HOMEWORK 2

Description of Tasks

1.

C) I have implemented 3 distinct methods for generating features from the given textual data for modelling.

1. **CountVectorizer –** Implemented to convert the collection of text documents into a matrix of token counts. This involves Tokenization, Vocabulary Building, Count Encoding.
This approach emphasizes the frequency of words within the news document but ignores the relative importance of words across the entire corpus.

2. **TFIDF Vectorizer (Term Frequency-Inverse Document Frequency) -** TFIDF offers a more nuanced representation by considering not just the occurrence of words in the single document but across all documents. It involves 2 steps - Term Frequency (TF), Inverse Document Frequency (IDF).
TFIDF scores are calculated by multiplying TF by IDF, resulting in features that reflect not only the presence of words but their significance in the context of the entire dataset.

3. GloVe (Global Vectors for Word Representation) - GloVe embeddings is implemented to provide a dense vector representation for words based on their co-occurrence information across the text corpus, capturing semantic and syntactic similarities between words.
The feature generation process using GloVe involves Loading Pre-trained Embeddings, Word Embedding Lookup and Averaging Word Embeddings.
This method generates features that capture deep linguistic and semantic information, making them powerful for many natural language processing tasks

In summary, these feature generation methods transform raw text into numerical formats suitable for machine learning models, each leveraging different aspects of the text data—from simple word counts to complex semantic representations.

2.

A) For the neural network model I have used the Multi-Layer Perceptron (MLP) classifier from scikit-learn for classification tasks on pre-processed training data.
Parameter Setting:
hidden_layer_sizes=(128, 128): This parameter defines the size and number of hidden layers in the neural network. In this case, the network is configured to have two hidden layers, each with 128 neurons. Each layer's size influences the model's capacity to learn from the data, with more neurons providing a higher capacity at the risk of overfitting if not properly regularized or if the training data is not sufficient. With two layers of 128 neurons each, the model is complex enough to capture underlying patterns and relationships in the data without being overly prone to overfitting

max_iter=300: This parameter specifies the maximum number of iterations (or epochs) over the entire dataset the solver will run during the model's training phase. This limit is set high enough to allow convergence to an optimal solution within reasonable computational time. If the model hasn't converged (i.e., continued to improve) by this

iteration, the training will stop. The 300 iterations is a balance to allow the algorithm enough time to adjust weights and biases to minimize the loss function, but not so long that training becomes computationally prohibitive

3.

A)

● The text data undergoes preprocessing, followed by feature generation primarily utilizing TF-IDF Vectorization.

● This method converts raw text into a matrix of TF-IDF features, highlighting significant words within a document compared to a corpus.

● The TfidfVectorizer from scikit-learn is trained on the training data to understand vocabulary and inverse document frequencies. It is subsequently employed on both training and test datasets to transform text into numerical features.

B)

● Opted for the count vectorizer model that produces favorable outcomes.

● Conducted experiments with different learning rates and optimizers to determine the optimal model

● The learning rates considered were: [0.0001, 0.0003, 0.003, 0.01, 0.03, 0.05, 0.1].

● Optimizers explored included: 'SGD', 'Adam', and 'RMSprop'.

● Found -0.03 LR to be giving optimal results and both RMSprop and Adam gives 97% accuracy

C)

● Opted for the Adam optimizer, which demonstrated strong performance, resulting in the highest accuracy of 98%.

● Configured a learning rate of 0.003 and activation functions as ReLu, with two hidden layers consisting of 128 cells each, and a Softmax layer with 5 cells, in addition to the input and output layers.