Final project: Sentiment Analysis with LSTMs for Mental Health Detection

Students (Team H)

- Stuart Lance Richards-Weir NIA: 288001
- Pau Peirats NIA: 254382
- Mireia Pou Oliveras NIA: 251725

Contents

GitHub

Here you have the link to our GitHub repository:

https://github.com/pdpau/DL2025-SentimentAnalysis

Introduction

Goal: The aim of this project is to build a sentiment analysis LSTM-based model focused on classifying text in order to detect potential mental health issues. The goal is to analyze sentences and relate them to mental states.

Motivation: We have chosen to do this project because we think it’s a nice way to start getting insights about one of the most important topics in medicine nowadays. Exploring solutions that technology can give to improve people's lives is the path to healthy human development.

Data: For our project, we are planning to use a “Kaggle” dataset [1] that includes people’s statements (small sentences) collected from social media, Reddit, Twitter, and others. Each statement is paired with one of the following seven mental health statuses: normal, depression, suicidal, anxiety, stress, bi-polar, or personality disorder. Initially, it has a shape of (53043, 3).

Specific goals include:

- Creating an LSTM baseline model that works as expected: classifying statements to mental health statuses
- Improving this baseline model by applying data preprocessing and architecture optimization.
- Trying to obtain better results than the benchmark models; that is, than what other people have done with our same dataset.

State of the art

Before digging into coding models, we made a deep research on what is being used recently to tackle sentimental analysis problems. We found that Recurrent Neural Networks (RNNs) and Transformers are the best approaches to this problem. To have a clear understanding of the advantages and disadvantages of each approach we have selected 3 different papers that address the issue of sentimental analysis in mental health text datasets.

Dash et al. [7] trained a single layer LSTM model with GloVe embeddings on 74.824 posts from Reddit communities about mental health. Given a post, the model must predict between 7 different categories (neutral, anxiety, bipolar, depression, PTSD, schizophrenia, suicidal).

Karamat et al. [8] used two transformer encoders (MentalBERT and MelBERT) pre-trained on mental health text and tuned for figurative language. This approach granted them a final accuracy of 93%, making transformers one of the best architectures for text classification problems.

Bendebane et al. [9] focused on combining convolutional layers for feature extraction and a Bidirectional GRU that captures the overall context of the text. This architecture has been applied to a Twitter dataset to classify tweets as normal, depressive or anxious. This hybrid approach is lighter in weight compared to transformers and obtained great results.

Standard techniques among all experiments are text preprocessing, pre-trained embeddings, dropout regularization and using accuracy and F1-score as main evaluation metrics. For our project, we have focused on LTSMs since it is the less computationally expensive architecture and still grants decent results.

Model	Classification	Accuracy	Reference
LSTM	7 classes	75-88%	[5]
MentalBERT	4 classes	93%	[6]
CNN+BiGRU	3 classes	93.38%	[7]

Table 1: State of the Art models

Benchmark

This section includes previous work using our dataset for sentiment analysis classification. There have been three different approaches to this problem, as summarized in Table 2.

Benchmark model	Architecture	Accuracy obtained	Reference
Transformers	bert-base-uncased model	0.849	[2]
LSTM-based	LSTM + two ReLU + Softmax	0.723	[3]
Machine Learning based	XGBoost	0.808	[4]

Table 2: Benchmark models

Methodology

The methodology followed for this problem is divided into three parts. First, we made an exploratory data analysis to identify key data features, detect outliers, and assess class imbalance. Following this, we developed an initial baseline model, which we will use as the base for future optimization; and, finally, we have developed some experiments to improve the baseline model.

Exploratory Data Analysis (EDA)

Our dataset consists of three columns: a unique identifier, a statement, and the corresponding mental health status. During the Exploratory Data Analysis (EDA), we focused on the last two columns.

