Wellbeing Watch (Group 4: Mentally Stable)

Pritam Dutta
A0248400N (E0925552)
Master of Computing
National University of Singapore
e0925552@u.nus.edu

Prabhat Ranjan
A0228355B (E0673180)
Master of Computing
National University of Singapore
e0673180@u.nus.edu

Tanamate Foo Yong Qin
A0237342J (E0767625)
Master of Computing
National University of Singapore
e0767625@u.nus.edu

Lalitha Ravi
A0268254X (E1101553)
Master of Computing
National University of Singapore
e1101553@u.nus.edu

***Abstract*—Mental wellbeing is a major issue not only in Singapore but also the rest of the world. This paper aims to find methods of classifying mental health problems in online text, in hopes that these problems can be detected early and if serious enough intervened. The focus of this report is classifying different mental health problems accurately. To do so different approaches have been taken, these include more traditional approaches such as tf-idf vectorization followed by logistic regression and deep learning methods which involve LSTM layers or BERT.**

***Keywords*—mental health, logistic regression, lstm, bert**

I. Introduction

This paper is created as an output for the final project of CS5246 – Text Mining at National University of Singapore by students of Group 4. The project offered a unique opportunity for the team members to put the learnt algorithms into practice. It also motivated us to look beyond the course curriculum for other solutions to practical problems of mental wellbeing.

In recent years, mental wellbeing has become a major health-related topic in Singapore. Mental health problems come in all shapes and forms, with affected people seeking help often not early enough, if at all. However, people share their thoughts, fears, worries, etc. across social media and/or online forums. In this project we explore classification of such posts in social media into different mental health categories.

The first part of the report mentions our motivation for the project and datasets selected. We then describe the details of EDA and preprocessing. The next part provides information on the different traditional and deep learning models explored. The results of prediction on validation and test data are mentioned next. Finally, we present the analysis of the predictions using explanation methods and conclude the report.

II. Motivation

Mental health is an increasingly important health related topic of discussion, with millions of people all over the world suffering from mental health issues such as depression and anxiety. Unfortunately, individuals who are suffering from mental health problems often do not seek help due to stigma, lack of awareness, stressful and busy lifestyle.

In recent past years, social media has become a go to platform for people to share their thoughts, feelings, and experiences, including those related to mental health. This provides us with a unique opportunity to make use of social media data to predict and monitor mental health of individuals and improve access to mental health care.

The goal of this project is to develop a classification model that can accurately predict whether the posts of an individual reflect an issue with mental wellbeing, and we also try to identify five broad classes of mental illnesses - depression, anxiety, bipolar disorder, ADHD (attention deficit hyperactivity disorder), PTSD (Post Traumatic Stress Disorder) and an additional 'None' class (which does not pertain to any mental illness).

There are many potential applications for a model like this, like it could be used for the early identification of people who are at the risk of depression, mental health campaigns, mental health research, personalized mental health support etc. Furthermore, we intend to explore the potential of this model to shortlist people who opt-in for self-mental health monitoring on social media platforms. We intend to alert them if their previous posts continuously reflect signs of depression.

By leveraging social media data and the latest deep learning techniques, we believe that there is potential to improve the overall awareness regarding mental health and its treatment among individuals and the society. The development of a classification model that can accurately predict depression using social media data could be highly instrumental in achieving this goal.

III. Data

A. Data Collection

Obtaining high-quality data that is appropriately annotated for mental health issues proved to be a challenging task, as numerous pertinent datasets are not publicly available and are typically loaned exclusively for research purposes. However, after careful consideration and evaluation of various options, we ultimately chose to utilize two datasets in this project. The first dataset [1] consists of tweets related to mental health issues. The second dataset was sourced from [2] and is a combination of subreddits and blogs.

B. Dataset

[1] was dropped as the data was highly unbalanced.

[2] dataset has a total of six classes associated with a mental illness namely:

1. 'adhd': Neurodevelopmental disorder characterized by inattention, hyperactivity, and impulsivity.
2. 'bipolar': Mood disorder marked by episodes of mania (elevated mood) and depression.
3. 'depression': Mental disorder characterized by persistent feelings of sadness, hopelessness, and loss of interest.
4. 'anxiety': Mental disorder characterized by excessive worry, fear, and nervousness.
5. 'ptsd': Mental disorder that can occur after experiencing or witnessing a traumatic event, causing

persistent symptoms of anxiety, avoidance, and distress.

6. ‘none’: No mental health issues.

The whole dataset was already divided into training, validation and training splits. The training, validation and test datasets have 13727, 1488 and 1488 data points, respectively.

IV. EXPLORATORY DATA ANALYSIS BEFORE PREPROCESSING

We started off with the EDA by checking if the classes are well balanced in the dataset. Though the dataset is not perfectly distributed, class labels are arguably well balanced in training dataset and equally distributed in the validation and test dataset

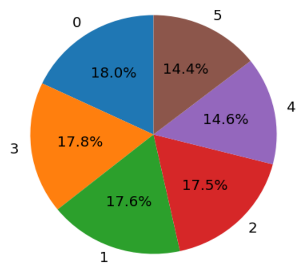


Fig. 1. Class Distribution Train Data

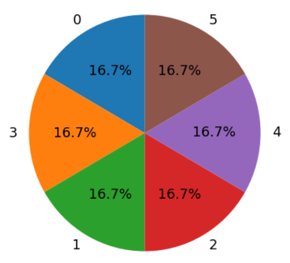


Fig. 2. Class Distribution Val Data

A. *Character Count/ Text Length*

Text length or number of characters range from 150 to 38261, with an average length of approximately 1133 and a standard deviation of approximately 1381. The majority of text lengths fall between 426 and 1306 approximately.

Table Head	**Train Text Length Summary**	
	Table column subhead	***Subhead***
1	count	13727.000000
2	mean	1133.409485
3	std	1381.169545
4	min	150
5	25%	427
6	50%	723
7	75%	1306
8	max	38261

Fig. 3. Text Length Summary

B. *Word Count*

Word counts range from 29 to 6973, with an average count of approximately 209 (Refer box plot for visual representation). Most text word counts fall between 80 and 245.

