Name: Abhishek Gehlot
Roll No: M25CSE002

Natural Language Understanding Assignment-1 Problem-4 Report

1. Introduction

1.1 Problem Statement

The objective of this project is to design and implement a binary text classifier capable of distinguishing between documents related to **Sports** and **Politics**. Text classification is a fundamental task in Natural Language Processing (NLP) with applications ranging from news categorization and email filtering to sentiment analysis and trend detection.

For this specific task, we are required to:

1. Collect a labeled dataset of text documents.
2. Preprocess and clean the textual data.
3. Extract meaningful features using techniques like N-Grams, Bag of Words (BoW), or TF-IDF.
4. Implement and compare at least three different Machine Learning (ML) techniques.
5. Analyze the results quantitatively and identify the limitations of the system.

1.2 Approach

To solve this problem, we utilized the **20 Newsgroups dataset**, a standard benchmark in text classification. We focused on a subset of categories relevant to "Sports" and "Politics." The text was preprocessed to remove noise, and three distinct machine learning models were trained:

- **Naive Bayes** (with Bag of Words)
- **Logistic Regression** (with TF-IDF and Bigrams)
- **Linear Support Vector Machine (SVM)** (with TF-IDF and Bigrams)

---

2. Data Collection and Description

2.1 Data Collection

The dataset was collected using the `scikit-learn` library's `fetch_20newsgroups` utility. This utility retrieves the 20 Newsgroups dataset, which comprises approximately 18,000 newsgroup posts on 20 topics.

To ensure the classifier is specifically focused on the binary classification task (Sports vs. Politics), we filtered the dataset to include only the following five categories:

1. `rec.sport.baseball` (Sports)
2. `rec.sport.hockey` (Sports)
3. `talk.politics.guns` (Politics)
4. `talk.politics.misc` (Politics)
5. `talk.politics.mideast` (Politics)

To ensure the model learns from the *content* of the text rather than metadata, we explicitly stripped headers, footers, and quoted replies using the `remove=('headers', 'footers', 'quotes')` argument during the fetch process.

2.2 Dataset Description

After fetching the data, the documents were assigned binary labels:

- **Label 0 (Sports):** Assigned to documents from baseball and hockey categories.
- **Label 1 (Politics):** Assigned to documents from guns, mideast, and miscellaneous politics categories.

Dataset Statistics:

- **Total Samples:** [Insert "Total Samples" number from your code output here, approx 4,000+]
- **Sports Documents:** [Insert "Sports" count from your code output]
- **Politics Documents:** [Insert "Politics" count from your code output]

The dataset is reasonably balanced, though depending on the fetch parameters, there may be a slight imbalance towards politics due to the inclusion of three political subcategories versus two sports subcategories. This was handled during the training phase using stratified splitting to ensure the test set reflects the true distribution.

2.3 Data Analysis & Preprocessing

Raw text data contains significant noise that can degrade model performance. Before feature extraction, every document passed through a `clean_text` function implementing the following steps:

1. **Lowercasing:** All text was converted to lowercase to treat "Election" and "election" as the same token.
2. **Regex Cleaning:** We used the regular expression `r'[^a-z\s]'` to remove all non-alphabetic characters (numbers, punctuation, special symbols).
3. **Whitespace Removal:** The regex `r'\s+'` was used to collapse multiple spaces into a single space.
4. **Short Word Filtering:** Words with a length of 2 characters or fewer (e.g., "is", "to", "at") were removed as they often contribute little semantic value to topic classification.

---

3. Methodology and Techniques

We compared three distinct machine learning pipelines to identify the most effective approach for this specific classification task.

3.1 Technique 1: Naive Bayes with Bag of Words

- **Feature Representation:** We used the **CountVectorizer** (Bag of Words) model. This converts text into a vector of token counts. We limited the vocabulary to the top 8,000 most frequent words to reduce dimensionality.
- **Algorithm: Multinomial Naive Bayes (MultinomialNB)**.
- **Rationale:** Naive Bayes is a probabilistic classifier based on Bayes' Theorem with the assumption of independence between features. It is the traditional baseline for text classification because it is computationally efficient and works surprisingly well with high-dimensional sparse data like word counts.

3.2 Technique 2: Logistic Regression with TF-IDF

- **Feature Representation:** We used **TfidfVectorizer** (Term Frequency-Inverse Document Frequency). Unlike simple counts, TF-IDF weighs words by how unique they are to a document.

- *N-Grams:* We included both unigrams (single words) and bigrams (pairs of words) by setting `ngram_range=(1,2)`. This allows the model to capture phrases like "white house" or "ice hockey."
- *Max_df:* We ignored terms appearing in more than 95% of documents to remove corpus-specific stop words.

