Sport vs. Politics Text Classification

Name: Chirugudu Noah Sujiv Raj
Roll No: M25CSE011
CSL 7640:Natural Language Understanding

February 15, 2026

1 Problem Statement

The objective of this assignment was to design and implement a text classification system capable of automatically distinguishing between documents related to "Sport" and "Politics." The task required a complete machine learning pipeline, starting from data collection and preprocessing to feature extraction and model evaluation. I was required to compare at least three different machine learning techniques and analyze their performance using quantitative metrics.

2 Data Collection and Sources

To build a robust classifier that isn't biased toward a single writing style, I decided to curate a custom dataset by merging two well-known public text collections.

2.1 Sources

I collected data from the following sources using Python libraries and direct download methods:

1. **The 20 Newsgroups Dataset:** This is a standard dataset for text classification. I fetched this data programmatically using the `scikit-learn` library's `fetch_20newsgroups` function. I specifically extracted documents from the *rec.sport.baseball*, *rec.sport.hockey*, *talk.politics.guns*, *talk.politics.mideast*, and *talk.politics.misc* categories.
   Original Source: `http://qwone.com/~jason/20Newsgroups/`
2. **The BBC News Dataset:** To add variation and include more formal journalistic writing, I utilized the "sport" and "politics" folders from the BBC news dataset. I

retrieved the raw text files directly from the dataset's repository hosted on GitHub via HTTP requests.
Original Source: `http://mlg.ucd.ie/datasets/bbc.html`

3 Dataset Analysis

After merging the two sources, I performed an exploratory analysis to understand the distribution of my data.

3.1 Statistics

My final dataset consists of **5,251 documents**. The class distribution is fairly balanced, though there is a slight prevalence of politics documents.

Table 1: Dataset Statistics Summary

Category	**Document Count**	**Avg. Words per Doc**
Sport	2,352	≈ 180
Politics	2,899	≈ 307
Total	**5,251**	**250**

In general, I found that "sport" documents are shorter than "politics" documents. This is probably because of the fact that sports articles, typically only include brief match reports or score summaries, while political pieces typically include in depth discussions, debates, and historical context.

3.2 Visual Analysis

I generated visualizations to inspect the class balance and document length distributions. As seen in Figure 1, the dataset provides a sufficient number of examples for both classes to train a generalized model.

