## Biggest Challenges and How I Overcame Them

I encountered several major challenges in that particular project, and here's how I overcame each of them:

### 1. Data Preprocessing for Bengali Text

- **Challenge**: Preprocessing Bengali text for sentiment analysis was quite challenging. Unlike English, Bengali requires specialized tools for tokenization, stemming, and cleaning. Managing issues like Unicode normalization and different types of punctuation also added complexity.
- **How I Overcame It**:
  - I used **BNLP's CleanText** for cleaning and **NLTKTokenizer** for tokenization to ensure the text was properly preprocessed.
  - I implemented regular expressions (`re.sub`) to handle multiple punctuation marks and remove unwanted phrases, ensuring consistent text formatting.
  - To further clean the text, I removed unnecessary stopwords and handled Bengali-specific quirks with custom processing rules.

### 2. Handling Multilingual or Non-English Text

- **Challenge**: Dealing with Bengali text in an NLP task presented a challenge, as it doesn't have the same readily available tools as English. The model had to understand context in a language with different grammar and characters.
- **How I Overcame It**:
  - I utilized the **BNLP toolkit**, which provided the necessary tools for text processing like cleaning, tokenization, and handling the Bengali script.
  - I customized the text preprocessing pipeline to account for unique elements of Bengali, such as handling punctuation ( ) and other script-specific nuances.

### 3. Feature Engineering and Dimensionality Reduction

- **Challenge**: Transforming Bengali text into a numerical format for machine learning models was another challenge. It was crucial to select the right method for feature extraction to capture essential information for sentiment analysis.
- **How I Overcame It**:
  - I applied **TF-IDF Vectorization**, which helped capture the significance of terms based on their frequency while discounting overly

common terms. This allowed me to represent the text effectively for machine learning models.
  - I calculated the **vocabulary size** to tune the LSTM model's `max_features` for better performance.

4. **Model Selection and Hyperparameter Tuning**

- **Challenge**: Choosing appropriate machine learning models and tuning their hyperparameters was tricky, especially with textual data. From traditional classifiers to more advanced techniques like LSTM, each model required thoughtful tuning.
- **How I Overcame It**:
  - I started with traditional classifiers such as **Logistic Regression** and **Random Forest** and later experimented with advanced models like **XGBoost**, **LightGBM**, and **LSTM** to handle more complex patterns in the data.
  - For hyperparameter tuning, I utilized **GridSearchCV** and tested different configurations, such as varying layers and batch sizes for models like LSTM.

5. **Fine-Tuning Large Language Models (LLMs)**

- **Challenge**: Fine-tuning large transformer models like **Llama 3.1** for Bengali sentiment analysis was resource-intensive and required careful tuning of training arguments to avoid overloading the system.
- **How I Overcame It**:
  - I used **HuggingFace's Trainer API** for fine-tuning the **Llama 3.1 model** and set parameters like **batch size**, **gradient accumulation**, and **mixed precision training (fp16)** to balance efficiency and performance.
  - To ensure the model could handle Bengali text, I used **AutoTokenizer** and configured the tokenization settings, such as padding and truncation, to avoid issues with sequence lengths.

6. **Handling Class Imbalance and Data Splitting**

- **Challenge**: The dataset likely had imbalanced sentiment classes, which could skew the model's performance. It was crucial to ensure that all sentiment classes were equally represented during training and evaluation.
- **How I Overcame It**:
  - I used **Stratified K-Fold Cross-Validation** to ensure that the training and test sets had a balanced distribution of sentiment labels.

  - During model evaluation, I monitored performance across each sentiment class, not just the overall accuracy, to ensure fair representation.

**7. Using Llama for Non-English Fine-Tuning**

- **Challenge**: Fine-tuning **Llama 3.1**, a large-scale language model, presented challenges such as managing computation, handling Bengali tokenization, and choosing the right training settings.
- **How I Overcame It**:
  - I used **AutoModelForSequenceClassification** and fine-tuned the pre-trained model by creating a custom training loop with **HuggingFace's Trainer**.
  - I set **TrainingArguments** to manage hardware constraints by using techniques like gradient accumulation, forcing CPU training (`no_cuda=True`), and enabling mixed precision (`fp16`) to optimize for available resources.

---

**Summary of Overcoming Challenges**

- **Leveraging Pre-built Libraries**: To overcome many challenges, I made use of specialized libraries such as **BNLP** for Bengali text preprocessing and **HuggingFace** for fine-tuning transformer models.
- **Custom Processing**: Where pre-built libraries fell short, I created custom cleaning and tokenization processes, especially for handling Bengali punctuation and grammar.
- **Efficient Resource Management**: I adapted my training strategies to work within hardware constraints, using techniques like gradient accumulation and mixed precision to make training more efficient.
- **Model Selection and Hyperparameter Tuning**: I experimented with both traditional machine learning models and state-of-the-art transformer models, carefully tuning each to achieve optimal performance.