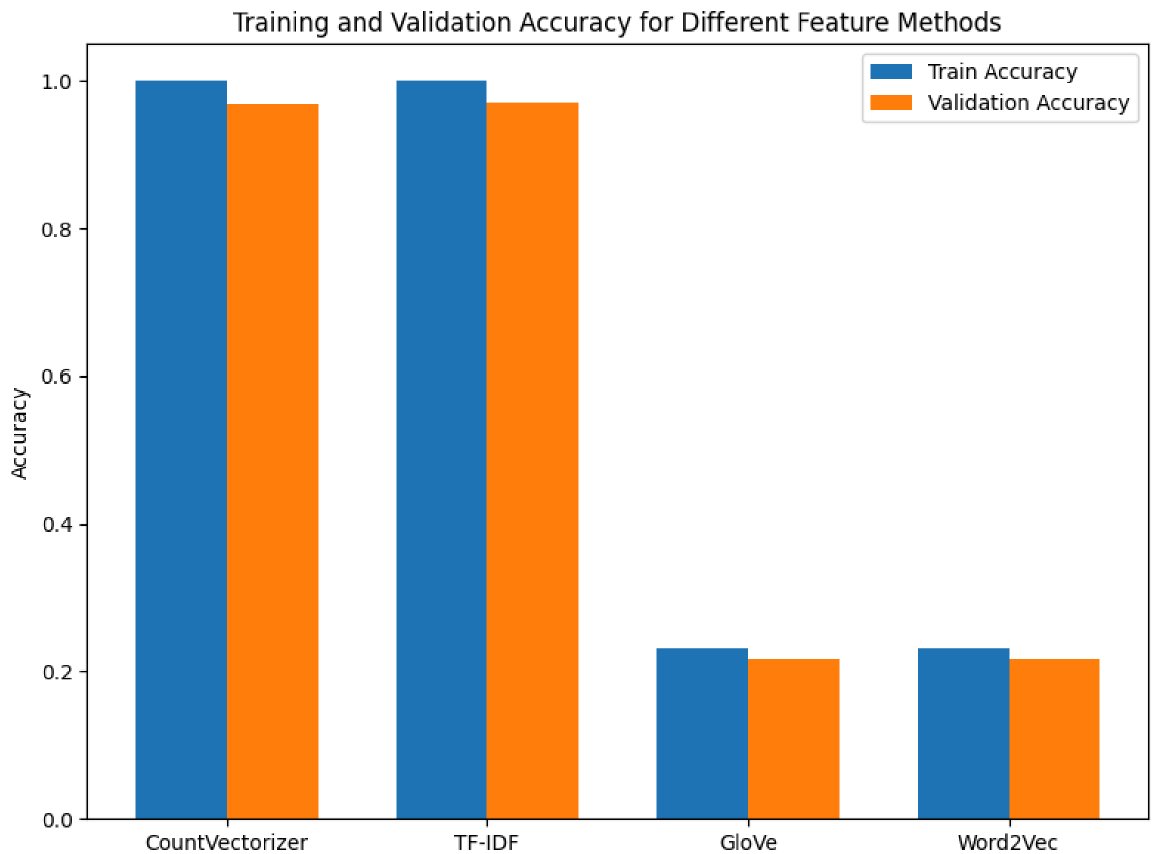
CSE 572: Data Mining (2024 Spring)
Homework 2

Prateek Mohan
pmohan9@asu.edu
ASU ID (emplid): 1225440970

Question 1:	**Preprocess the raw training data. You can use the code from Homework 1. You are required to construct other features, such as n-grams or keyword extractions. (15pt)**
a : Run Neural Networks with the 2-hidden layers, each has 128 neurons, extracting features by CountVectorizer() as the original features. Use 5-fold cross-validation to evaluate the performance.	This section preprocesses the raw text data by: 1. Tokenizing 2. Lowercasing 3. Removing punctuation 4. Removing non-alphabetic tokens 5. Removing stopwords 6. Stemming 7. It then extracts CountVectorizer features and runs a 2-layer neural network with 5-fold cross-validation.
b. Feature exploration. Use other features like TFIDF, or any word embeddings provided by other packages like GloVe with gensim, or BERT. Use 5-fold cross-validation to evaluate the performance of your Neural Network.	This section explores additional feature engineering techniques: 1. Generating n-grams (bigrams) 2. Creating TF-IDF vectors 3. Using pre-trained GloVe embeddings 4. Training a Word2Vec model on the corpus 5. It runs 5-fold cross validation with each feature set to compare performance.
c. Describe how you generate features. (5pt)	The features are generated as follows: ● CountVectorizer: Counts the frequency of each word in each document ● N-grams: Generates bigrams from the preprocessed text ● TF-IDF: Computes TF-IDF weights for each word in each document ● GloVe: Averages the pre-trained GloVe embeddings for each word in a document to get a document embedding ● Word2Vec: Trains a Word2Vec model on the corpus and averages the learned word embeddings for each document

