Report on

SENTIMENT ANALYSIS ON MOVIE REVIEWS

By

Khushi Bora – 202410116100099

Monika Tyagi – 202410116100123

Himanshi Sharma – 202410116100091

Harshita chauhan – 202410116100090

Session:2024-2026 (II Semester)

Under the supervision of

MR.APPORAV JAIN (Assistant Professor)

DEPARTMENT OF COMPUTER APPLICATIONS

KIET GROUP OF INSTITUTIONS,DELHI-NCR
GHAZIABAD- 201206

INTRODUCTION

With the explosion of user-generated content on the internet, sentiment analysis has become a powerful tool for understanding public opinion. Sentiment Analysis, also known as opinion mining, refers to the use of natural language processing, text analysis, and computational linguistics to identify and extract subjective information from textual data.

This project focuses on performing sentiment analysis on movie reviews using Python and machine learning techniques. Movie reviews are chosen due to their availability, relevance, and rich emotional content. By categorizing these reviews into positive, negative, or neutral sentiments, we can derive meaningful insights into audience opinions, which can be valuable for filmmakers, marketers, and streaming platforms.

The project demonstrates how sentiment analysis can be implemented in Python using libraries such as NLTK, Scikit-learn, and pandas. It also introduces concepts of text preprocessing, feature extraction (TF-IDF, Bag of Words), and classification algorithms like Naive Bayes, Logistic Regression, and Support Vector Machine (SVM).

OBJECTIVES OF THE PROJECT

The primary objectives of this project are:-

1. Understand the Basics of Sentiment Analysis

- Dive deep into the theoretical background of sentiment analysis, including polarity (positive, negative, neutral) and subjectivity.
- Explore real-world applications such as product reviews, social media monitoring, and public opinion tracking.
- Study the challenges involved, such as sarcasm detection, context interpretation, and handling negations.

2. Collect and Preprocess Movie Review Data

- Identify reliable sources of labeled movie reviews (e.g., IMDb, Rotten Tomatoes, Kaggle).
- Convert raw review data into a format suitable for machine learning by cleaning text (removing HTML tags, special characters), normalizing (lowercasing), and correcting inconsistencies.
- Apply natural language processing (NLP) techniques such as tokenization, stopword removal, stemming, and lemmatization to improve model readiness.

3. Feature Engineering and Vectorization

- Transform textual data into numerical format using feature extraction methods such as:
- Bag of Words (BoW): Represent each review as a frequency vector of known words.

- TF-IDF: Weigh the importance of terms by considering how common or rare they are across documents.
- Analyze the impact of using n-grams (bigrams/trigrams) to capture context and improve model performance.

4. Implement Machine Learning Models

- Train multiple classifiers (e.g., Naive Bayes, Logistic Regression, SVM) to determine which is best suited for text classification.
- Tune hyperparameters using grid search or cross-validation to optimize model behavior.
- Apply cross-validation to avoid overfitting and ensure generalization to unseen reviews. **5.**

Evaluate Model Performance

- Use classification metrics to assess the quality of predictions:
- Accuracy: Overall correctness of the model.
- Precision: Proportion of predicted positives that are actually positive.
- Recall: Proportion of actual positives correctly identified.
- F1-Score: Harmonic mean of precision and recall.
- Generate confusion matrices to visually analyze misclassifications and understand areas of improvement.

6. Visualize Results

- Create data visualizations that help interpret both the input and model output:
- Word clouds to display frequent terms in positive and negative reviews.
- Sentiment distribution charts to observe dataset balance.
- Heatmaps of confusion matrices for easier interpretation of evaluation results.

METHODOLOGY

The methodology for this sentiment analysis project is structured into clear, logical steps that ensure thorough processing of data from raw input to insightful results. The entire process is implemented using Python and prominent libraries such as NLTK, Scikit-learn, pandas, and matplotlib/seaborn for visualization.

Step 1: Data Collection

- **Source of Data:-**
  The dataset is typically sourced from public repositories like IMDb (Internet Movie Database), Kaggle (e.g., IMDb Large Movie Review Dataset), or Rotten Tomatoes.
- **Structure of the Dataset:-**
  It contains thousands of movie reviews, each labeled with a sentiment (positive or negative). Some datasets also provide neutral or mixed labels.
- **Data Format:-**
  Reviews are usually stored in .csv or .tsv format, with columns like Review (text) and Sentiment (label).

Step 2: Data Preprocessing

Preprocessing is crucial to clean and standardize the text data for machine learning models.

