Lab Assignment No. 4- NLP Preprocessing And Text Classification

Deep Learning

Team members:

Ayush Bhalerao (202302090015)

Om Vitole (202302090016)

Bhavesh Jadhav (202302090021)

Batch: T4

Problem Statement -

The objective of this assignment is to implement NLP preprocessing techniques and build a text classification model using machine learning techniques.

Learning Outcomes:

1. Understand and apply NLP preprocessing techniques such as tokenization, stopword removal, stemming, and lemmatization.
2. Implement text vectorization techniques such as TF-IDF and CountVectorizer.
3. Develop a text classification model using a machine learning algorithm.
4. Evaluate the performance of the model using suitable metrics.

Google Colab Link -

https://colab.research.google.com/drive/1zX34gBsEGVB890RQWFxGCMlaivf5uMDH#scrollTo=qfwaeuSgcupl

Github Link https://github.com/omvitole7/Lab_4_NLP

NLP Lab_Assignmnent_4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

Files

Analyze your files with code written by Gemini Upload

- bin
- boot
- content
  - .config
  - .ipynb_checkpoints
  - ag_news
    - test.csv
    - train.csv
  - drive
  - sample_data
  - archive (2).zip
- datalab
- dev
- etc
- home
- kaggle
- lib
- lib32
- lib64

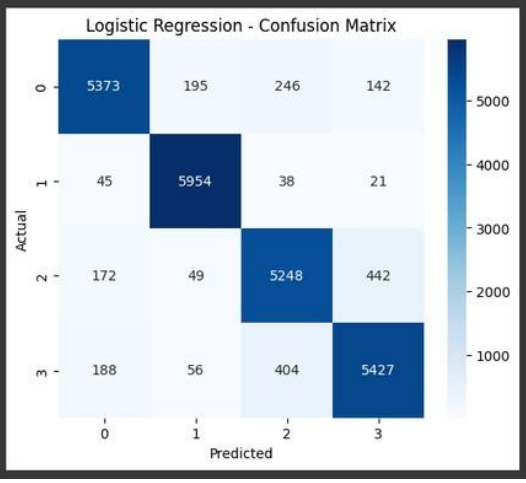
Disk 70.71 GB available

62

Logistic Regression
F1 Score: 0.9167
Accuracy: 0.9167 (91.67%)

Classification Report:

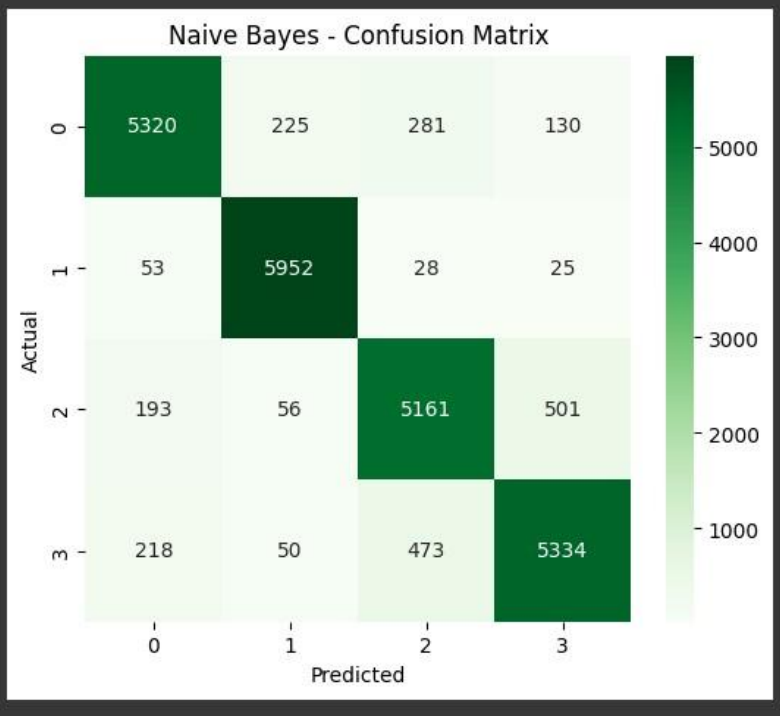
	precision	recall	f1-score	support
1	0.93	0.90	0.92	5956
2	0.95	0.98	0.97	6058
3	0.88	0.89	0.89	5911
4	0.90	0.89	0.90	6075
accuracy			0.92	24000
macro avg	0.92	0.92	0.92	24000
weighted avg	0.92	0.92	0.92	24000

