# Documentation of Sentiment Analysis Model Development

---

## Process and Method Overview:

### 1. Data Loading:

- Loaded the provided dataset, which contained user conversation posts labeled with three sentiment categories: positive, negative, and neutral.
- Checking the dataset info and missing values.

### 2. Data Preprocessing:

- **Text Cleaning:**
  - Removed unnecessary text elements such as punctuation, special characters, etc.. Converted text to lowercase to standardize the input.
- **Stopword Removal:**
  - Removed Bengali stopwords using an NLTK Bengali stopword list.

### 3. Feature Engineering:

- **TF-IDF (Term Frequency-Inverse Document Frequency):**
  - This technique was applied to convert the text data into a numerical format that reflects the importance of words in the dataset. It allows the machine learning model to understand the textual data numerically.

### 4. Exploratory Data Analysis (EDA):

- **Word Cloud:**
  - Generated word clouds to visualize the most frequently occurring words across different sentiment classes, helping identify common themes or topics in positive, negative, and neutral posts.
- **Bar Plot & Histogram:**
  - Bar plotted the distribution of sentiment classes in the dataset to observe class imbalance, where the "neutral" class was dominant. Histogram provides a better understanding of the relationship between sentiment and review conversation text length

### 5. Handling Imbalanced Data:

- **SMOTE (Synthetic Minority Over-sampling Technique):**

    - This technique helped balance the dataset and improve model performance.

**6. Model Development:**

- **Logistic Regression:** A statistical model used for binary and multiclass classification.
- **Naive Bayes:** A probabilistic classifier based on Bayes' theorem with strong independence assumptions.
- **Llama-3.1-8B:** Llama-3.1 8b and Unsloth 2x faster finetuning

**7. Model Evaluation:**

- **Metrics Used:**
    - Accuracy: To measure the overall correctness of the model.
    - Precision, Recall, F1-Score: These metrics were used to assess the balance between precision and recall, especially in light of class imbalance.
    - Confusion Matrix: A table used to evaluate the performance of classification models.

---

## Challenges Faced:

**1. Preprocessing Bengali Text:**

- Removing Bengali stopwords was challenging due to limited support in NLP libraries.

**2. Class Imbalance:**

- The dataset was small, containing only 99 rows and highly imbalanced, with a larger proportion of neutral sentiments. This imbalance negatively affected model performance until SMOTE was applied to oversample the minority classes.

**3. Bangla Font Issue in Word Cloud Visualization:**

- While generating word clouds, Bengali text was not displayed correctly due to font issues. To overcome this, the **Kalpurush** font was downloaded and applied to properly render the Bengali characters in the visualization.

**4. Low Accuracy:**

- Initial models (logistic regression and Naive Bayes) yielded low accuracy (60% and 65%), indicating room for improvement in feature extraction, model tuning, or data augmentation.

**5. Fine-Tuning LLaMA:**

- **Model Size**: LLaMA is a large model, which can easily exceed available GPU or CPU memory, leading to OOM errors, especially on consumer-grade hardware.
- **Batch Size**: When training large models, batch sizes often need to be reduced to prevent memory overflow. LLaMA's attention layers require substantial memory.
- **Gradient Accumulation**: Large models like LLaMA struggle with memory management, especially when trained with large datasets or longer sequences.

---

## Performance Analysis:

- **Baseline Model (Logistic Regression, Naive Bayes):**
  - Accuracy: Logistic Regression 60%, Naive Bayes 65%
  - Precision, Recall, and F1-Score indicated the model was struggling to differentiate between neutral and other classes.
- **SMOTE + Logistic Regression, Naive Bayes:**
  - Naive Bayes handles the class imbalance effectively after applying SMOTE, delivering better performance 80% accuracy for underrepresented classes like neutral sentiment.
  - Logistic regression improved slightly with SMOTE but did not perform as well as Naive Bayes due to its difficulty in predicting the neutral sentiment class
- **Llama-3.1-8B:**
  - The evaluation loss of 0.818 indicates how well the model performed on the validation set. A lower loss value suggests better model performance. Compared to the training loss (0.948), this lower evaluation loss signifies that the model generalized well to the validation data.
  - **Evaluation Metrics:**
    - **Evaluation Loss:** 0.818 (lower is better, indicating good generalization)
    - **Runtime:** 10.2 seconds
    - **Samples Per Second:** 1.96
    - **Steps Per Second:** 0.295

- **Training Metrics:**
    - **Training Loss:** 0.948
    - **Runtime:** 163.3 seconds
    - **Samples Per Second:** 0.735
    - **Steps Per Second:** 0.367

---

## Recommendations for Improvement:

1. **Data Augmentation:**
    - Collecting more labeled data would provide the model with more training examples, especially for underrepresented sentiment classes. This could significantly improve classification performance.
2. **Experiment with Deep Learning Models:**
    - Replacing traditional ML models with deep learning techniques like LSTM, GRU, or transformers would allow for capturing the temporal and semantic dependencies in the text, leading to better performance.
3. **Hyperparameter Tuning:**
    - Exploring a wider range of hyperparameters (e.g., regularization strength, learning rate) through grid search or random search could enhance model performance.
4. **Cross-Validation:**
    - Implementing cross-validation to ensure that the model generalizes well across different subsets of the data and is not overfitting to the training data.

---

# How to Run the Project

---

### 1. Download the Notebook Files

- **Download from the Drive Link:**
    - Obtain the notebook files from the provided Google Drive link and save them to your local machine.

### 2. Upload the Notebook Files to Google Colab

- **Open Google Colab:**
    - Go to [Google Colab](Google Colab).
- **Upload the Notebook Files:**
    - Click on the **"File"** menu.
    - Select **"Upload notebook"**.
    - Click on **"Choose File"** and select the notebook file you downloaded.
    - Repeat this step if you have multiple notebook files to upload.
- **Upload Dataset and Bangla Font Files:**
    - Click on the **"Files"** icon (folder icon) on the left sidebar.
    - Click on **"Upload"** and select your dataset files and Bangla font file from your local machine.
    - Once uploaded, note the file paths. You will need to adjust these paths in your notebook to match the uploaded files.
- **Change Paths in the Notebook:**
    - Open the uploaded notebook file(s).
    - Locate the sections where the dataset and Bangla font file paths are specified.
    - Update these paths to match the location of the files you just uploaded.

### 3. Switch the Runtime to GPU for Fine-Tuning LLaMA 3.1 8B Notebook

- **Change Runtime Type:**
    - In your Colab notebook, click on the **"Runtime"** menu.
    - Select **"Change runtime type"**.
    - In the pop-up window, select **"GPU"** from the **"Hardware accelerator"** dropdown menu.
    - Click **"Save"**.

### 4. Run All Cells

- **Run Cells Sequentially:**

- Click on **"Runtime"** in the top menu.
- Select **"Run all"** from the dropdown. This will execute all cells in the notebook sequentially.