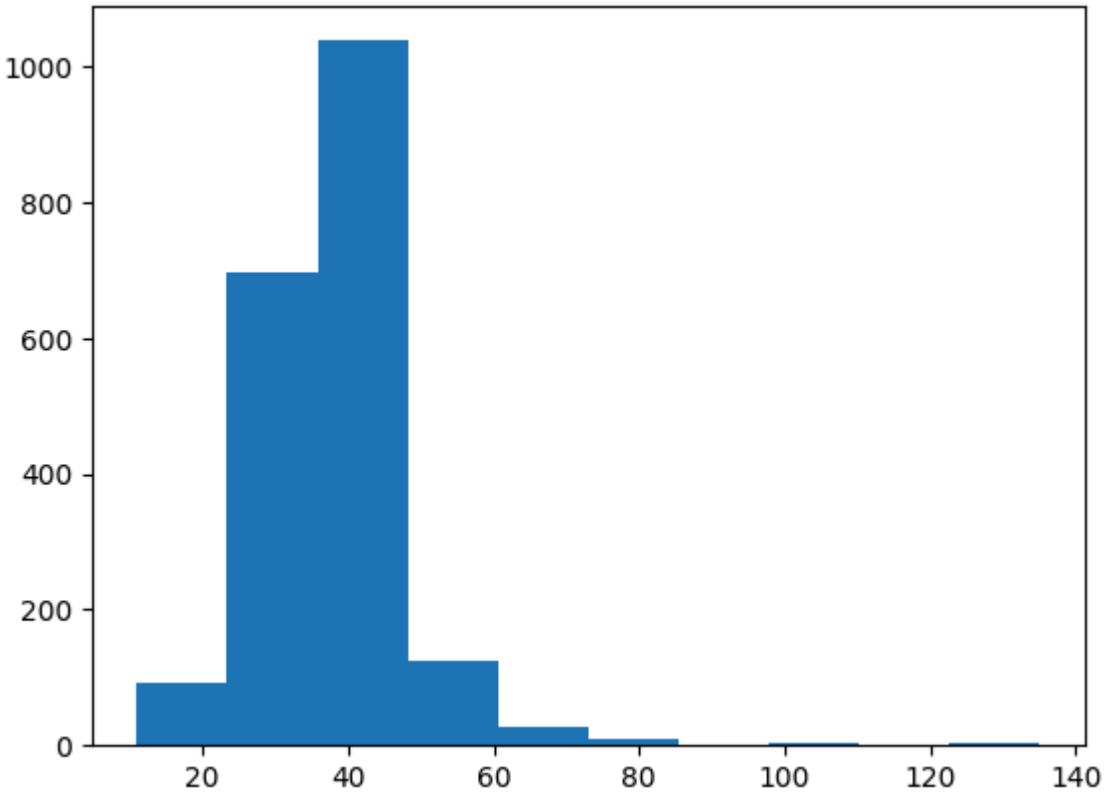
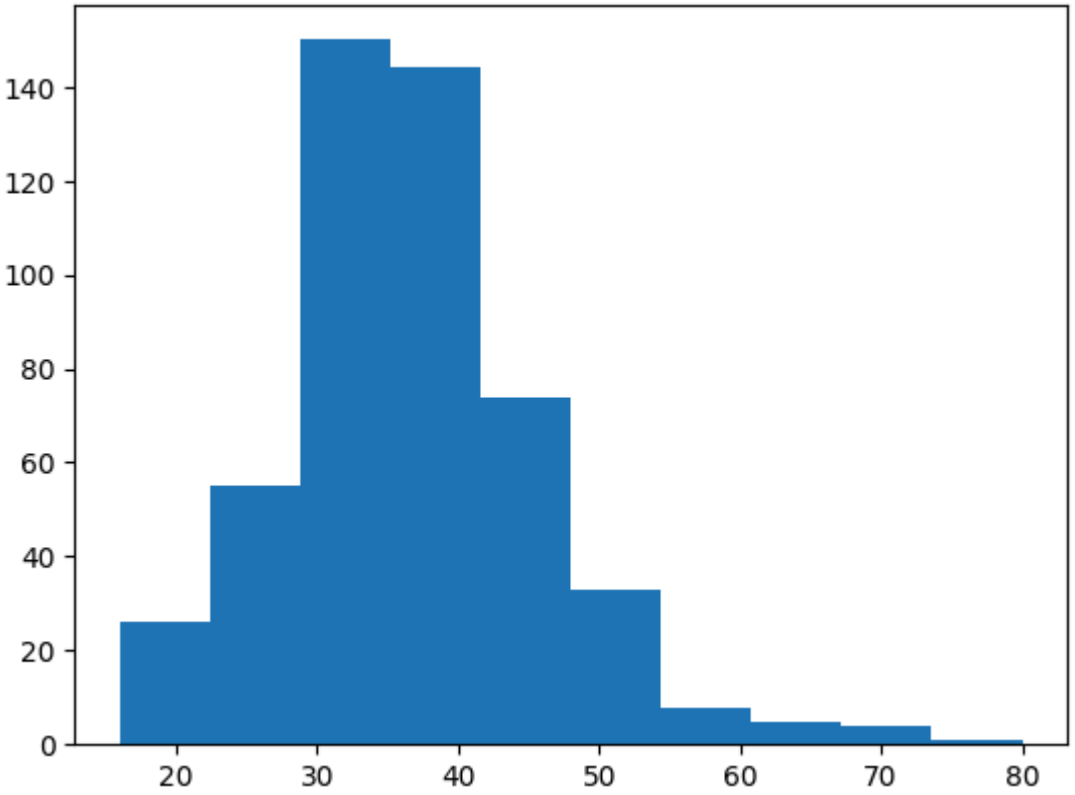
TRANSFORMER-based BERT model for the TEXT classification