- **Text Cleaning:-**
  - Remove HTML tags using regex or BeautifulSoup.
  - Strip punctuation, numbers, and special characters.
  - Convert all text to lowercase for uniformity.
- **Tokenization:-**
  - Break down each review into individual words or tokens using NLTK's word_tokenize() or similar tools.
- **Stopword Removal:-**
  - Remove common words like "is," "the," "and," that do not contribute to sentiment using NLTK’s stopword corpus.
- **Stemming/Lemmatization:-**
  - Stemming reduces words to their base form (e.g., "liked" → "like") using PorterStemmer.
  - Lemmatization uses context-aware reduction (e.g., "better" → "good") via WordNetLemmatizer.

Step 3: Feature Extraction

Text must be converted into a numerical format before feeding into machine learning models.

- **Bag of Words (BoW):**

    - Creates a matrix of word counts for each review. Each word becomes a feature, and its frequency in a review is the value.
- **TF-IDF (Term Frequency–Inverse Document Frequency):**
    - Weighs words by their frequency in a specific review relative to their occurrence across all reviews. It helps emphasize more important, review-specific words.
- **N-Grams (optional):**
    - Considers combinations of words (bigrams/trigrams) for capturing context (e.g., "not good" as a negative phrase).
- **Vectorizers:**
    - Use CountVectorizer for BoW and TfidfVectorizer for TF-IDF implementation in Scikit-learn.

Step 4: Train-Test Split

 Splitting the Dataset:
    - Typically, 80% of the data is used for training and 20% for testing using train_test_split() from Scikit-learn.
    - Ensures the model is evaluated on unseen data.

Step 5: Model Building

Various machine learning models are used to classify the sentiment of reviews:

- **Naive Bayes Classifier (MultinomialNB):**
    - Works well with high-dimensional text data and assumes independence between features.
- **Logistic Regression:**
    - A probabilistic model that predicts the likelihood of a review being positive or negative.
- **Support Vector Machine (SVM):**
    - Finds the optimal hyperplane to separate classes and performs well on sparse, high-dimensional data like text.
- **Model Selection:**
    - Models are chosen based on initial test performance and further refined using hyperparameter tuning.

Step 6: Model Evaluation

To assess the performance of the models:

- **Metrics Used:**
    - Accuracy: Proportion of total correct predictions.
    - Precision: Correct positive predictions out of total predicted positives.
    - Recall: Correct positive predictions out of actual positives.

    - F1-Score: Harmonic mean of precision and recall for balanced evaluation.
    - Confusion Matrix: A 2x2 matrix that shows true positives, true negatives, false positives, and false negatives.
- **Cross-validation (optional):**
    - Further validation using techniques like k-fold cross-validation to ensure consistency.

Step 7: Visualization

Helps in understanding both the dataset and model results visually:

- **Word Clouds:**
    - Shows frequently used words in positive and negative reviews separately.
- **Bar Graphs:**
    - Display the distribution of sentiments and feature importance.
- **Heatmaps:**
    - Represent confusion matrix for better interpretation.