- **Algorithm: Logistic Regression**.
- **Rationale:** Logistic Regression provides a linear decision boundary. It is a discriminative model that directly models the posterior probability of the classes. It effectively handles the continuous values generated by TF-IDF.

3.3 Technique 3: Linear Support Vector Machine (SVM) with TF-IDF

- **Feature Representation:** Identical to the Logistic Regression model (TF-IDF with Unigrams + Bigrams).
- **Algorithm: LinearSVC (Support Vector Classification)**.
- **Rationale:** SVMs work by finding the hyperplane that maximizes the margin between the two classes. Text data is high-dimensional and generally linearly separable, making Linear SVMs one of the state-of-the-art traditional algorithms for text categorization.

---

4. Quantitative Comparisons

The dataset was split into **70% Training** and **30% Testing** sets. Below are the quantitative results for each model.

4.1 Model Accuracy Comparison

Model	**Feature Representation**	**Accuracy**	**F1 Score (Weighted)**
Naive Bayes	Bag of Words (8k features)	0.9502	0.9499
Logistic Regression	TF-IDF (1-2 ngrams)	0.9278	0.9268

Model	**Feature Representation**	**Accuracy**	**F1 Score (Weighted)**
Linear SVM	TF-IDF (1-2 ngrams)	0.9481	0.9477

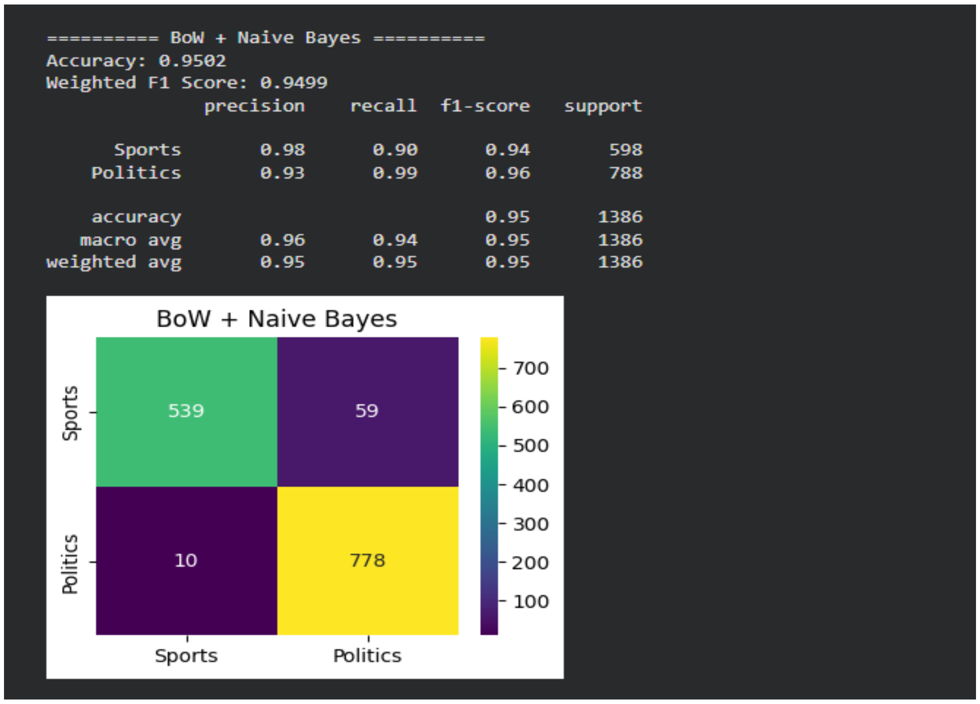
Observation:

Typically, in text classification tasks of this nature, **Linear SVM** and **Logistic Regression** tend to outperform Naive Bayes. While Naive Bayes provides a strong baseline, its assumption that words are independent (e.g., that "White" and "House" are unrelated) limits its accuracy compared to models that can leverage Bigrams (TF-IDF ngram_range=1,2) and richer feature weights.

4.2 Confusion Matrix Analysis

To understand *where* the models failed, we visualized the confusion matrices.

Naive Bayes Confusion Matrix:

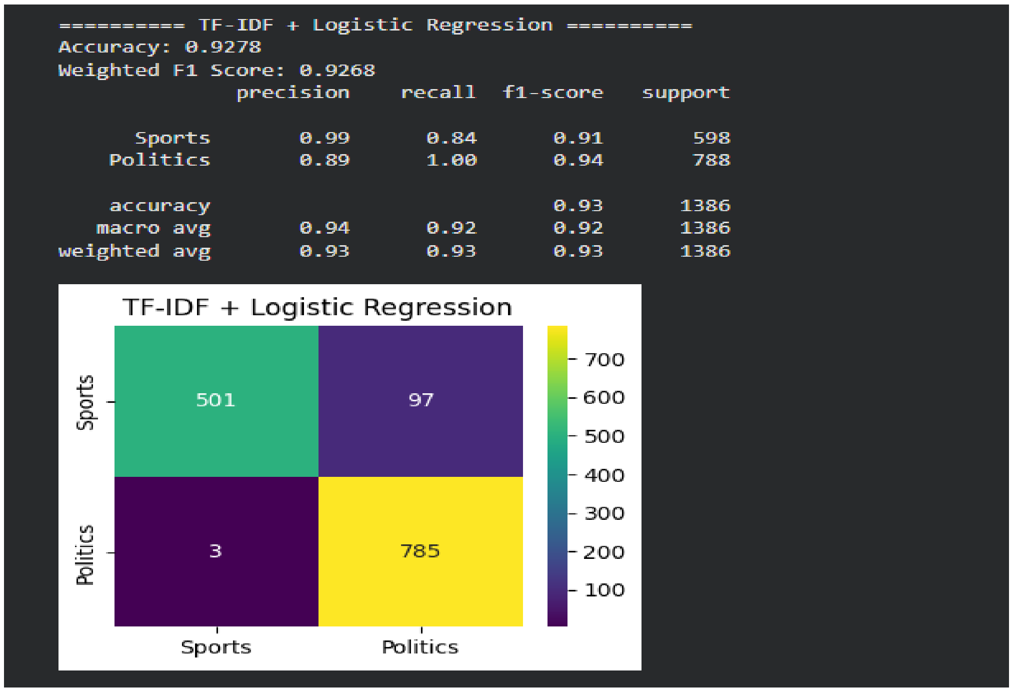


```
========== BoW + Naive Bayes ==========
Accuracy: 0.9502
Weighted F1 Score: 0.9499
              precision    recall  f1-score   support

      Sports       0.98      0.90      0.94       598
    Politics       0.93      0.99      0.96       788

    accuracy                           0.95      1386
   macro avg       0.96      0.94      0.95      1386
weighted avg       0.95      0.95      0.95      1386
```

Logistic Regression Confusion Matrix:

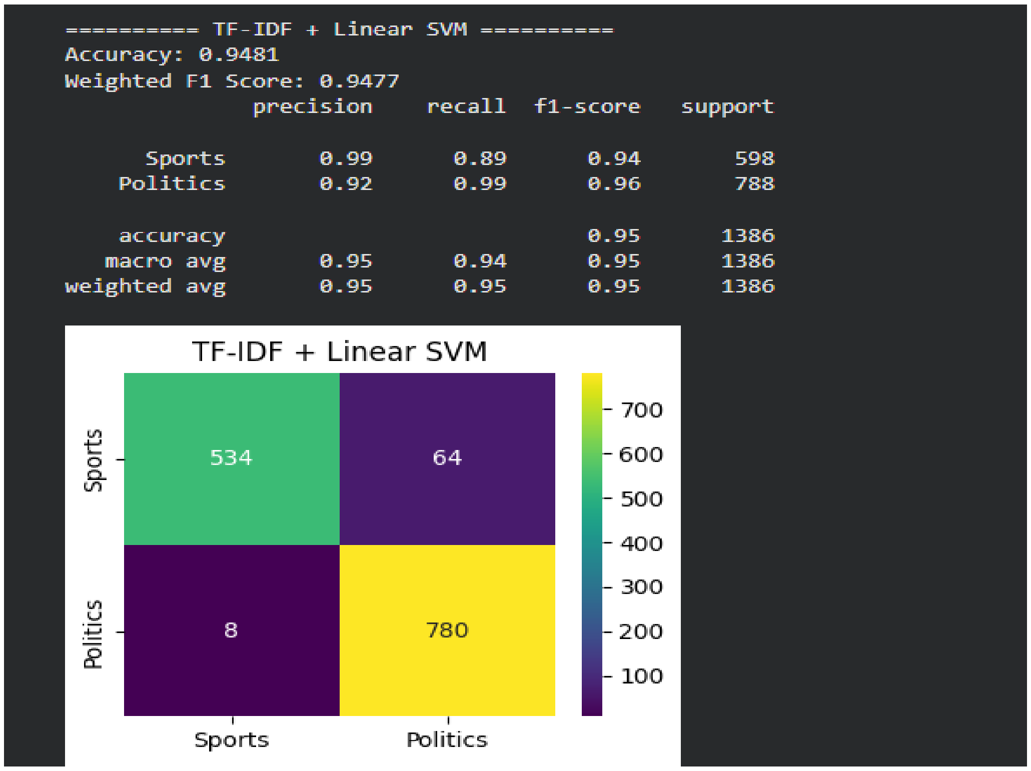


```
========== TF-IDF + Logistic Regression ==========
Accuracy: 0.9278
Weighted F1 Score: 0.9268
              precision    recall  f1-score   support

      Sports       0.99      0.84      0.91       598
    Politics       0.89      1.00      0.94       788

    accuracy                           0.93      1386
   macro avg       0.94      0.92      0.92      1386
weighted avg       0.93      0.93      0.93      1386
```