STEPS to solve the Task of TEXT classification using BERT MODEL:

1. First, all the important libraries and hugging face transformer libraries are imported to use the pre-trained Roberta Model and tokenizer.
2. Both train and test data are imported in the form of CSV and they are preprocessed which includes making the text lowercase, removing unnecessary symbols, and Replacing all \\ with " ". I haven't removed the stopwords as it is told not to remove them.
3. I have calculated the word count of each sentence in the training dataset to see the distribution of no. of words in the sentences and fix the maximum sequence length. The minimum, maximum, and average no. of words in a sentence in a training dataset is 11,135 and 38 and the distribution is as follows:



4. The minimum, maximum, and average no. of words in a sentence in a test dataset is 16, 80, and 36 and the distribution is as follows:



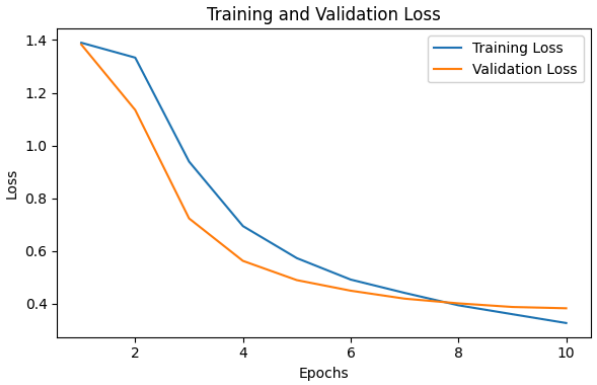
5. As the maximum length in the training dataset is 138 while for the test dataset, it is 80 so, I have set the **maximum sequence length as 150.** This ensures that there is no data loss while still staying within the limits of RoBERTa's tokenization constraints which are 512. And based on this length each sentence in the dataset is padded. Also, I have used the **batch size = 8** as I have tried with other batch size values like 4,16,32 but the best results are obtained for the batch size =8.
6. I have defined a custom dataset class and then using this padding is done to all sentences and then all the datasets are converted to dataloaders.
7. Then a model class is defined where 1st layer is pre trained Roberta model with a "RoBERTa-base" checkpoint imported from the hugging face Transformers library. I have added a 2 layer classification head-on to this Roberta model to perform the classification task. I**nstead of choosing the pre-trained model directly and doing the prediction, I have used a 2-layer classification head which helps make the prediction better and increase the accuracy.** The vectors embeddings come from Roberta of size 768 and then we have added a hidden layer with 64 neurons and then there is another final linear layer to connect these 64 neurons to 4 neurons
8. A standard training loop of pytorch is used where the optimizer is **Adam** with **learning rate = 1e-6**, and the **criterion** is **cross-entropy loss.** The model is trained for **10 epochs** as I have found that the model starts overfitting after that. So, want good prediction model should not be overfitted so I choose no of epochs as 10. Also, the best model is decided based on the validation loss value.

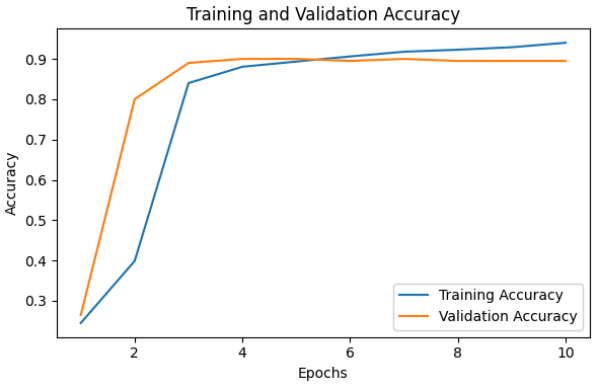
9. On train dataset, train loss, train accuracy, validation loss, and validation accuracy for the best model are as follows:

```
Best epoch :9,
Best Validation Loss: 0.38308261752128603
Validation accuracy at epoch no. 9 is 0.895
Training loss at epoch no. 9 is 0.361
Training accuracy at epoch no. 9 is 0.929
```

We can see that the **validation loss is 0.38** and the **train loss is 0.361** which are good values showing the model is trained well. Also, the **training accuracy is 0.929** while **validation accuracy is 0.895** which seems good showing that the model can perform well on test datasets also.