Table 1	**Train Data Word Count Summary**	
	Table column subhead	***Subhead***
1	count	13727.000000
2	mean	209.842937
3	std	250.509970
4	min	29.000000
5	25%	80.000000
6	50%	135.000000
7	75%	245.000000
8	max	6973.000000

Fig. 4. Word Count Summary

C. *Visualising N Grams for different Mental Health Problems*

We visualise the top 5 one, bi and tri grams for each class label excluding the stop words (This is not part of pre-processing.) Adhd, Anxiety, Depression, Bipolar & Ptsd

-

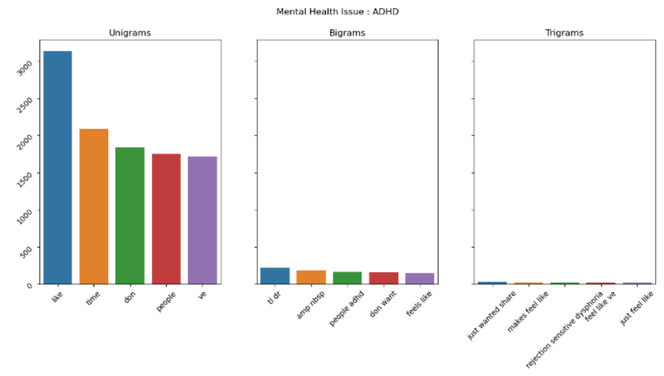


Fig. 5. ADHD N-Grams

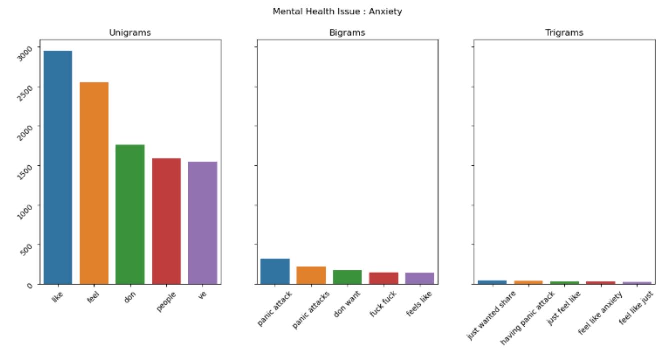


Fig. 6. Anxiety N-Grams

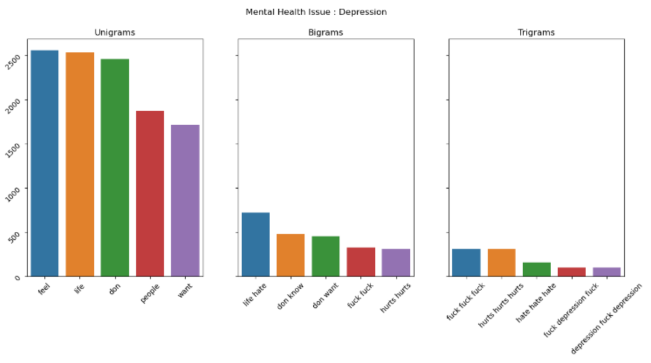


Fig. 7. Depression N-Grams

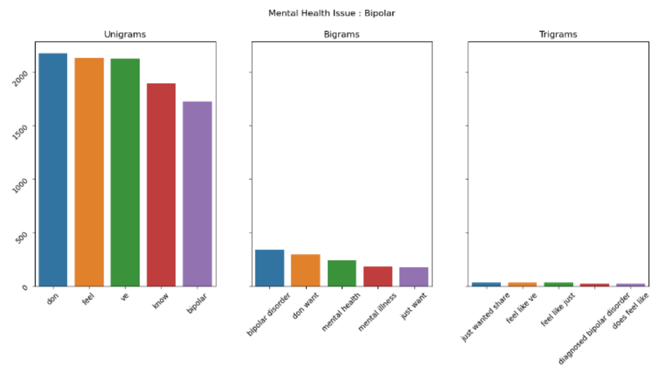


Fig. 8. Bipolar N-Grams

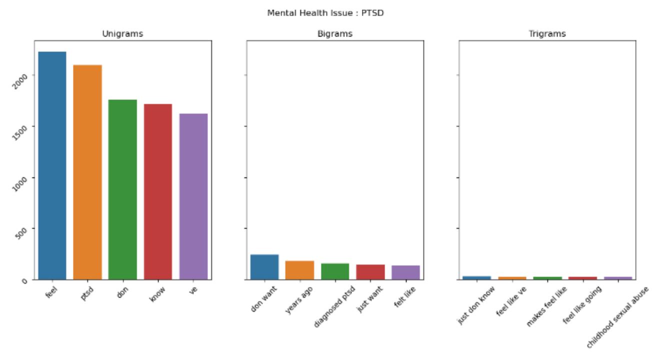


Fig. 9. PTSD N-Grams

V. DATA PREPROCESSING

Below enumerated steps were followed during pre-processing.

1. As social media text may contain a lot of emojis, we first converted emojis and emoticons to their text meaning. We used emoji and emot libraries respectively for this purpose.
2. Contractions like I've were converted to "I have" (Library used: text hammer)
3. Emails were removed (Library used: text hammer)
4. Html tags were removed (Library used: text hammer)
5. Special Characters were removed. (Library used: text hammer)
6. Accented characters were removed (Library used: text hammer).
7. Some most common words with multiple occurrences of a letter, example "coooool" turns into --> cool was replaced. (Regex)
8. Some common acronyms, typos and abbreviations were corrected (Regex).
9. Some common words with multiple occurrences of a letter, which were not handled previously are taken care of using regex.
10. Punctuation marks other than exclamation and Question mark were removed. These were retained to check the strong sentiment expressed by exclamation mark and doubt or uncertainty expressed by question mark.
11. Ellipses were removed.
12. Stop words were removed. Note: Negation words and pronouns were retained. Pronouns were retained as some of the researches have indicated that depressed people often use more pronouns like I, me , myself.
13. Lemmatisation was performed using spacy.

VI. EXPLORATORY DATA ANALYSIS AFTER PREPROCESSING

A. Most Frequent Words

10 most frequent words for each mental issue were identified and plotted, our preprocessing was further refined to avoid typos for such words.