To understand whether all mental health statuses (last column) are equally represented, we examined the class distribution, illustrated in Figure 1. As you can observe, the distribution is unbalanced. This is an issue we will need to treat when building our models, to prevent bias towards majority classes (such as normal or depression).

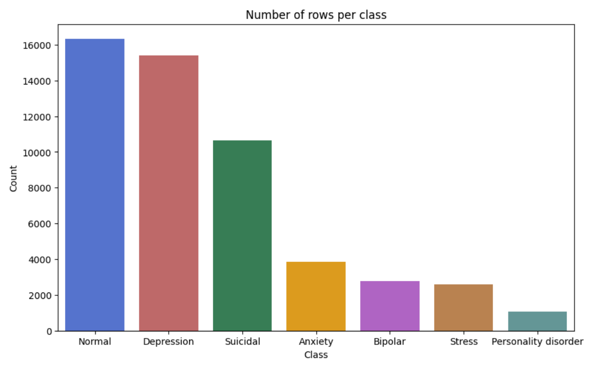

Figure 1: Class distribution.

For the second column (statements), we looked at several characteristics to better understand our data. First, we checked for null or duplicated statements, which we'll need to decide whether to remove or keep. In total, we found 362 null statements and 1403 duplicated ones.

Next, we explored a few more aspects: statement length, text noise, the most common words per class, and the most common bigrams per class. Since our data comes from social media, these insights help us understand what kind of preprocessing is needed and how the statements are typically structured. After checking all of these aspects, we arrived at the following conclusions:

1. The length of the statements across all classes is generally similar, but there are 6 outliers that could negatively impact model performance. This is shown in *Figure 2 (left)*.
2. Our dataset contains text noise; that is, parts of the text that either the model cannot interpret or that don't contribute meaningfully. Addressing this noise is crucial during preprocessing. Specifically, our text contains emojis, mentions, hashtags, etc.

3. Some classes include the class name among their most common words. This suggests that people often explicitly mention their mental health condition, which could help distinguish between classes. This is shown in *Figure 2 (right)*, a word cloud of the anxiety class, where anxiety is the most common word.

4. By examining the bigrams (the most common pairs of words in each class), we noticed that several classes share overlapping language. In other words, different mental health statuses sometimes use similar vocabulary. This could be a downside when making the classification. Overlapping language between Depression and Normal classes is shown in *Figure 3*.

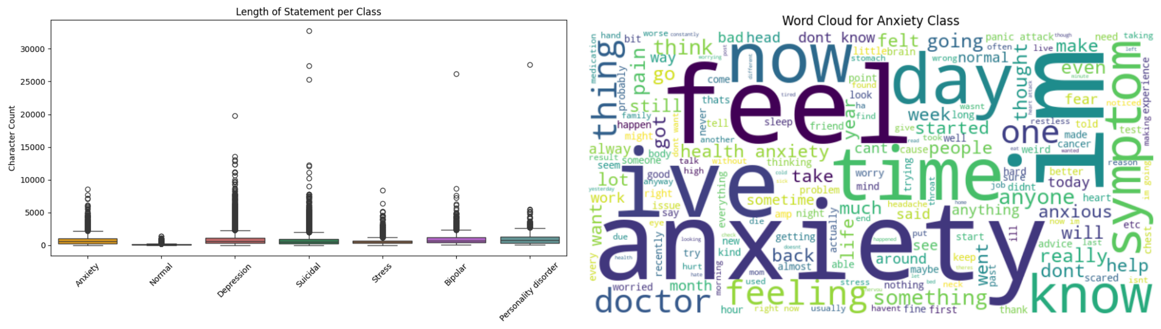

Figure 2: Length of each statement per class (left) and most common words of the Anxiety class (right).

