International Centre for Education and Research (ICER)

VIT – Bangalore

# Sentiment Analysis Using Python and Machine Learning

Project - I Report

*Submitted by*

**Devjyot Singh Sidhu**

**(24MSP3075)**

***In partial fulfilment for the award of the degree of***

**Post Graduate Programme**

**International Centre for Education and Research**

**VIT – Bangalore**

**December, 2024**

## International Centre for Education and Research (ICER)

## VIT – Bangalore

## Bonafide Certificate

Certified that this project report “**Sentiment Analysis Using Python and Machine Learning**” is the bonafide record of work done by “**Devjyot Singh Sidhu – 24MSP3075**” who carried the project work under my supervision.

| | |
|---|---|
| **Signature of Supervisor** | **Signature of Director** |
| **Prof. Kartheek GCR** | **Prof. Prema M** |
| ICER | **Director,** |
| VIT – Bangalore | ICER |
| | VIT - Bangalore |

**Evaluation Date:** 13th December, 2024

## International Centre for Education and Research (ICER)

## VIT – Bangalore

### Acknowledgements

I express my sincere gratitude to our director of ICER, VIT - Bangalore Prof. Prema M. for their support and for providing the required facilities for carrying out this study.

I wish to thank my faculty supervisor, "Prof. Kartheek GCR", ICER, VIT - Bangalore for extending help and encouragement throughout the project. Without his/her continuous guidance and persistent help, this project would not have been a success for me.

I am grateful to all the members of ICER, VIT – Bangalore, my beloved parents, and friends for extending the support, who helped us to overcome obstacles in the study.

**Devjyot Singh Sidhu**

**24MSP3075**

## Contents

## List of Figures

<u>**List of Code Snippets**</u>

## 1. <u>Abstract</u>

Sentiment analysis is an essential tool in natural language processing that determines the sentiment of a given piece of text. This project focuses on analysing sentiments from textual data using Python and machine learning techniques. The IMDB movie review dataset, containing 50,000 reviews, was used to build and evaluate a model that classifies reviews as positive or negative. We employed text preprocessing, feature extraction using the Bag of Words approach, and trained a Naive Bayes classifier. The model achieved 85% accuracy, demonstrating its effectiveness in distinguishing between positive and negative reviews. However, limitations were observed in handling sarcasm and ambiguous text, which could be addressed in future work.

# 2. Problem Statement

## 2.1 Business Context

Sentiment analysis is widely used in industries to gauge customer satisfaction, monitor brand reputation, and derive actionable insights. Companies rely on user-generated data, such as reviews and social media posts, to understand customer sentiments and improve their products or services.

## 2.2 Challenges

1. Ambiguity in language (e.g., sarcasm or double negatives).
2. Handling large-scale textual data efficiently.
3. Preprocessing to standardize text without losing context.
4. Building a scalable model that generalizes well to unseen data.

## 2.3 Objectives

The project aims to:

1. Classify textual reviews as positive or negative.
2. Achieve high accuracy and reliability.
3. Lay the groundwork for further improvements, such as handling complex linguistic nuances.

# 3. Literature Review

## 3.1 Traditional Approaches[1]

Early sentiment analysis relied on lexicon-based techniques, where predefined dictionaries of words with positive or negative sentiments were used. While simple, these methods lacked contextual understanding.

## 3.2 Machine Learning Approaches[2][4][5]

Machine learning introduced algorithms like Naive Bayes, Support Vector Machines, and Logistic Regression. These models use features extracted from text (e.g., frequency of words) to classify sentiments, offering better accuracy than lexicon-based methods.

## 3.3 Modern Deep Learning Approaches[3]

Recent advancements like Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and transformers (e.g., BERT) enable deeper contextual understanding. However, these methods require substantial computational resources.

## 3.4 Naïve Bayes Classification

The Naive Bayes Classifier is a simple yet powerful probabilistic machine learning algorithm based on Bayes' Theorem. It is primarily used for classification tasks, such as spam detection, sentiment analysis, and medical diagnosis.

It is based on Baye’s Theorem which allows one to calculate the probability of a hypothesis given some evidence and it is given as follows:

$$P(A|B) = \frac{P(B|A) \bullet P(A)}{P(B)}$$

Where,

- P(A|B): Probability of hypothesis A happening given data B.
- P(B|A): Probability of hypothesis B happening given data A.
- P(A): Prior probability of hypothesis A.
- P(B): Total probability of data B.