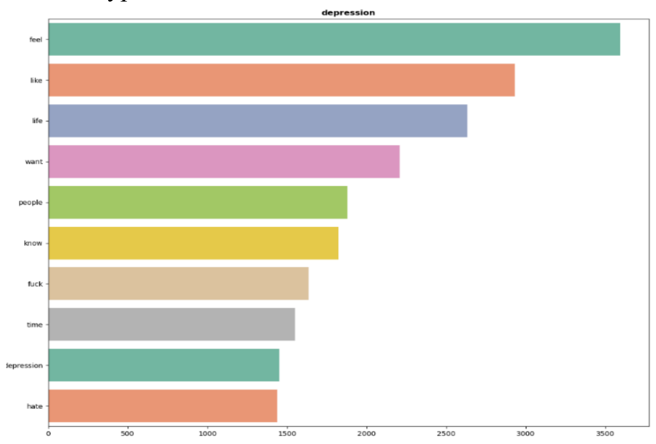


Fig. 10. Depression 10 most frequent words

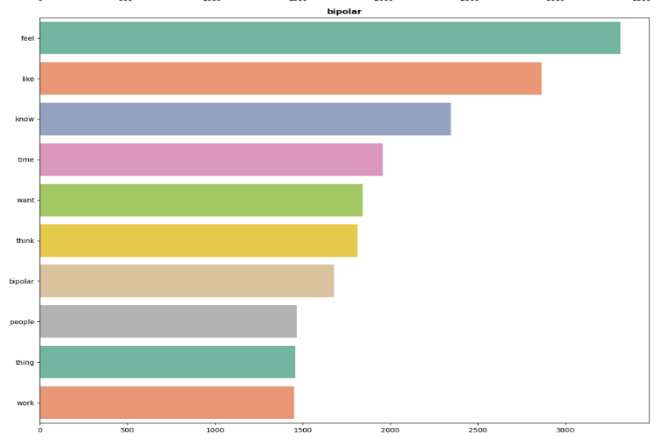


Fig. 11. Bipolar 10 most frequent words

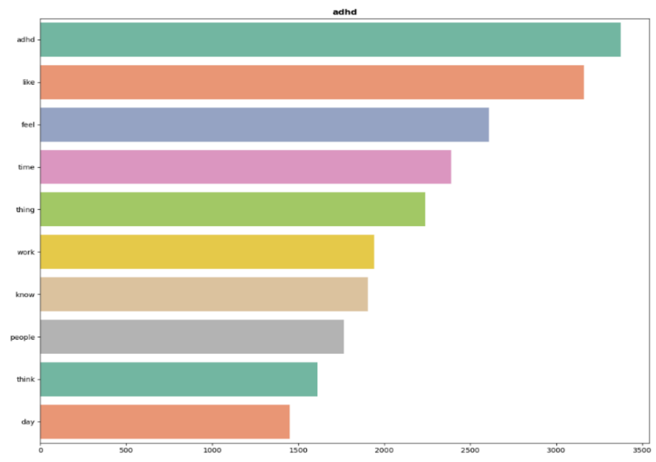


Fig. 12. ADHD 10 most frequent words

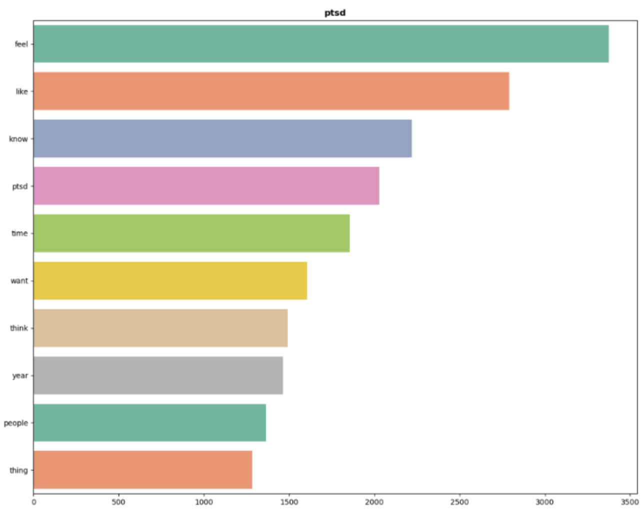


Fig. 13. PTSD 10 most frequent words

B. *Word Count (Tokens)*

Tokens word counts range from 149 to 3963, with an average count of approximately 125(Refer box plot for visual representation). The majority of text word counts fall between 48 and 146. Given that majority of the words lie in this range we have chosen the word embedding size for one of our models to be 200

C. *Average Word Count For Each Class Label*

When we calculated the average word count for each class label and plotted it we noticed that the idea that a person exhibiting more mental issues tend to write longer posts was not reflected in this dataset, so we will not be considering this as an important feature in the currentscope.

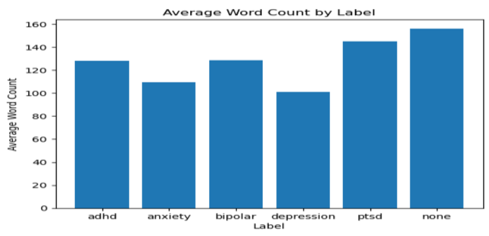


Fig. 14. Average Word Count by Label

VII. METHODOLOGY

This section describes the methodology used in the project to train and test the models. Both traditional as well as deep learning models were used.

A. *Traditional Models*

We used the following base models during this project:

- K-Nearest Neighbor (KNN)
- Naive Bayes
- Decision Tree
- Random Forest
- Logistic Regression

KNN helps capture local patterns by considering k-nearest data points. Naive Bayes was used for the assumption that the features are conditionally independent given the class label. Decision Trees and ensemble Random Forest was used to check for non-linear relationships and handle high-dimensional noisy data. Logistic Regression was used to get. interpretable coefficients that represent the effect of each term on the prediction.

All the above models were trained using the default parameters and performance was measured. The performance of these models was measured by calculating Accuracy, Precision, Recall and F1-score of predictions on validation and test data. Since these are the standard metrics for Machine Learning models, we will not elaborate on this selection.

We then selected the best performing model for further hyperparameter tuning. The detailed results of the models are covered in subsequent sections. Amongst the traditional models, Logistic Regression had the highest accuracy, precision, recall and f1 score. GridSearchCV was used to find the best parameters for Logistic regression model. Three options for parameter, 'C' was fed to GridSearchCV. A smaller value of 0.1 was added to see the results from stronger regularization. A large value of 10 was added with the hope that larger coefficient values can better fit the training data. A medium value of 1.0 was also checked to rule out the possibilities of underfitting and overfitting from the two extreme values respectively.

The Tfidf vectorizer was used before feeding the training data to the base models. The vocabulary built had 41,185 features. For the further tuning of best model, we limited the vectorizer to produce a maximum of 20,000 features. This was done to reduce dimensionality and prevent risks of overfitting. We experimented with the following 4 Vectorizers to find the best option:

- Count Vectorizer
- Tfidf Vectorizer
- Count Vectorizer with ngram range (1,2)
- Tfidf Vectorizer with ngram range (1,2)

Count Vectorizer was selected to determine how well the model performs in terms of capturing term frequency or count of individual tokens. Tfidf was selected to seek important terms based on both frequency and rarity within the reviews. The n-grams were explored to capture local contextual information in the text.

Finally, we used the Feature Importance Explanation method to analyze the results. The absolute values of coefficients produced by the Logistic regression model was used as feature importance. We selected this approach because it helped us determine the top 20 features representing each class. A larger magnitude of the coefficient indicates a stronger influence of the feature or term on model's prediction. We were able to interpret both the correct and incorrect predictions using this knowledge.

B. *Deep Learning Models*

1) *Recurrent Neural Networks*

We also used Recurrent Neural Network (RNN) based text classifiers such as LSTM (Long-Short Term Memory) and Bidirectional LSTM (BiLSTM) on our dataset to compare the results with those obtained using traditional Machine Learning models. We know that LSTM is a variety of RNN that can learn long-term dependencies, especially in sequence-based prediction problems. LSTM neural networks can solve numerous tasks that are not solvable by vanilla RNNs. Long-term temporal dependencies can be captured effectively by LSTM, without suffering much optimization hurdles.

The BiLSTM is an upgrade on top of the basic LSTM. In BiLSTM, the information is obtained by processing the sequence in both forward and backward directions within two hidden layers. Both the sequences are connected to the same output layer. BiLSTM has complete information about every point in each sequence, everything before and after it, thus, placing them above LSTMs with respect to performance in text classification tasks.

We use LSTM and BiLSTM models with the below structures for this project. The number of neurons in the output layer is 6 because there are a total of 6 class labels.

