Documentation for Empathetic Sentiment Analysis Using TinyLlama-1.1B

1. Approach:

The goal of this project was to build an empathetic sentiment analysis model using the **TinyLlama-1.1B-Chat-v1.0** model. The focus was to analyze empathetic dialogues and their relationship to emotions expressed in conversations. The process involved data preparation, model fine-tuning, and performance evaluation, all structured in a well-defined project folder.

2. Data Preparation Techniques:

1. **Dataset:**
    - The dataset was sourced from the Kaggle dataset **atharvjairath/empathetic-dialogues-facebook-ai**, containing **64,636 rows** and **7 features**, which included:
        - situation, emotion, empathetic_dialogues, labels, and 3 unnamed columns.
2. **Data Cleaning:**
    - **Missing values** were handled by replacing them with the mode of the respective column and dropping irrelevant rows.
    - **Duplicates** were removed to avoid overfitting.
    - **Text Standardization:** Removed unwanted characters (e.g., "customer:") and made the text lowercase to maintain consistency.
    - Created a new feature, text_length, to assess any correlation between the length of situation and the associated emotion.
3. **Exploratory Data Analysis (EDA):**
    - **Emotion Distribution:** Visualized using countplots to understand the frequency of each emotion.
    - **Text Length Distribution:** Visualized using histograms to determine the consistency of text lengths across emotions.
    - **Word Cloud Generation:** Created word clouds for situation and empathetic_dialogues to analyze commonly used words.
    - The data was then split into **training** and **testing** datasets, with cleaned and renamed features for convenience.

3. Model Choices:

1. **TinyLlama-1.1B-Chat-v1.0:**
    - Chosen for its efficiency, as it is lightweight and easy to fine-tune even with limited GPU resources.
    - The model is based on causal language modeling, suitable for dialogue generation and empathy-based sentiment analysis.
2. **Fine-Tuning with BitsAndBytes and PEFT:**
    - **BitsAndBytes Configuration:** This approach enables the model to be loaded in **4-bit precision**, reducing memory consumption significantly while maintaining efficiency in fine-tuning.
    - **PEFT (Performance-Efficient Fine-Tuning):** Specifically, **LoRA** (Low-Rank Adaptation) was used, updating only certain layers (causal_lm) while freezing the rest of the model's weights to make training more efficient and scalable.

- The fine-tuning was performed using **fp16 precision** for speed optimization.

3. **Trainer Setup:**
    - Utilized the **Supervised Fine-Tuning (SFT) Trainer** for model training, using the prepared dataset.
    - Fine-tuning was done with **AdamW optimizer**, **learning rate scheduler**, and a total of **2 epochs** for better generalization.

4. Model Folder Structure:

1. **src Folder:**
    - **Common Scripts:** Contains exception_handler.py, logger.py, and utility functions for logging and handling errors.
2. **Components Folder:**
    - **dataingestion.py:** Converts the pre-processed CSV into a Hugging Face dataset format and splits it into training and evaluation datasets.
    - **llm_trainer.py:** Responsible for loading and fine-tuning the TinyLlama model. Utilizes the **BitsAndBytes configuration** and **PEFT** for efficient training.
3. **Pipeline Folder:**
    - **training_main.py:** Manages the training pipeline, integrating the trainer and data preprocessing.
    - **model_evaluator.py:** Handles evaluation of the fine-tuned model using the test dataset. Calculates accuracy, precision, recall, F1-score, and generates a confusion matrix. It also includes error analysis, displaying the true and predicted labels for each text.
    - **inference.py:** Provides an interface for generating conversational responses from the fine-tuned model.

5. Challenges Faced:

- **Handling Imbalanced Data:** Emotions were not uniformly distributed, leading to some classes being underrepresented. To handle this, data augmentation techniques could be implemented in future iterations.
- **Memory Constraints:** Fine-tuning large models with limited GPU resources was challenging. Using 4-bit precision and PEFT (LoRA) helped mitigate these issues.
- **Evaluation Metrics:** The evaluation process was hindered by the high class imbalance, and further optimization of evaluation strategies (e.g., using weighted metrics) could improve results.

6. Results:

- The model achieved promising results in predicting the **emotion** based on dialogues, with good precision, recall, and F1-scores.
- Confusion matrices showed clear distinctions between major emotions but highlighted some difficulty in predicting more nuanced emotional expressions.

7. Reflection & Further Improvements:

1. **Improving Cultural Sensitivity:**
    - The model could be improved by incorporating a more diverse dataset that covers a broader range of cultural contexts and expressions. Current datasets may lack the nuance needed to handle diverse emotional expressions across different cultures.

2. **Fine-Tuning Techniques:**
    - More advanced techniques, such as **prompt-tuning** or using domain-specific pre-trained models, could improve the model's ability to detect nuanced emotions better.
3. **Enhanced Evaluation Metrics**