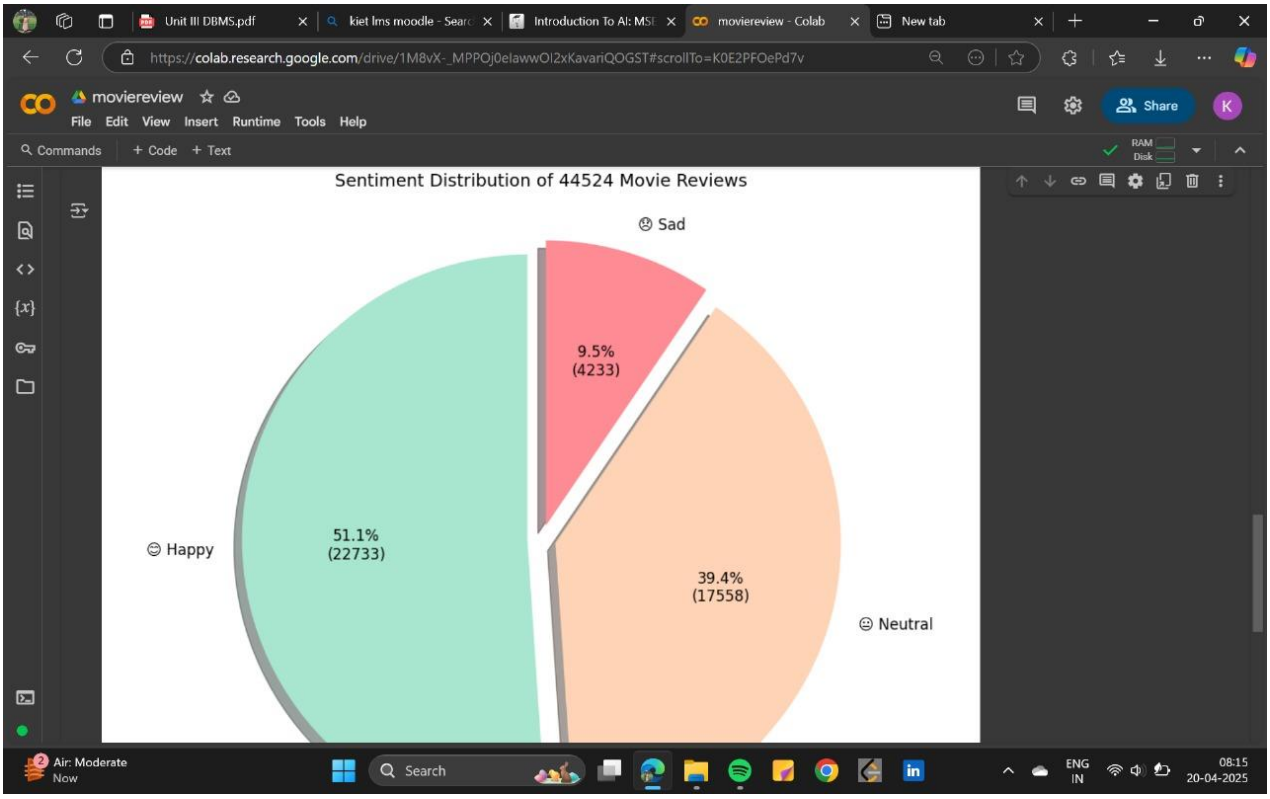
CODE & OUTPUT

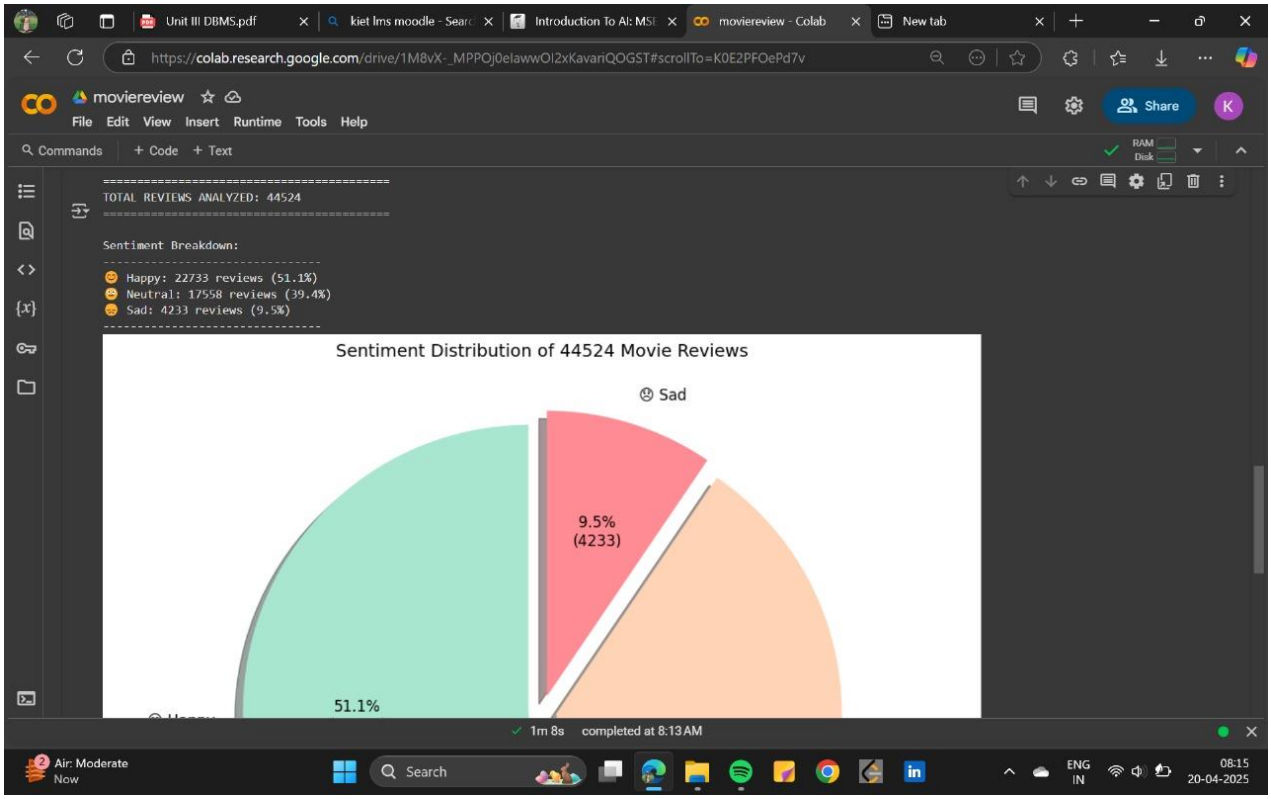
```
    import pandas as pd
from textblob import TextBlob
import matplotlib.pyplot as plt

# Function to classify review sentiment
def classify_sentiment(review):
   analysis = TextBlob(str(review))
   polarity = analysis.sentiment.polarity

   if polarity > 0.1:
      return '       Happy'
   elif polarity < -0.1:
      return '     Sad'
   else:
      return '       Neutral'

csv_path = 'IMDB Dataset.csv'

# Load the dataset
df = pd.read_csv(csv_path, on_bad_lines='skip', encoding='utf-8')

# Ensure the review column is named correctly
if 'review' not in df.columns:
   text_columns = [col for col in df.columns if 'text' in col.lower() or 'review' in col.lower()
or 'comment' in col.lower()]
   df.rename(columns={text_columns[0]: 'review'}, inplace=True)

# Apply sentiment analysis
df['sentiment'] = df['review'].apply(classify_sentiment)
total_reviews = len(df)

# Count sentiment results
sentiment_counts = df['sentiment'].value_counts()
percentages = (sentiment_counts / total_reviews) * 100

# Prepare result summary
results_df = pd.DataFrame({
   'Sentiment': ['       Happy', '       Neutral', '     Sad'],
   'Count': [
```

```
        sentiment_counts.get('       Happy', 0),
        sentiment_counts.get('       Neutral', 0),
        sentiment_counts.get('     Sad', 0)
    ],
2
    'Percentage': [
        round(percentages.get('       Happy', 0), 1),
        round(percentages.get('       Neutral', 0), 1),
        round(percentages.get('     Sad', 0), 1)
    ]
})

# Print sentiment breakdown
print("=========================================")
print(f"TOTAL REVIEWS ANALYZED: {total_reviews}")
print("=========================================")
print("\nSentiment Breakdown:")
print("---------------------------------")
for _, row in results_df.iterrows():
    print(f"{row['Sentiment']}: {row['Count']} reviews ({row['Percentage']}%)")
print("---------------------------------")