- *LSTM*: The model is structured in the order: Embedding layer [Vocabulary size x 300 (word embedding length)], dropout, LSTM layer [300 x 64], Linear layer [64 x 6].
- *BiLSTM*: The model is structured in the order: Embedding layer [Vocabulary size x 300 (word embedding length)], Bidirectional LSTM layer [300 x 64], Linear layer [256 x 64], Linear layer [64 x 6]. ReLU is used as the activation function and dropout is used.

Pretrained word embeddings from Global Vectors of length 300 have been used to convert the texts to their corresponding feature vectors for the LSTM-based models. All unique words from the training dataset are collected to make up the master vocabulary. For unknown or Out-of-vocabulary (OOV) words, a vector of length 300 comprising of random numbers between -0.25 and 0.25 is generated and used as the embedding. An encoding is generated for each text record with the indexes of the constituent words from the master vocabulary. The encoding length is chosen to be 400 based on the mean, maximum and 90th quartile of the number of words (tokens) in each record of the preprocessed dataset.

2) BERT Model

Based on a lot of anecdotal evidence as well, recommendations from blog articles as well as the final lecture discussing the topic, transformer-based models, most notably- BERT trained models which were superior to embedded LSTM models as well as Tfidf models in classifying data accurately. This is due to BERT's model having a self-attention mechanism which leads it to having no locality bias and provides equal opportunity towards all contexts - from long distance to short distance context. Thus, we decided to use BERT to see if higher accuracies could be achieved.

a) Pre-processing (BERT)

Pre-processing steps for training BERT were mostly the same as the others with a few key differences. These include:

1. Removal of stop-words
2. Removal of punctuation
3. Tokenization
4. Balancing the dataset (for the depression severity dataset)

Intuitively, as the benefit of BERT over more traditional approaches is that it learns to compute text representations in context, removal of stop words and punctuation will not improve and in some cases negatively affect the model. For example, consider the following sentence before and after removal of stop words and punctuation:

Input: "*I was depressed. But now I feel fine!*"
Result: "depressed. feel fine!"

If only considering the output it is ambiguous for a human, let alone a machine to tell the state of depression let alone the severity. As a result, stop-words were not removed. The same goes for punctuation: a question mark can certainly change the overall meaning of a sentence. Therefore, removing stop words and punctuation would just imply removing context which BERT could have used to get better results. We must consider that the benefit of BERT is that it learns to compute text representations in context.

Tokenization is unnecessary in the pre-processing step as BERT does that anyways during the encoding step when it makes the input ids and the attention masks. Encoding was slightly different to the LSTM model. To train the model, text must be encoded into attention masks and input ids. Attention masks are padding or truncation to direct models to the same number of tokens, so they have the same input shape. Input ids are a mapping between tokens and their respective ids.

Balancing the dataset for training the model is self-explanatory. The depression severity dataset had over 90% cases as non-depressed and we were worried if trained on this, the accuracy metric may not have reliably informed us on how well the model performed.

b) Training

To train the model the encoded data was passed into the model and depending on the results needed the output layer was adjusted accordingly.

We will split the reporting into two parts based on the 2 different datasets.

1. Mental Illness Multi-Classification Dataset.
2. Extras: Depression Severity Dataset

VIII. PREDICTIONS AND PERFORMANCE

A. Traditional Models

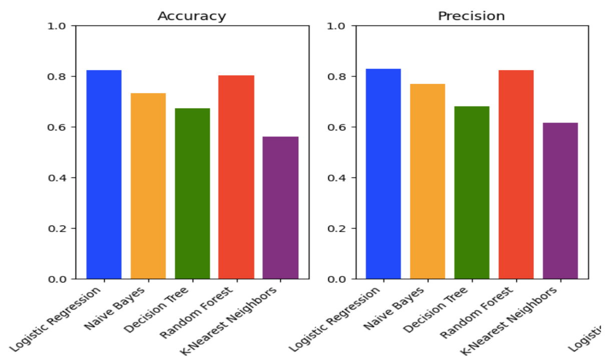


Fig. 15. Accuracy and Precision on Validation data

The trained models on default parameters were used to run predictions on both validation and test data. The results from the validation data are shown above in Figure 15. It is seen that the Logistic Regression model had the best performance amongst the traditional models with an accuracy of 0.81 followed by Random Forest at 0.8. This is because both these models can capture complex interactions between words. They are also robust to handling incomplete sentences and outliers in text data.

On the other hand, KNN had the lowest score of 0.5 because it can easily suffer from curse of dimensionality.

Since the number of words in the default training was higher than 40,000; the feature space was high dimensional. KNN relies on the distances between data points and as the count of dimensions increase, the distances tend to converge. This reduces the discriminatory power of KNN and leads to a lower accuracy.

The results for Recall and F1-Score are shown in the image below. The results for all the four metrics are very close to each other for each model. This is because the dataset is balanced between each class as seen during the EDA. The results of the validation data and test data are also almost same with similar scores for each model. It shows the consistency of the models' performance. So, we are not mentioning it separately.

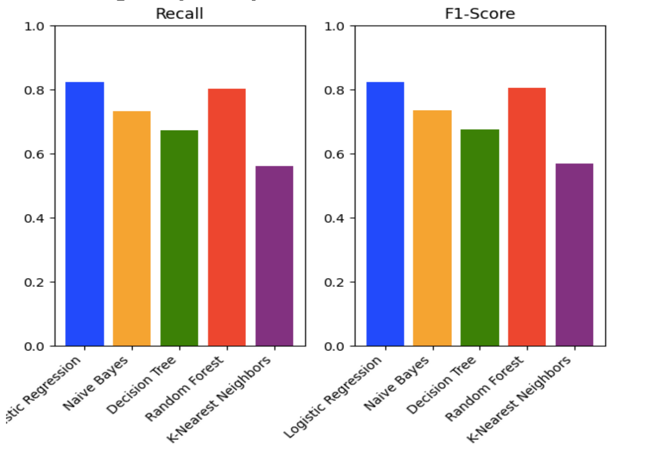


Fig. 16. Recall and F1-Score on Validation data

It is expected that the results could have improved for KNN by choosing an appropriate value for hyperparameter K. However, since Logistic Regression had the best scores, we decided to select it for further optimization. The table below shows the best value of parameter, 'C' that was obtained for the different vectorizer methods.

We observe that the best value for both types of Count Vectorizer is obtained as 0.1. This is because the features extracted from the text just based on frequency of their occurrence are less informative. So, a stronger regularization is required to find the best classification accuracy. In the case, Tfidf, a larger value of C indicates that the model performed better with weaker regularization as the features extracted from Tfidf are more informative and have higher predictive power as compared to Count Vectorizer.

Hyperparameter 'C' and Different Vectorizers	
Vectorizer	***C***
Count Vectorizer	0.1
Tfidf Vectorizer	1.0
Count Vectorizer with ngram range (1,2)	0.1
Tfidf Vectorizer with ngram range (1,2)	10

Fig. 17. Best values for Logistic Regression 'C' using different vectorizers

We observe that the best value for both types of Count Vectorizer is obtained as 0.1. This is because the features extracted from the text just based on frequency of their occurrence are less informative. So, a stronger regularization is required to find the best classification accuracy. In the case, Tfidf, a larger value of C indicates that the model performed better with weaker regularization as the features extracted from Tfidf are more informative and have higher predictive power as compared to Count Vectorizer

We also observe the difference in value of 'C' obtained when using the ngram range with Tfidf vectorizer. The inclusion of bigrams increased the feature space, which meant the model required even a weaker regularization or a high value of 'C' as 10 to avoid underfitting and a better capture of contextual information.