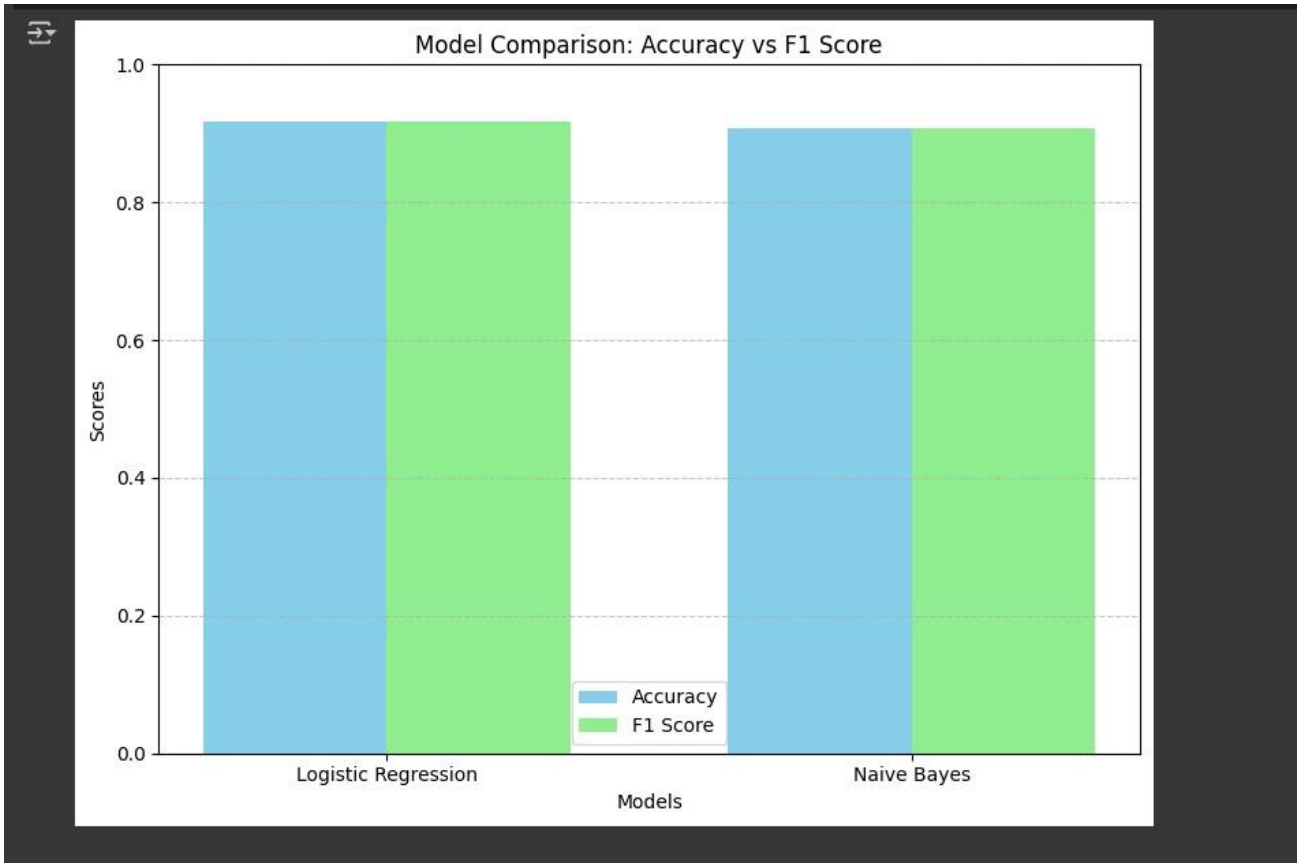


Naive Bayes
F1 Score: 0.9070
Accuracy: 0.9070 (90.70%)

Classification Report:

	precision	recall	f1-score	support
1	0.92	0.89	0.91	5956
2	0.95	0.98	0.96	6058
3	0.87	0.87	0.87	5911
4	0.89	0.88	0.88	6075
accuracy			0.91	24000
macro avg	0.91	0.91	0.91	24000
weighted avg	0.91	0.91	0.91	24000





Objective

The goal of this assignment is to preprocess text data using NLP techniques and build classification models (Logistic Regression and Naive Bayes) to categorize news articles into predefined classes. This involves understanding and implementing the entire text classification pipeline: data loading, cleaning, transformation, modeling, and evaluation.

Step-by-Step Theoretical Explanation

1. Dataset Loading and Exploration

- The AG News dataset is a collection of news articles categorized into 4 classes: World, Sports, Business, and Science/Technology.
- The dataset is loaded using Pandas into DataFrames.
- Title and Description columns are combined into a single text column for processing.
- Basic dataset inspection is performed: checking shapes, null values, and class distribution. 2.

NLP Preprocessing

a. Text Normalization

- The text is converted to lowercase to avoid treating the same word in different cases as different tokens (e.g., "Apple" vs "apple").

b. Tokenization

- Each sentence is split into words (tokens) using nltk.word_tokenize(). This converts raw text into a list of individual words.

c. Stopword Removal

- Commonly used words that don't carry much meaning (like "the", "is", "in") are removed using NLTK's stopwords list.

d. Stemming

- Words are reduced to their root forms using PorterStemmer (e.g., "running" becomes "run"). This helps group similar terms.

e. Lemmatization

- Words are converted to their dictionary base form using WordNetLemmatizer (e.g., "better" becomes "good"). Lemmatization is more accurate linguistically than stemming.

3. Text Vectorization

Since ML models can't understand raw text, the data is converted to numeric form using two popular methods:

a. CountVectorizer

- Converts the collection of text into a matrix of token counts.

b. TF-IDF Vectorizer

- Converts text into a matrix of term frequency–inverse document frequency values. It downweights commonly used words and boosts rare but informative ones.

4. Splitting the Data

- The transformed data is split into training and testing sets (e.g., 80% training, 20% testing) to evaluate how well the model generalizes to unseen data.

5. Model Building

Two models are trained using scikit-learn:

☑ Logistic Regression:

- A linear model used for classification problems. It models the probability of class membership.

☑ Multinomial Naive Bayes:

- A probabilistic classifier well-suited for text classification tasks. It uses word frequencies to predict class probabilities.

6. Model Evaluation

- Predictions are generated on the test set using both models.
- Evaluation Metrics used:
  - Accuracy – Overall correctness of the model.
  - F1 Score – Harmonic mean of precision and recall, useful for imbalanced datasets.
  - Classification Report – Includes precision, recall, and F1 score per class.
  - Confusion Matrix – A visual representation of correct vs incorrect predictions.
  - Bar Graph – Used to visually compare the F1 Score and Accuracy of both models.

📊 Summary of Findings

- Both models were evaluated and compared side-by-side.
- Visual tools like bar plots and confusion matrices were used to illustrate the results.
- This helped determine which model performed better on the AG News classification task.