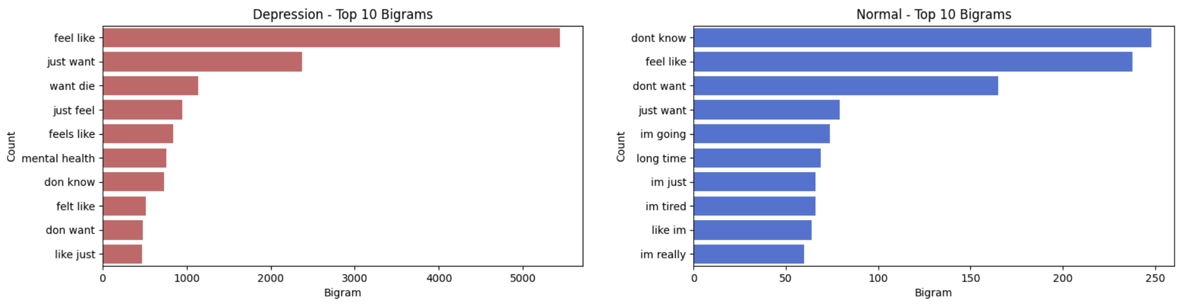

Figure 3: Most common bigrams in the Depression (left) and Normal (right) classes.

Baseline model

After completing the EDA, we removed non-useful information such as null statements and the ID column. However, aside from these basic cleanups, the baseline model is trained on **raw data**, without applying preprocessing. Its architecture is intentionally simple, serving as a starting point for further improvements. It includes the following components:

- **Embedding layer:** which helps the model convert text into a format it can understand by mapping words to vector representations. It has been implemented using a basic PyTorch embedding layer.
- **LSTM (Long Short-Term Memory) layer:** which allows the model to capture sequential dependencies and context within the statements.
- **Fully connected (linear) layer:** at the end, used to perform the classification into mental health status categories.

For the loss function, we use **Cross-Entropy Loss**, which is standard for multi-class classification problems. To evaluate the model, we rely on two main metrics: **accuracy and F1-score.** The accuracy gives us an overall idea of performance and serves as a benchmark comparison. The F1-score is more informative in our case because our dataset is unbalanced and we want to account for performance across all classes.

The experiments developed during this project are divided into two parts. First, we improved our baseline model by modifying the input data and tuning the model's hyperparameters. Then, we optimized the model by changing its architecture. The experiments are shown in the following section.

Experiments

Model improvement through data preprocessing

We made five experiments by applying different data preprocessing steps, while keeping the same model architecture. These experiments are:

1. **Data cleaning:** In this experiment, we wanted to reduce irrelevant information and help the model focus on the actual content of the statements. We removed noisy text such as URLs, emojis, HTML tags, mentions, extra spaces, and more. Additionally, we excluded the statement length outliers identified during the EDA.

2. **Stopword removal:** In this experiment we wanted to further clean the data by removing stopwords. Stopword removal is a widely used text preprocessing technique that involves taking out commonly used words that don't have any significant meaning, such as articles and prepositions. We used the English stopword list from NLTK (Natural Language Toolkit) and modified it by keeping negations, as these hold important meaning in the mental health context.

3. **Embedding selection:** Since our baseline model used a basic embedding layer, we tested whether more sophisticated embeddings would improve performance. In this experiment, we compared two word embeddings:
   - First we tried Word2Vec because it's a simple, fast, and widely used embedding method. However, as you will see in the results, this model didn't improve the baseline one.
   - Instead we tried GloVe, a more sophisticated, state-of-the-art embedding that captures semantic relationships between words more effectively.

4. **Data augmentation:** To improve generalization and treat class imbalance, we applied data augmentation to the minority classes. First, we used back translation (translating a sentence into another language and back to English) to create differently written versions with the same meaning. Since this didn't work as expected, we tried synonym replacement, where words in a sentence are replaced with their synonyms while preserving the same meaning.

5. **Grid Search:** As our final preprocessing experiment, we fine-tuned hyperparameters to better adapt the training process to our model. We used grid search to explore different combinations of hidden layer size, batch size, and learning rate.