The figure below compares the performance of each of the vectorizer techniques with the best values of 'C' obtained for them. In order to highlight the difference, the scale has been reduced to show the bars between scores of 0.8 and 0.85 only.

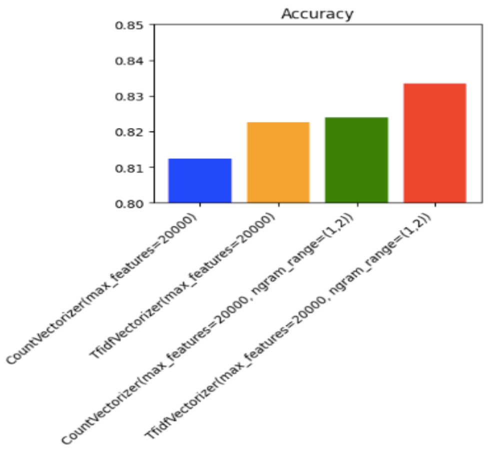


Fig. 18. Performance of Tuned Logistic regression after different vectorization

We observe that although the performance of all the 4 types is very close to each other, the Tfidf Vectorizer with ngram range has the best accuracy of 83.3%. This is because it can capture more contextual information using the bigrams. It resulted in having features that had more predictive power and could capture multi-word expressions.

B. *Deep Learning Models*

1) *Recurrent Neural Networks*

As mentioned before, the dataset was already split into training, validation and test from the start. Both the LSTM and BiLSTM models were trained on the training dataset and validated against the validation dataset. The BiLSTM model was found to be converging faster than the LSTM model. We trained the LSTM model for 40 epochs and the BiLSTM model for 20 epochs. Learning rates of 0.005 and 0.0001 with an Adam optimizer were used for the LSTM and the BiLSTM models, respectively. We chose to use the Adam optimizer as it is adaptive in comparison to other optimization algorithms like Stochastic Gradient Descent (SGD) and is known to yield faster and better results. The Cross-Entropy Loss and accuracy/f1-score metrics were recorded for both the training and validation splits at the end of every epoch. The pretrained weights used for the models and the state dictionary of the trained model were saved to disk for testing the performance of the model on the test dataset.

In order to analyze the effect of dropout rates on these models, we use dropout values of 0.2, 0.3 and 0.5 to train, validate and test the LSTM and BiLSTM models. A consolidated view of the results obtained can be seen in Table I. For every scenario, we use the model state dictionary for the epoch after which the validation loss becomes greater than the training loss and starts to increase. Based on this approach, we use the state dictionaries of epoch 7, 19 and 29 as the final trained models for LSTM with dropout values of 0.2, 0.3 and 0.5, respectively. Similarly, we use the state dictionaries of epoch 8, 9 and 12 as the final trained models for BiLSTM with dropout values of 0.2, 0.3 and 0.5, respectively.

Table 3 Validation and Test results for LSTM-based models

Model (dropout)	*Validation Data*		*Test Data*	
	Accuracy	*f1-score*	*Accuracy*	*f1-score*
LSTM (0.2)	0.733	0.737	0.710	0.716
LSTM (0.3)	0.790	0.795	0.775	0.778
LSTM (0.5)	0.790	0.794	0.755	0.758
BiLSTM (0.2)	0.819	0.820	0.798	0.800
BiLSTM (0.3)	0.816	0.817	0.807	0.807
BiLSTM (0.5)	0.823	0.823	0.811	0.810

Fig. 19. Validation and Test results for LSTM-based models

The LSTM model demonstrates the best performance with a dropout value of 0.3. It achieves a validation accuracy and f1-score of 0.790 and 0.795, respectively and a test accuracy and f1-score of 0.775 and 0.778, respectively. On the other hand, the BiLSTM model gives the best performance with a dropout value of 0.5. We get a validation accuracy and f1-score of 0.823 and 0.823, respectively and a test accuracy and f1-score of 0.811 and 0.810, respectively. Between the two models, the BiLSTM consistently demonstrates better performance, as is expected based on previous research on the subject.

2) *BERT Model*

a) Mental Illness Multi-Classification Results:

To compare accuracies to the other 2 approaches, Tfidf logistical regression as well as LSTM approaches models, our BERT model was fitted to this dataset as well. We truncated the text to a maximum of 1500 length which was a more than 5 standard deviations away from the mean text length accounting for more than 99% of the data, which we thought was sufficient. Encoding and training this data took a little under 2 hours after purchasing collab pros compute units. These are the results:

Validation and Test results for BERT-based models

Model (dropout)	*Validation Data*		*Test Data*	
	Accuracy	*f1-score*	*Accuracy*	*f1-score*
Bert-base-uncased (0.15)	0.847	0.848	0.843	0.844

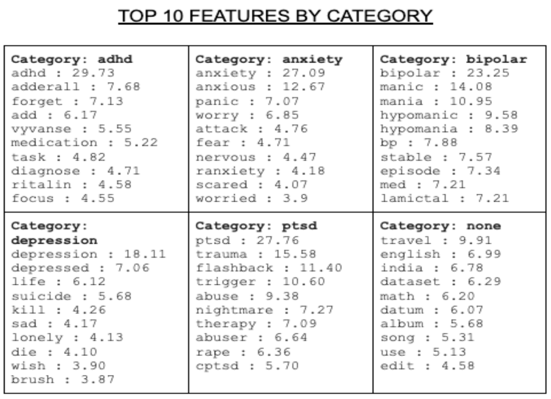
As we can see BERT performs extremely well with an accuracy of 84-85% and we see the f1-scores at 0.84-0.85 which is highest among all other prediction methods.

IX. EXPLANATIONS AND ANALYSIS

This section covers the analysis and discussion on how the predictions of the model can be explained.

A. Traditional Models

Feature Importance was used to test the interpretability of the tuned Logistic Regression model. The top ten features obtained for each class is mentioned in the figure below.



