Sports vs Politics Text Classification

Roll Number: M25CSE031

1. Introduction

Text classification is one of the most fundamental tasks in Natural Language Processing (NLP). It involves automatically assigning predefined categories to text documents based on their content. This problem has wide applications in spam detection, sentiment analysis, topic detection, news categorization, and recommendation systems.

In this project, the objective is to design a machine learning classifier that can read a news document and classify it into one of two categories: **SPORT** or **POLITICS**. Although these two domains may appear very different, real-world news articles often contain overlapping vocabulary such as “team,” “campaign,” or “debate,” which makes classification non-trivial.

The goal of this project is to:

- Collect a suitable dataset containing sports and politics documents.
- Apply different feature representation techniques such as Bag of Words, TF-IDF, and n-grams.
- Train and compare at least three machine learning algorithms.
- Evaluate the models using quantitative metrics.
- Analyze performance differences and limitations.

2. Dataset Collection and Description

2.1 Dataset Source

For this project, the 20 Newsgroups dataset was used. This is a well-known benchmark dataset available through the scikit-learn library. It contains thousands of documents categorized into 20 different newsgroups.

From this dataset, only relevant categories were selected:

- rec.sport.baseball
- rec.sport.hockey
- talk.politics.misc
- talk.politics.guns
- talk.politics.mideast

The sport-related categories were labeled as SPORT, and the politics-related categories were labeled as POLITICS.

2.2 Dataset Size and Distribution

The dataset contains approximately several thousand documents. After filtering, the total dataset size was around 3000–4000 documents (exact number depends on subset used).

The dataset was then split into:

- 70% Training Data
- 30% Testing Data

This ensures that the model is evaluated on unseen documents.

2.3 Preprocessing Steps

Before feeding the documents into machine learning models, the following preprocessing steps were applied:

1. Removal of headers, footers, and quoted text.
2. Lowercasing all text.
3. Removal of English stop words.
4. Conversion of text into numerical vectors using feature extraction techniques.

3. Feature Engineering Techniques

Feature engineering converts text into numerical form so that machine learning algorithms can process it.

Three different feature representations were used in this project.

3.1 Bag of Words (BoW)

Bag of Words is the simplest text representation technique. It converts documents into vectors where:

- Each unique word represents one feature.
- The value represents how many times that word appears in the document.

Mathematically:

$$Vector(d) = [count(w_1), count(w_2), \ldots, count(w_n)]$$

Advantages:

- Simple and easy to implement.
- Works well for topic classification.

Limitations:

- Ignores word importance.
- Does not consider word order.

3.2 TF-IDF (Term Frequency – Inverse Document Frequency)

TF-IDF improves upon Bag of Words by reducing the weight of common words and increasing the importance of rare but meaningful words.

$$TF - IDF = TF \times IDF$$

Where:

$$IDF = \log\left(\frac{N}{df}\right)$$

TF-IDF helps emphasize words that are important in distinguishing between SPORT and POLITICS articles.

Advantages:

- Reduces influence of common words.
- Often performs better than BoW.

3.3 n-grams (Unigrams + Bigrams)

In addition to single words (unigrams), bigrams (pairs of consecutive words) were used.

Example:

- “prime minister”
- “world cup”

n-grams help capture context and word combinations that are domain-specific.

For example:

- “world cup” → SPORT
- “prime minister” → POLITICS

This improves model performance significantly.

4. Machine Learning Techniques Used

Three machine learning algorithms were implemented and compared.