10. Also, the plots of training and validation loss vs. epochs as well as training and validation accuracy vs. epochs help us to visualize the training and decide the hyperparameter values like no. of epochs to set and the learning rate value of the Adam.





11. The best-trained model is tested on the test dataset. The results of test accuracy, f1-score, and the classification report showing the precision, recall, and f1-score for each of the 4 classes are as follows:

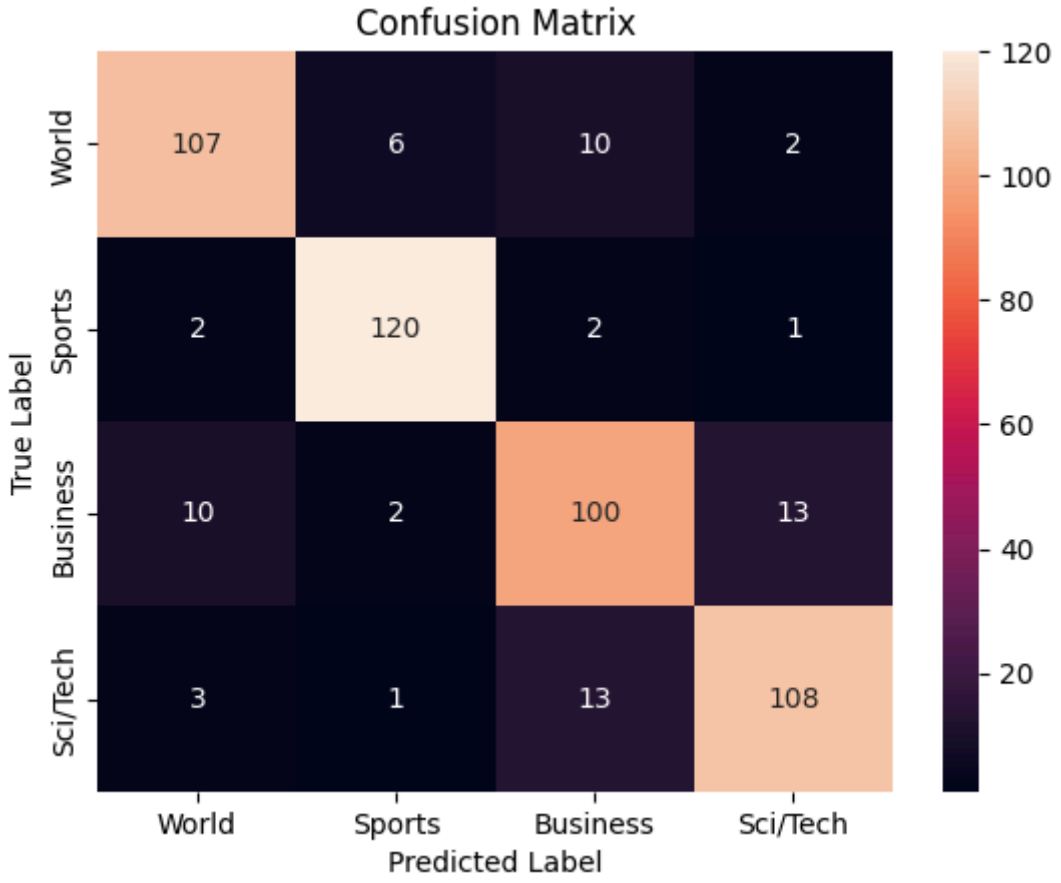
```
Test Accuracy: 0.870
Macro F1 Score: 0.8697
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.86      0.87       125
           1       0.93      0.96      0.94       125
           2       0.80      0.80      0.80       125
           3       0.87      0.86      0.87       125

    accuracy                           0.87       500
   macro avg       0.87      0.87      0.87       500
weighted avg       0.87      0.87      0.87       500
```

We can see that **the test accuracy is 0.870** which is quite good for a training accuracy of 0.929. Also, the **f1-score is 0.8697** which is high showing that the model is performing well across the dataset as a whole in terms of both precision and recall.

12. The confusion matrix for the test datasets is as follows.



We can see that most of the predictions are along the diagonal which shows correct prediction as well and incorrect predictions are also more near to the diagonal rather than away from the diagonal. So, we can say that the model is performing well on the test dataset.

13. The predictions on the test set are saved in a CSV file and the file is downloaded.