TOP 10 FEATURES BY CATEGORY

Category: adhd	**Category: anxiety**	**Category: bipolar**
adhd : 29.73	anxiety : 27.09	bipolar : 23.25
adderall : 7.68	anxious : 12.67	manic : 14.08
forget : 7.13	panic : 7.07	mania : 10.95
add : 6.17	worry : 6.85	hypomanic : 9.58
vyvanse : 5.55	attack : 4.76	hypomania : 8.39
medication : 5.22	fear : 4.71	bp : 7.88
task : 4.82	nervous : 4.47	stable : 7.57
diagnose : 4.71	ranxiety : 4.18	episode : 7.34
ritalin : 4.58	scared : 4.07	med : 7.21
focus : 4.55	worried : 3.9	lamictal : 7.21
Category: depression	**Category: ptsd**	**Category: none**
depression : 18.11	ptsd : 27.76	travel : 9.91
depressed : 7.06	trauma : 15.58	english : 6.99
life : 6.12	flashback : 11.40	india : 6.78
suicide : 5.68	trigger : 10.60	dataset : 6.29
kill : 4.26	abuse : 9.38	math : 6.20
sad : 4.17	nightmare : 7.27	datum : 6.07
lonely : 4.13	therapy : 7.09	album : 5.68
die : 4.10	abuser : 6.64	song : 5.31
wish : 3.90	rape : 6.36	use : 5.13
brush : 3.87	cptsd : 5.70	edit : 4.58

Fig. 20. Performance of Tuned Logistic regression after different vectorization

It is observed that the term mentioning the name of the class has the highest importance which is expected. For example, the term, 'adhd' has the importance of 29.73 which is the highest for the class, 'adhd'. The category, 'none' contains generic terms that are not used in the context of a mental wellness. Two documents from each class were selected randomly to further analyze the prediction. We present analysis of two such results here, one where the prediction matched the actual class and another where the prediction was different. We will refer them as Test A and Test B.

The text from user's post for Test A was the following: *"does anyone here not take medication? i was diagnosed a few months ago. i really don't know if the story is important. we are dead broke and all i can do is work as much as i can to try and stay fed and sheltered. because of recent years incomes from my wife's business wwe don't qualify for assistance. i am now not struggling with any mania or paralyzing depression, probably mostly due to hard physical work that i recently started doing. lithium made me feel panic attacks like my limbs were turning to stone and a little creature was trying to burrow out the front of my skull. my wife verifies i am much calmer, am not paranoid or delusional-seeming, and that medication seemed to harm more than help. i know it is simple-minded to assume that one experience with a medication means i won't ever benefit. if the nukes don't get us and i continue to climb out of the dark trench toward better times, i don't know whether there is a need to seek medication. i would love to know if there are others who for whatever reason aren't taking anything (pharmaceuticals, weed, etc.) and are happier for it."*

The actual class as mentioned in the dataset is 'bipolar' and the prediction of the model is also 'bipolar'. The prediction probability of the different classes for this test are shown in the table <Add number>. We observe that the post has the term ‘depression’ that has high feature importance for ‘depression’ class. However, it also has the term ‘mania’ which has high feature importance for the ‘bipolar’ class.

Table 4	**PREDICTION PROBABILITY**		
	Class	***Test A***	***Test B***
1.	adhd	0.07150	0.00048
2.	anxiety	0.01766	0.00008
3.	bipolar	0.87412	0.00072
4.	depression	0.00520	0.00036
5.	ptsd	0.03016	0.99830
6.	none	0.00133	0.00005

Fig. 21. Probability assigned by Logistic regression for each class

The text from user’s post selected as Test B is shown below: *“do you guys get nightmares? i had the worst one of my life tonight. i was psychotic and had a flashback to my attempt. multiple times. my hand was talking to me and telling me bad things. i have 3-4 nightmares a week and they aren't getting better. i don't know why this is happening.”*

The actual class mentioned in the dataset is ‘bipolar’ but the predict class is ‘ptsd’. The prediction probability of the different classes for this test are shown in the table above. We observe a big difference in the prediction probability for the actual and predicted class where ‘ptsd’ received a probability of 99% as compared to ‘bipolar’ that has 0.07%. This is because the document contains words like, ‘flashback’ and ‘nightmare’ that have high importance for the class ‘ptsd’ as seen in the ‘Top 10 Features by category’ table.

B. Deep Learning Models

1) Recurrent Neural Networks

The plots depicting the variation of training and validation loss/f1-score versus epochs for the various dropout values were captured for both the LSTM and the BiLSTM models. The plots for the LSTM model with a dropout of 0.3 and the BiLSTM model with a dropout of 0.5 can be seen in Figure 22, (a) - (d).

The LSTM and BiLSTM models, in their pursuit of trying too hard to learn different features from the training data, can sometimes learn the statistical noise in the training dataset. This can definitely improve the model performance on the training dataset, but the trained model would then fail massively on unseen data (e.g. test dataset). This would indicate that the model is overfitting on the training data. Hence, we use dropout, which is the practice of randomly dropping out the nodes in the input and hidden layers of a neural network. All the forward and backwards connections with a dropped node are temporarily removed, thus creating a new network architecture out of the parent network. The nodes are dropped by the dropout probability which we specify when creating the model. This helps the models to prevent the overfitting problem.

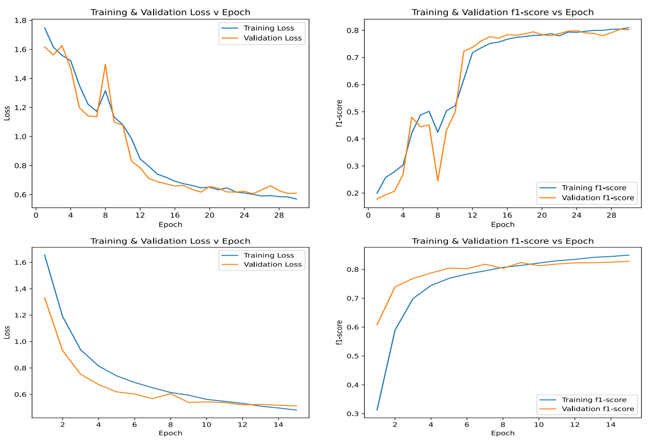


Fig. 22. (a) Training & Validation Loss vs Epoch for LSTM with a dropout of 0.3 (b) Training & Validation f1-score vs Epoch for LSTM with a dropout of 0.3 (c) Training & Validation Loss vs Epoch for BiLSTM with a dropout of 0.5 (d) Training & Validation f1-score vs Epoch for BiLSTM with a dropout of 0.5

In general, we found the training and validation loss trajectories to be smoother for the BiLSTM model than it was for the LSTM model. Each epoch takes longer to complete for the BiLSTM model than the LSTM model as the input sequences is processed in both the forward and backward directions in two hidden layers in the former case. However, since the BiLSTM model converges faster, they only need to be trained for a shorter number of epochs as compared to the LSTM model.

We used the best performing LSTM and BiLSTM models to classify the test dataset and derive the precision and recall metrics too, in addition to the accuracy and f1-scores stated earlier. We get a precision score of 0.793 and a recall score of 0.775 using the LSTM model with a dropout of 0.3 and a precision score of 0.812 and a recall score of 0.811 using the BiLSTM model with a dropout of 0.5. Confusion Matrices normalized over the actual counts of the various class labels were generated for both scenarios to better analyze the performance of the trained models on the test data and to see if the models performed better on certain class labels than it did on the others. The generated Confusion Matrix for the best LSTM model can be seen in Figure 23(a) and the same for the best BiLSTM model can be seen in Figure 23(b).

The LSTM model with a dropout of 0.3 performed very well for the ‘none’ and the ‘depression’ class labels, classifying them with an accuracy of 87.5% and 82.7%, respectively. The worst performance of this model was seen for the ‘bipolar’ class label. It was only able to achieve an accuracy of 64.5% for this class. We also see that 18.1% of the test data points under the ‘bipolar’ class label get misclassified under the ‘depression’ class label. Similarly, 14.9% and 11.3% of the test data points under the ‘anxiety’ and ‘ptsd’ class labels, respectively, get misclassified under the ‘depression’ class label.

