CSL7640: Natural Language Understanding
Assignment 1
Sports vs Politics Text Classification

Namya Dhingra
Roll No: B23CS1040

1 Introduction

The objective of this assignment is to design and implement a machine learning classifier that reads a text document and classifies it into one of two categories: **Sports** or **Politics**. The system must utilize standard NLP feature representations such as Bag of Words, TF-IDF, or n-grams, and compare at least three machine learning models.

This report presents the complete pipeline including:

- Data collection
- URL filtering
- Web scraping
- Dataset construction
- Text preprocessing (implemented from scratch)
- Feature extraction (manual Bag of Words)
- Model training and evaluation

2 Overall System Architecture

The complete workflow of the system is illustrated below:

1. Raw query dataset: `india_news.csv`
2. Query filtering → `filter_queries.py`
3. URL dataset → `url_indian_news.csv`
4. Web scraping → `scrape_articles.py`
5. Full text dataset → `indian_news_with_text.csv`
6. Structural cleaning → `load_data.py`
7. Manual NLP preprocessing → `preprocess.py`
8. Feature extraction + model training → `train_models.py`

3 Datasets Used

3.1 India News Dataset

The file `india_news.csv` contains:

- Search queries
- Associated news URLs

Using rule-based filtering, only relevant Sports and Politics queries were retained. After filtering, 255 URLs were extracted (see Screenshot 1).

3.2 BBC Text Dataset

The file `bbc-text.csv` was obtained from an online public dataset repository (BBC News dataset, commonly available on Kaggle). It contains 994 news articles categorized into multiple classes including Sports and Politics.

This dataset was used to supplement the training data and ensure better diversity.

4 Query Filtering

The script `filter_queries.py` performs substring-based category detection. Instead of exact matching, substring matching was used for robustness.

Example:

- Queries containing "politic" or "election" → Politics
- Queries containing "cricket" or "sport" → Sports

This resulted in:

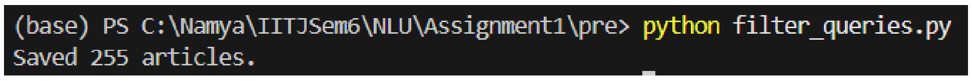
255 filtered URLs

5 Web Scraping

The script `scrape_articles.py` performs:

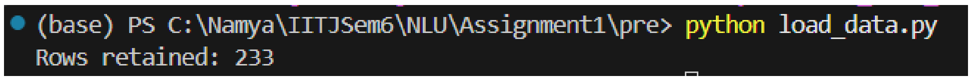
1. HTTP request using `requests`
2. HTML parsing using `BeautifulSoup`
3. Extraction of text inside `<p>` tags
4. Filtering articles shorter than 200 characters
5. Writing category, URL, and extracted text to CSV

A 1-second delay was introduced between requests to ensure ethical scraping practices. This produced: `indian_news_with_text.csv`



```
(base) PS C:\Namya\IITJSem6\NLU\Assignment1\pre> python filter_queries.py
Saved 255 articles.
```

Figure 1: Output of filter_queries.py



```
(base) PS C:\Namya\IITJSem6\NLU\Assignment1\pre> python load_data.py
Rows retained: 233
```

Figure 2: Output of scrape_articles.py

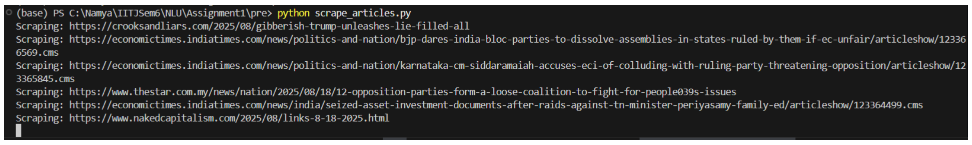
6 Data Loading and Structural Cleaning

The file `load_data.py` performs:

- Loading CSV using pandas
- Keeping only required columns: category and text
- Removing rows with missing values

After cleaning:

Total Samples: 233



```
(base) PS C:\Namya\IITJSem6\NLU\Assignment1\pre> python scrape_articles.py
Scraping: https://crooksandliars.com/2025/08/gibberish-trump-unleashes-lie-filled-all
Scraping: https://economictimes.indiatimes.com/news/politics-and-nation/bjp-dares-india-bloc-parties-to-dissolve-assemblies-in-states-ruled-by-them-if-ec-unfair/articleshow/12336
6569.cms
Scraping: https://economictimes.indiatimes.com/news/politics-and-nation/karnataka-cm-siddaramaiah-accuses-eci-of-colluding-with-ruling-party-threatening-opposition/articleshow/12
3365845.cms
Scraping: https://www.thestar.com.my/news/nation/2025/08/18/12-opposition-parties-form-a-loose-coalition-to-fight-for-people039s-issues
Scraping: https://economictimes.indiatimes.com/news/india/seized-asset-investment-documents-after-raids-against-tn-minister-periyasamy-family-ed/articleshow/123364499.cms
Scraping: https://www.nakedcapitalism.com/2025/08/links-8-18-2025.html
```

Figure 3: Output of load_data.py

Class distribution:

- Politics: 152
- Sports: 81

7 Manual NLP Preprocessing (From Scratch)

Tokenization, stopword removal, and Bag of Words were implemented manually without using NLTK tokenizers or sklearn vectorizers.

7.1 Tokenization

Implemented using regular expressions:

- Convert text to lowercase
- Extract alphabetic sequences using regex

7.2 Stopword Removal

A manually defined stopword list was used. Words shorter than 3 characters were removed.

7.3 Bag of Words Construction

The following steps were implemented:

1. Build vocabulary mapping: word $\rightarrow$ index
2. For each document:
    - Count word frequencies
    - Construct vector of vocabulary size

Vocabulary size:

16,940 unique words

8 Train-Test Split

A stratified 80-20 split was used:

- Training samples: 186
- Testing samples: 47

Stratification ensures class proportions are maintained.

9 Models Used

The following models from `sklearn` were used:

9.1 Multinomial Naive Bayes

Assumes conditional independence of features.

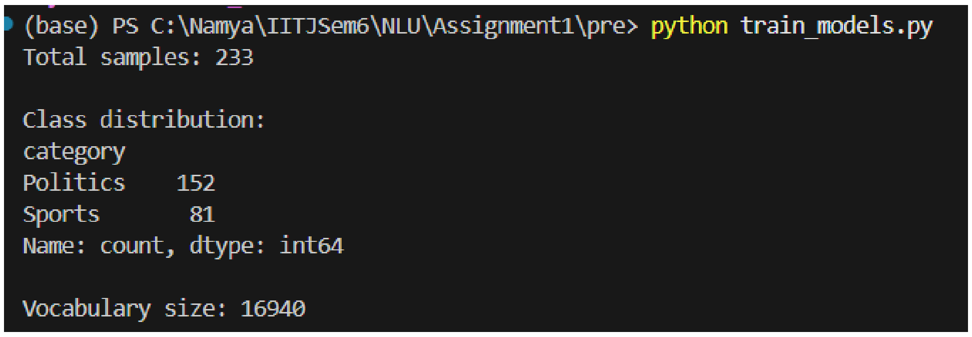
9.2 Logistic Regression

Linear classifier using log-loss optimization. Class imbalance handled using `class_weight="balanced"`.

9.3 Linear Support Vector Machine

Maximizes margin between classes. Also used with balanced class weights.

10 Results



```
(base) PS C:\Namya\IITJSem6\NLU\Assignment1\pre> python train_models.py
Total samples: 233

Class distribution:
category
Politics    152
Sports       81
Name: count, dtype: int64

Vocabulary size: 16940
```

