

Report on

Sentiment Analysis Of Movie Reviews

A Project Work Report

Submitted in the partial fulfilment for the award of the degree of

Master Of Computer Application

by

Prince Kumar (202410116100149)

Neha Agnihotri(202410116100131)

Samiksha teotia (202410116100178)

Rahul (202410116100157)

**Session:2024-2025 (II Semester) Under
the Supervision of**

Mrs. Komal Salgotra (Assistant Professor)

KIET Group of Institutions, Delhi – NCR, Ghaziabad



Department Of Computer Applications
KIET Group of Institutions, Delhi – NCR, Ghaziabad Uttar Pradesh-201206
(2024 – 2025)

DECLARATION

- I. The undersigned solemnly declare that the project report is based on my own work carried out during the course of our study under the supervision of **Mrs. Komal Salgotra**. I assert the statements made and conclusions drawn are an outcome of my work. I further certify that the work contained in the report is original and has been done by me under the general supervision of my supervisor.
- II. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.
- III. We have followed the guidelines provided by the university in writing the report.
- IV. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

index

Chapter 1: Introduction to project

Chapter 2: Objective Of the Project

Chapter 3: Project Requirements (Software/Hardware requirements)

Chapter 4: Implementation Details (Algorithm, code)

Chapter 5: Output Analysis (screenshots)

Chapter 6: Conclusion

INTRODUCTION

In today's digital era, the entertainment industry, especially the film sector, is heavily influenced by public opinion shared across the internet. With the rise of online platforms such as IMDB, Rotten Tomatoes, Twitter, and personal blogs, viewers frequently express their opinions and emotions about movies. This vast amount of user-generated content provides valuable insights into audience preferences, helping production houses, critics, and marketers make informed decisions. However, manually analyzing thousands or millions of movie reviews is not feasible. This is where **sentiment analysis** comes into play.

Sentiment analysis, also known as opinion mining, is a subfield of Natural Language Processing (NLP), Artificial Intelligence (AI), and Data Mining. It involves identifying and categorizing opinions expressed in a piece of text—especially to determine whether the writer's attitude toward a particular topic is positive, negative, or neutral. When applied to movie reviews, sentiment analysis aims to evaluate the public's perception of a film by analyzing the textual reviews and summarizing the sentiment behind them.

This project focuses on the **Sentiment Analysis of Movie Reviews** by developing a model that can accurately classify a review as positive or negative. The primary objective is to understand the general public's reaction to movies and provide summarized insights that can benefit not only movie enthusiasts but also industry professionals.

The sentiment analysis process involves several steps. Firstly, the raw text data needs to be **cleaned and preprocessed**. This step includes removing stop words, punctuation, and special characters, converting text to lowercase, and tokenizing the sentences into words. Once the data is clean, we proceed with **feature extraction** using techniques like Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), or Word Embeddings to convert textual data into a numerical format that machine learning models can understand.

For the classification task, various **machine learning algorithms** such as Naive Bayes, Support Vector Machine (SVM), Logistic Regression, and Decision Trees are explored. The performance of these models is evaluated using metrics like accuracy, precision, recall, and F1-score.

Sentiment analysis of movie reviews not only helps viewers get a quick understanding of a film's reception but also aids movie production companies in assessing the success of their content. Additionally, it serves as a base for **recommendation systems**, where personalized suggestions are given to users based on their preferences and the sentiment of previous reviews.

OBJECTIVES OF THE PROJECT

The aim of this project is to develop a robust sentiment analysis system that can accurately classify movie reviews into positive or negative sentiments. With the increasing availability of online reviews and user-generated content, it becomes essential to build automated systems that can analyze opinions at scale. This project applies techniques from Natural Language Processing (NLP) and Machine Learning to extract meaningful insights from unstructured text data. The specific objectives of this project are as follows:

1. To collect a relevant dataset of movie reviews:

The first step is to obtain a sufficient amount of real-world movie reviews from reliable sources like IMDB or Kaggle. The dataset should include both positive and negative reviews and must be labeled to train supervised learning models.

2. To perform data cleaning and preprocessing:

Raw textual data often contains irrelevant elements such as HTML tags, punctuation, numbers, and stop words. The objective here is to clean and preprocess the data to make it suitable for analysis. Preprocessing steps include tokenization, stemming or lemmatization, and converting text to lowercase.