<table>
<tr>
<td>d. Report the average training and validation accuracy, and their standard deviation for different feature construction (organize the results in a table). (5pt)</td>
<td>The run_cross_validation function performs 5-fold cross validation with a given feature set and reports the average
training and validation accuracies along with standard deviations.
The results for each feature set are printed in a table.
<table>
<tr><th>Feature Method</th><th>Train Accuracy</th><th>Train Std</th><th>Val Accuracy</th><th>Val Std</th></tr>
<tr><td>CountVectorizer</td><td>1.000</td><td>0.000</td><td>0.968</td><td>0.012</td></tr>
<tr><td>TF-IDF</td><td>1.000</td><td>0.000</td><td>0.971</td><td>0.016</td></tr>
<tr><td>GloVe</td><td>0.232</td><td>0.006</td><td>0.217</td><td>0.026</td></tr>
<tr><td>Word2Vec</td><td>0.230</td><td>0.008</td><td>0.217</td><td>0.026</td></tr>
</table>
</td>
</tr>
<tr>
<td>e. Draw a bar figure showing the training and validation result, x-axis should be the parameter values, y-axis should be the training and validation accuracy. (5pt)</td>
<td></td>
</tr>
</table>

	
Question 2	**Explore the Neural Network model on pre-processed training data. (25pt)**
a. Describe your parameter setting. (5pt)	The key parameters for the neural network model are: ● Parameter Settings for Learning Rate Experimentation ● Neural Network Architecture: 2 hidden layers with 128 neurons each ● Activation Function: ReLU ● Output Layer: Softmax (implicitly applied with CrossEntropyLoss) ● Loss Function: CrossEntropyLoss ● Batch Size: 32 ● Number of Epochs: 10 (or more, depending on convergence) ● Learning Rates to Explore: [0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1]

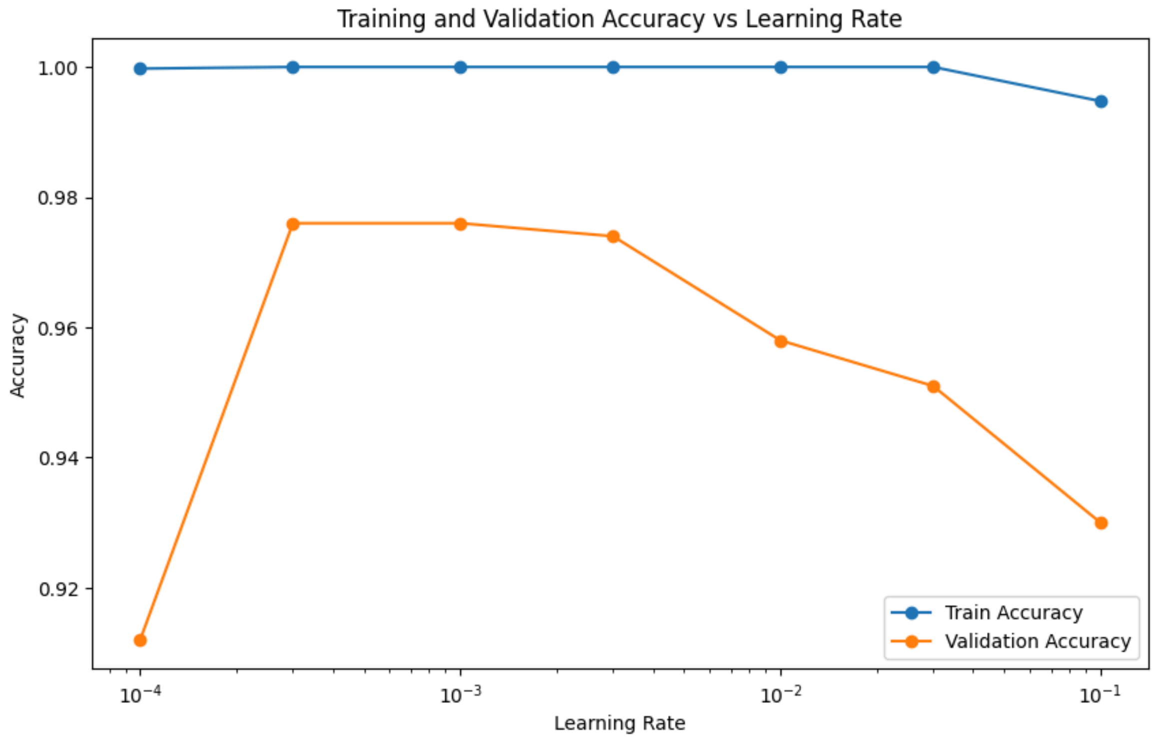
b. Use 5-fold cross-validation to evaluate the performance w.r.t. the learning rates (η), you could use the feature engineering method that has the best performance from Question 1. Recommended candidate values: [0.0001,0.0003,0.001, 0.003,0.01,0.03,0.1]

