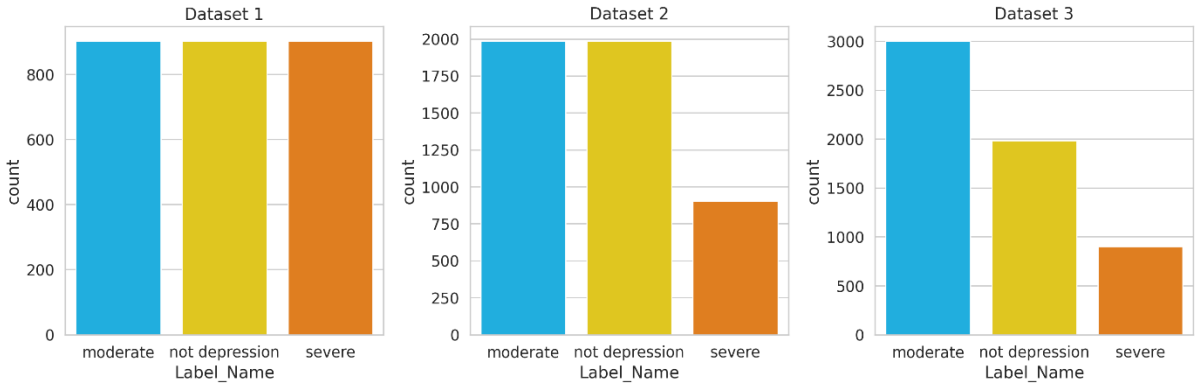
In order to address the issue of class imbalance in our data, we have constructed three distinct datasets. Each dataset has been carefully curated to ensure a more balanced representation of the three sentiment classes: 'moderate', 'not depression', and 'severe'. By doing so, we aim to improve the performance of our models and ensure that they are not biased towards the majority class. These datasets will serve as the foundation for our experiments in fine-tuning various models to achieve the best possible results.

	Moderate	Not Depression	Severe
1	902	902	902
2	1985	1985	902
3	3000	1985	902



I have successfully established a comprehensive pipeline for fine-tuning, training, and validating any pre-trained model on our datasets. This flexible framework allows us to easily substitute different pre-trained models and adjust various hyperparameters to optimize performance. This versatility enables us to experiment with a wide range of configurations and to adapt quickly to new models and techniques as they become available.

Report: Data_set_1

I fine-tuned four different BERT models: bert_cased, bert_uncased, PHS_Bert, and Mental_Bert on dataset 1. The models were trained with a maximum length of 300, a batch size of 18, and for 15 epochs.

The bert_cased model achieved a training accuracy of 99.68% and a validation accuracy of 83.95% by the 13th epoch. However, the model's performance on the validation set in terms of precision, recall, and F1-score varied across the classes, with the 'moderate' class achieving the highest F1-score of 0.61.

The bert_uncased model achieved similar training accuracy as the bert_cased model but had a slightly lower validation accuracy of 81.18% by the 15th epoch. The F1-scores for the classes were also similar to the bert_cased model.

The PHS_Bert model achieved a slightly higher training accuracy of 99.77% and a validation accuracy of 85.24% by the 15th epoch. The F1-scores for the classes were slightly improved compared to the previous models.

The Mental_Bert model achieved a training accuracy of 99.74% and the highest validation accuracy of 85.74% by the 15th epoch. The F1-scores for the classes were also the highest among all the models, indicating that the Mental_Bert model performed the best on dataset 1.

Table:

Model	Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	F1-Score (Moderate)	F1-Score (Not Depression)	F1-Score (Depression)
bert_cased	13	0.0114	99.68%	1.3137	83.95%	0.61	0.40	0.28
bert_uncased	15	0.0079	99.68%	1.2366	81.18%	0.56	0.45	0.30
PHS_Bert	15	0.0044	99.77%	1.2450	85.24%	0.60	0.42	0.32
Mental_Bert	15	0.0077	99.74%	1.0089	85.74%	0.60	0.47	0.37

This table summarizes the performance of each model at the end of their respective training. It includes the training loss, training accuracy, validation loss, validation accuracy, and the F1-scores for each class.

Report:Dataset2

For dataset 2, we fine-tuned two different BERT models: PHS_Bert and Mental_Bert. The models were trained with different maximum lengths, a batch size of 18, and for 10 epochs.

The PHS_Bert model was trained with a maximum length of 200. It achieved a training accuracy of 99.72% and a validation accuracy of 86.26% by the 10th epoch. However, the model's performance on the validation set in terms of precision, recall, and F1-score varied across the classes, with the 'moderate' class achieving the highest F1-score of 0.51.