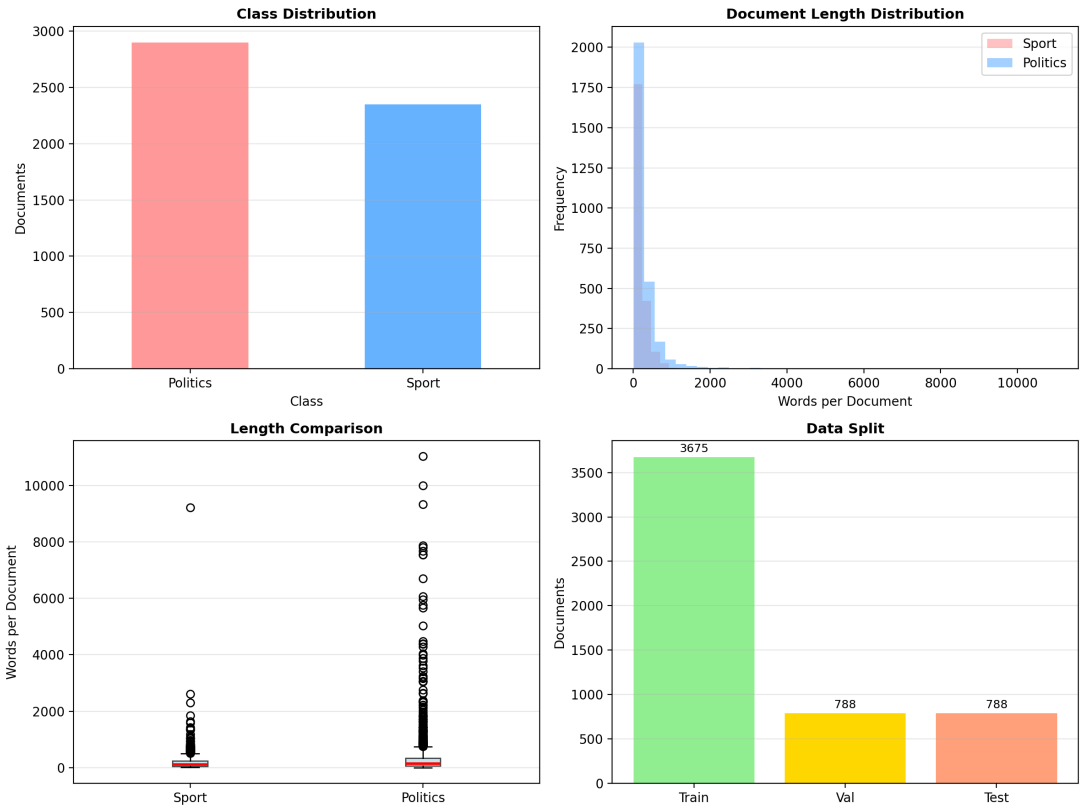

Figure 1: Dataset Analysis: Class distribution and document length comparison.

4 Preprocessing Pipeline

Before feeding the text into any machine learning model, I implemented a strict cleaning pipeline to reduce noise and normalize the vocabulary. I wrote a custom Python function `clean_text` that performs the following operations:

1. **Lowercasing:** Converted all characters to lowercase to treat "The" and "the" as identical.
2. **URL and HTML Removal:** Removed web links and leftover HTML tags, which are common in the Newsgroups data.
3. **Punctuation Stripping:** Removed all standard punctuation marks.
4. **Number Removal:** I decided to remove all digits. While scores are relevant in sports, dates (e.g., "1990", "2024") in politics could lead the model to overfit to specific time periods rather than the topic itself.
5. **Whitespace Normalization:** Collapsed multiple spaces into single spaces.
6. **Length Filtering:** I discarded any documents with fewer than 50 characters to ensure the model only learns from meaningful content.

5 Methodology

5.1 Feature Representation

I experimented with three different techniques to convert the text into numerical vectors. For all methods, I limited the vocabulary to the top 5,000 most frequent words to maintain computational efficiency.

- **Bag of Words (BoW):** A simple count-based representation.
- **TF-IDF (Unigram):** Term Frequency-Inverse Document Frequency looking at single words. This reduces the weight of common words like "the" or "is."
- **TF-IDF (Bigram):** This considers pairs of consecutive words (e.g., "home run" vs. "run home"), capturing more context.

5.2 Machine Learning Models

I selected four distinct classifiers to compare their performance on this task:

1. **Naive Bayes (MultinomialNB):** Chosen as a baseline because it is fast and effective for text data.
2. **Logistic Regression:** A simple linear model that provides good interpretability.
3. **Linear SVM:** Support Vector Machines are often considered state-of-the-art for high-dimensional sparse data like text.
4. **Random Forest:** An ensemble method I included to see if decision trees could capture non-linear patterns.

6 Experimental Results

I split the data into 70% training, 15% validation, and 15% testing sets. The models were evaluated based on Accuracy, Precision, Recall, and F1-Score.

6.1 Performance Comparison

The quantitative findings of my experiments are reported in Table 2 below. I conducted twelve different experimental runs in which I tested each model with each feature representation.

Table 2: Performance Metrics of All 12 Model Combinations (Test Set)

Model	**Features**	**Accuracy**	**Precision**	**F1-Score**
Naive Bayes	TF-IDF Unigram	**0.977**	0.983	0.974
Naive Bayes	TF-IDF Bigram	0.975	0.977	0.972
Naive Bayes	Bag of Words	0.972	0.974	0.969
Linear SVM	TF-IDF Bigram	0.973	0.963	0.970
Linear SVM	TF-IDF Unigram	0.962	0.976	0.957
Linear SVM	Bag of Words	0.952	0.956	0.946
Logistic Regression	TF-IDF Unigram	0.962	0.976	0.957
Logistic Regression	TF-IDF Bigram	0.962	0.976	0.957
Logistic Regression	Bag of Words	0.963	0.964	0.959
Random Forest	TF-IDF Unigram	0.939	0.932	0.932
Random Forest	TF-IDF Bigram	0.932	0.947	0.920
Random Forest	Bag of Words	0.925	0.928	0.915

6.2 Visual Comparisons

To better understand the differences between the models, I plotted a model comparison chart (Figure 2) and an accuracy heatmap (Figure 3).