Model improvement through architecture change

After optimizing the model's input data, we did three architectural improvements on the LSTM plus a simpler RNN (GRU) architecture. These are the experiments:

1. **Dropout:** Our previous experiments were giving decent results, although they all had something in common, they were overfitting around epochs 5-10 depending on the experiment. To treat this problem we applied dropout to the LSTM layer, a mechanism that randomly "drops out" some % of neurons preventing them from becoming too specialized or overly dependent on some features of the training data.

2. **Stacked LSTM:** After dropout solved the issue of overfitting, we wanted the model to understand more complex relationships by stacking 2-5 LSTM layers together. It did improve the single layer model but not substantially.

3. **Attention layer:** Now that we had the strongest model since the start of the experiments, we wanted to add an attention layer after each LSTM layer to help the output layer with the

classification. An attention layer gives weights to the most meaningful words in the sentence, therefore reducing the network's confusion.

4. **GRU:** Having obtained our good results with LSTM architecture, for our last experiment we tried a slightly different approach. GRUs are also RNN but they have only two gates, simpler than LSTMs. We wanted to see if a less complex model could work better with our dataset.

To be able to compare all models, we used Pytorch and random seeds.

Results

In this section, we present the results of all conducted experiments. First, you will find a table with the model name, the best and final validation accuracies obtained, the final validation loss, and the weighted F1-score. The table highlights in green all experiments that improved upon the previous experiment.

	Model name	Best validation accuracy	Final validation accuracy	Final validation loss	Weighted F1-score
0	Baseline	0.7524	0.7332	1.5110	0.7355
1	Data cleaning	0.7566	0.7408	1.3936	0.7412
2	Stopword removal	0.7539	0.7394	1.4413	0.7384
3.1	Embedding (Word2Vec)	0.7493	0.7364	0.8504	0.7357
3.2	Embedding (GloVe)	0.7779	0.7744	0.6665	0.7753
4	Data augmentation	0.7590	0.7442	0.8882	0.7476
5	Grid Search	0.7818	0.7735	0.6010	0.7769
6	Dropout	0.7790	0.7773	0.5892	0.7774
7	Stacked LSTM	0.7820	0.7789	0.5835	0.7795
8	Attention	0.7804	0.7715	0.6420	0.7732
9	Baseline GRU	0.7627	0.7388	1.7622	0.7376
10	Improved GRU	0.7932	0.7684	0.9345	0.7682

Table 3: Results table.

Now we present all the model's loss and accuracy curves, and their confusion matrix (Figures 4-13):

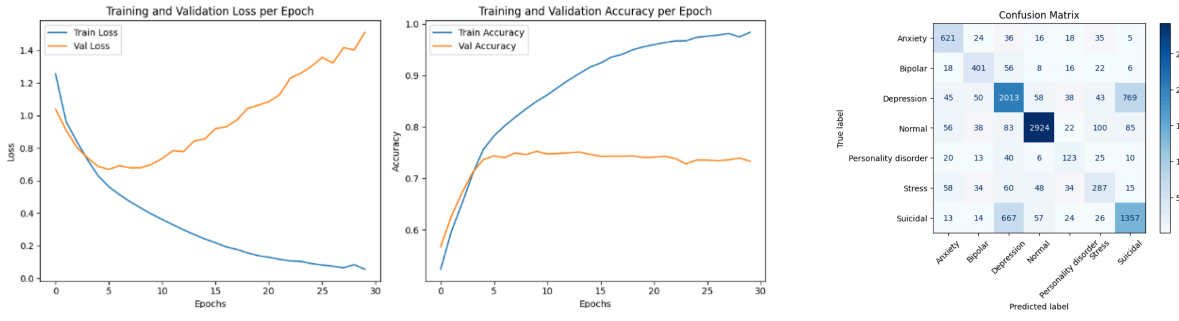

Figure 4: **Baseline model**. Loss curve (left), accuracy curve(middle) and confusion matrix (right)