The Mental_Bert model was trained with a maximum length of 300. It achieved a training accuracy of 99.69% and a validation accuracy of 85.85% by the 15th epoch. The F1-scores for the classes were similar to the PHS_Bert model.

It was observed that the cased and uncased versions of the BERT model did not perform well on dataset 2, hence they were not fine-tuned.

Table:

Model	Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	F1-Score (Moderate)	F1-Score (Not Depression)	F1-Score (Depression)
PHS_Bert	10	0.0078	99.72%	0.9021	86.26%	0.51	0.48	0.38
Mental_Bert	15	0.0077	99.69%	1.0517	85.85%	0.54	0.48	0.38

This table summarizes the performance of each model at the end of their respective training. It includes the training loss, training accuracy, validation loss, validation accuracy, and the F1-scores for each class.

Report:Dataset3

For dataset 3, we fine-tuned two different BERT models: Mental_Bert and PHS_Bert. The models were trained with a maximum length of 200, a batch size of 18, and for 10 epochs.

The Mental_Bert model achieved a training accuracy of 99.66% and a validation accuracy of 88.29% by the 10th epoch. However, the model's performance on the validation set in terms of precision, recall, and F1-score varied across the classes, with the 'moderate' class achieving the highest F1-score of 0.62.

The PHS_Bert model achieved a slightly higher training accuracy of 99.72% and a validation accuracy of 89.73% by the 10th epoch. The F1-scores for the classes were slightly improved compared to the Mental_Bert model, indicating that the PHS_Bert model performed better on dataset 3.

Model	Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	F1-Score (Moderate)	F1-Score (Not Depression)	F1-Score (Depression)
Mental_Bert	10	0.0123	99.66%	0.7635	88.29%	0.62	0.43	0.35
PHS_Bert	10	0.0063	99.72%	0.7848	89.73%	0.63	0.44	0.36

Table: This table summarizes the performance of each model at the end of their respective training. It includes the training loss, training accuracy, validation loss, validation accuracy, and the F1-scores for each class.

In our experiments, the PHS_Bert model demonstrated superior performance on dataset 3 compared to the other models and datasets. This suggests that PHS_Bert is particularly well-suited for this dataset and could be the optimal choice for further analysis and deployment.

Future Work:

- Experiment with Pretrained Model Outputs: I plan to experiment with the pooled outputs from the pre-trained models. By feeding these outputs into different classification algorithms, I hope to discover more effective ways to leverage the power of these models.
- Cost-Sensitive Learning: To better handle imbalanced labels, I intend to apply cost-sensitive learning techniques. This approach assigns a higher cost to misclassifying minority classes, which can help improve the performance of my models on these classes.
- Data Augmentation: I will also explore data augmentation techniques to address class imbalance. By creating synthetic examples, I can increase the size of the minority classes and potentially improve the robustness of my models.
- Preprocessing Techniques: I will experiment with different preprocessing techniques tailored to my specific NLP tasks. This could include different tokenization methods or using lemmatization instead of stemming.
- Ensemble Methods: I plan to combine the predictions of multiple models using ensemble methods. This could lead to better performance than any single model.
- Hyperparameter Optimization: I will employ systematic methods like grid search or random search to find the optimal hyperparameters for my models.
- Transfer Learning: I aim to leverage transfer learning, where a model trained on one task is repurposed on a second related task.
- Multimodal Models: If my data includes non-textual features, I will experiment with multimodal models that can process and learn from these different types of data.

Concerns and Request for Guidance:

As I continue to delve deeper into my work, I have encountered a few challenges that I would like to discuss and seek your guidance on.

- Handling Long Documents: Some documents in my dataset have more than 512 tokens, which makes it difficult to train models on platforms like Google Colab or even on a local system due to memory constraints. I understand that this is a common issue in NLP tasks, and I am exploring various strategies to handle it, such as truncation, segmentation, or using sliding windows. However, I would appreciate your insights and suggestions on the best practices to handle this issue.

- Class Imbalance: Another significant challenge is the class imbalance in my dataset, with the minority class having around 900 instances and the majority class having more than 6000. I have tried several techniques to address this, such as resampling and using class weights, but the problem persists. I would be grateful for your advice on other effective strategies to handle class imbalance, and how to implement them in my current workflow.

- Efficiency Improvements: Lastly, I am always looking for ways to improve the efficiency of my work. If you have any suggestions on methods, architectures, or tools that could help me work more efficiently, I would be eager to learn about them.

Your guidance and support have been invaluable to me so far, and I am confident that with your continued mentorship, I will be able to overcome these challenges and make significant progress in my work.