Task: Question Pair Similarity Classification

Goal:

Classify pairs of questions as either "duplicate" (i.e., semantically similar) or "not duplicate" based on the provided dataset.

Dataset Information:

- **train.csv**: Contains labeled question pairs from Quora.
    - **id**: Unique identifier for each question pair.
    - **qid1**: Unique identifier for the first question in the pair.
    - **qid2**: Unique identifier for the second question in the pair.
    - **question1**: Full text of the first question.
    - **question2**: Full text of the second question.
    - **is_duplicate**: The target variable. It is set to 1 if the two questions are semantically similar (duplicate), and 0 otherwise.
- **Test Set**: Contains question pairs that you need to predict whether they are duplicates or not.

Task Overview:

1. **Step 1: Exploratory Data Analysis (EDA)**
    - **Objective**: Explore the dataset to understand the distribution of the target variable (`is_duplicate`), relationships between features (question text), and any potential patterns in the data.
    - **Tasks**:
        - Visualize the distribution of `is_duplicate`.
        - Analyze the text columns (`question1` and `question2`) using word clouds, sentence length, common words, etc.
        - Check for missing values or outliers.
        - Investigate if there are any significant correlations between features like question text length, character count, etc.
2. **Step 2: Text Preprocessing**
    - **Objective**: Preprocess the text data to prepare it for machine learning. This will involve cleaning and transforming the questions into a suitable format for model training.
    - **Tasks**:
        - Tokenization: Split the questions into words.
        - Lowercasing: Convert all text to lowercase to standardize.

        - Removing stopwords, special characters, and unnecessary punctuation.
        - Stemming or Lemmatization: Normalize words to their root form.
        - Feature Extraction: Use methods like **TF-IDF (Term Frequency-Inverse Document Frequency)** or **Word Embeddings** (Word2Vec, GloVe) to transform the questions into numerical representations.
3. **Step 3: Model Creation**

    - **Objective**: Build a model to classify whether two questions are duplicates or not. You are encouraged to try multiple models, but the core requirement is to create a **Neural Network (ANN)** model using Keras or TensorFlow for text similarity classification.
    - **Tasks**:
        - Use a **Custom Artificial Neural Network (ANN)** or pre-built models like **Siamese Networks** (two neural networks comparing the two question pairs) for semantic similarity.
        - Use **LSTM** (Long Short-Term Memory) or **GRU** (Gated Recurrent Units) networks, as they are effective for sequence-based data like text.
        - Alternatively, you can experiment with a simpler **Logistic Regression** or **Support Vector Machine (SVM)** as a baseline model.
        - If desired, try transfer learning using pre-trained models like **BERT** or **DistilBERT**.
4. **Step 4: Model Evaluation**

    - **Objective**: Evaluate the performance of your model on a validation/test set.
    - **Tasks**:
        - Evaluate the model using common classification metrics: **accuracy**, **precision**, **recall**, **F1-score**.
        - Plot a **Confusion Matrix** to understand the number of true positives, true negatives, false positives, and false negatives.
        - **AUC-ROC** curve to visualize the tradeoff between true positive rate and false positive rate.
5. **Step 5: Model Tuning and Hyperparameter Optimization**

    - **Objective**: Tune your model for better performance.
    - **Tasks**:
        - Experiment with different architectures, activation functions (ReLU, sigmoid, etc.), and optimizers (Adam, SGD, etc.).
        - Try hyperparameter optimization (e.g., learning rate, batch size, number of epochs) using **Grid Search** or **Random Search**.

Model Evaluation Criteria:

- **Exploratory Data Analysis**: The depth of analysis in understanding the data.
- **Text Preprocessing**: The quality of your text preprocessing pipeline.

- **Model Choice**: The suitability of the model for the task, including the use of advanced models.
- **Performance Metrics**: Achieved accuracy and classification metrics (precision, recall, F1-score).
- **Code Quality**: The clarity, efficiency, and modularity of your code.
- **Justification of Choices**: Provide reasons behind your choice of model, evaluation metric, and tuning steps.

Expected Deliverables:

1. **Code Implementation**:
    - Jupyter notebook or Google Colab with your complete code, including the following sections:
        - EDA (with visualizations)
        - Text preprocessing pipeline
        - Model building and training
        - Evaluation of model performance
        - Model tuning (if applicable)
    - Comments and markdown cells explaining your steps and thought process.
2. **Evaluation Metrics**:
    - Report the performance of your model using accuracy, precision, recall, F1-score, and AUC-ROC.
    - Provide a Confusion Matrix to demonstrate the classification results.
3. **Submission**:
    - Upload the Jupyter notebook or Colab file.
    - Include any additional resources or explanations you think are necessary to understand your approach.

Submission Requirements:

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