# 4. Methodology

## 4.1 Dataset Description

The IMDB dataset contains 50,000 movie reviews equally divided into positive and negative sentiments. Each review is a string of text, varying from a few words to several paragraphs.

Sample Data:

| Review | Sentiment |
|---|---|
| "The movie was fantastic and thrilling!" | Positive |
| "It was a waste of time, completely dull." | Negative |

## 4.2 Exploratory Data Analysis (EDA)

1. **Class Distribution**: The dataset is balanced, with 25,000 positive and 25,000 negative reviews.
2. **Review Length Analysis**: Most reviews are between 100 and 200 words.

```
data['sentiment'].value_counts().plot(kind='bar', color=['green', 'red'])
plt.title('Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```

*Code 1: Driver code for review length analysis*

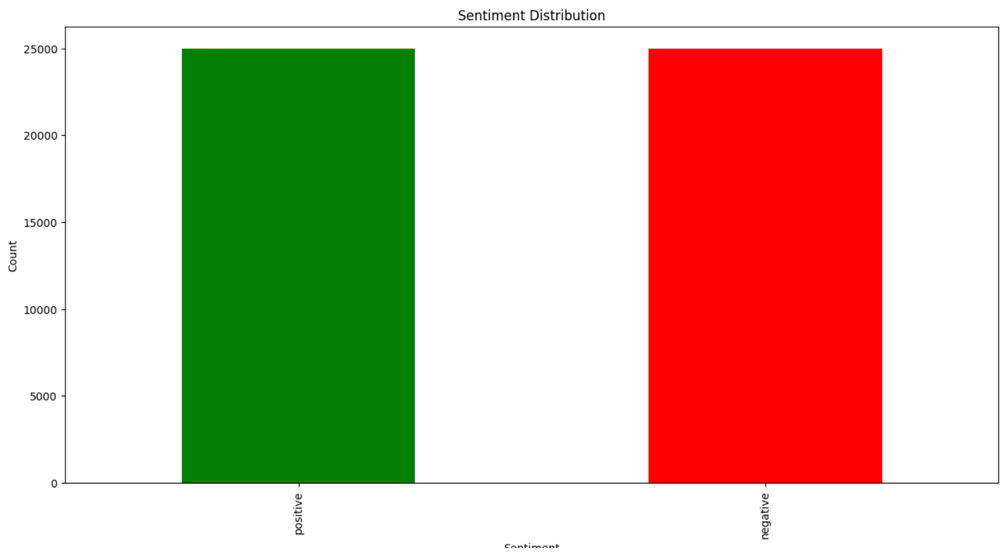


*Fig 1: Review Length Analysis for the dataset*

## 4.3 Data Preprocessing

1. Text Cleaning:
    - Remove HTML tags, special characters, and extra spaces.
    - Convert text to lowercase.
2. Tokenisation: Split txt into words for processing.
3. Stopword Removal: Remove common words like "is," "the," and "and" that don’t add meaning.

## 4.4 Feature Extraction

We used the Bag of Words (BoW) technique to convert text into a numerical matrix. The vectorization step transformed each review into a fixed-length representation, where each column represents the frequency of a specific word in the vocabulary.

## 4.5 Training a Naive Bayes Classifier:

- Ideal for text classification tasks.
- Efficient with small datasets and sparse matrices like BoW.

# 5. Code Implementation

### 5.1 Importing necessary modules

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
```

*Code 2: Importing modules*

### 5.2 Data Loading and Cleaning

```python
data = pd.read_csv('IMDB Dataset.csv')
data['review'] = data['review'].str.lower()
```

*Code 3: Driver code for data loading and cleaning*

### 5.3 Vectorisation

```python
vectorizer = CountVectorizer(stop_words='english')
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

*Code 4: Driver code for vectorisation*

### 5.4 Model Training and Evaluation

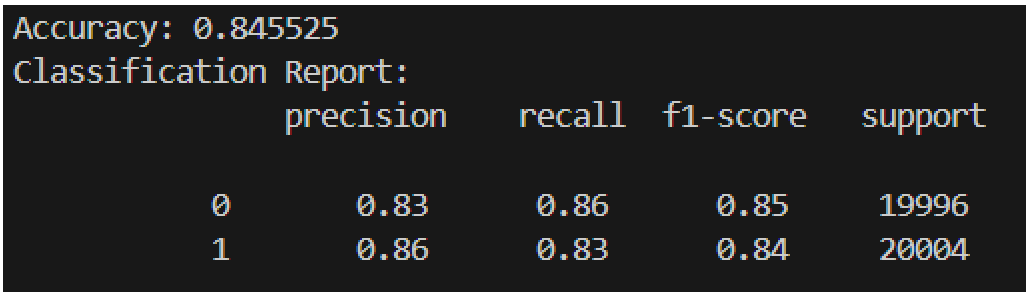
```python
model = MultinomialNB()
model.fit(X_train_vec, y_train)