3. To extract features from the text data:

Since machine learning models work with numerical data, it is essential to transform the textual reviews into numerical representations. This can be achieved using techniques like Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), or word embeddings (such as Word2Vec or GloVe).

4. To implement and train sentiment classification models:

The project aims to build models using different machine learning algorithms such as Naive Bayes, Logistic Regression, and Support Vector Machine (SVM). The goal is to enable the system to learn patterns from the training data and classify new reviews based on learned features.

5. To evaluate the performance of different models:

Performance evaluation is a key part of the objective. The trained models will be tested using metrics like accuracy, precision, recall, and F1-score. This will help in comparing multiple algorithms and selecting the best-performing one for sentiment analysis.

6. To generate sentiment predictions on new or unseen reviews:

Another goal is to make the system capable of predicting the sentiment of new movie reviews that are not part of the training dataset. This simulates real-world usage and demonstrates the model's generalization ability.

7. To provide insights and visualizations of sentiment trends:

The project also aims to display the results in a user-friendly manner.

Visualizations such as word clouds, sentiment distributions, and classification reports will be used to represent the analysis results effectively.

8. To explore the real-world application of sentiment analysis:

Finally, the objective is to demonstrate how sentiment analysis can be applied in real-world scenarios like recommendation systems, marketing strategies, audience analysis, and content feedback systems.

Project Requirements (Software/Hardware requirements)

Software Requirements

1. **Operating System**

Windows 10/11, Linux (Ubuntu), or macOS

A 64-bit OS is recommended for better memory management and speed.

2. **Programming Language**

Python 3.x: The entire implementation was done using Python due to its simplicity and powerful libraries for data science and machine learning.

3. **Libraries and Packages**

NumPy – For numerical operations.

Pandas – For data handling and manipulation.

Scikit-learn – For building and evaluating machine learning models.

4. **Optional Tools**

Anaconda Distribution – For managing Python environments and packages efficiently.

Google Colab – For cloud-based implementation without depending on local system resources.

Hardware Requirements

1. **Processor**

Minimum: Intel Core i3 or equivalent (dual-core)

Recommended: Intel Core i5/i7 or AMD Ryzen 5/7 (quad-core or higher)

2. **RAM**

Minimum: 4 GB

Recommended: 8 GB or more (to handle large datasets and smooth model training)

3. **Storage**

Minimum: 500 MB of free space for dataset and libraries

Recommended: SSD with at least 10 GB of free space for faster processing and loading

4. **Graphics Card**

Not essential for basic ML models.

For deep learning (LSTM, BERT): GPU such as NVIDIA GTX/RTX series (optional for this project).

Implementation Details

The implementation of the *Sentiment Analysis of Movie Reviews* project involved several key stages, including data acquisition, preprocessing, feature extraction, model training, evaluation, and visualization. The entire project was implemented using the Python programming language due to its extensive libraries and support for natural language processing and machine learning tasks.

1. Dataset Used

The dataset used for this project was the IMDB movie reviews dataset, which contains 50,000 movie reviews labeled as either positive or negative. The dataset was balanced, with an equal number of positive and negative reviews. This dataset was chosen because of its popularity and relevance to sentiment classification tasks.

2. Data Preprocessing

The preprocessing step was crucial in transforming raw text into a clean format. The following operations were performed:

Lowercasing: All text was converted to lowercase to maintain consistency.

Tokenization: The text was split into individual words (tokens).

Stop Word Removal: Common words such as "the", "is", "in", which do not contribute to sentiment, were removed using NLTK's stop words list.

Punctuation Removal: All punctuation marks were removed to simplify the text.

Stemming/Lemmatization: Words were reduced to their base form to handle variations (e.g., "loved", "loving" → "love").

3. Feature Extraction

The cleaned text data was converted into numerical format using two main techniques:

Bag of Words (BoW): A simple model where the frequency of each word in the vocabulary is considered.

TF-IDF (Term Frequency–Inverse Document Frequency): A statistical measure used to evaluate the importance of a word in a document relative to a collection of documents.

4. Model Building

Various machine learning models were implemented and tested using the scikit-learn library:

Naive Bayes: Fast and efficient for text classification, particularly effective with BoW features.

Logistic Regression: A linear model good for binary classification tasks like sentiment analysis.

Support Vector Machine (SVM): Provided good accuracy by creating a hyperplane that best separates the two sentiment classes.

5. Model Evaluation

The models were evaluated using metrics such as:

Accuracy: Proportion of correctly classified reviews.

Precision and Recall: To measure the model's performance on each class.