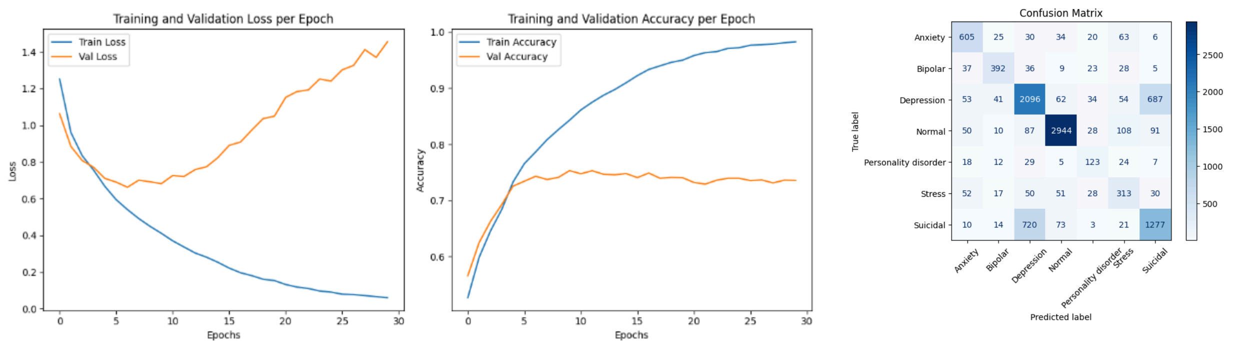

Figure 5: **Data cleaning model**. Loss curve (left), accuracy curve(middle) and confusion matrix (right)

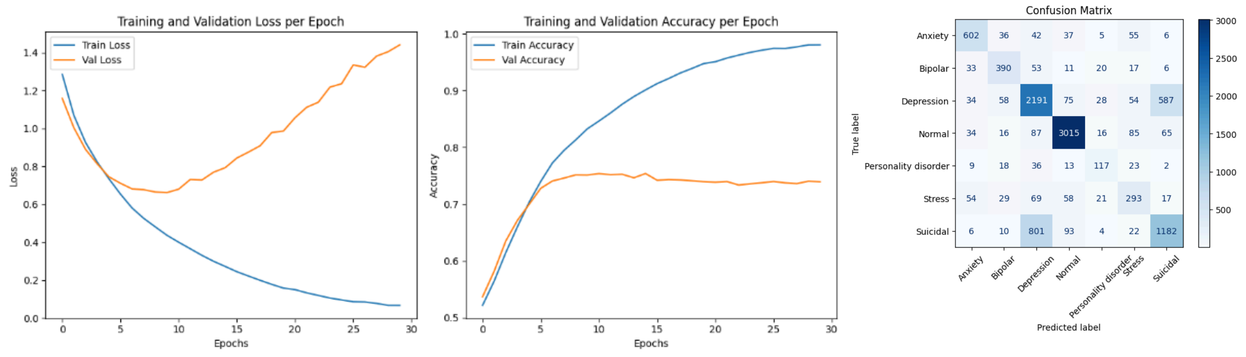

Figure 6: **Stopword removal model**. Loss curve (left), accuracy curve(middle) and confusion matrix (right)