This section runs 5-fold cross-validation with different learning rates to find the optimal value.
It uses the best-performing feature set from Question 1, which is TFIDF.
The run_cross_validation_lr function trains and evaluates the model with each learning rate.
The results are reported in a table and visualized in a line plot of accuracy vs learning rate.

B.1:

Learning Rate	Train Accuracy	Train Std	Validation Accuracy	Validation Std
0.0001	1.000	0.000	0.912	0.033
0.0003	1.000	0.000	0.976	0.009
0.001	1.000	0.000	0.976	0.009
0.003	1.000	0.000	0.974	0.007
0.01	1.000	0.000	0.958	0.026
0.03	1.000	0.000	0.951	0.014
0.1	0.995	0.003	0.930	0.024

Part B.2:



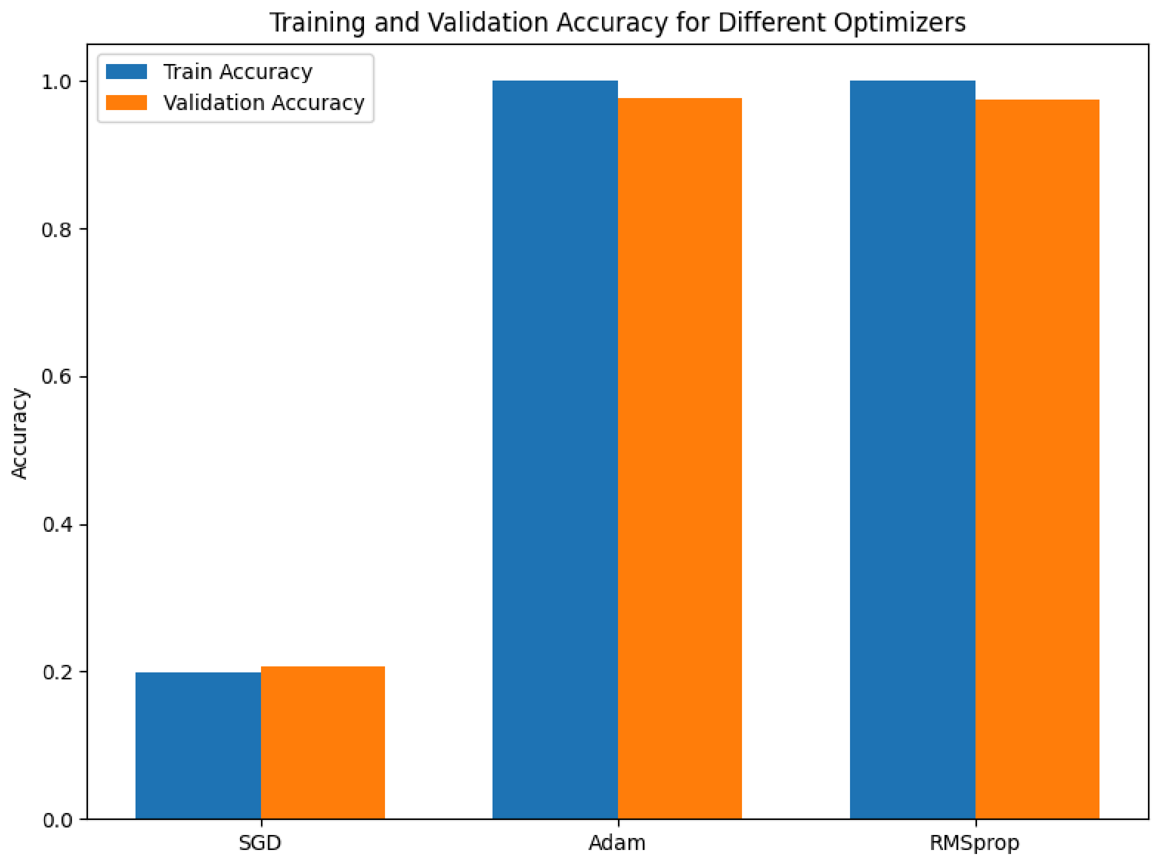
c. Use 5-fold cross-validation to evaluate the performance w.r.t. optimizers, you could use the feature engineering method that has the best performance from Question 1. Recommended candidate values: [SGD, Adam, RMSprop] (see PyTorch or Tensorflow)