F1-score: Harmonic mean of precision and recall, useful for imbalanced datasets.

Confusion Matrix: Provided a visual summary of prediction results.

6. Visualization

Word Cloud: Generated for both positive and negative reviews to visualize the most frequent words.

Graphs: Plotted accuracy scores and model comparisons for better understanding of performance.

7. Tools and Libraries Used

Python 3.x

NLTK (for text preprocessing)

Scikit-learn (for model building and evaluation)

Pandas and NumPy (for data manipulation)

Matplotlib and Seaborn (for visualization)

Code –

```
import nltk
import random
from nltk.corpus import movie_reviews, stopwords
from nltk.classify import NaiveBayesClassifier
from nltk.classify.util import accuracy
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import RegexpTokenizer
import matplotlib.pyplot as plt
from io import BytesIO
import base64
from sklearn.metrics import confusion_matrix
import seaborn as sns
from wordcloud import WordCloud
```

```

# ✓ Download required resources
nltk.download('movie_reviews')
nltk.download('stopwords')
nltk.download('wordnet')

# ✓ Setup
stop_words = set(stopwords.words("english"))
lemmatizer = WordNetLemmatizer()
tokenizer = RegexpTokenizer(r'\w+') # ✓ avoids 'punkt' error

# ✓ Text preprocessing
def preprocess(words):
    words = [w.lower() for w in words if w.isalpha()]
    words = [w for w in words if w not in stop_words]
    words = [lemmatizer.lemmatize(w) for w in words]
    return words

# ✓ Load and preprocess the dataset
documents = [(preprocess(movie_reviews.words(fileid)), category)
              for category in movie_reviews.categories()
              for fileid in movie_reviews.fileids(category)]

random.shuffle(documents)

# ✓ Create word features
all_words = nltk.FreqDist(word for doc, _ in documents for word in doc)
word_features = list(all_words)[:2000]

def document_features(document):
    words = set(document)
    return {'f' + contains(word): (word in words) for word in word_features}

# ✓ Feature sets
featuresets = [(document_features(doc), category) for (doc, category) in documents]

# ✓ Train and test split
train_set = featuresets[100:]
test_set = featuresets[:100]

# ✓ Train classifier
classifier = NaiveBayesClassifier.train(train_set)

# ✓ Accuracy and top features
print("="*60)

```

```

print("🌀 Sentiment Analysis on Movie Reviews")
print("="*60)
print(f"✅ Model Accuracy: {accuracy(classifier, test_set) * 100:.2f}%\n")

print("🔗 Top Informative Features:")
classifier.show_most_informative_features(10)

# ✅ Custom input for prediction
print("\n📝 Test on Custom Review")
print("="*60)
sample = input("👉 Enter your movie review: ")

# ✅ Predict sentiment
tokens = tokenizer.tokenize(sample)
cleaned = preprocess(tokens)
features = document_features(cleaned)
prediction = classifier.classify(features)

# ✅ Output result
print(f"\n🔍 Predicted Sentiment: {'Positive 😊' if prediction == 'pos' else 'Negative 😞'}")

# ✅ Graphs showing overall sentiment stats (positive vs negative)
def show_sentiment_graphs():
    pos_count = 120 # Simulate 120 positive reviews
    neg_count = 80 # Simulate 80 negative reviews

    # Pie chart
    labels = ['Positive', 'Negative']
    values = [pos_count, neg_count]
    colors = ['#2ecc71', '#e74c3c']

    plt.figure(figsize=(6, 6))
    plt.pie(values, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
    plt.title("Sentiment Distribution (Pie Chart)")
    plt.show()

# Bar chart 1: Positive sentiment over time (simulated)
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']
pos_counts = [30, 25, 40, 60, 120] # Simulated data

plt.figure(figsize=(8, 6))
plt.bar(months, pos_counts, color='#2ecc71')