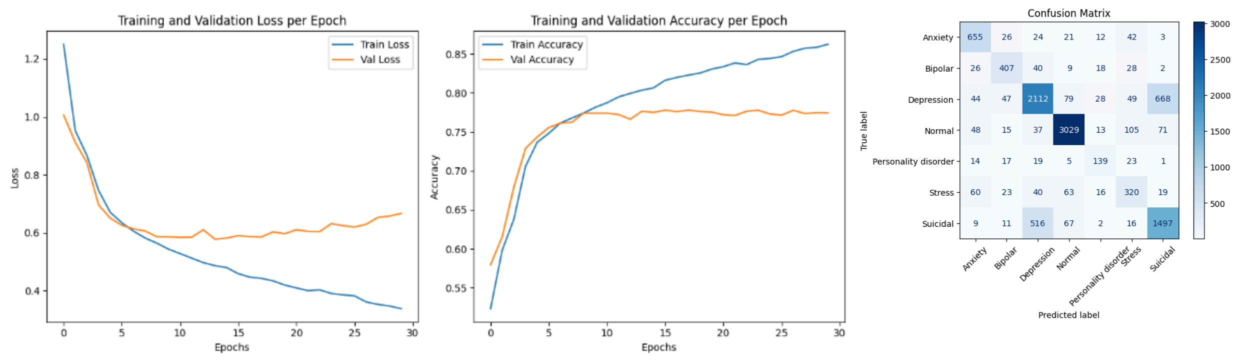

Figure 7: **Embedding (GloVe) model**. Loss curve (left), accuracy curve(middle) and confusion matrix (right)

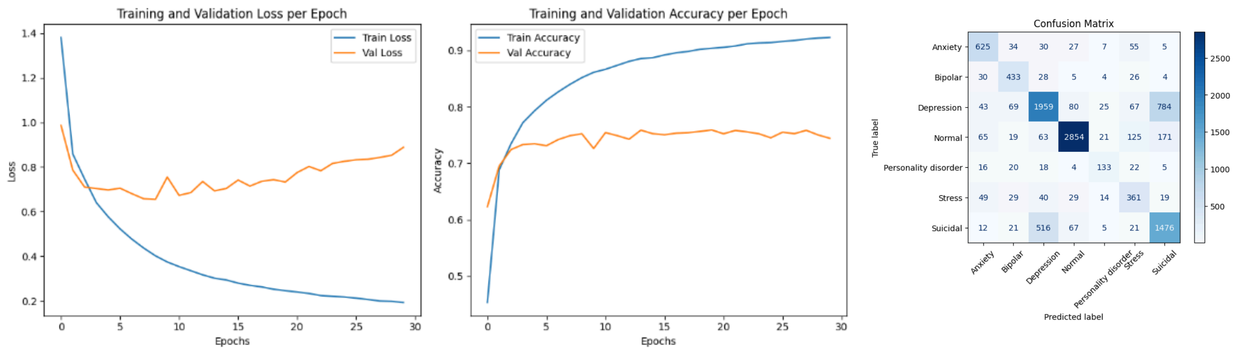

Figure 8: **Data augmentation model**. Loss curve (left), accuracy curve(middle) and confusion matrix (right).

For the grid search model, we have found that the best hyperparameters were:

- Hidden dimension=64
- Batch size=128
- Learning rate=0.001

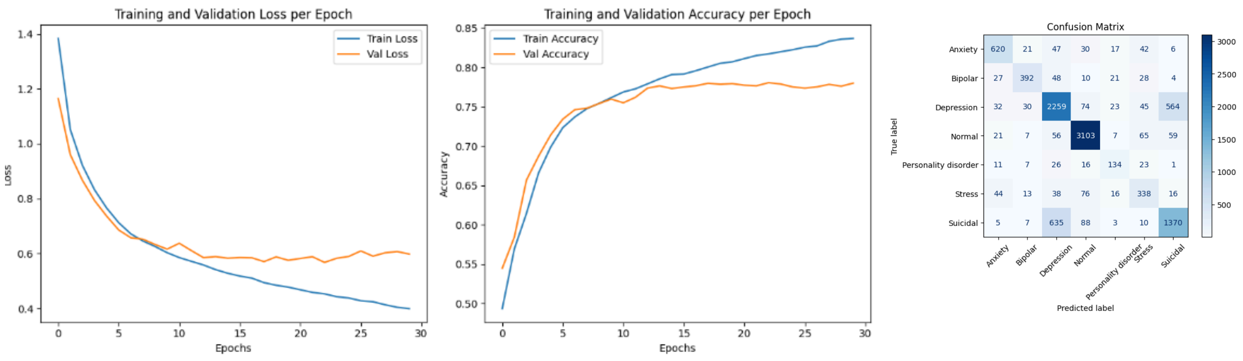

Figure 9: **Grid search model**. Loss curve (left), accuracy curve(middle) and confusion matrix (right).