# Pie chart
plt.figure(figsize=(10, 8))
colors = ['#a8e6cf', '#ffd3b6', '#ff8b94']
explode = (0.05, 0.05, 0.05)

def make_autopct(values):
    def my_autopct(pct):
        total = sum(values)
        val = int(round(pct * total / 100.0))
        return f'{pct:.1f}%\n({val})'
    return my_autopct

plt.pie(
    results_df['Count'],
    labels=results_df['Sentiment'],
    colors=colors,
    explode=explode,
    autopct=make_autopct(results_df['Count']),
    startangle=90,
    shadow=True,
    textprops={'fontsize': 12}
```

)

```
plt.axis('equal')
plt.title(f'Sentiment Distribution of {total_reviews} Movie Reviews', pad=20, fontsize=14)
plt.tight_layout()
plt.show()
```




Unit III DBMS.pdf
kiet lms moodle - Searc
Introduction To AI: MSE
moviereview - Colab
New tab
https://colab.research.google.com/drive/1M8vX-_MPPOj0elawwOI2xKavariQOGST#scrollTo=K0E2PFOePd7v
moviereview
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text
RAM
Disk
==========
TOTAL REVIEWS ANALYZED: 44524
==========
Sentiment Breakdown:
Happy: 22733 reviews (51.1%)
Neutral: 17558 reviews (39.4%)
Sad: 4233 reviews (9.5%)
Sentiment Distribution of 44524 Movie Reviews
Sad
9.5%
(4233)
51.1%
1m 8s completed at 8:13 AM
Air: Moderate
Now
Search
ENG
IN
08:15
20-04-2025

CONCLUSION

The Iris flower classification problem demonstrates the power of machine learning in species identification. Various classification techniques were explored and evaluated, with Random Forest and Support Vector Machine emerging as top-performing models. The project highlights the importance of feature selection, model training, and evaluation in predictive analytics.

Future work may include integrating additional plant species, leveraging neural networks for better classification, and developing a user-friendly interface for real-time predictions. This project serves as an excellent introduction to classification problems and showcases the effectiveness of different machine learning models in solving real-world problems.