**Text Classification Using Naive Bayes and Sentiment Analysis on Blog Posts**

**Overview**

This assignment focuses on applying Natural Language Processing (NLP) techniques for text classification and sentiment analysis using the **"blogs\_categories.csv"** dataset. The dataset consists of 2000 blog posts categorized into various themes. The primary objectives were:

* Build a **Naive Bayes classifier** to categorize blog posts.
* Perform **sentiment analysis** to understand the tone of these posts.

**Dataset**

* **Total Records:** 2000
* **Columns:**
  + Text – Blog post content.
  + Category – Category label (20 unique classes).
* **Missing Values:** None
* **Top Category:** alt.atheism (100 entries)
* **Data Types:** Both columns are of type object.

**1. Data Exploration and Preprocessing**

**Exploration Highlights:**

* The dataset was clean with no missing values.
* Categories were fairly distributed, although some class imbalance was noted.

**Preprocessing Steps:**

* Converted text to lowercase.
* Removed punctuation and special characters.
* Tokenized text and removed stopwords.
* Applied TF-IDF vectorization for feature extraction.

# Sample Output After Preprocessing

print(df[['Text', 'Processed\_Text', 'Category']].head())

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

**Model Implementation:**

* Split data into training and test sets.
* Used TfidfVectorizer for text feature representation.
* Trained a **Multinomial Naive Bayes** classifier using scikit-learn.

**Evaluation Metrics:**

* **Accuracy:** 80.50%
* **Classification Report:** Showed good balance across precision, recall, and F1-score.

**Challenges Encountered:**

* **Text Complexity:** Informal language and varied writing styles posed modeling challenges.
* **Class Imbalance:** Some categories had fewer examples, affecting predictive accuracy.
* **Naive Bayes Assumptions:** Independence assumption doesn’t hold in natural language but still performed well.
* **Preprocessing Sensitivity:** Results varied based on stopword list and tokenization choices.

**3. Sentiment Analysis**

**Method Used:**

* Employed the **VADER** sentiment analysis tool.
* Assigned each post a sentiment label: **Positive, Negative, or Neutral**.

**Category-wise Sentiment Distribution:**

print(sentiment\_by\_category)

**Insights:**

* Some categories had higher positive sentiment — indicating well-received content.
* Others leaned more negative — possibly related to sensitive or controversial discussions.
* Many posts fell under **neutral**, especially when content was factual or technical.

⚠️ Note: VADER is optimized for social media and may interpret formal blog content as neutral more often.

**4. Conclusion**

This project successfully demonstrated:

* Building a **Naive Bayes text classifier** using TF-IDF.
* Performing **sentiment analysis** on blog posts.
* Evaluating both model performance and content tone.

**Key Takeaway:** Even with limitations, Naive Bayes can effectively classify text data, and sentiment analysis reveals valuable thematic insights.

**Appendix: Tools & Libraries Used**

* pandas
* scikit-learn
* nltk
* matplotlib / seaborn (optional for visualization)
* VADER for sentiment analysis