Report: IndoML Datathon 2024 Phase 1 - Team DUCS

1. Team Information

- **Team Name**: DUCS
- **Team Members**:
    - Harsh Yadav [harshyamsc23@cs.du.ac.in]

2. Detailed Approach for Solving the Problem

a. Model Architecture and Details

- **Model**: T5 (Text-to-Text Transfer Transformer)
- **Architecture**:
    - **Tokenizer**: "T5Tokenizer" from the "transformers" library.
    - **Model**: "T5ForConditionalGeneration" from the "transformers" library.
    - **Configuration**:
        - **Pre-trained Model**: "t5-small'
        - **Input Format**: Texts formatted as "title: {title} store: {store} details_Manufacturer: {details_Manufacturer}"
        - **Output Format:** Texts formatted as "details_Brand: {details_Brand} L0_category: {L0_category} L1_category: {L1_category} L2_category: {L2_category} L3_category: {L3_category} L4_category: {L4_category}"
- **Training Arguments:**
    - **Learning Rate**: "2e-3'
    - **Batch Size**: "500'
    - **Epochs**: "50'
    - **Weight Decay**: "0.01'
    - **Evaluation Strategy:** "epoch'
    - **Logging Steps:** "20'

b. Reason Behind Model Selection

- **Choice**: The T5 model was chosen for its capability in sequence-to-sequence tasks where both input and output are text-based. Its architecture allows for the effective handling of various text generation tasks, including conditional generation, which aligns well with the problem of predicting product details based on input attributes.

c. Techniques to Improve Scores

- Techniques:
    - **Data Preprocessing:**
        - Concatenated multiple features into a single input string and similarly formatted the target text to include all relevant categories.
    - **Data Augmentation**: Used standard tokenization and padding strategies to ensure consistent input length.
    - **Custom Callback:** Implemented a callback to print training progress and metrics at each step, which helps monitor the training process in real time.

- **Fine-Tuning:** Utilized pre-trained T5 weights to accelerate convergence and improve the model's performance.

3. Setup for Reproducing Results

a. Specific Setup

- **Folder Structure:**
  - "data/" - Contains raw data files
  - "results/" - Output from model training
  - "logs/" - Training logs
- **Dependencies:**
  - "transformers" library
  - "datasets" library
  - "torch" library
  - "pandas" library
  - "numpy" library
  - "tqdm" library
- **Instructions to run the code:**
  - Start Notebook and run all the cells sequentially.

b. Computational Setup

Hardware:

- **GPU**: NVIDIA A100 80GB
- **PROCESSOR**: Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz 2.89 GHz (2 processors)
- **RAM**: 512GB

4. Training Log

a. Training Data

- **Training Size**: 4,43,499 samples
- **Validation Size**: 95,035 samples

b. Hyperparameters

- **Learning Rate**: 2e-3
- **Batch Size**: 500
- **Epochs**: 50

Additional Details

- Inference and Evaluation:
  - The fine-tuned model was used to generate predictions for the test dataset.
  - Extracted generated details and formatted them according to required categories.
  - Results were saved as a JSONL file and zipped for submission.

- Final Submission:
  - Prediction File: "full_50epochs.zip" containing "full_50epochs.predict'