News Classification Project Report

Problem Statement

The goal of this project is to classify news articles into one of the following four categories:

- **World**
- **Sports**
- **Business**
- **Science and Technology**

This is a **multiclass classification problem**, where each article is labeled with a class index from 1 to 4.

Dataset Overview

- **Training Dataset:** Loaded from `train.csv`
- **Testing Dataset:** Loaded from `test.csv`
- Both datasets contain the following relevant columns:
  - `Title` — the headline of the article
  - `Description` — the summary or snippet
  - `Class Index` — target label from 1 to 4
- The class labels were mapped to readable labels for interpretability.
- Combined `Title` and `Description` to form a single text column: `Summary`.

Text Preprocessing

- Combined title and description to create the feature column: `Summary`.
- Handled missing values (if any).
- Converted class indices into categorical string labels using a dictionary map.
- Used visualizations (bar charts) to confirm class balance in the dataset.
- Created Word Clouds for each category to analyze frequent words.
- **Stopword Handling:**
  - Used NLTK's predefined stopwords list and customized it.
  - Removed general words like "said", "br", and spaces.

  - Retained "not" and "no" as they are important for sentiment and meaning.

- **Tokenization and Lemmatization:**
  - Applied NLTK's `word_tokenize` for splitting text into tokens.
  - Used `WordNetLemmatizer` for converting words to their root form.

- **Vectorization:** (In later steps, assumed based on pipeline)
  - Likely used TF-IDF or CountVectorizer (details may follow in later cells).

Exploratory Data Analysis

- Visualized the most common words in each class using WordClouds.
- Verified that the dataset is balanced across the four categories.

Next Steps

The remaining part of the report will include:

- Feature extraction techniques (e.g., TF-IDF).
- Classifier models tried (e.g., Logistic Regression, Naive Bayes, XGBoost, MLP).
- Hyperparameter tuning and evaluation (e.g., GridSearchCV).
- Validation/test performance and model comparison.

Models and Results

1. Logistic Regression

- Grid Search Best Params: `C=10`, `solver=newton-cg`, `max_iter=100`
- Best Cross-Validation Score: `0.285`

2. XGBoost Classifier

- Objective: `multi:softmax`
- Best Parameters: `learning_rate=0.3`, `max_depth=5`, `n_estimators=100`
- Best CV Score: `0.853`
- Performed best among all evaluated models.

3. Random Forest

- Hyperparameters tuned using GridSearchCV
- Best CV Accuracy: `0.814`

4. Multinomial Naive Bayes

- CV Accuracy: `0.877`
- One of the top-performing models for this dataset.

5. MLP Classifier

- Train Accuracy: `0.814`
- Good performance, slightly lower than Naive Bayes and XGBoost.

Evaluation Strategy

- Train-Validation-Test split used for model development and testing.
- Accuracy used as the primary metric.

Insights

- Simpler models like Multinomial Naive Bayes can outperform complex ones on textual data.
- Logistic Regression underperformed suboptimal decision boundary or needs more text pre processing.
- XGBoost provided the highest CV score, showing its strength on structured high-dimensional data.

Conclusion

The XGBoost and Naive Bayes classifiers performed best on this multiclass text classification task. Further work could involve deeper neural models or fine-tuned transformers.