Text Classification using Naive Bayes and Sentiment Analysis on Blog Posts

**Overview**

In this assignment, you will work on the "blogs\_categories.csv" dataset, which contains blog posts categorized into various themes. Your task will be to build a text classification model using the Naive Bayes algorithm to categorize the blog posts accurately. Furthermore, you will perform sentiment analysis to understand the general sentiment (positive, negative, neutral) expressed in these posts. This assignment will enhance your understanding of text classification, sentiment analysis, and the practical application of the Naive Bayes algorithm in Natural Language Processing (NLP).

**Dataset**

The provided dataset, "blogs\_categories.csv", consists of blog posts along with their associated categories. Each row represents a blog post with the following columns:

* **Text**: The content of the blog post. Column name: Data
* **Category**: The category to which the blog post belongs. Column name: Labels

**Tasks**

**1. Data Exploration and Preprocessing**

* Load the "blogs\_categories.csv" dataset and perform an exploratory data analysis to understand its structure and content.
* Preprocess the data by cleaning the text (removing punctuation, converting to lowercase, etc.), tokenizing, and removing stopwords.
* Perform feature extraction to convert text data into a format that can be used by the Naive Bayes model, using techniques such as TF-IDF.

**2. Naive Bayes Model for Text Classification**

* Split the data into training and test sets.
* Implement a Naive Bayes classifier to categorize the blog posts into their respective categories. You can use libraries like scikit-learn for this purpose.
* Train the model on the training set and make predictions on the test set.

**3. Sentiment Analysis**

* Choose a suitable library or method for performing sentiment analysis on the blog post texts.
* Analyze the sentiments expressed in the blog posts and categorize them as positive, negative, or neutral. Consider only the Data column and get the sentiment for each blog.
* Examine the distribution of sentiments across different categories and summarize your findings.

**4. Evaluation**

* Evaluate the performance of your Naive Bayes classifier using metrics such as accuracy, precision, recall, and F1-score.
* Discuss the performance of the model and any challenges encountered during the classification process.
* Reflect on the sentiment analysis results and their implications regarding the content of the blog posts.

**Submission Guidelines**

* Your submission should include a comprehensive report and the complete codebase.
* Your code should be well-documented and include comments explaining the major steps.

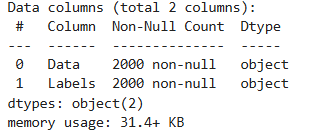
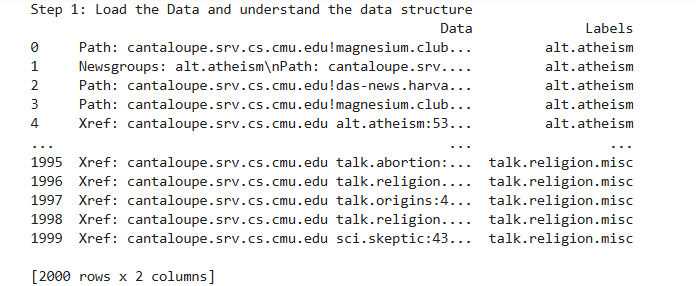
**Evaluation Criteria**

* Correct implementation of data preprocessing and feature extraction.
* Accuracy and robustness of the Naive Bayes classification model.
* Depth and insightfulness of the sentiment analysis.
* Clarity and thoroughness of the evaluation and discussion sections.
* Overall quality and organization of the report and code.

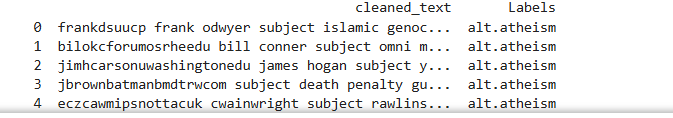
Good luck, and we look forward to your insightful analysis of the blog posts dataset!

**Observations:**

* 1. **Data Exploration and Preprocessing**
* Load the "blogs\_categories.csv" dataset and perform an exploratory data analysis to understand its structure and content.



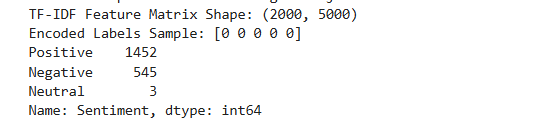
* **2000 rows** and **2 columns** (Data and Labels).
* Each text entry in the Data column likely represents a blog post or message.
* The Labels column has **20 unique categories**, with each category appearing **100 times**.
* Preprocess the data by cleaning the text (removing punctuation, converting to lowercase, etc.), tokenizing, and removing stopwords.