Figure 4: Final Output - 1

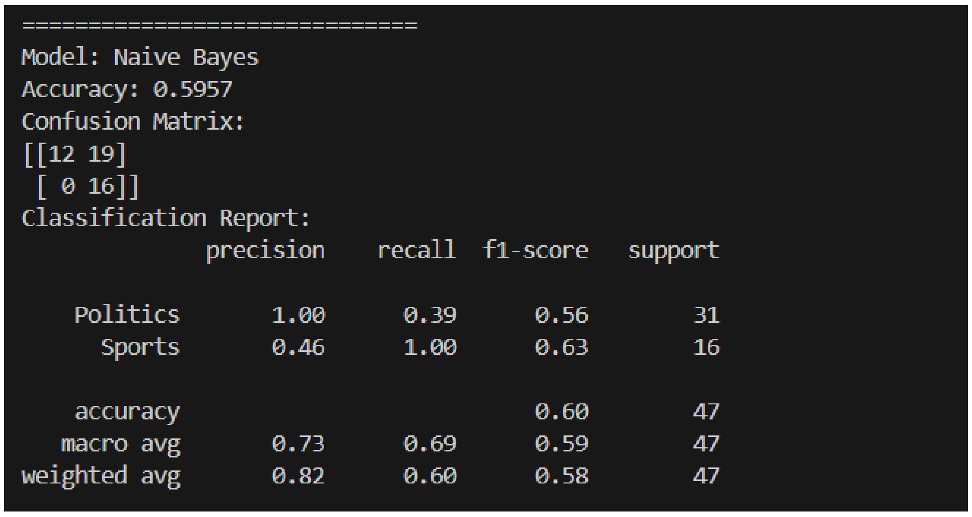
10.1 Naive Bayes

Accuracy: 59.57%

Confusion Matrix:

$$\begin{bmatrix} 12 & 19 \\ 0 & 16 \end{bmatrix}$$

NB performed poorly due to strong independence assumptions and high vocabulary sparsity.



```
==============================
Model: Naive Bayes
Accuracy: 0.5957
Confusion Matrix:
[[12 19]
 [ 0 16]]
Classification Report:
              precision    recall  f1-score   support

    Politics       1.00      0.39      0.56        31
      Sports       0.46      1.00      0.63        16

    accuracy                           0.60        47
   macro avg       0.73      0.69      0.59        47
weighted avg       0.82      0.60      0.58        47
```

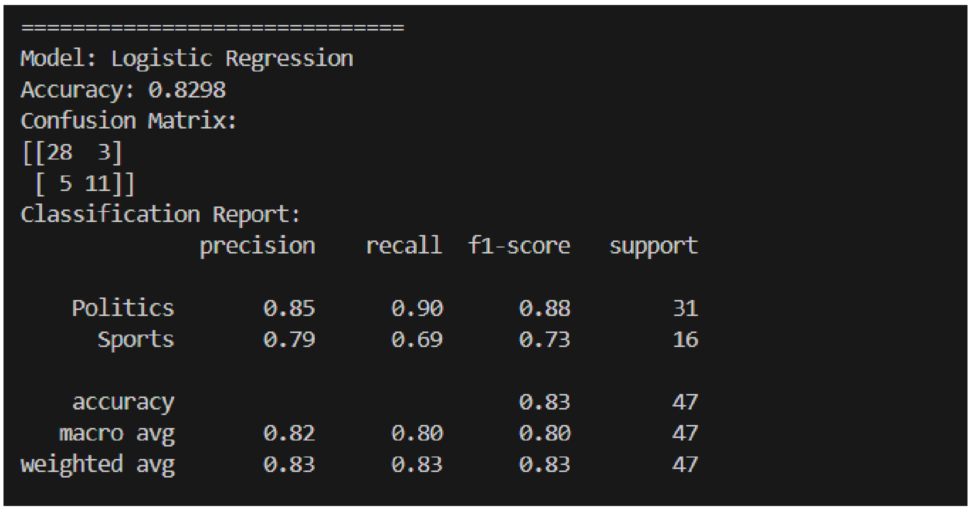
Figure 5: Final Output - Naive Bayes

10.2 Logistic Regression

Accuracy: 82.98% Confusion Matrix:

$$\begin{bmatrix} 28 & 3 \\ 5 & 11 \end{bmatrix}$$

This model achieved the best overall performance with balanced precision and recall.



```
==============================
Model: Logistic Regression
Accuracy: 0.8298
Confusion Matrix:
[[28  3]
 [ 5 11]]
Classification Report:
              precision    recall  f1-score   support

    Politics       0.85      0.90      0.88        31
      Sports       0.79      0.69      0.73        16

    accuracy                           0.83        47
   macro avg       0.82      0.80      0.80        47
weighted avg       0.83      0.83      0.83        47
```

