Interview Task: Sentence Contradiction Classification

Goal:

Classify pairs of sentences as "contradiction," "entailment," or "neutral" based on their meaning. The task requires building a model that can understand semantic relationships between text pairs.

Dataset Information:

Data Files:

- **train.csv** (Labeled Training Data)
  - id: Unique identifier for each sentence pair.
  - sentence1: The first sentence in the pair (Premise).
  - sentence2: The second sentence in the pair (Hypothesis).
  - label: The relationship between the two sentences:
    - **0** = Contradiction (Sentences have opposite meanings)
    - **1** = Neutral (Sentences are related but do not imply each other)
    - **2** = Entailment (One sentence logically follows from the other)
- **test.csv** (Unlabeled data for prediction)

Task Overview:

Step 1: Exploratory Data Analysis (EDA)

Objective: Analyze the dataset to understand class distribution and text patterns.

Tasks:

- Visualize the distribution of Contradiction, Entailment, and Neutral labels.
- Analyze sentence structure (length, word distribution, common words).
- Check for missing values or outliers.

Step 2: Text Preprocessing

Objective: Clean and transform text for model training.

Tasks:

- Tokenization: Split sentences into words.
- Lowercasing: Convert text to lowercase.
- Remove stop words, special characters, and punctuation.
- Stemming/Lemmatization: Normalize words to their root form.

- Feature Extraction: Convert text into numeric representations using **TF-IDF, Word2Vec, or Transformer embeddings** (e.g., BERT, XLM-R).

Step 3: Model Creation

Objective: Train a machine learning model to classify sentence relationships.

Tasks:

- **Baseline Model:** Try Logistic Regression, Random Forest, Decision trees, XGB.
- **Neural Networks:** Implement a Custom Artificial Neural Network (ANN).
- **Advanced Models:** Train LSTM/GRU models for sequence-based learning.
- **Transformer-Based Models:** Fine-tune BERT/XLM-R for contextual understanding.

Step 4: Model Evaluation

Objective: Measure model performance using classification metrics.

Tasks:

- Compute **accuracy, precision, recall, and F1-score**.
- Plot a **Confusion Matrix** to analyze misclassifications.
- Generate an **AUC-ROC curve** to evaluate classification performance.

Step 5: Model Tuning and Optimization

Objective: Improve model performance through tuning.

Tasks:

- Experiment with different **optimizers (Adam, SGD, etc.)** and activation functions.
- Adjust **learning rate, batch size, and number of epochs**.
- Use **Grid Search or Random Search** for hyperparameter tuning.

Model Evaluation Criteria:

- **EDA Quality** – Depth of dataset understanding.
- **Text Preprocessing** – Effectiveness of data cleaning techniques.
- **Model Choice** – Selection of a suitable architecture.
- **Performance Metrics** – Classification results.
- **Code Quality** – Clarity, efficiency, and modularity.
- **Justification** – Explanation of modeling choices.

Expected Deliverables:

1. Code Implementation

Submit a **Jupyter Notebook (.ipynb)** or **Google Colab** file containing:

- **EDA (with visualizations)**
- **Text preprocessing pipeline**
- **Model training and evaluation**
- **Model tuning (if applicable)**

2. Evaluation Metrics

- **Report accuracy, precision, recall, F1-score, and AUC-ROC.**
- **Include a Confusion Matrix.**

3. Submission Requirements

- **Time Limit:** 3 Hours
- **Format:**
    - Code file (.ipynb or .py)
    - Performance evaluation report