For the dropout model, we have found that the best dropout value for the LSTM was 0.3.

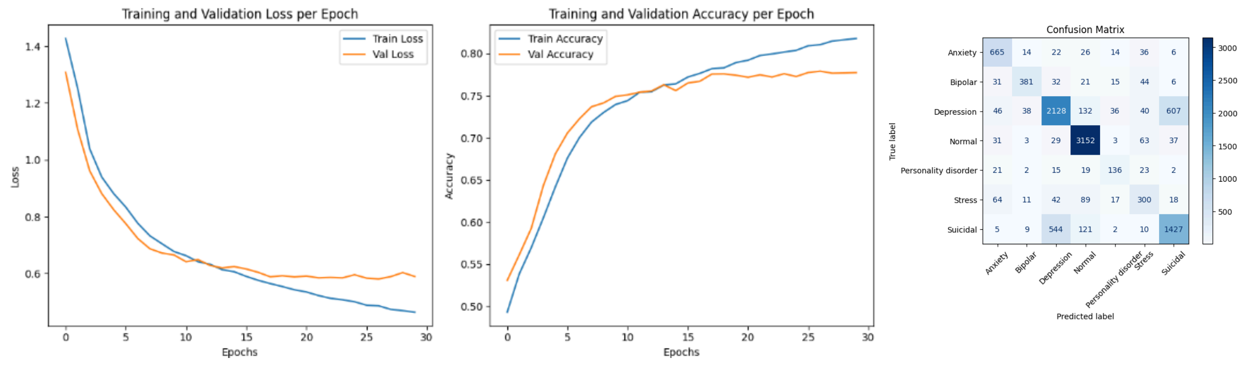

Figure 10: **Dropout model**. Loss curve (left), accuracy curve(middle) and confusion matrix (right).

For the Stacked LSTM model, we have found that the best number of layers were 4.

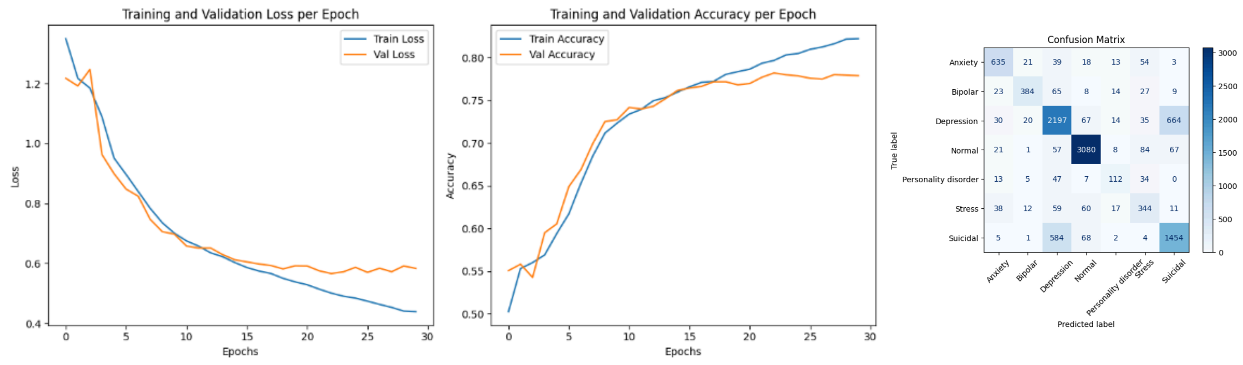

Figure 11: **Stacked LSTM model**. Loss curve (left), accuracy curve(middle) and confusion matrix (right).