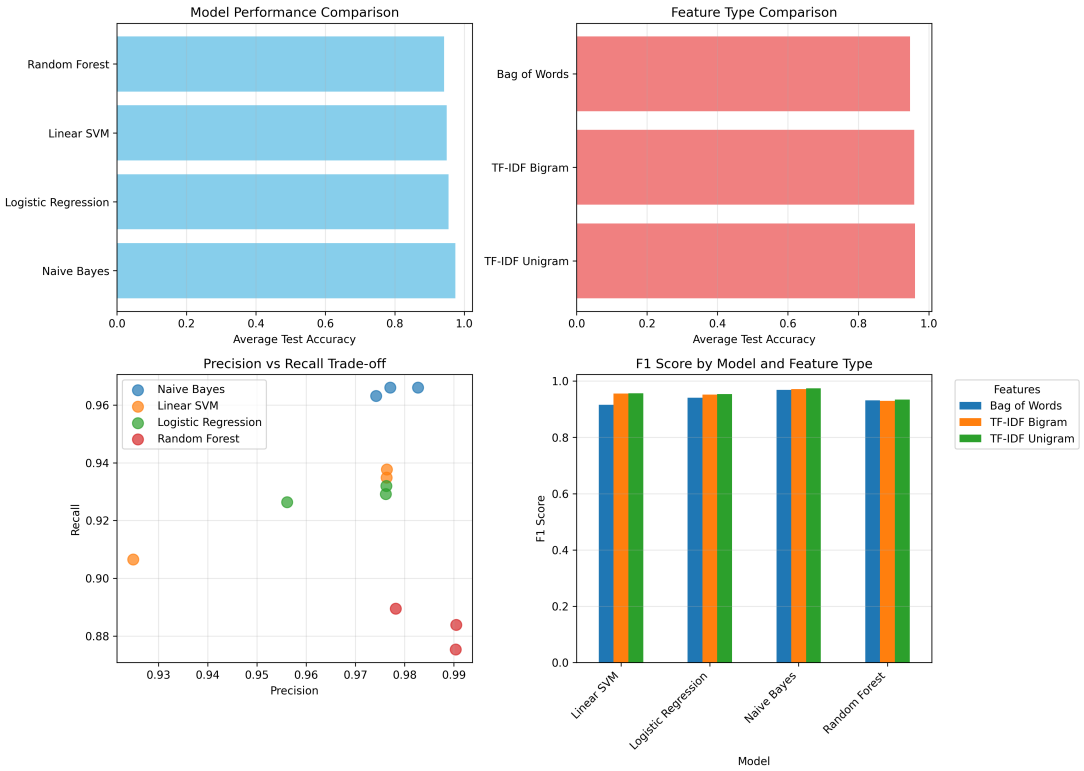

Figure 2: Bar chart comparing the accuracy of different models.