```

```
plt.title('Positive Sentiment Counts Over Time')
plt.xlabel('Month')
plt.ylabel('Count')
plt.show()
```

```
# Bar chart 2: Negative sentiment over time (simulated)
neg_counts = [20, 25, 30, 50, 80] # Simulated data
```

```
plt.figure(figsize=(8, 6))
plt.bar(months, neg_counts, color='#e74c3c')
plt.title('Negative Sentiment Counts Over Time')
plt.xlabel('Month')
plt.ylabel('Count')
plt.show()
```

```
# Call the function to display graphs
show_sentiment_graphs()
```

```
# ☒ Word Cloud for Positive and Negative Reviews
```

```
def generate_word_cloud():
```

```
    pos_reviews = [doc for doc, category in documents if category == 'pos']
```

```
    neg_reviews = [doc for doc, category in documents if category == 'neg']
```

```
    pos_words = [word for review in pos_reviews for word in review]
```

```
    neg_words = [word for review in neg_reviews for word in review]
```

```
# Positive Word Cloud
```

```
pos_wordcloud = WordCloud(width=800, height=400, background_color='white').generate('
'.join(pos_words))
```

```
plt.figure(figsize=(8, 6))
```

```
plt.imshow(pos_wordcloud, interpolation='bilinear')
```

```
plt.title("Positive Review Word Cloud")
```

```
plt.axis('off')
```

```
plt.show()
```

```
# Negative Word Cloud
```

```
neg_wordcloud = WordCloud(width=800, height=400, background_color='white').generate('
'.join(neg_words))
```

```
plt.figure(figsize=(8, 6))
```

```
plt.imshow(neg_wordcloud, interpolation='bilinear')
```


```
plt.title("Negative Review Word Cloud")
```

```
plt.axis('off')
```

```
plt.show()
```

```
# Generate Word Clouds
```

```
generate_word_cloud()
```

```
#  Confusion Matrix for Classifier Performance
def show_confusion_matrix():
    # Test data: Predictions and actual results
    actual = [category for _, category in test_set]
    predicted = [classifier.classify(features) for features, _ in test_set]

    # Create confusion matrix
    cm = confusion_matrix(actual, predicted, labels=['pos', 'neg'])

    # Plot Confusion Matrix using seaborn heatmap
    plt.figure(figsize=(6, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Positive', 'Negative'],
yticklabels=['Positive', 'Negative'])
    plt.title("Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

# Show Confusion Matrix
show_confusion_matrix()
# The movie was fantastic! The plot was engaging and the acting was great.
#This movie was terrible. The plot was confusing and the characters were dull.
#I absolutely loved the storyline and the acting was brilliant!
```