This section compares the performance of different optimizers (SGD, Adam, RMSprop) using 5-fold cross-validation.
It uses the best-performing feature set from Question 1, which is TFIDF.
The run_cross_validation_optimizer function trains and evaluates the model with each optimizer.
The results are reported in a table and visualized in a bar chart of train and validation accuracy for each optimizer.

C.1

Optimizer	Train Accuracy	Train Std	Validation Accuracy	Validation Std
SGD	0.199	0.021	0.206	0.029
Adam	1.000	0.000	0.977	0.009
RMSprop	1.000	0.000	0.975	0.010

C.2:

<table>
<tr><td></td><td>
</td></tr>
<tr><td></td><td></td></tr>
<tr><td>QUESTION 3</td><td>3. Predict the labels for the testing data (using raw training data and raw testing data). (60pt)</td></tr>
<tr><td>a. Describe how you pre-process the data to generate features. (5pt)</td><td>The test data is preprocessed using the same steps as the training data:
1. Tokenizing the text into words
2. Converting to lowercase
3. Removing punctuation
4. Removing non-alphabetic tokens
5. Removing stopwords
6. Stemming the words

After preprocessing, the CountVectorizer is used to generate bag-of-words features from the preprocessed text.

The CountVectorizer creates a vocabulary of unique words from the training data and then generates a vector for each document indicating the count of each word.</td></tr>
<tr><td>b. Describe how you choose the model and parameters. (5pt)</td><td>The final model used is a 2-layer neural network with 128 hidden units in each layer. This architecture was specified in the homework requirements.
The key parameters are chosen based on the cross-validation experiments:
Feature set: CountVectorizer (best validation accuracy in Q1)</td></tr>
</table>

	Learning rate: 0.001 (best validation accuracy in Q2b) Optimizer: Adam (best validation accuracy in Q2c)
c. Describe the performance of your chosen model and parameters on the training data. (5pt)	The performance of the final model on the full training set is shown by training it for 10 epochs and printing the training loss and accuracy for each epoch: Training Loss: 0.7579, Training Accuracy: 0.8520 Epoch 1/10 - Train Loss: 0.7579, Train Acc: 0.8520 Training Loss: 0.0209, Training Accuracy: 0.9960 Epoch 2/10 - Train Loss: 0.0209, Train Acc: 0.9960 Training Loss: 0.0026, Training Accuracy: 1.0000 Epoch 3/10 - Train Loss: 0.0026, Train Acc: 1.0000 Training Loss: 0.0014, Training Accuracy: 1.0000 Epoch 4/10 - Train Loss: 0.0014, Train Acc: 1.0000 Training Loss: 0.0007, Training Accuracy: 1.0000 Epoch 5/10 - Train Loss: 0.0007, Train Acc: 1.0000 Training Loss: 0.0005, Training Accuracy: 1.0000 Epoch 6/10 - Train Loss: 0.0005, Train Acc: 1.0000 Training Loss: 0.0004, Training Accuracy: 1.0000 Epoch 7/10 - Train Loss: 0.0004, Train Acc: 1.0000 Training Loss: 0.0003, Training Accuracy: 1.0000 Epoch 8/10 - Train Loss: 0.0003, Train Acc: 1.0000 Training Loss: 0.0003, Training Accuracy: 1.0000 Epoch 9/10 - Train Loss: 0.0003, Train Acc: 1.0000 Training Loss: 0.0002, Training Accuracy: 1.0000 Epoch 10/10 - Train Loss: 0.0002, Train Acc: 1.0000
d. The final classification models to be used in this question are limited to random forest, neural networks, and ensemble methods. (45pt)	The code uses a neural network model for the final classification task on the test set. After training the model on the full training set, it generates predictions on the test set as follows: 1. Load the test data features (count_test) into a Dataset and DataLoader 2. Put the model in evaluation mode 3. Use torch.no_grad() to disable gradient tracking 4. Loop over the test batches 5. Generate predictions by passing inputs through the model 6. Take the class with maximum probability as the predicted label 7. Convert the numerical predictions back to category labels 8. Print out the article ID and predicted label for each test case