The BiLSTM with a dropout of 0.5 demonstrated the best performance for the ‘none’ class label, achieving an accuracy of 94.4%, followed by an accuracy of 86.7% for the ‘adhd’ class label. The BiLSTM model can classify the test data points of the ‘anxiety’ and ‘ptsd’ class labels much better. We see only 6% of the ‘anxiety’ data points and 3.2% of the ‘ptsd’ data points get misclassified under the ‘depression’ class label.

Interestingly, the accuracy of the 'depression' class label goes down to 69% for the BiLSTM model in contrast to 82.7% for the LSTM model. We see that 8.1% and 7.7% of the 'depression' data samples get misclassified under the 'bipolar' and 'anxiety' class labels, respectively by the BiLSTM model.

It is easy to see why see such high accuracies for the 'none' class label achieved by both models. The words present in posts belonging to this category would generally be different as compared to the words that would be commonly present in posts belonging to any of the other class labels. On the other hand, the other class labels, namely 'adhd', 'anxiety', 'bipolar', 'depression' and 'ptsd', are very closely related and would have a very high probability of common words occurring in the posts tagged to each of these labels. Consequently, both the models occasionally tend to misclassify data points belonging to these five class labels, although the BiLSTM model performs consistently performs better than the LSTM model except for the 'depression' class.

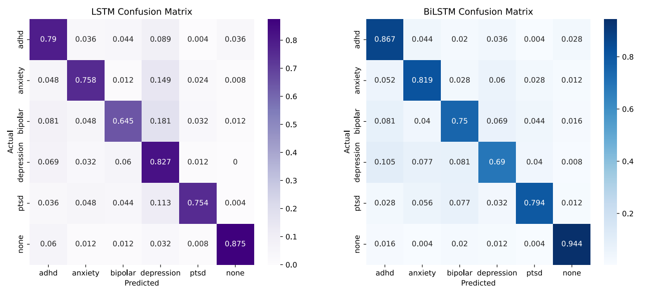


Fig. 23. (a) Normalized Confusion Matrix generated after classifying the test dataset using the trained LSTM model with a dropout of 0.3 (b) Normalized Confusion Matrix generated after classifying the test dataset using the trained BiLSTM model with a dropout of 0.5

2) *BERT Model*

Due to training being extremely time-consuming as well as computationally expensive we only trained BERT for 3 epochs. In addition, training the BERT model for over 3 epochs seemed to end up overfitting our data shown by the increase in loss function.

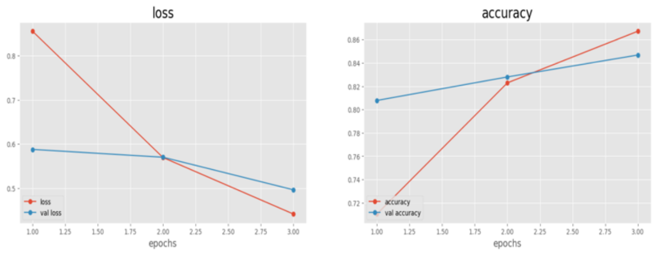


Fig. 24. Training & Validation Loss vs Epoch for BERT

Like LSTM, our BERT model performed poorest when detecting bipolar classes. Also similar was that BERT performed best when detecting the "none" class as well as ADHD. Our model performed worst in classifying anxiety as well as bipolarism, misclassifying most data into depression, anxiety with 12% and bipolar with 9%. We suspect that it is because these mental health conditions have overlapping symptoms and are difficult for even professionals to detect.

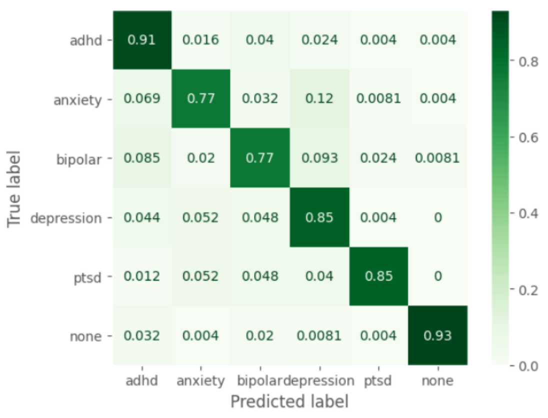


Fig. 25. Normalized Confusion Matrix for Mental Illness generated after classifying the test dataset using the trained BERT model with a dropout of 0.15

For example, bipolarism and depression are similar in that both include periods of feeling low mood or lack of in everyday activities and previously bipolar disorder was named "manic depression".

X. Conclusion

The tasks designed for the project were fulfilled. We were able to experiment with multiple traditional and deep learning models. We were also able to analyze the results and determine the reasons for predictions and other metric values.

It was found that Logistic regression performed the best amongst various other traditional models. We also discovered that Tfidf Vectorizer with a ngram range gives better results than Count Vectorizer. Feature importance based on coefficients of Logistic Regression model can explain the predictions made by the model.

We found that the BiLSTM model gives better results when compared to the LSTM model for this test classification task, possibly because it processes the input sequence in both forward and backward directions within two hidden layers. Although it takes slightly longer to train the BiLSTM for one epoch, it converges faster and overall yields a smoother learning curve. Dropout is extremely essential to prevent overfitting and ensure generalization of the results.

Our Bert had the best results by a small margin. Beating our next best model Logistical regression with an 84-85% accuracy. Possible improvements could involve using a different BERT model, for example bert-large instead of bert-based-uncased.

XI. Extras: Depression Severity

Our successes in predicting mental health status from the mental illness database were after a failed attempt at classifying another dataset during the first part of our report submission. Following our failed attempt to classify our depression dataset during the first milestone of our report, we needed a better model to train the data. Using tf-idf vectorization followed by logistic regression to classify depression severity was going to be difficult. Especially because data retrieval of the depression severity dataset involved obtaining tweets that contained symptoms of depression or their synonyms.

Thus, options moving forward were:

1. Find a new dataset which did not retrieve datasets based off existing keywords.
2. Find a more sophisticated model to train our data.

A. Depression Severity Dataset Results:

As using BERT was not taught in class, many mistakes were made during configuration encoding as well as the configurations of the final layer of the model. Thus, before we jumped right into multi class classification, the data was split into binary classes, depressed and non-depressed for what we thought may have been a simpler model to train. The results are shown below:

1) Binary Depression

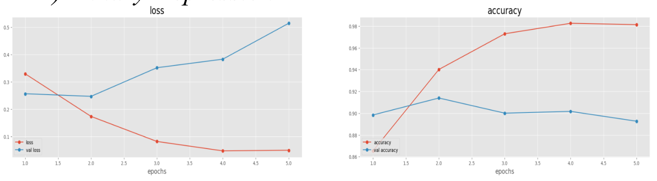


Fig. 26. Depression Severity Binary: Training & Validation Loss vs Epoch for BERT

As we can see from the graph the model stayed around 90% accuracy. And this was after the data was balanced by removing many of the non-depressed cases. Once we had these results, we felt that it was time to move on to identify the severity of depression using the same dataset.