Figure 6: Final Output - Logistic Regression

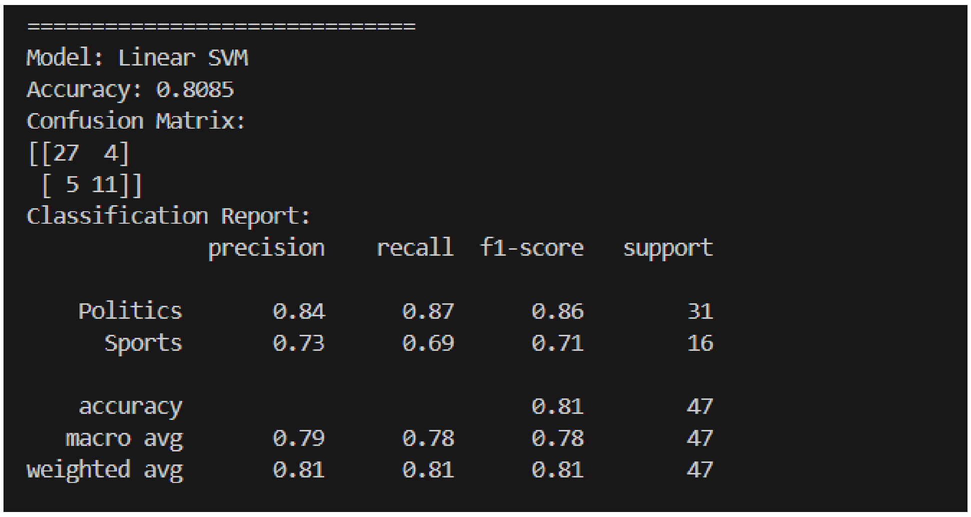
10.3 Linear SVM

Accuracy: 80.85%

Confusion Matrix:

$$\begin{bmatrix} 27 & 4 \\ 5 & 11 \end{bmatrix}$$

SVM performed comparably to Logistic Regression but slightly lower.



```
==============================
Model: Linear SVM
Accuracy: 0.8085
Confusion Matrix:
[[27  4]
 [ 5 11]]
Classification Report:
              precision    recall  f1-score   support

    Politics       0.84      0.87      0.86        31
      Sports       0.73      0.69      0.71        16

    accuracy                           0.81        47
   macro avg       0.79      0.78      0.78        47
weighted avg       0.81      0.81      0.81        47
```

Figure 7: Final Output - Linear SVM

11 Model Comparison

Model	Accuracy	Macro F1
Naive Bayes	0.60	0.59
Logistic Regression	**0.83**	**0.80**
Linear SVM	0.81	0.78

11.1 Analysis of Results

From the experimental results, Logistic Regression achieved the highest overall accuracy of 82.98%, followed closely by Linear SVM at 80.85%. Naive Bayes performed significantly worse, achieving only 59.57% accuracy.

The confusion matrices reveal that Naive Bayes struggled to correctly classify Politics articles, misclassifying many of them as Sports. This behavior can be attributed to the independence assumption of Naive Bayes and the high dimensional sparse feature space created by the manually constructed Bag-of-Words representation.

Logistic Regression performed best due to its robustness in handling high-dimensional sparse data and the use of class weighting to address dataset imbalance. The macro F1-score of 0.80 indicates balanced performance across both classes.

Although Linear SVM performed competitively, it showed slightly lower recall for the Sports class compared to Logistic Regression.

Overall, linear models demonstrated superior performance over probabilistic Naive Bayes for this task.

Logistic Regression achieved the highest overall performance.

12 Limitations

- Small dataset size (233 samples)
- Moderate class imbalance
- Large vocabulary leading to sparsity
- No stemming or lemmatization
- Web scraping may include noise
- Model may overfit to India-specific vocabulary

13 Conclusion

A complete NLP classification pipeline was developed from data acquisition to evaluation. Manual implementation of tokenization and Bag of Words demonstrated foundational understanding of NLP mechanics. Among the evaluated models, Logistic Regression achieved the best performance with an accuracy of 82.98%.

The system successfully classifies news articles into Sports and Politics categories while demonstrating both practical and theoretical understanding of NLP workflows.

14 GitHub Repository

The complete source code, datasets, and implementation details for this project are available at:

`https://github.com/namyadhingra/NLP_Classifies-SportsVsPolitics`

The repository contains:

- `filter_queries.py`
- `scrape_articles.py`
- `build_dataset.py`
- `load_data.py`
- `preprocess.py`
- `train_models.py`
- All intermediate CSV files
- This report