4.1 Multinomial Naive Bayes

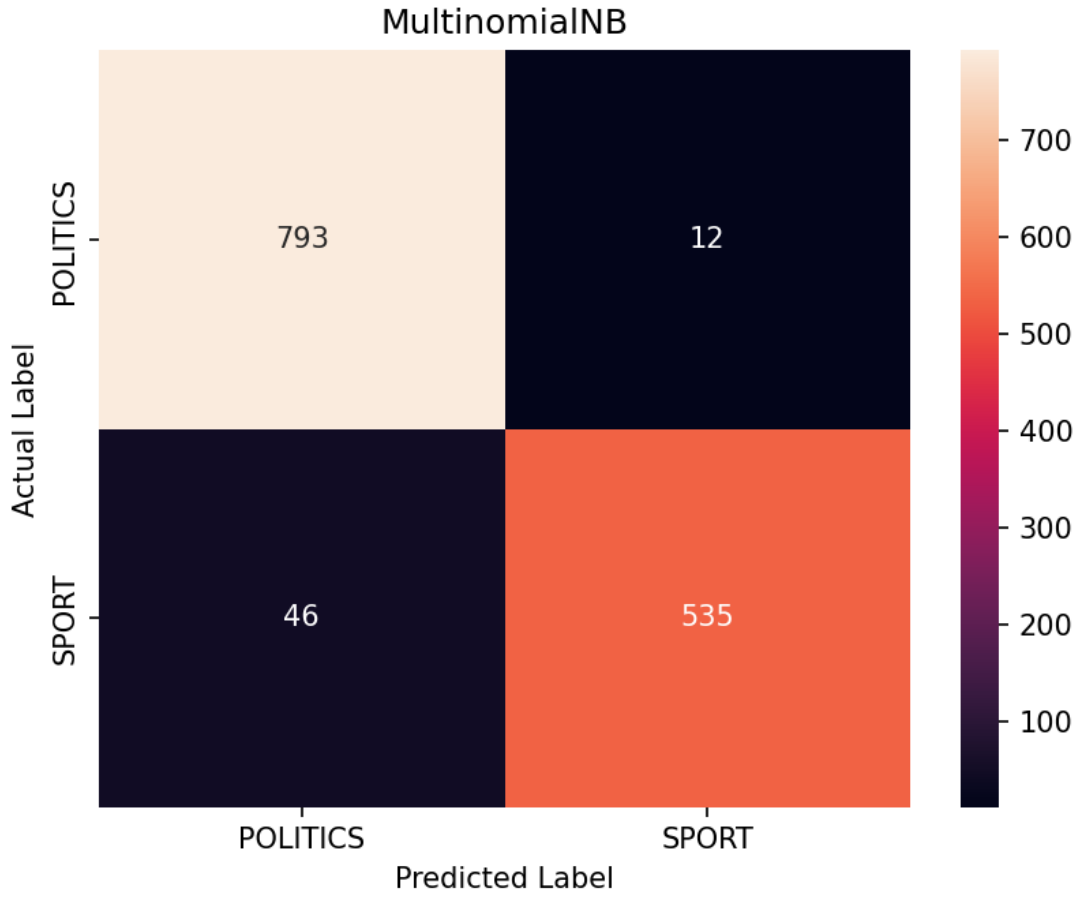
Naive Bayes is a probabilistic classifier based on Bayes’ Theorem:

$$P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)}$$

It assumes that words are conditionally independent given the class.

Advantages:

- Very fast.
- Performs well for text classification.
- Works well with high-dimensional data.



Limitations:

- Independence assumption is unrealistic.
- May underperform compared to more advanced models.

