(1) Introduction

This task aims to develop and assess an NLP system that can identify semantic similarity between question pairs. The report details the complete development workflow, encompassing Exploratory Data Analysis, Text Preprocessing, Model Creation, Model Evaluation and Model Tuning and Hyperparameter Optimization.

(2) Environment Settings

- **Platform:** Kaggle platform is used for the assigned task.
- **Dataset:** Provided dataset train.csv and additional dataset glove.6B.100d.txt is used for embeddings.
- **Keras Conflict:** To avoid keras conflict with Kaggle system install the intended version (Provided in the python code)

(3) Exploratory Data Analysis

The following process are followed for Data Analysis:

- **Class Distribution:** The distribution of the target variable was examined to determine whether there was a class imbalance.
- **Word Cloud:** Word cloud is used to visualize the most frequent words in a text, making dominant topics easy to spot and also helps with quick exploratory analysis, revealing key terms, noise, and patterns in the dataset.
- **Sentence Length Analysis:** Helps to understand the complexity and structure of text. It also helps detect outliers, noise, or unusual patterns that may affect model performance.
- **Common Words Analysis:** Helps to identify frequent terms, stopwords, and dominant themes that influence text patterns.
- **Missing Value Analysis:** Detects gaps or incomplete entries so we can clean or impute them to avoid biased or unreliable NLP models.
- **Correlation Analysis:** Helps to measure the strength and direction of relationships between variables in the dataset. It also identify how features influence each.

(4) Text Preprocessing

Each input sequence underwent the following transformations:

- **Lowercasing:** Standardizes the text by converting all characters to lowercase.
- **Special Characters, Unnecessary Punctuation Removal:** Cleans the text by removing special characters and unnecessary punctuation.
- **Extra Spaces Removal:** Standardizes spacing by removing extra spaces between words.
- **Tokenization:** Segments the text into individual tokens or words for further processing.
- **Removing Stopwords:** Eliminates common stopwords to retain only meaningful words in the text.
- **Lemitization:** Reduces words to their base or root form to standardize the text.
- **Feature Extraction:** Transforms the processed text into numerical features suitable for model input. Following two method are used.

    - **TF-IDF:** Represents text by weighting terms based on their frequency and importance across the corpus**.**
    - **Glove Embeddings**: Maps words to dense vector representations that capture their semantic meaning.

(5) Model Creation

The following models are implemented:

- **Baseline Model:** Logistic Regression is used as a benchmark model by utilizing the Hybrid (TF-IDF + Glove Embedding) features. It is a statistical model that predicts the probability of a binary outcome using a logistic (sigmoid) function.

- **Deep Learning Model:** Siamese + LSTM Model is implemented. A Siamese LSTM model learns to measure similarity between two sequences by encoding them with shared LSTM layers and comparing their representations.

- **Artificial Neural Network (ANN):** ANN is also implemented. An ANN is a computational model inspired by the human brain, consisting of interconnected layers of nodes (neurons) that learn to map inputs to outputs by adjusting connection weights.

- **Transformer Learning Model:**

    I. **BERT**: BERT is a deep learning model that uses a transformer architecture to generate contextualized word embeddings by reading text bidirectionally, capturing meaning from both left and right context.
    II. **DistilBERT**: DistilBERT is a smaller, faster, and lighter version of BERT that retains most of BERT's language understanding by using knowledge distillation, making it more efficient for practical applications.

(6) Model Evaluation

- Each model is evaluated thorough classification metrics (Accuracy, precision, recall, F1 score), Confusion matrix and AUC-ROC curve (Provied in the python code)
- Model's epoch is reduced to save time

(7) Model Tuning and Hyperparameter Optimization

i. Different architectures, activation functions (sigmoid), and optimizers (Adam) are used.
ii. Learning rate, batch size, number of epochs can be changeable in the code to achieve better result

(8) Conclusion

Successfully developed and validated a robust NLP framework capable of recognizing duplicate questions with high accuracy.