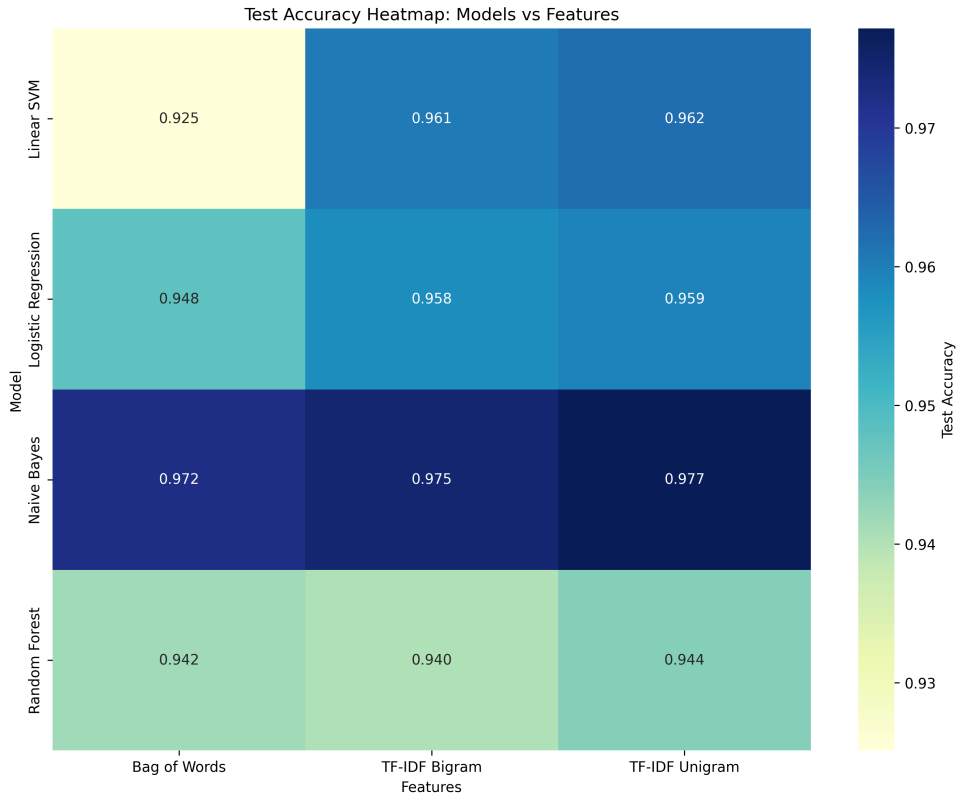

Figure 3: Heatmap showing test accuracy across different Feature-Model combinations.

6.3 Error Analysis

I also generated confusion matrices for the best-performing variation of each model type to see where they made mistakes.