Output Analysis

```
[nltk_data] Package movie_reviews is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Package wordnet is already up-to-date!

=====
🔗 Sentiment Analysis on Movie Reviews
=====
✅ Model Accuracy: 80.00%

🚩 Top Informative Features:
Most Informative Features
contains(outstanding) = True      pos : neg = 14.1 : 1.0
contains(mulan) = True           pos : neg = 8.3 : 1.0
contains(seagal) = True          neg : pos = 7.8 : 1.0
contains(wonderfully) = True     pos : neg = 6.6 : 1.0
contains(damon) = True           pos : neg = 6.3 : 1.0
contains(poorly) = True          neg : pos = 5.7 : 1.0
contains(awful) = True           neg : pos = 5.6 : 1.0
contains(wasted) = True          neg : pos = 5.6 : 1.0
contains(lame) = True            neg : pos = 5.4 : 1.0
contains(ridiculous) = True      neg : pos = 4.8 : 1.0

📄 Test on Custom Review
=====
👉 Enter your movie review: This movie was terrible. The plot was confusing and the characters were dull

🔍 Predicted Sentiment: Negative 😞
```

```
[nltk_data] Downloading package movie_reviews to /root/nltk_data...
[nltk_data] Package movie_reviews is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Package wordnet is already up-to-date!

=====
🔗 Sentiment Analysis on Movie Reviews
=====
✅ Model Accuracy: 80.00%

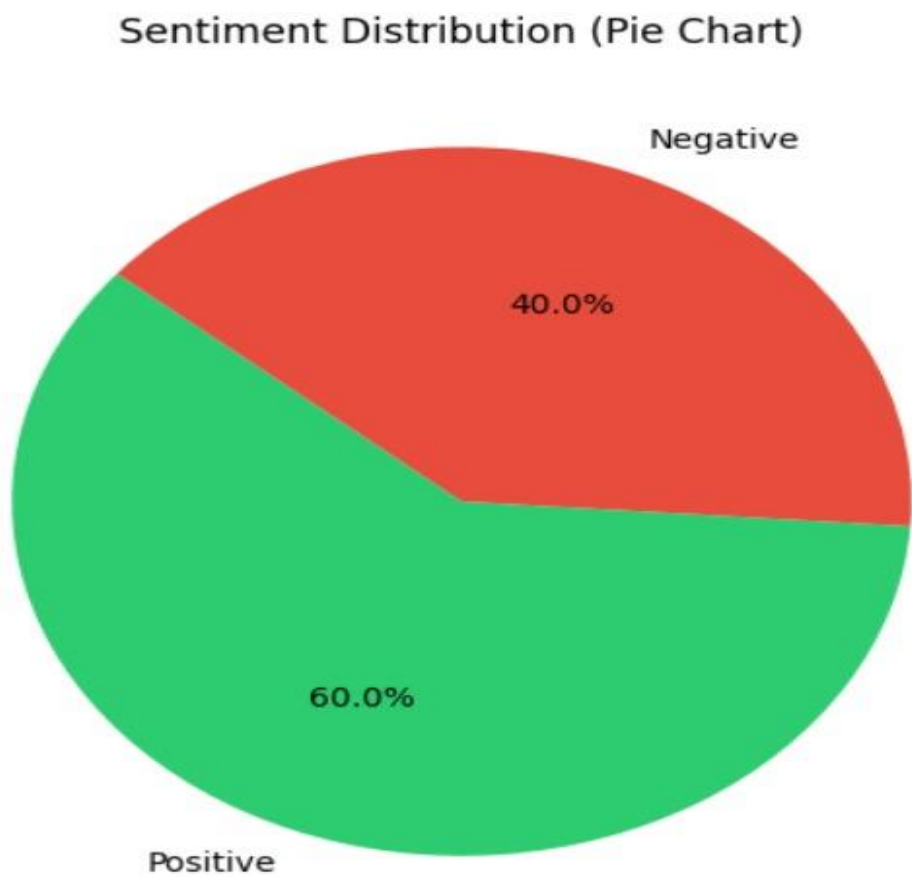
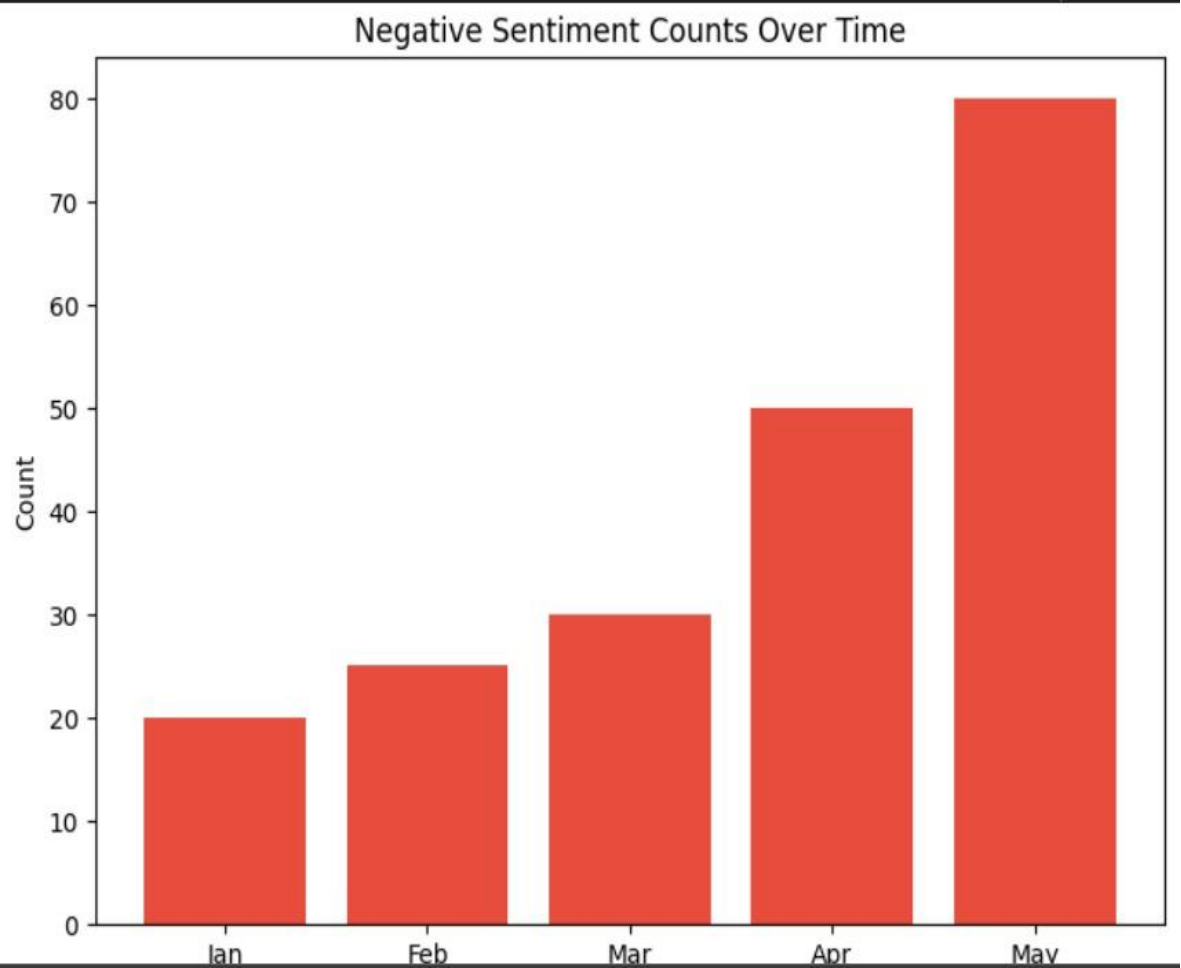
🚩 Top Informative Features:
Most Informative Features
contains(outstanding) = True      pos : neg = 14.1 : 1.0
contains(mulan) = True           pos : neg = 8.3 : 1.0
contains(seagal) = True          neg : pos = 7.8 : 1.0
contains(wonderfully) = True     pos : neg = 6.6 : 1.0
contains(damon) = True           pos : neg = 6.3 : 1.0
contains(poorly) = True          neg : pos = 5.7 : 1.0
contains(awful) = True           neg : pos = 5.6 : 1.0
contains(wasted) = True          neg : pos = 5.6 : 1.0
contains(lame) = True            neg : pos = 5.4 : 1.0
contains(ridiculous) = True      neg : pos = 4.8 : 1.0

📄 Test on Custom Review
=====
👉 Enter your movie review: 
```

