Project Report: Classifying News Articles into Sports and Politics

A Machine Learning Approach

Submitted To:

Department of Computer Science

Course: Natural Language Understanding (NLU)

Assignment: Problem 4

Submitted By:

Name: **Akshat Jain**

Roll Number: **M25CSA003**

Date: February 15, 2026

GitHub Repository:

`https://github.com/m25csa003-glitch/NLU-Assignment-1`

Abstract

For this assignment, I was tasked with building a system that can automatically read a news article and decide whether it is about **Sports** or **Politics**. I used a dataset of BBC news articles to train three different machine learning models: Multinomial Naive Bayes, Logistic Regression, and Random Forest. My goal was to see if a computer could learn the difference between words like “election” and “tournament” just by looking at examples. After training the models, I tested them on new data. To my surprise, the simpler models (Naive Bayes and Logistic Regression) performed perfectly, achieving 100% accuracy. This report documents my entire process, from cleaning the messy text data to evaluating the final results.

Contents

1 Introduction

We see news everywhere—on our phones, on TV, and in newspapers. Apps like Google News automatically sort these stories into categories. If an article is about the Prime Minister, it goes to "Politics." If it's about the Cricket World Cup, it goes to "Sports." But how does a computer know which is which? It can't actually "read" or understand English.

This project is my attempt to build a similar system on a smaller scale. The core problem is **Binary Classification**. I have two buckets (Sports and Politics), and I need to throw incoming text documents into the correct bucket.

To do this, I used **Machine Learning**. Instead of writing thousands of rules (like "if the word is 'goal', put it in sports"), I fed the computer labeled examples and let it figure out the patterns on its own. I compared three very different algorithms to see which one learns the best. This report explains every step I took, the challenges I faced with the data, and why I think the results turned out the way they did.

2 The Dataset

2.1 Source of Data

I used the BBC News Dataset provided for this assignment (`bbc_news_text_complexity_summarizatio`
This is a well-known collection of news stories from the BBC website.

2.2 Understanding the Data

When I first opened the CSV file, I saw that it contained articles from five different topics: Business, Entertainment, Politics, Sport, and Tech. Since the problem statement specifically asked for a **Sports vs. Politics** classifier, my first step was to filter the data.

- I wrote a script to keep only the rows labeled `sport` or `politics`.
- I threw away everything else.

2.3 Data Statistics

After filtering, I was left with **908 articles**.

- **Sports Articles:** 505 (approx. 55%)
- **Politics Articles:** 403 (approx. 45%)

I noticed that the dataset is fairly balanced. There are slightly more sports articles, but the difference isn't big enough to cause major problems. If it were heavily imbalanced (e.g., 90% sports), I would have needed to use special techniques to fix it, but here it wasn't necessary.

I also checked for duplicates (identical articles appearing twice) and null values (empty rows). Fortunately, the dataset was clean. The text column contained full paragraphs of news reporting, which is great because longer text usually gives the model more clues to work with.

3 Methodology

Building this classifier wasn't just about importing a library and clicking "run." I had to prepare the data first. Computers understand numbers, not words, so a big part of this project was converting text into a numeric format.

3.1 Step 1: Preprocessing (Cleaning the Text)

Raw text is messy. It has capital letters, punctuation, and weird symbols. If I didn't clean it, the model would think "Election" and "election" are two different words.

- **Lowercasing:** I converted every single letter to lowercase.
- **Removing Noise:** I stripped out newlines (`\n`) and extra spaces.
- **Removing Stop Words:** Words like "the", "is", "at", and "on" appear in almost every sentence. They don't tell us anything about the topic. Whether it's sports or politics, the word "the" will be there. So, I removed these "stop words" to let the model focus on the important stuff like "parliament" or "stadium."

3.2 Step 2: Feature Extraction (TF-IDF)

This was the most critical step. I used a technique called **TF-IDF** (Term Frequency-Inverse Document Frequency).

Here is how I understand it:

> Imagine two words: "said" and "penalty". The word "said" appears in almost every news report. It has a high frequency, but it's not useful. The word "penalty" is rare, but when it appears, it strongly suggests the article is about Sports.

TF-IDF gives a low score to common words like "said" and a high score to rare, meaningful words like "penalty." This helps the model pay attention to the words that actually matter. I limited the system to look at the **top 5,000 most important words** to keep things fast.

3.3 Step 3: Splitting the Data

I couldn't train the model on all the data, or I wouldn't have any way to test it fairly. So, I split the 908 articles into two sets:

- **Training Set (80%):** I gave this to the models to learn from.
- **Test Set (20%):** I hid this data and only used it at the very end to check the accuracy.

4 The Models (The Contestants)

I decided to pit three different algorithms against each other.

4.1 Multinomial Naive Bayes

This is the "classic" choice for text classification. It works on probability. It basically asks: "Given that I see the word 'vote', what is the probability this is Politics?"

- **Why I chose it:** It is super fast and usually works very well with text.

4.2 Logistic Regression

Don't let the name confuse you; it's used for classification, not regression. It tries to draw a straight line through the data. Anything on one side is "Sports," and anything on the other is "Politics."

- **Why I chose it:** It is a linear model, which means it's great at separating things when the keywords are very distinct.

4.3 Random Forest

This is a more complex model. Imagine asking 100 people to guess the category of an article, and then going with the majority vote. Random Forest does exactly that but with 100 "Decision Trees."

- **Why I chose it:** It is usually very powerful and can capture complex patterns that simpler models miss.

5 Results

I evaluated the models using **Accuracy** (how many they got right overall) and **F1-Score** (a balance of precision and recall). Here is what happened on the Test Set:

Model	Accuracy Score
Multinomial Naive Bayes	100%
Logistic Regression	100%
Random Forest	98.9%

Table 1: Model Performance on Test Data

5.1 Analysis of the Results

Honestly, I was surprised to see 100% accuracy. Usually, machine learning models make at least a few mistakes.

- **Naive Bayes & Logistic Regression:** Both got every single prediction correct.
- **Random Forest:** It made a mistake on just 1 or 2 articles out of 182.

Why did they perform so well? I think the reason is the nature of the topics. "Sports" and "Politics" use very different vocabularies.

- **Sports words:** *match, win, loss, cup, champion, olympic, coach, team.*
- **Politics words:** *election, minister, tax, law, government, parliament, vote.*

There is very little overlap. You rarely see the word "touchdown" in a political debate, and you rarely see "legislation" in a football match. Because the words are so distinct, the simpler models (Linear models) found it very easy to draw a boundary between the two classes.

6 Limitations

Even though I got a perfect score, I know my system isn't perfect in the real world.

1. **It's too formal:** The model learned from BBC News, which uses perfect English. If I tried to use this on Twitter tweets, which are full of slang and typos, it would probably fail.
2. **Context:** The model is "dumb" in a way—it only looks for keywords. If a politician says, "*We need to tackle this problem like a linebacker*," the model might see "tackle" and "linebacker" and wrongly guess Sports.

3. **Data Size:** 900 articles is a small dataset. For a production-level system, I would need millions of articles to ensure it's robust.

7 Conclusion

This project taught me that sometimes, simple is better. We often think we need complex Deep Learning or massive Neural Networks to solve AI problems. But for a task like distinguishing Sports from Politics, a simple algorithm like **Naive Bayes** is not only much faster but also perfectly accurate. The clear difference in vocabulary between these two domains made it an ideal problem for standard text classification techniques.

All the source code, dataset, and implementation details for this project are available at the following GitHub repository:
`https://github.com/m25csa003-glitch/NLU-Assignment-1`