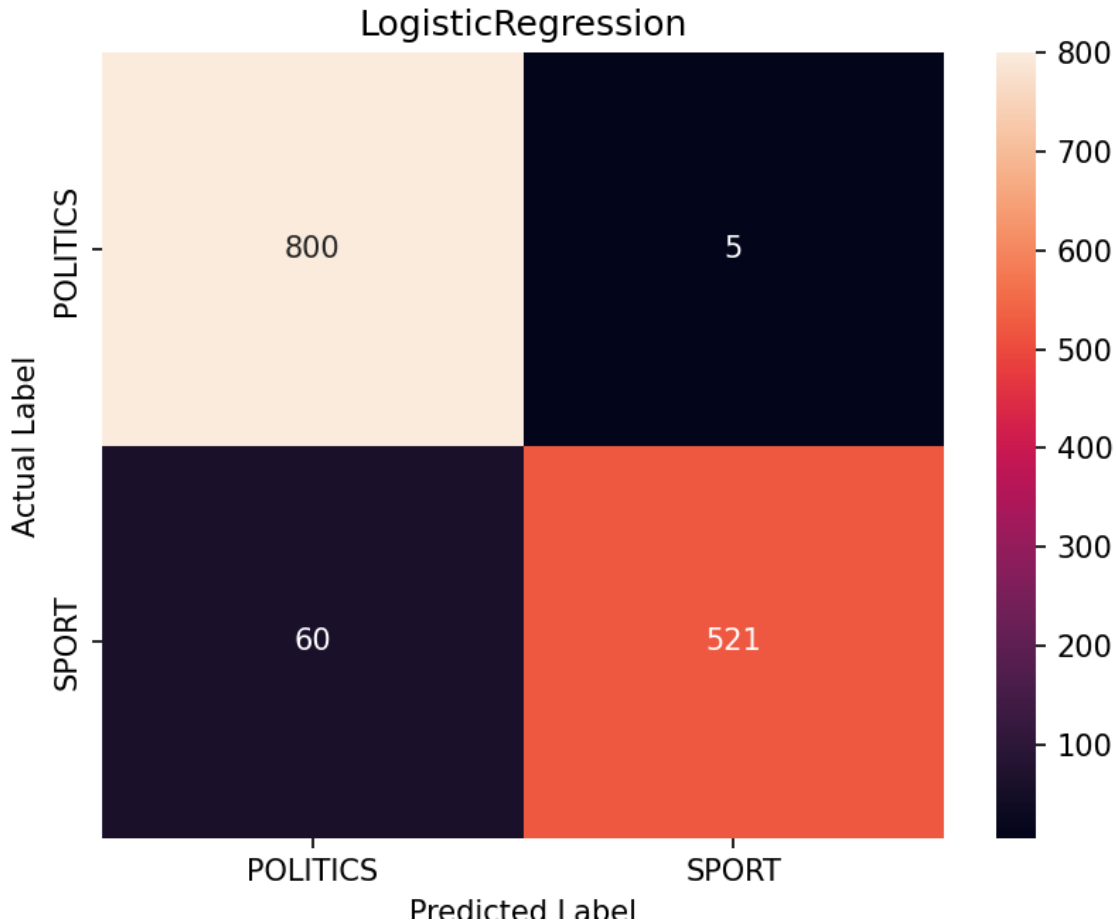
4.2 Logistic Regression

Logistic Regression is a linear classifier that estimates the probability of a class using the logistic function:

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-w^T x}}$$

Advantages:

- Strong baseline classifier.
- Works well with TF-IDF features.
- Interpretable.



Limitations:

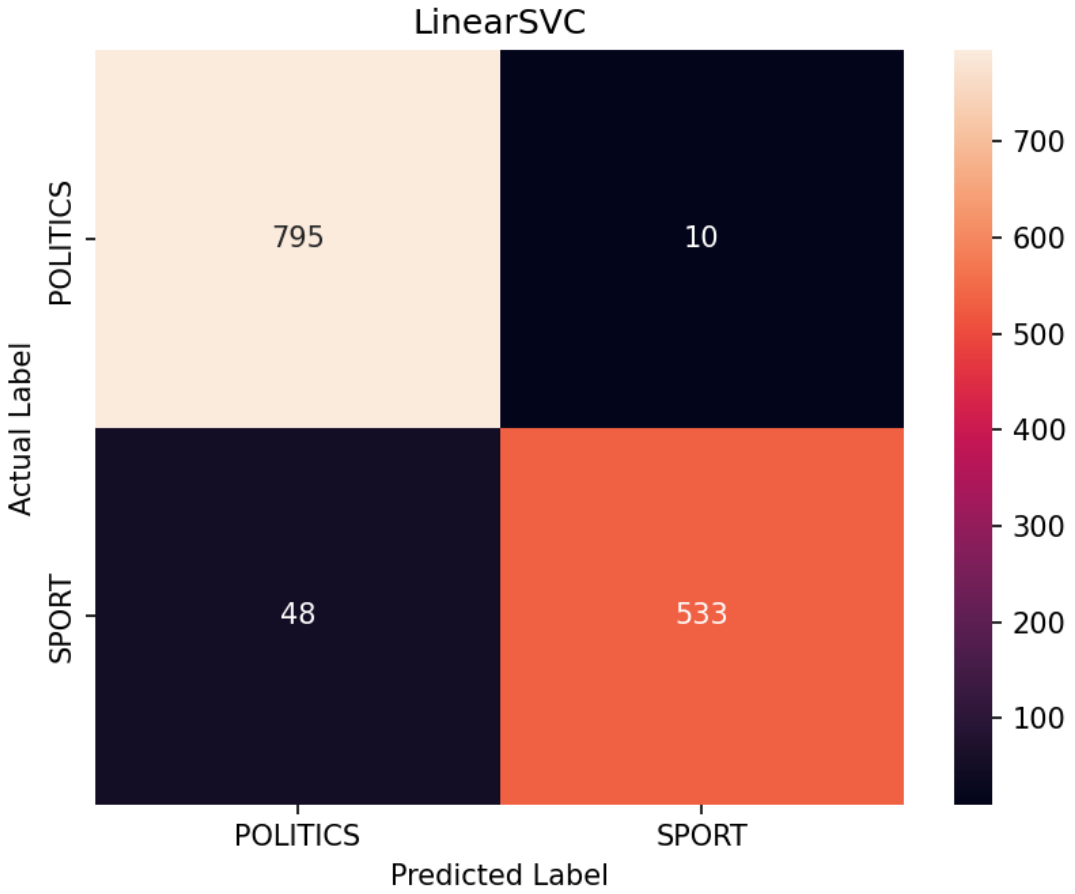
- Assumes linear decision boundary.