* **Text Cleaning:** Removing metadata, special characters, and digits.
* **Lowercasing:** Ensures text uniformity.
* **Tokenization:** Splits the text into individual words.
* **Stopword Removal:** Filters out common words like "the", "and", etc.
* Perform feature extraction to convert text data into a format that can be used by the Naive Bayes model, using techniques such as TF-IDF.

**some important parameters to improve the feature extraction process:**

* max\_features=5000 → Limits the vocabulary to the top 5000 most important words.
* ngram\_range=(1, 2) → Uses both unigrams (single words) and bigrams (two-word combinations) to capture context.
* stop\_words='english' → Automatically removes common English stopwords.



**TF-IDF Feature Matrix Shape**

* **Shape: (2000, 5000)**  
  ➔ This indicates that the dataset contains **2000 blog posts** and has been transformed into a **5000-dimensional feature space** using TF-IDF.  
  ➔ The max\_features=5000 parameter limited the vocabulary to the **top 5000 most significant terms**.

**Encoded Labels Sample**

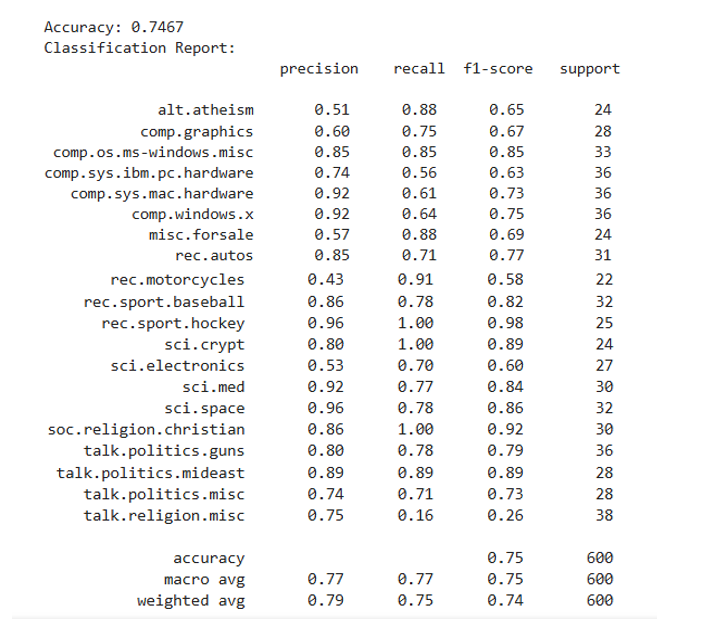
* The encoded labels [0 0 0 0 0] suggest that the first few rows all belong to the same category.
* This might indicate an **imbalance in the dataset** or that these entries belong to a frequently occurring category.

**2. Naive Bayes Model for Text Classification**

* Split the data into training and test sets.
* Implement a Naive Bayes classifier to categorize the blog posts into their respective categories. You can use libraries like scikit-learn for this purpose.
* Train the model on the training set and make predictions on the test set.

**Insights from Model Performance**

The overall **accuracy** is **0.7467** (≈ 75%), which is decent for a text classification task with 20 categories.

