**Project Report**

**Title:** Sentiment Analysis of Movie Reviews

**Group Members:**

* *Sijal Fatima (L1F21BSDS0030)*
* *Kainat Ijaz (L1F21BSDS0038)*

**1. Introduction:**

Sentiment analysis is a subfield of natural language processing (NLP) which aims to classify text data based on the sentiment it conveys, such as positive, negative, or neutral. The objective of this project is to develop a model capable of accurately predicting the sentiment of text data. This project’s scope includes preprocessing text data, selecting appropriate modeling techniques, and evaluating their performance to derive meaningful insights.

**2. Dataset Overview:**

• **Source of the Dataset:** The dataset used in this project is sourced from Kaggle (https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews).

• **Key Features and Target Variables:**

1. **Text Data:** The main feature, containing user reviews or feedback.
2. **Sentiment Labels:** The target variable categorizes each text as Positive, Negative, or Neutral.

**3. Methodology:**

The following steps outline the methodology employed for sentiment analysis:

* **Data Preprocessing:**
  + - **Reading the Dataset:** The Pandas library (pd.read\_csv) is used to import the IMDB Dataset into a DataFrame.
    - **Text Lowercasing:** All review text is converted to lowercase using df['review'] = df['review'].str.lower() for case-insensitive processing.
    - **HTML Tag Removal:** A function remove\_html\_tag is implemented using regular expressions to eliminate HTML tags from the reviews.
    - **Punctuation Removal:** The remove\_pun function utilizes string.punctuation to remove punctuation marks.
* **Stemming:** The Porter Stemmer (PorterStemmer) is used to reduce words to their root forms (e.g., "running" becomes "run"). However, consider using Lemmatization (WordNetLemmatizer) for potentially better results, preserving the word's meaning (e.g., "running" stays "running").
* **Sentiment Label Encoding:** The sentiment labels ("positive" and "negative") are converted into numerical values (1 for positive, 0 for negative) using df.sentiment.replace.
* **TF-IDF Vectorization:** The TfidfVectorizer from scikit-learn is employed to create TF-IDF vectors from the preprocessed reviews. TF-IDF considers both the word frequency (TF) and inverse document frequency (IDF) to weigh the importance of words in a document.
* **Model Training and Evaluation:**
  1. **Train-Test Split:** The data is divided into training and testing sets using train\_test\_split from scikit-learn. The training set is used to train the models, and the testing set is used to evaluate their performance.
  2. **Multinomial Naive Bayes (MNB):** A Multinomial Naive Bayes classifier (MultinomialNB) is trained on the TF-IDF features. It is a probabilistic classifier that assumes independence between features.
  3. **Logistic Regression (LR):** A Logistic Regression model (LogisticRegression) is also trained for comparison. Logistic regression is a linear model that predicts the probability of a binary outcome (positive or negative sentiment in this case).
  4. **Linear SVC:** A Linear Support Vector Classifier (LinearSVC) is trained with a regularization parameter (C) of 0.5 and a random state of 42 for comparison. This model uses support vectors to create a hyperplane that separates the positive and negative classes.
  5. **Evaluation Metrics:** The performance of each model is evaluated using the following metrics:
     + **Confusion Matrix:** This matrix visualizes the number of correct and incorrect predictions for each class.
     + **Classification Report:** This report provides precision, recall, F1-score, and support for each class. These metrics provide insights into how well the models are classifying positive and negative reviews.

**4. Code Implementation:**

 **Imports:** Imports necessary libraries for data manipulation, text processing, machine learning, and evaluation.

 **Data Loading and Preprocessing:** Loads the IMDB dataset using pandas and applies the preprocessing steps: lowercasing, HTML tag removal, punctuation removal, combined text cleaning, and stemming.

 **Sentiment Encoding:** Converts the string labels ("positive", "negative") to numerical labels (1, 0).

 **TF-IDF Vectorization:** Creates TF-IDF vectors from the preprocessed text data. vect.fit\_transform(X) both fits the vectorizer to the training data (learns the vocabulary and IDF values) and transforms the data into TF-IDF vectors.

 **Train-Test Split:** Splits the data into training and testing sets to evaluate the model's performance on unseen data. random\_state ensures consistent splitting for reproducibility.

 **Model Training and Evaluation:**

* For each model (Multinomial Naive Bayes, Logistic Regression, Linear SVC):
  1. Initializes the model.
  2. Trains the model using fit(x\_train, y\_train).
  3. Makes predictions on the test set using predict(x\_test).
  4. Prints the confusion matrix and classification report to evaluate performance.
* **Challenges Faced:**
  1. Stemming Function Error
  2. Logistic Regression Convergence

**5. Results and Analysis:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Precision (0)** | **Recall (0)** | **F1-Score (0)** | **Precision (1)** | **Recall (1)** | **F1-Score (1)** | **Accuracy** |
| Multinomial Naive Bayes | 0.86 | 0.87 | 0.86 | 0.87 | 0.86 | 0.86 | 0.86 |
| Logistic Regression | 0.90 | 0.88 | 0.89 | 0.88 | 0.91 | 0.90 | 0.89 |
| Linear SVC | 0.91 | 0.89 | 0.90 | 0.89 | 0.91 | 0.90 | 0.90 |

**Analysis:**

* **Linear SVC:** The Linear SVC model achieved the highest accuracy (0.90) and generally better precision, recall, and F1-scores for both positive and negative sentiment compared to the other models. This suggests that the hyperplane created by the SVC is effectively separating the two classes in the TF-IDF feature space.
* **Logistic Regression:** Logistic Regression performed very closely to Linear SVC, with an accuracy of 0.89. It also shows a good balance between precision and recall.
* **Multinomial Naive Bayes:** Multinomial Naive Bayes had the lowest performance among the three, with an accuracy of 0.86. While it's computationally efficient, its assumption of feature independence might not hold true for text data, where word occurrences are often correlated.

**Insights and Alignment with Expectations:**

It was expected that linear models like Logistic Regression and Linear SVC would perform well on this text classification task, as they are known to be effective with high-dimensional data like TF-IDF vectors. The results align with this expectation, with both models achieving high accuracy. Multinomial Naive Bayes, while simpler, is often a good baseline but was outperformed by the other two.

**6. Conclusion:**

This report demonstrated the application of sentiment analysis on the IMDB Dataset using three machine learning algorithms: Multinomial Naive Bayes, Logistic Regression, and Linear SVC. The Linear SVC model achieved the best performance, followed closely by Logistic Regression, indicating their suitability for this text classification task.

**Future Work and Improvements:**

* **Lemmatization:** Instead of stemming, using lemmatization could potentially improve performance by preserving word meanings.
* **Hyperparameter Tuning:** Optimizing the hyperparameters of each model (e.g., the C parameter for Linear SVC, regularization parameters for Logistic Regression) using techniques like grid search or cross-validation could lead to further improvements.
* **Ensemble Methods:** Exploring ensemble methods like Random Forest or Gradient Boosting could potentially combine the strengths of different models and achieve higher accuracy.
* **Deep Learning Models:** Investigating deep learning models like Recurrent Neural Networks (RNNs) or Transformers could capture more complex patterns in the text data and potentially yield better results.