y_pred = model.predict(X_test_vec)

accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

print()
print(f"Accuracy: {accuracy}")
print("Classification Report:\n", report)
print("Confusion Matrix:")
print(cm)
```

*Code 5: Model training, accuracy and evaluation driver code*

# 6. <u>Results and Analysis</u>

## 6.1 Evaluation Metrics

- Accuracy is about 85%
- Precision and Recall are both high for positive as well as negative classes.

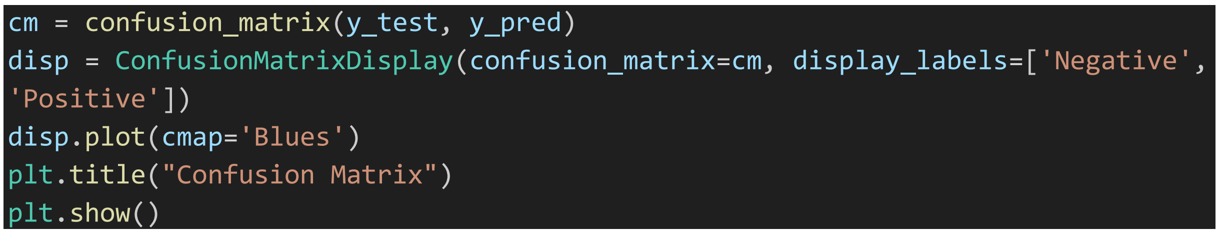


```
Accuracy: 0.845525
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.86      0.85     19996
           1       0.86      0.83      0.84     20004
```

*Fig 2: Evaluation Metrices*

## 6.2 Confusion Matrix

Visualising the results using a confusion matrix:



```
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Negative',
'Positive'])
disp.plot(cmap='Blues')
plt.title("Confusion Matrix")
plt.show()
```

*Code 6: Driver code for showing the confusion matrix*

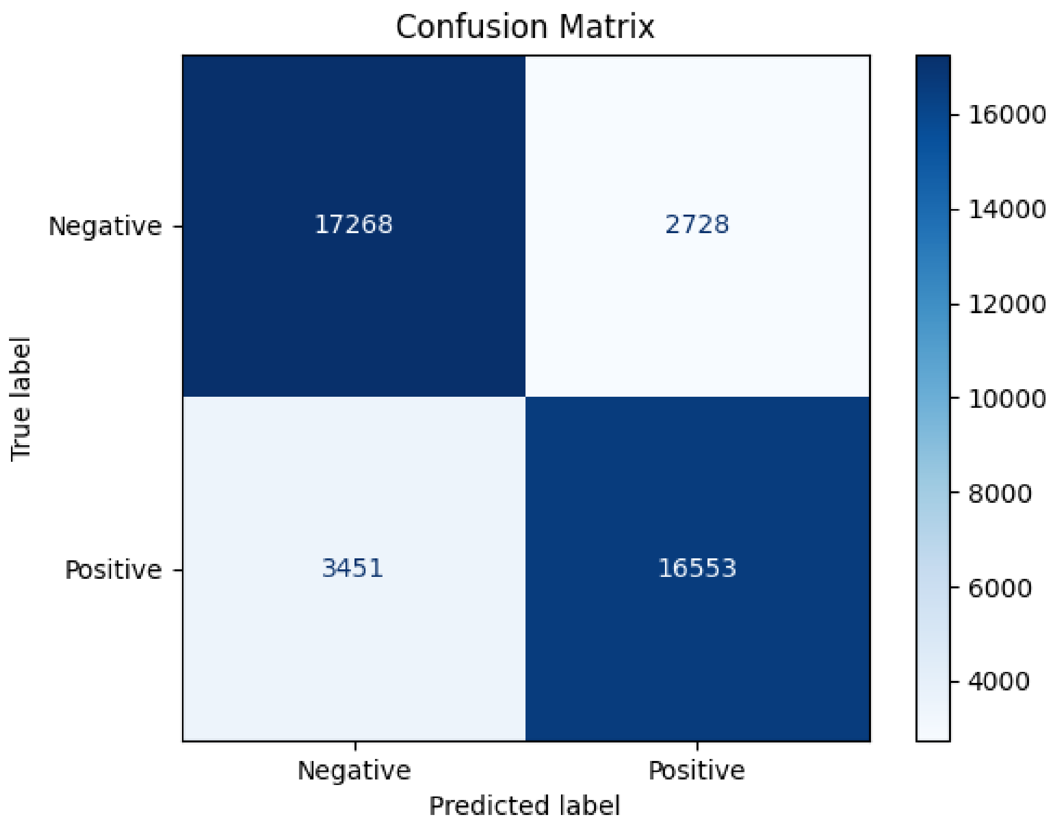


*Fig 3: Confusion Matrix*

# 7. Discussion

## 7.1 Strengths

1. High accuracy for classifying text reviews.
2. Scalable preprocessing pipeline.

## 7.2 Limitations

1. Struggles with ambiguous sentences and sarcasm.
2. Cannot generalize to languages other than English.

## 7.3 Future Improvements

1. Explore pre-trained models like BERT or GPT for contextual understanding.
2. Implement techniques to handle sarcasm and multilingual datasets.

## 8. Conclusion

This project successfully demonstrates the implementation of sentiment analysis using the IMDB dataset. The Naive Bayes classifier provides a baseline model with good accuracy and scalability. Future work could explore deep learning models like BERT for better contextual understanding.

This sentiment analysis can be moreover, scaled to customer reviews of multiple things, from applications to products. This will allow companies to have a better and more efficient feedback system without utilizing much of manpower or time.

# 9. Appendix

## 9.1 Full Code

