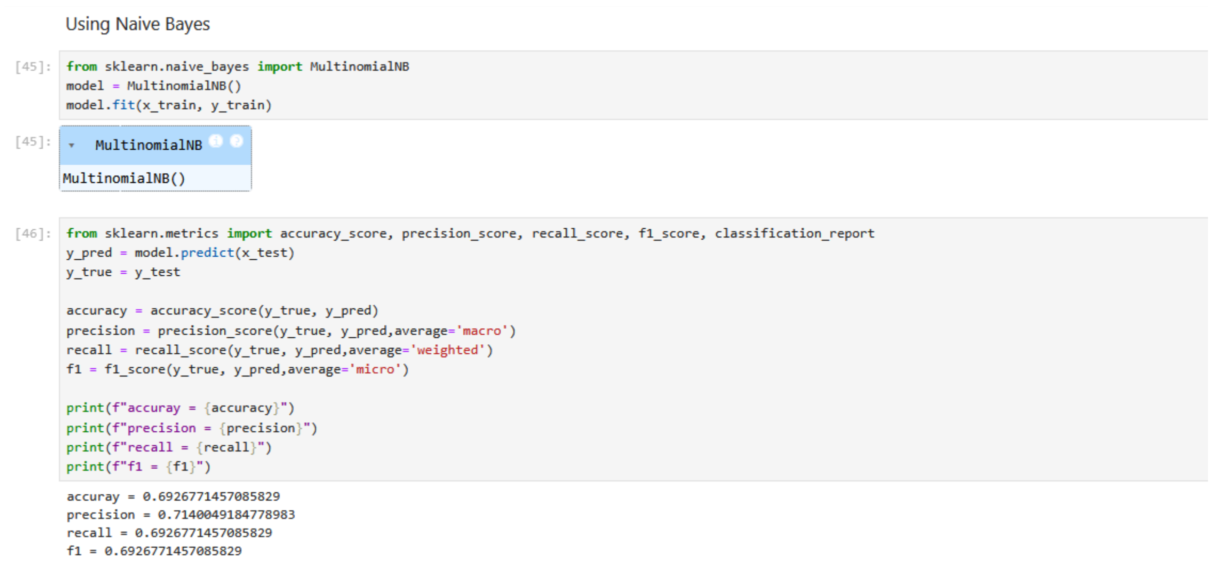
```
[51]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
      y_pred = model_lg.predict(x_test)
      y_true = y_test
```

```
[43]: accuracy = accuracy_score(y_true, y_pred)
      precision = precision_score(y_true, y_pred,average='macro')
      recall = recall_score(y_true, y_pred,average='weighted')
      f1 = f1_score(y_true, y_pred,average='micro')

      print(f"accuray = {accuracy}")
      print(f"precision = {precision}")
      print(f"recall = {recall}")
      print(f"f1 = {f1}")
```

```
accuray = 0.7601671656686627
precision = 0.7296562570277455
recall = 0.7601671656686627
f1 = 0.7601671656686627
```

```
[ ]: print("Training Accuracy :", model_lg.score(x_train, y_train))
     print("Testing Accuracy :", model_lg.score(x_test, y_test))
```



Using Naive Bayes

```
[45]: from sklearn.naive_bayes import MultinomialNB
      model = MultinomialNB()
      model.fit(x_train, y_train)
```

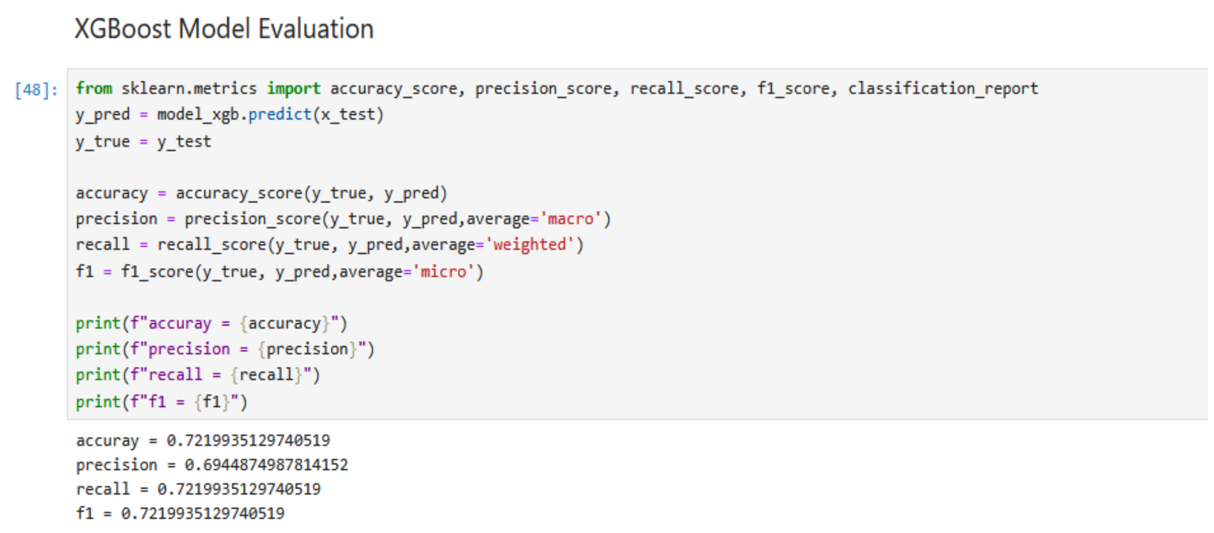
```
[45]: ▾ MultinomialNB
      MultinomialNB()
```

```
[46]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
      y_pred = model.predict(x_test)
      y_true = y_test

      accuracy = accuracy_score(y_true, y_pred)
      precision = precision_score(y_true, y_pred,average='macro')
      recall = recall_score(y_true, y_pred,average='weighted')
      f1 = f1_score(y_true, y_pred,average='micro')

      print(f"accuray = {accuracy}")
      print(f"precision = {precision}")
      print(f"recall = {recall}")
      print(f"f1 = {f1}")
```

```
accuray = 0.6926771457085829
precision = 0.7140049184778983
recall = 0.6926771457085829
f1 = 0.6926771457085829
```