2) Multi-Class Depression Severity

a) Balanced Dataset.

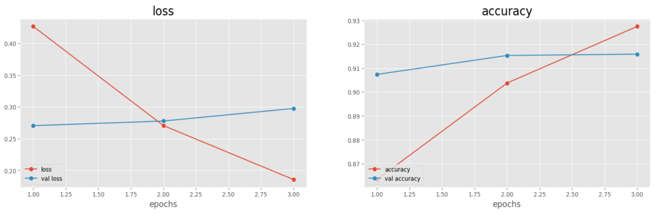
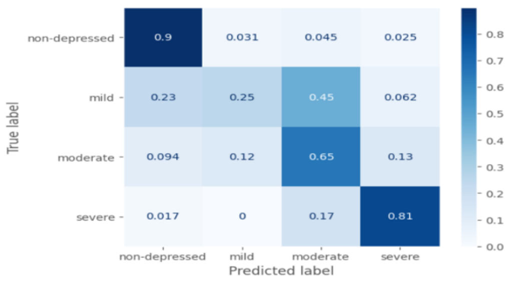


Fig. 27. Depression Severity Multi Class Balanced: Training & Validation Loss vs Epoch for BERT

As we can see the accuracy is at an acceptable 75%. We thought the model could have been improved. If you study the confusion matrix below, we will see that true severe cases are predicted as mild or non-depressed 1.7% of the time. True non-depressed cases are labelled as moderate or severe 7% of the time.



True label \ Predicted label	non-depressed	mild	moderate	severe
non-depressed	0.9	0.031	0.045	0.025
mild	0.23	0.25	0.45	0.062
moderate	0.094	0.12	0.65	0.13
severe	0.017	0	0.17	0.81

Fig. 28. Normalized Confusion Matrix for balanced depression severity dataset generated after classifying the test dataset using the trained BERT model with a dropout of 0.15

Fig. 29.

b) Non-Balanced Dataset

We hypothesised that false positives (having non-depressed classified as moderate/severe) could be removed by training the model on the imbalanced dataset where non-depressed cases were not removed for balancing. Here are the results.

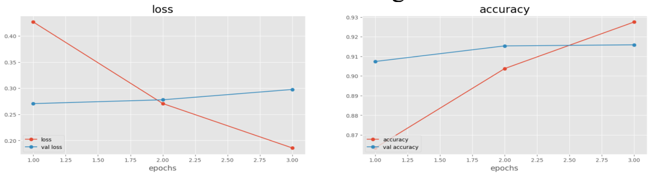
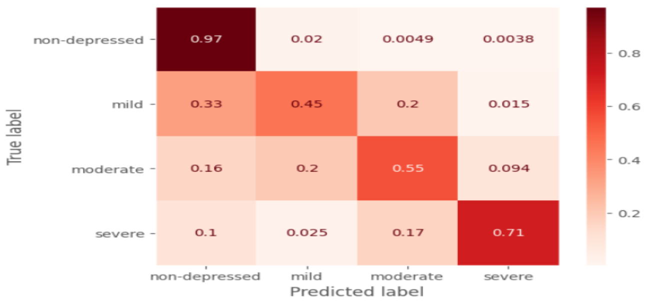


Fig. 30. Depression Severity Multi Class Balanced: Training & Validation Loss vs Epoch for BERT



True label \ Predicted label	non-depressed	mild	moderate	severe
non-depressed	0.97	0.02	0.0049	0.0038
mild	0.33	0.45	0.2	0.015
moderate	0.16	0.2	0.55	0.094
severe	0.1	0.025	0.17	0.71

Fig. 31. Normalized Confusion Matrix for imbalanced depression severity dataset generated after classifying the test dataset using the trained BERT model with a dropout of 0.15

As we can see the number of cases which were true non-depressed but predicted moderate or severe dropped to 0.0087%, down from 7% of the balanced model. However, true severe cases were detected as mild or non-depressed 12% of the time compared to the balanced model at only 1.7% more than 5 times more.

In summary the imbalanced dataset created a bias, favoring detection towards the mild and moderate cases of depression.

XII. Future Work

While the models developed provide valuable insights and good accuracy, we would still like to explore various other relevant models like Recurrent CNNs and check how well our current models are working with unknown data. The current model focusses on some of the mental issues, we intend to broaden our horizon and, if possible, develop an application that could track and notify a person about his/her mental health condition (if provided with necessary permissions).

Including Explainable AI would be a very important aspect of our future work as it improves trustworthiness and transparency and provides important insights as to how the model is making predictions or classifications, which can help the user and mental health professionals better understand the factors contributing to the mental health issues.

XIII. Acknowledgment

We would like to express our sincerest gratitude to our Prof. Christian Don Verth for his invaluable guidance throughout the Text Mining course and project which was instrumental in shaping the direction of this project.

XIV. References

[1] Mohsinul Kabir a, Tasnim Ahmed a, Md. Bakhtiar Hasan a, Md Tahmid Rahman Laskar b c, Tarun Kumar Joarder d, Hasan Mahmud a and Kamrul Hasan, "DEPTWEET: A Typology for Social Media Texts to Detect Depression Severities".

[2] Ankit Murarka, Balaji Radhakrishnan and Sushma Ravichandran, "Detection and Classification of mental illnesses on social media using RoBERTa".

[3] Jacob Devlin Ming-Wei Chang Kenton Lee and Kristina Toutanova "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

[4] Lecture NoteBook 8 (Text Mining Lecture Notes)

XV. Appendix A

How to run the code

A. *EDA and Pre-processing*

1. Ensure that the data is placed in Data folder which is in the same directory as the textPreprocessing.ipynb file.

```
train_dir = os.path.join(os.getcwd(), 'Data/both_train.csv')
val_dir = os.path.join(os.getcwd(), 'Data/both_val.csv')
test_dir = os.path.join(os.getcwd(), 'Data/both_test.csv')
```

B. *Traditional Models and Feature Importance*

1. Open the notebook named, "Traditional models - Mental Health Classification.ipynb"
2. Update the paths to the following preprocessed files:
   preprocessed_train.csv
   preprocessed_val.csv
   preprocessed_test.csv
3. Execute the cells in the notebook to see the output

C. *LSTM*

1. Open the folder *DeepNN_MentalHealthClassification* and ensure that the *data* folder, containing the *preprocessed_train.csv*, *proceprocessed_val.csv* and *preprocessed_test.csv* files is inside
2. Run *lstm_classifier.py*

D. *Bert*

1. Open notebook named, "BERT - MentalyStable.ipynb"
2. Update paths of the following files:
   a. bert_preprocessed_train.csv
   b. bert_preprocessed_val.csv
   c. bert_preprocessed_test.csv
   d. deptweet_dataset.xlsx
3. Execute code in notebook to see output. You may need a gpu to train the models otherwise it may be slow.