In this assignment we have a dataset of around 1700 email messages labelled according to their genres. Our objective is to classify a given email message in one of the 6 genre classes.

First, we download our zipped data file to colab workspace by reading it through its URL. Then we unzip it and remove the earlier downloaded zipped file as it is no longer needed. The file directory is as follows: we have 8 subfolders each of which contain emails and their corresponding labels. We are interested in 1-6 subfolders only. Emails are in .txt format and its label are in .cats format. Since these subfolders belong to each individual class, we use label information directly from here instead of extracting it from .cats file. Now as we have our emails and their label information, we create a pandas dataframe with labels in 1st column and the emails as string in 2nd column. Now we observe the number of samples in each class, and we find that dataset is imbalanced. We will deal with this after data cleaning.

Now we start data cleaning. Apart from the main message each email has multiple information like Message-ID, Date, From, To, Subject, etc. For our modelling process we will be using concatenation of subject and main message as other details have almost no relevance in classification. To extract the subject and message body we use Parser class from email.parser module. Now as we have these as two separate strings, we simply concatenate them together. Moving on, it is found that those emails which were forwarded or have been replied to appear as quoted message below the current message. So, we need to remove these quoted messages. To remove them we define a function which removes everything after "<" or "<<" or "<<<" or ">" or ">>" or ">>>" or "Forwarded by" or "Original Message". Now, we remove any trailing or leading whitespaces. We remove extra whitespace, if there is any and convert everything to lowercase. We remove numbers, punctuations, URLs and stopwords from our data.

Now we start preprocessing. We split our dataset into training, validation and test set. We split it in the ratio of 80:10:10. As found earlier, there is an imbalance in the dataset. To deal with this we upsample all the minority classes in training set to the size of the biggest sized class in training set. For this we use ContextualWordEmbsAug class from the nlpaug.augmenter.word.context_word_embs module which does word level augmentation using contextual word embeddings. After upsampling all the classes to same size, we save our data in google drive as it took very long time to augment. So now we don't need to augment every time we run our code. Before this

upsampling, we used one more technique to deal with imbalanced data, but it didn't work as expected. We will come back to this. Now, we find that our augmented data has some samples which have the text string in a list which is then enclosed in another string. To deal with this we use literal_eval function from ast module. Also, we find that we have some missing values in the augmented data. We drop the rows containing missing values. Then we lemmatize our data. Now, we tokenize our strings using BertTokenizer class from Hugging Face Transformers library with bert-base-uncased configuration. We set max_length = 100 for this process and truncate any longer sentence by setting truncation = True, also we add adequate paddings if needed by setting padding = True. After this we convert the labels corresponding to training, validation and test set to tensors. Then we prepare our dataset using TensorDataset and pass it to the DataLoader. We set batch_size = 32 in DataLoader.

Now we start the modelling process. We use BertForSequenceClassification class from Hugging Face Transformers library with bert-base-uncased configuration and num_labels = 6. We use Adam optimizer with lr = 3e-5. We use Cross Entropy Loss as our loss function (Before upsampling of minority class, we tried to deal with the problem of data imbalance by assigning high weights to minority class and low weights to majority class while calculating Cross Entropy Loss, but it wasn't much effective). Finally, we train and validate our model for 3 epochs. After the training is done, we test our model on test set.

Performance on test set is as following:

- Overall Accuracy: 0.91566
- Overall Precision: 0.93727
- Overall Recall: 0.91474
- Overall F1 Score: 0.91566

Classwise Precision, Recall and F1 Score are as following:

Class	Precision	Recall	F1 Score
1.1	0.94	0.91	0.93
1.2	1.00	1.00	1.00
1.3	0.82	0.90	0.86
1.4	0.87	0.94	0.90
1.5	1.00	0.89	0.94
1.6	1.00	0.85	0.92

Confusion matrix is as following:

Class	1.1	1.2	1.3	1.4	1.5	1.6
1.1	75	0	2	5	0	0
1.2	0	3	0	0	0	0
1.3	0	0	9	1	0	0
1.4	3	0	0	46	0	0
1.5	1	0	0	0	8	0
1.6	1	0	0	1	0	11