Part A: IMDb Sentiment Classification

Introduction

In the age of digital media, user-generated reviews significantly influence the public perception of movies. Platforms like IMDb collect thousands of reviews from users, expressing both positive and negative sentiments. These reviews hold valuable insights for producers, studios, and streaming platforms seeking to understand audience preferences.

However, manually analyzing such a large volume of textual data is impractical. This makes **Natural Language Processing (NLP)** an essential tool to automate and streamline sentiment analysis. Through this project, we use machine learning and NLP techniques to classify IMDb reviews into two categories: **positive** or **negative**, based on the text content.

Objective

The main objectives of this project are:

1. **To build an end-to-end sentiment classification model** that can analyze and classify movie reviews from the IMDb dataset.
2. **To apply NLP techniques** such as tokenization, stopword removal, stemming, and lemmatization to clean the raw review data.
3. **To extract features** using methods like **Bag-of-Words (BoW)**, **TF-IDF**, and **Word Embeddings** (Word2Vec).
4. **To train and compare multiple machine learning models** like Logistic Regression, Naive Bayes, and Support Vector Machines (SVM).
5. **To evaluate the model's performance** using standard classification metrics such as accuracy, precision, recall, F1-score, and confusion matrix.
6. **To visualize insights** derived from the data such as frequent words, word clouds, and feature importance, and communicate the findings effectively

Task 1: Data Exploration and Preprocessing

In this task, we performed a detailed exploration of the IMDb dataset, which includes movie reviews labeled as positive or negative. We began by checking for missing values, outliers, and class imbalances. After confirming class distribution was slightly skewed, we applied text cleaning steps such as removing HTML tags, lowercasing, removing punctuation, and eliminating stopwords. Tokenization was used to split the text into words, followed by stemming and lemmatization to normalize words.

Insights:
- Reviews had informal language and required thorough cleaning.
- Tokenization helped in identifying and standardizing words.
- Lemmatization preserved word meanings better than stemming.

What we learned:
- Preprocessing is vital to convert raw, noisy text into structured input for modeling.

Task 2: Feature Engineering

After preprocessing, we transformed text data into numerical features using Bag-of-Words (BoW) and TF-IDF vectorization. We also applied Word2Vec to capture semantic similarities between words. Additionally, basic textual features like word count and average word length were extracted.

Insights:
- TF-IDF effectively emphasized meaningful terms in a review.
- Word2Vec captured contextual relationships but needed more computational power.

What we learned:
- Feature engineering plays a central role in improving model input and performance.

Task 3: Model Development

We trained several classification models including Logistic Regression, Naive Bayes, and Support Vector Machines (SVM). Models were trained using features derived from TF-IDF and Word2Vec. Cross-validation was used for robust performance evaluation.

Insights:
- SVM performed the best with TF-IDF features.
- Naive Bayes worked fast but had slightly lower precision.

What we learned:
- Different models react differently to feature types and dataset structures.

Task 4: Model Evaluation

We evaluated the models using metrics like accuracy, precision, recall, and F1-score. The confusion matrix provided a visual comparison of true vs. predicted labels.

Insights:
- SVM achieved the highest F1-score.

Some misclassifications occurred due to vague reviews.
What we learned:
- Evaluating models with multiple metrics gives a comprehensive view of performance.

We Learned in This Project

- **Importance of Preprocessing**
    - Text data is unstructured and noisy. Cleaning, tokenization, lemmatization, and removing stopwords are essential to prepare meaningful input.
- **Feature Extraction Techniques**
    - Learned how **Bag-of-Words** and **TF-IDF** capture term frequency and relevance, and how **Word2Vec** brings semantic understanding to models.
- **Model Building and Comparison**
    - Gained hands-on experience with text classification using **Logistic Regression, Naive Bayes, and SVM**, and saw how different models behave with different features.
- **Model Evaluation Metrics**
    - Understood how to use **Accuracy, Precision, Recall, F1-score**, and **Confusion Matrix** to evaluate model performance fairly.
- **Pipeline Building**
    - Learned to create a full pipeline from raw data to prediction, including vectorization, training, and evaluation

Pros:

- **Well-defined Binary Problem:** Only two classes (positive/negative) make it easier to model and evaluate.
- **Clean Dataset:** IMDb dataset is structured and widely used, enabling reproducibility.
- **Text Length Uniformity:** Helps feature extraction methods like TF-IDF and Word2Vec.
- **Performance Boost with SVM:** Support Vector Machine gave strong results with sparse text data.

- **Preprocessing Insights:** Lemmatization and stopword removal improved text clarity.

Cons:

- **Subjectivity in Reviews:** Ambiguous sentiments can confuse models.
- **Imbalanced Phrases:** Neutral or sarcastic reviews are hard to classify.
- **Computational Cost:** Word embeddings like Word2Vec are resource-intensive.
- **Overfitting Risk:** Complex models can overfit on small datasets if not tuned properly