Review Summarization :

1. Collected the dataset from the site given in the question.
2. Imported all the packages needed to preprocess reviews like nltk, numpy and pd etc.
3. Lowered texts.
4. Tokenized texts.
5. Removed stop words.
6. Corrected the spacing for the output generated and then removed punctuation.
7. Finally, performed stemming and lemmatization.
8. I have applied the preprocessing to Text and Summary columns.
9. Then saved the result to a new csv file '`final_csv_after_preprocessing.csv`'.
10. Then took the first 6667 rows of the dataset as the training was taking a lot of time.
11. Splitted the data into train and test set and saved both dataframes to csv files train.csv and test.csv respectively.

Custom dataset class:

- SummaryDataset()
    - I have initialized the block size.
    - I have initialized a data frame after conversion from csv file and then two other data frames for Text and Summary respectively.
    - Initialized a tokenizer.
    - Defined **len** function.
    - Defined __get__ function.
        - In which I have first converted a specific row data into encodings.
        - Then returned the Input IDs and attention masks and the embedding of respective labels.
- Training of Model.
    - Fine tuned the model on the food review dataset.
    - In the train function:
        - I have loaded the model and the tokenizer and The DataCollator as there might be a chance that inputs are not padded to the same length, so to pad them we need a collator.
        - Then passed the training arguments like number of epoch, batch size and at which checkpoint should we save the model, etc.
        - Trained the model and saved the trained model in `'final_model_aditya' directory` as this directory changes as we tune the model.
- Load and Generation of output
    - Then I loaded the saved model and the tokenizer and created a new function named "generate_text" and then generated the output from the saved model using generated ids from tokenizer.
    - I have formed a list '`summaries_and_generated`' in which I am saving the test label given and their corresponding generated summary.
- Evaluation.

- Calculated the Rouge score of unigram, bigram and longest common subsequence(for entire testing set).

Hyper Parameter Tuning:

- Code flow is the same.
- Model and loading and generation of text is the same.
- 2nd model I have changed the block size to 512 and the number of epochs to 3.
- 3rd model I have again changed the block size to 128 and increased the number of epochs to 8.

Result Analysis:

Result for model 3 was the best. The reason I think is because of increasing the number of epochs the model ran for more number of times so the loss decreased, hence outputting the best result.

Result 3 > Result 2 > Result 1