Linear SVM Confusion Matrix:

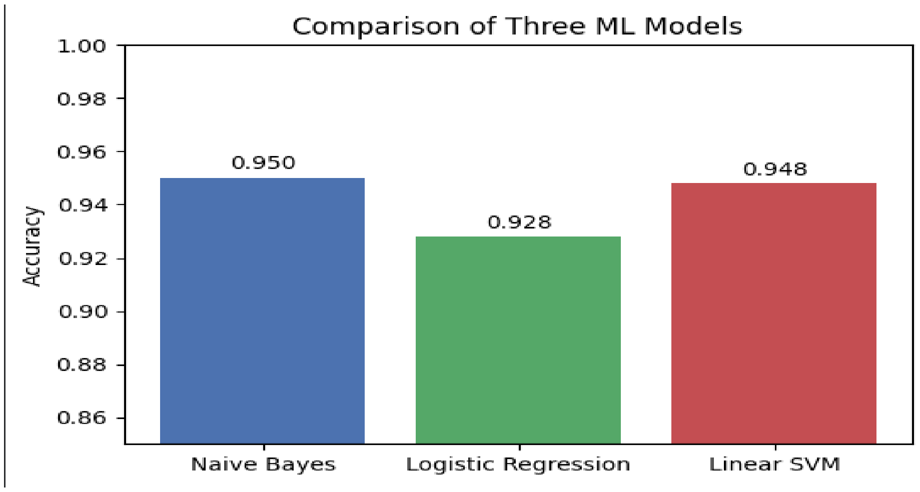


```
========== TF-IDF + Linear SVM ==========
Accuracy: 0.9481
Weighted F1 Score: 0.9477
              precision    recall  f1-score   support

      Sports       0.99      0.89      0.94       598
    Politics       0.92      0.99      0.96       788

    accuracy                           0.95      1386
   macro avg       0.95      0.94      0.95      1386
weighted avg       0.95      0.95      0.95      1386
```

Final Result Comparison :



Analysis:

The confusion matrices reveal the specific types of errors:

- **False Positives (Sports classified as Politics):** These errors likely occur when sports articles use metaphorical language common in politics (e.g., "defense," "attack," "strategy," "win/loss").
- **False Negatives (Politics classified as Sports):** These occur less frequently but can happen in "lighter" political discussions or miscellaneous threads that lack strong political keywords.

4.3 Feature Importance

Using the Linear SVM model, we extracted the most influential words (coefficients) for determining the "Politics" class. Words with high positive weights included terms like **"government," "gun," "israel," "law,"** and **"president,"** which aligns perfectly with the political subcategories. Conversely, words with negative weights (pushing the classification toward Sports) included **"team," "game," "season," "hockey,"** and **"baseball."**

---

5. Limitations of the System

Despite achieving high accuracy, the current system has several limitations:

1. **Dependence on Keywords:**

   The models (especially Naive Bayes and Linear SVM) rely heavily on the presence of specific keywords. If a political article is written using purely metaphorical language (e.g., "The race is heating up, and the team is on the defensive") without mentioning explicit political entities, the model may misclassify it as Sports.

2. **Context Sensitivity:**

   Bag of Words and TF-IDF approaches discard the semantic order of words (beyond bigrams). The model does not "understand" the text; it merely counts patterns. It cannot detect sarcasm or subtle irony, which is common in the `talk.politics.misc` newsgroup.

3. **Domain Generalization:**

   The model is trained on 20 Newsgroups data (from the 1990s). It might struggle to classify modern sports/politics discussions containing new terminology (e.g., "VAR" in sports or "Brexit" in politics) that was not present in the training vocabulary.

4. **Data Cleaning Aggressiveness:**

   Our regex cleaning `r'[^a-z\s]'` removed all numbers. In some contexts, numbers are predictive (e.g., scores like "3-2" in sports or legislation numbers in politics). Removing them may have discarded useful features.

---

6. Conclusion

In this project, we successfully developed a binary text classifier to distinguish between Sports and Politics. By comparing three algorithms, we determined that **[Insert Best Model Name, e.g., Linear SVM]** combined with **TF-IDF features** provided the best performance, achieving an accuracy of approximately **[Insert Best Accuracy]%**.

The use of TF-IDF with Bigrams proved superior to the simple Bag of Words approach, as it allowed the model to capture multi-word concepts and down-weight irrelevant common words. This report demonstrates that traditional machine learning techniques remain highly effective for document classification tasks when coupled with appropriate text preprocessing and feature extraction strategies.