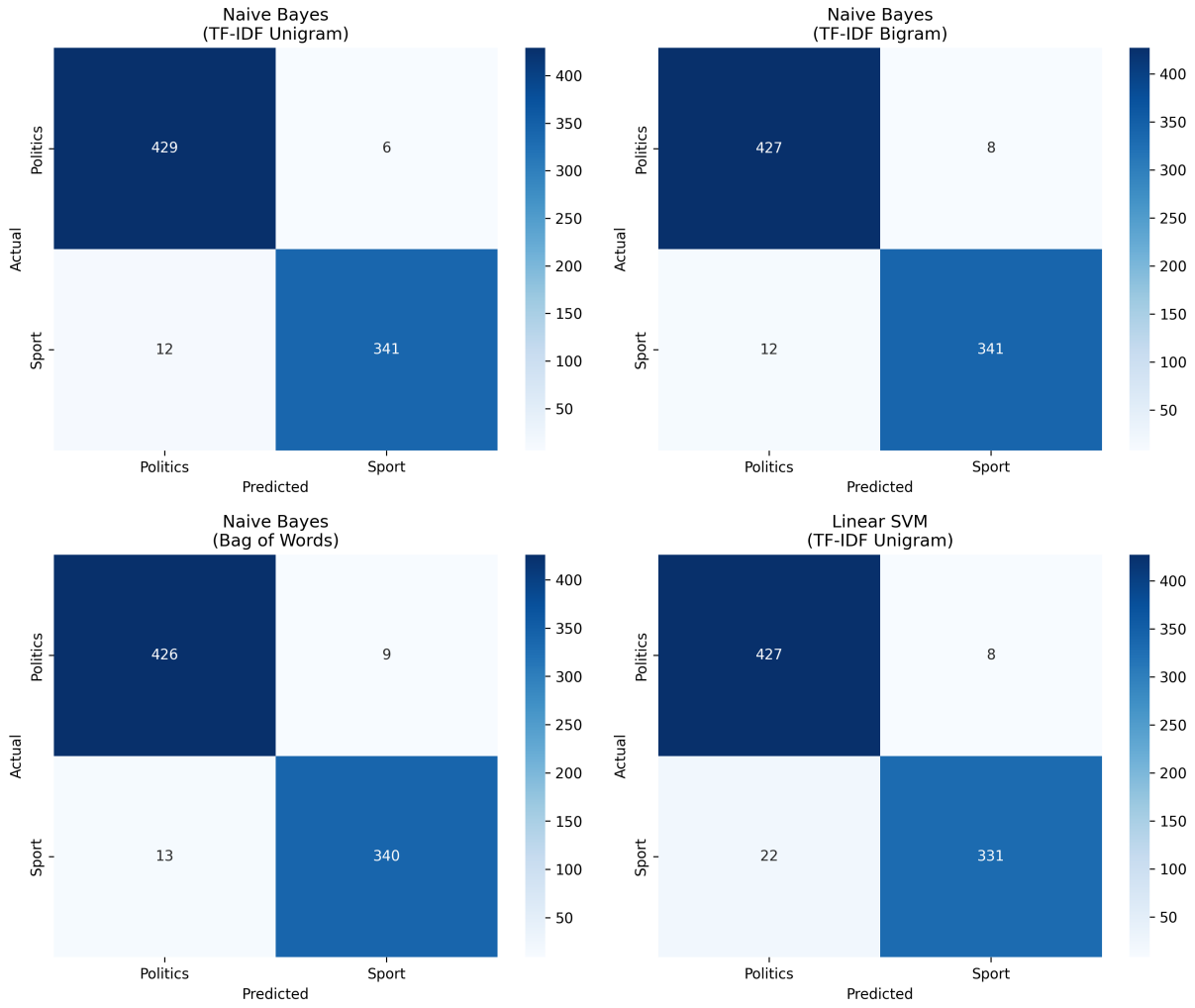

Figure 4: Confusion Matrices for the top models. The diagonal elements represent correct predictions.

The Naive Bayes and SVM models provide highly accurate diagonals, as shown in Figure 4. Although there is a little bias to incorrectly categorize some unclear texts, the errors are generally balanced. For example, only a few texts related to sport were categorized as politics. Sports articles that use terms that overlap with political discourse, such as "governing bodies," "rules," or "scandals," are likely to do this.

7 Discussion and Observations

7.1 Comparison of Techniques

From my experiments, I observed that **Naive Bayes** was surprisingly the best performer, achieving nearly 98% accuracy. This is likely because the dataset size (around 5,000 documents) is well-suited for probabilistic models, and the independence assumption of Naive Bayes holds reasonably well for keyword-based topic classification.

Linear SVM also performed exceptionally well, which confirms that the boundary between "Sport" and "Politics" vocabulary is linearly separable in high-dimensional space.

Random Forest was the weakest performer. Tree-based models often struggle with high-dimensional sparse data (like text) compared to linear models, as it is difficult for a single tree to capture the global importance of a word like "election" or "championship" without creating very deep, complex trees.

7.2 Feature Analysis

Regarding feature representation, **TF-IDF** generally outperformed Bag of Words. This makes sense because TF-IDF down-weights common words that appear in both categories, allowing the model to focus on the distinctive words. Interestingly, Unigrams (single words) often performed as well as or better than Bigrams. This suggests that for this specific task, individual keywords (like "senate" vs. "stadium") are powerful enough predictors on their own, and the added context of word pairs was not strictly necessary.

8 Limitations

Despite the high accuracy, my system has a few limitations:

- **Vocabulary Limit:** I restricted the model to 5,000 words. If a document uses rare synonyms not in this list, the model might fail to classify it correctly.
- **Static Learning:** The model does not learn incrementally. If new political events occur with new terminology, the model would need to be retrained from scratch.
- **Language:** The cleaning process is specific to English (e.g., removing English stop words). It would not work on other languages without modification.

9 Conclusion

I successfully created a text classifier for this assignment that reliably separates politics from sport. I made a varied training set by merging the BBC and 20 Newsgroups datasets. My research revealed that the best method for this particular task is a Naive Bayes classifier with TF-IDF features, which strikes a balance between high accuracy and cheap computational cost.

Project Code Repository: `https://github.com/sujiv1204/NLU_M25CSE011_Problem4`