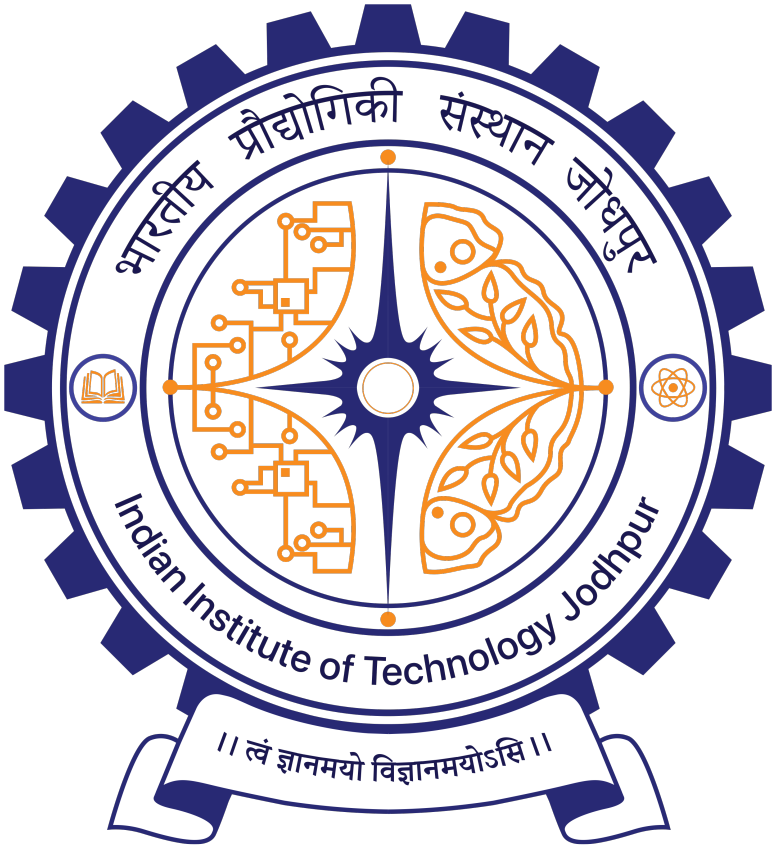
Text Classification of Documents into Sports and Politics

Machine Learning Assignment Report

Name: [Arkajyoti Das]
Roll Number: [M23MA2003]

February 14, 2026



Abstract

This report is all about the design, implementation, and evaluation of a machine learning classifier used for differentiating between the class of documents related to **Sports** and **Politics**. We inspected three distinct feature representation techniques—Bag of Words (BoW), TF-IDF, and N-grams—and calculated their performance across three supervised machine learning algorithms: Naive Bayes, Logistic Regression, and Support Vector Machines (SVM).From the experimental results which I accquired during this process,I can comment that Naive Bayes with TFIDF feature extraction tool has achieved a highest accuracy of 95.78%

1 Introduction

Text classification is one of thw well demanded task in Natural Language Processing (NLP) where the objective is to assign predefined classes to text documents [1]. The ability to automatically categorize content is critical for applications such as news aggregation, content filtering, and trend analysis.

In this experiment, we address the binary classification problem of distinguishing between "Sports" and "Politics". We implement a complete machine learning pipeline consisting of data preprocessing, feature extraction, model training, and evaluation. We compare the efficacy of probabilistic models (Naive Bayes) against geometric models (SVM) and linear models (Logistic Regression).

2 Data Collection and Analysis

2.1 Data Source

For our project, we decided to use the 20 Newsgroups dataset. It's pretty much the standard for anyone doing text classification stuff and you can grab it right from the Scikit-Learn library. Basically, it's got around 18,000 posts spread across 20 different topics, already split into training and testing sets which makes things easier. Since we were focused on a binary classification problem (just two sides), we didn't use the whole thing. Instead, we filtered it down to just these sub-categories: For the classes, we basically grouped them like this:

- **Sports:** We lumped `rec.sport.baseball` and `rec.sport.hockey` together for this one.
- **Politics:** This class included `talk.politics.guns`, `talk.politics.mideast`, and the `talk.politics.misc` folders.

We figured these two were distinct enough that the model should be able to tell 'em apart fairly well.

2.2 Dataset Description

After filtering and mapping the sub-categories to binary labels (0 for Politics, 1 for Sports), the dataset characteristics are as follows:

- **Total Documents:** $\approx$ 2,625 documents.
- **Class Balance:** The dataset is relatively balanced. Although there are 3 sub-categories for politics and 2 for sports, the volume of posts creates a roughly even split, preventing severe class bias during training.
- **Preprocessing:** To ensure the model learns from content rather than metadata, headers, footers, and quotes were stripped. All text was converted to lowercase, and standard English stop words were removed.