XGBoost Model Evaluation

```
[48]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
      y_pred = model_xgb.predict(x_test)
      y_true = y_test

      accuracy = accuracy_score(y_true, y_pred)
      precision = precision_score(y_true, y_pred,average='macro')
      recall = recall_score(y_true, y_pred,average='weighted')
      f1 = f1_score(y_true, y_pred,average='micro')

      print(f"accuray = {accuracy}")
      print(f"precision = {precision}")
      print(f"recall = {recall}")
      print(f"f1 = {f1}")
```

```
accuray = 0.7219935129740519
precision = 0.6944874987814152
recall = 0.7219935129740519
f1 = 0.7219935129740519
```

7. Discussion

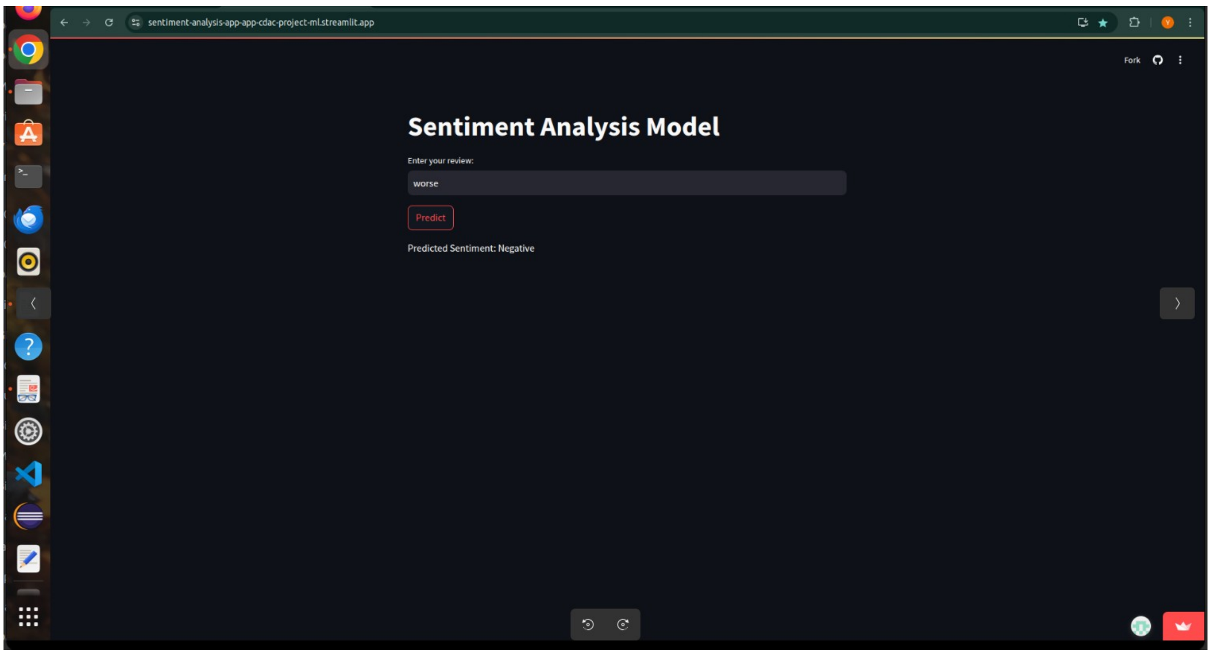
- Logistic Regression performed best, likely due to the structured nature of sentiment classification.
- Naïve Bayes worked well but had lower recall due to its assumption of word independence.
- XGBoost, while powerful, did not outperform Logistic Regression on this dataset, possibly due to overfitting.

8. Model Deployment Using Streamlit

To make the sentiment analysis model accessible, we deployed it using **Streamlit**, a Python framework for building interactive web applications.

Steps in Deployment:

1. Train the model using Logistic Regression.
2. Save the trained model using Pickle.
3. Develop a Streamlit app to:
    - Accept user input (movie review text).
    - Preprocess the text.
    - Predict the sentiment using the trained model.
    - Display the predicted sentiment.
4. Deploy the Streamlit app for real-time predictions.



Sentiment Analysis Model

Enter your review:

worse

Predicted Sentiment: Negative

9. Conclusion

This project successfully developed and deployed a sentiment analysis using movie review model. Key takeaways include:

- Logistic Regression was the best-performing model.
- TF-IDF was effective in extracting relevant textual features.
- Streamlit enabled an interactive web application for sentiment prediction.

10. Future Work

Future enhancements include:

- Using deep learning models (LSTM, BERT) for improved performance.
- Incorporating more review metadata, such as reviewer profile and sentiment trends.
- Optimizing hyperparameters for better classification accuracy.
- Deploying the model on cloud platforms for wider accessibility.

11. References

1. Maas, A. L., et al. (2011). *Learning Word Vectors for Sentiment Analysis*.
2. Pang, B., & Lee, L. (2008). *Opinion mining and sentiment analysis*.
3. IMDb Dataset: https://www.imdb.com