```
'''
This script performs textual sentiment anaysis on movie revies posted by users
on IMDB.
The dataset can be found at
https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-
reviews
'''

#Module importing
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

#Creating pandas dataframe of the dataset
data = pd.read_csv('IMDB Dataset.csv')
print(data.head())

# Review Length Analysis
data['sentiment'].value_counts().plot(kind='bar', color=['green', 'red'])
plt.title('Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()

#Preprocessing data
data['review'] = data['review'].str.lower()
data['sentiment'] = data['sentiment'].map({'positive': 1, 'negative': 0})

# using train_test_split to test fit
X = data['review']
y = data['sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8,
random_state=42)

# vectorisation of data
vectorizer = CountVectorizer(stop_words='english')
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# fitting the data onto Naive Bayes Classifier
model = MultinomialNB()
```

```python
model.fit(X_train_vec, y_train)

# model prediction data
y_pred = model.predict(X_test_vec)

# accuracy metrics
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# printing accuracy scores
print()
print(f"Accuracy: {accuracy}")
print("Classification Report:\n", report)
print("Confusion Matrix:")
print(cm)

# visualising confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Negative',
'Positive'])
disp.plot(cmap='Blues')
plt.title("Confusion Matrix")
plt.show()
```

*Code 7: Full driver code*

# 10.References

## 10.1 Research Papers

1. *Shaukat, Z., Zulfiqar, A.A., Xiao, C. et al. Sentiment analysis on IMDB using lexicon and neural networks. SN Appl. Sci. 2, 148 (2020)*
2. *Pang, B., & Lee, L. (2004). "A Sentimental Education: Sentiment Analysis Using Machine Learning Techniques." Presented at the Association for Computational Linguistics (ACL).*
3. *Zhang, L., Wang, S., & Liu, B. (2018). Deep Learning for Sentiment Analysis: A Survey. Wiley Interdisciplinary Reviews.*
4. *"Sentiment Analysis Using Machine Learning and Deep Learning Models on IMDB Dataset." IEEE Xplore, 2024.*
5. *"A Literature Review on Application of Sentiment Analysis Using Machine Learning." SSRN Papers, 2023.*
6. *"Sentiment Analysis: Machine Learning Approaches Comparison." IEEE Xplore, 2024.*
7. *Tumasjan, A., et al. (2010). "Predicting Elections with Twitter: What 140 Characters Reveal About Political Sentiment." Proceedings of the International Conference on Web and Social Media (ICWSM).*
8. *"Optimization of Sentiment Analysis Using Machine Learning Classifiers." SpringerOpen, 2023.*

## 10.2 Other References

1. Kaggle IMDB Dataset: *https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews*
2. "Introduction to Machine Learning with Python" by Andreas C. Müller.
3. "Natural Language Processing with Python" by Steven Bird.
4. Scikit-learn Documentation: *https://scikit-learn.org*