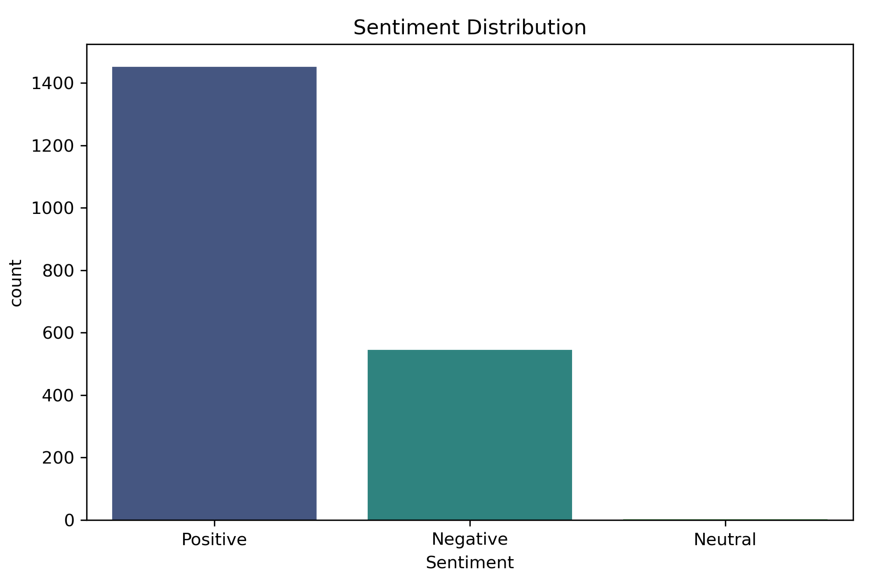


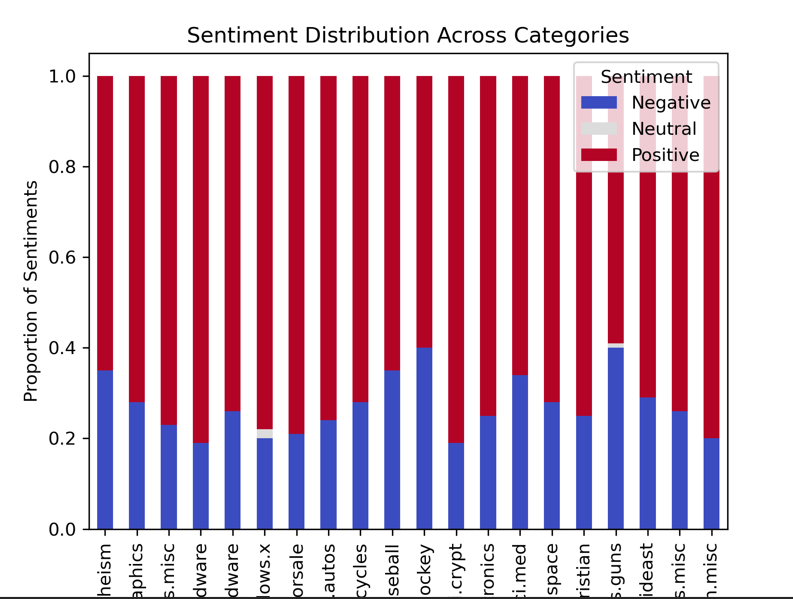
**Sentiment Analysis**

* Choose a suitable library or method for performing sentiment analysis on the blog post texts.
* Analyze the sentiments expressed in the blog posts and categorize them as positive, negative, or neutral. Consider only the Data column and get the sentiment for each blog.
* Examine the distribution of sentiments across different categories and summarize your findings.

**Sentiment Distribution**

* **Positive: 1452 (72.6%)**
* **Negative: 545 (27.2%)**
* **Neutral: 3 (0.2%)**





* The dataset is **heavily skewed toward positive sentiment**.
* **Neutral sentiments are extremely rare** (just 3 posts), which may indicate that the dataset lacks balanced neutral content or that the sentiment detection tool (TextBlob) is less effective at capturing neutrality in text.
* The **imbalance** in sentiment distribution could affect model performance, especially if neutral predictions are require

**Evaluation**

* Evaluate the performance of your Naive Bayes classifier using metrics such as accuracy, precision, recall, and F1-score.
* Discuss the performance of the model and any challenges encountered during the classification process.
* Reflect on the sentiment analysis results and their implications regarding the content of the blog posts.

**Key Performance Observations**

**Strong Performing Categories:**

* **sci.crypt**, **rec.sport.hockey**, and **soc.religion.christian** achieved **very high recall** (close to 100%) and strong precision.
* These categories may have distinct vocabulary or unique language patterns that the Naive Bayes model successfully captured.

**Moderately Performing Categories:**

* **comp.graphics**, **rec.autos**, and **talk.politics.guns** show balanced precision and recall, indicating stable performance.
* These categories likely contain some overlapping terms but still maintain distinctive features.

**Weak Performing Categories:**

* **talk.religion.misc** has **low recall (0.16)** and **low F1-score (0.26)**, indicating frequent misclassifications.
* **rec.motorcycles** has **high recall (0.91)** but **low precision (0.43)**, meaning the model identifies most motorcycle-related posts but incorrectly labels unrelated posts as motorcycles.

**Precision vs. Recall Analysis**

* Categories with **high recall but low precision** (e.g., rec.motorcycles) tend to capture most relevant data but with frequent false positives.
* Categories with **high precision but low recall** (e.g., comp.sys.mac.hardware) are selective and may miss important examples.

**Macro vs. Weighted Average**

* **Macro Average (0.77)** — Treats all classes equally, giving insights into overall balance.
* **Weighted Average (0.79)** — Favors larger classes, hence closer to the accuracy score.

The Naive Bayes model performed reasonably well, especially for categories with distinct language patterns. However, addressing **class imbalance** and enhancing text preprocessing could further improve performance. Sentiment analysis suggests that the dataset contains emotionally charged discussions, particularly in sensitive topics like politics and religion.

