# Comparative Analysis: Sentiment Analysis Using BERT, LSTM, GRU, and RNN

## Objective

Perform sentiment analysis on the given dataset using multiple deep learning models: **BERT**, **LSTM**, **GRU**, and **RNN**, and conduct a comparative analysis based on their performance metrics.

[Dataset]

## Implementation Plan

### 1. Data Preprocessing

- **Load the Dataset:** Import the CSV file and extract the relevant columns (`polarity`, `text`).
- **Clean the Text:** Remove URLs, special characters, numbers, and extra spaces. Convert text to lowercase.
- **Tokenization:** Split the text into individual tokens using a tokenizer suitable for each model (e.g., Word2Vec for RNN-based models, BERT tokenizer for BERT).
- **Class Mapping:** Map sentiment labels:
  - `0` → Negative
  - `2` → Neutral
  - `4` → Positive
- **Train-Test Split:** Divide the data into training, validation, and test sets.

### 2. Feature Engineering

- **BERT Tokenization:**

    - Use a pre-trained BERT tokenizer to convert the text into input IDs, attention masks, and token type IDs.
- **Embedding for RNN-based Models:**
    - Generate word embeddings using GloVe or Word2Vec.
    - Pad sequences to a fixed length for uniformity.

**3. Model Implementation**

- **BERT:**
    - Use a pre-trained BERT model from the Hugging Face library.
    - Add a classification head (e.g., a dense layer with softmax activation) to fine-tune the model.
- **LSTM:**
    - Use a sequential model with embedding, LSTM layers, and a dense output layer.
- **GRU:**
    - Similar to LSTM but replace LSTM layers with GRU layers for comparison.
- **RNN:**
    - Use simple RNN layers instead of LSTM/GRU for baseline comparison.

**4. Evaluation Metrics**

- **Accuracy:** Overall percentage of correct predictions.
- **Precision, Recall, F1-Score:** Evaluate per class (negative, neutral, positive).
- **Confusion Matrix:** Show performance across all classes.
- **ROC-AUC Score:** Measure the ability of the model to distinguish between classes.

**5. Comparative Analysis**

- Compare the models on:
    - Performance metrics (accuracy, precision, recall, F1-score).
    - Computational requirements (training time, memory usage).
    - Complexity of implementation.
- Generate visualizations:
    - Bar chart comparing F1-scores for all models.

- Line plot showing training/validation loss and accuracy over epochs.
- Confusion matrix heatmaps for each model.

**6. Expected Outcome**

- **BERT:** Likely to outperform RNN-based models due to its pre-trained contextual embeddings and transformer architecture.
- **LSTM/GRU:** Expected to perform better than simple RNN due to their ability to handle long-term dependencies and avoid vanishing gradient problems.
- **RNN:** May provide a baseline but is likely to underperform compared to other models.

**7. Deliverables**

- Code implementation for each model in Python (using libraries like TensorFlow, PyTorch, Hugging Face).
- Comparative analysis report with:
  - Metric tables
  - Charts and graphs
  - Insights on model performance.
- Recommendations on the best model for deployment based on trade-offs between performance and resource usage.