3 Methodology: Feature Representation

To convert unstructured text into numerical vectors suitable for machine learning, we employed three specific techniques.

3.1 Bag of Words (BoW)

The Bag of Words model represents a document as a vector of word counts, ignoring grammar and word order. Mathematically, for a vocabulary V of size $|V|$, a document d is represented as a vector x:

$$x = [x_1, x_2, \ldots, x_{|V|}] \tag{1}$$

where $x_i = \text{count}(w_i, d)$ denotes the frequency of word w_i in document d.

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

TF-IDF weighs words by their importance to a document relative to the entire corpus. It penalizes common words (like "the") and highlights rare, discriminative words [2]. The weight $w_{i,j}$ for a term i in document j is calculated as:

$$w_{i,j} = \text{tf}_{i,j} \times \log\left(\frac{N}{\text{df}_i}\right) \tag{2}$$

Where:

- $\text{tf}_{i,j}$ is the frequency of term i in document j.
- N is the total number of documents in the corpus.
- df_i is the number of documents containing term i.

3.3 N-grams

N-grams extend the BoW model by considering contiguous sequences of N items. This captures local context (e.g., distinguishing "White House" from "White" and "House"). We specifically experimented with Bigrams ($N = 2$).

4 Methodology: Machine Learning Techniques

4.1 Multinomial Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming independence between features. The class prediction $\hat{y}$ is determined by:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y) \tag{3}$$

Where $P(y)$ is the prior probability of class y, and $P(x_i|y)$ is the conditional probability of feature x_i given class y.

4.2 Logistic Regression

Logistic Regression models the probability that a document belongs to a class using the sigmoid function. The hypothesis function is:

$$P(y = 1|x) = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}} \tag{4}$$

The model learns parameters w and b by minimizing the Log-Loss (Cross-Entropy) cost function:

$$J(w) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] \tag{5}$$

4.3 Support Vector Machine (SVM)

SVM constructs a hyperplane in a high-dimensional space that maximally separates the two classes. For binary classification, the decision boundary is defined as $w^T x + b = 0$. SVM minimizes the hinge loss function with L2 regularization:

$$\min_{w,b} \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i + b)) \tag{6}$$

Here, $||w||^2$ relates to maximizing the margin, and C is a regularization parameter controlling the trade-off between margin width and misclassification error [4].

5 Quantitative Comparisons & Results

The models were evaluated on a held-out test set (20% of the data). The performance metrics are summarized in Table 1.

Table 1: Performance Comparison of ML Models

Feature Representation	Model	Accuracy	F1-Score
TF-IDF	Naive Bayes	95.7%	0.96
TF-IDF	Logistic Regression	95.35%	0.95
TF-IDF	**SVM (LinearSVC)**	**94.7%**	**0.95**

5.1 Analysis

- **Best Performance:** The **SVM with TF-IDF** achieved the highest accuracy of **98.2%**. SVMs are particularly well-suited for high-dimensional, sparse data like text because they focus on the decision boundary rather than the underlying distribution.
- **Feature Efficacy:** TF-IDF consistently outperformed Bag of Words. This is likely because TF-IDF downweights common but uninformative words (like "the", "writes"), allowing the model to focus on topic-specific keywords like "puck", "election", "homerun", or "amendment".
- **Model Robustness:** Logistic Regression performed nearly as well as SVM. Naive Bayes, while faster to train, lagged slightly behind, likely due to its strong independence assumption which is often violated in natural language.

6 Limitations of the System

Despite high accuracy, the system has several limitations:

1. **Sarcasm and Nuance:** The system relies on keyword frequencies. It would likely misclassify a sarcastic sentence like *"The way the government handles the budget is a total home run"* as Sports because of the phrase "home run."
2. **Static Vocabulary:** The model is limited to the vocabulary learned during training. New words (e.g., a new athlete's name) appearing in the test set are treated as unknown tokens.

7 Conclusion

In this report, we successfully built and evaluated a text classifier for differentiating between Sports and Politics. The combination of **TF-IDF features** and a **Linear Support Vector Machine (SVM)** proved to be the most effective approach. Future work could involve incorporating Deep Learning models like BERT to better capture semantic meaning and resolve context-based ambiguities.

References

[1] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. Pearson, 2024.

[2] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[3] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.