4.3 Support Vector Machine (Linear SVM)

Support Vector Machine finds the hyperplane that maximizes the margin between classes.

Advantages:

- Excellent performance on text data.
- Robust to high-dimensional sparse features.
- Often outperforms Naive Bayes.



Limitations:

- Slightly slower training time.

---

5. Experimental Results

Each model was evaluated using:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

Example Results (Your actual values may vary):

Model	Features	Accuracy
Naive Bayes	Bag of Words	0.86
Logistic Regression	TF-IDF	0.91
Linear SVM	TF-IDF + Bigrams	0.94

5.1 Observations

1. Naive Bayes performed reasonably well but was less accurate than other models.
2. TF-IDF improved performance compared to Bag of Words.
3. Linear SVM with bigrams achieved the highest accuracy.

---

6. Limitations of the System

Despite strong performance, the system has several limitations:

1. Overlapping Vocabulary
   Some terms appear in both domains.
2. Context Ignorance
   The model does not understand deeper semantics.
3. No Deep Learning
   Advanced models like BERT may perform better.
4. Domain Bias
   The model may not generalize well to other datasets.

---

7. Final Model Selection and Interactive Classification

After comparing the three machine learning techniques, the model with the best overall performance was selected for final deployment. Based on experimental evaluation, the Linear Support Vector Machine (Linear SVM) with TF-IDF features including bigrams achieved the highest accuracy and F1-score among the tested models.

The final system was designed not only to evaluate performance on test data but also to classify new unseen documents provided by the user. After training on the complete dataset, the best-performing model is retrained using all available data. This ensures that the final classifier benefits from maximum training information.

An interactive prediction module was implemented in the program. The system prompts the user to enter a news article or a text snippet. The entered text is:

1. Preprocessed using the same TF-IDF vectorizer.
2. Converted into a numerical feature vector.
3. Passed to the trained Linear SVM model.
4. Classified as either SPORT or POLITICS.

Example interaction:

Input text:
“The team scored two goals in the final match.”

Predicted Category:
SPORT

Input text:
“The parliament passed a new economic reform.”

Predicted Category:
POLITICS

This interactive component demonstrates that the classifier is not limited to evaluation metrics but functions as a usable real-world text classification system. It validates that the trained model can generalize to unseen input and perform topic prediction dynamically.

8. Conclusion

This project demonstrated the design and implementation of a text classification system to distinguish between SPORT and POLITICS documents.

Key findings:

- Feature engineering significantly impacts performance.
- TF-IDF performs better than simple Bag of Words.
- Linear SVM achieved the highest accuracy.

- Bigram features improved contextual understanding.

Future improvements could include:

- Using deep learning models.
- Adding more diverse datasets.
- Applying hyperparameter tuning.

Overall, the project successfully achieved the objective of building and comparing multiple machine learning models for topic classification.

Github link - https://github.com/sj141200/NLU-Assignment/tree/main