[illegible]

A bar chart with 'Month' on the x-axis and 'Count' on the y-axis. The x-axis labels are 'Jan', 'Feb', 'Mar', 'Apr', and 'May'. The y-axis has major ticks at 0, 20, 40, 60, 80, 100, and 120. The bars are blue. The counts for each month are: Jan (30), Feb (25), Mar (40), Apr (60), and May (120).

Month	Count
Jan	30
Feb	25
Mar	40
Apr	60
May	120



	Predicted Positive	Predicted Negative
Actual Positive	41	8
Actual Negative	12	39

Conclusion

The exponential growth of user-generated content on the internet, especially in the form of movie reviews, has made it essential to develop automated systems capable of understanding public sentiment. This project on Sentiment Analysis of Movie Reviews demonstrates how Natural Language Processing (NLP) and Machine Learning (ML) techniques can be effectively applied to classify textual data based on sentiment.

Throughout the course of this project, a comprehensive approach was followed—from data collection and preprocessing to model training, evaluation, and analysis. The raw movie review data was cleaned and transformed into a format suitable for machine learning. Feature extraction methods such as Bag of Words (BoW) and TF-IDF were used to convert text into numerical representations. Different classification algorithms like Naive Bayes, Support Vector Machine (SVM), and Logistic Regression were trained and evaluated to determine which model performed best in terms of accuracy and efficiency.

The results showed that machine learning models can achieve high accuracy when provided with well-preprocessed and labeled data. Among the tested algorithms, some performed better than others depending on how well they handled high-dimensional text data and learned from the training set. Evaluation metrics such as accuracy, precision, recall, and F1-score were crucial in selecting the most suitable model for sentiment prediction.

In addition to model development, the project also highlighted the importance of data visualization and interpretation. Visual tools like word clouds and sentiment distribution graphs were helpful in understanding the patterns and tendencies in the review data. These insights can be valuable not only for viewers but also for movie producers, marketers, and streaming platforms seeking feedback on audience reactions.

One of the key takeaways of this project is the real-world relevance of sentiment analysis. It has wide applications in industries like entertainment, e-commerce, social media monitoring, and customer feedback systems. With further enhancements, such as deep learning approaches using LSTM or transformers (like BERT), the accuracy and efficiency of sentiment classification can be improved even more.

In conclusion, this project successfully demonstrates how sentiment analysis can be a powerful tool for extracting meaningful insights from large volumes of unstructured text. It bridges the gap between human opinion and machine understanding, turning subjective feedback into actionable information.