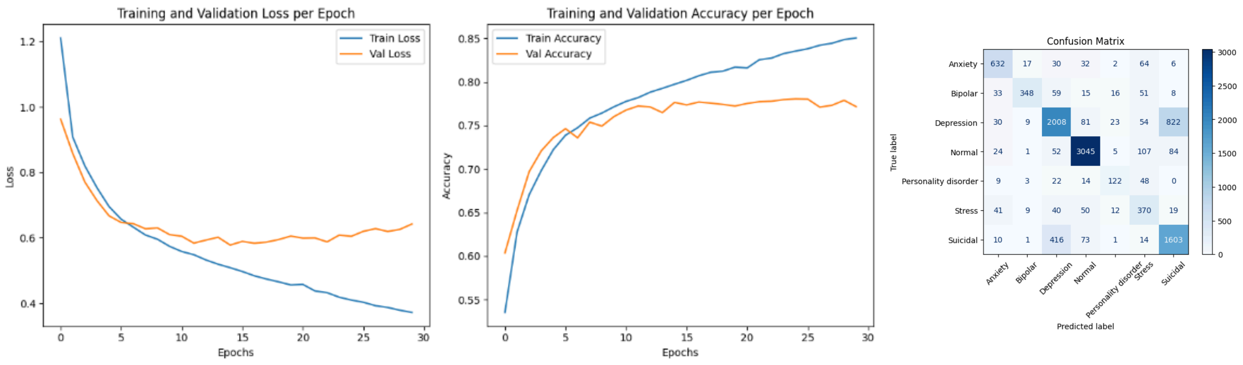

Figure 12: **Attention model**. Loss curve (left), accuracy curve(middle) and confusion matrix (right).

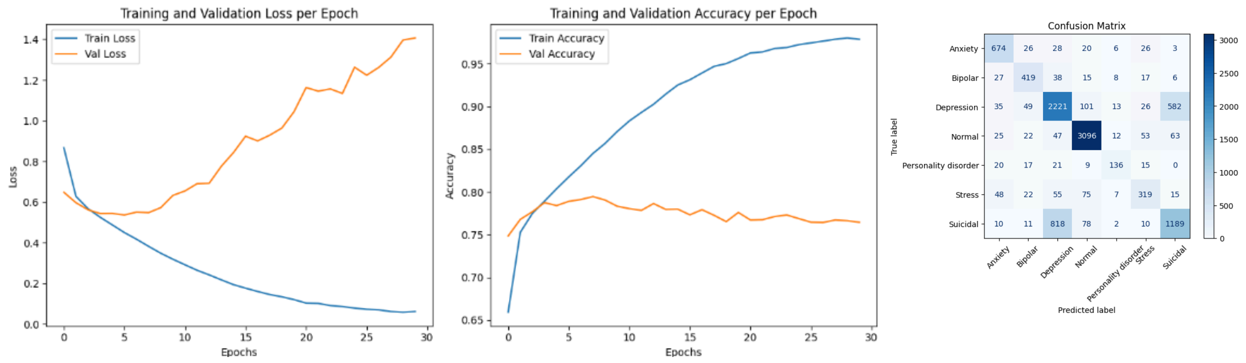

Figure 13: **Improved GRU model**. Loss curve (left), accuracy curve(middle) and confusion matrix (right).

<u>Discussion</u>

We are really happy to say that all of our models, even those that didn't outperform their immediate previous, showed improvements over the baseline model. This means that even small changes in data processing or configuration can positively impact the model's performance.

One of the most important things we have seen throughout the experiments is how important data preprocessing is. With the first five experiments (focusing on data changes and model tuning), we were able to:

- Improve the final validation accuracy from 0.7332 to 0.7735
- Improve the weighted F1-score from 0.7355 to 0.7769
- Decrease the final loss from 1.5110 to 0.6010

These gains were achieved without altering the model architecture, showing how impactful it can be to simply give the model cleaner and more meaningful data. That said, not all preprocessing steps had the same effect. For example, stopword removal didn't help overall performance. In fact, it slightly harmed it. This is likely because, in the context of mental health, even common words like pronouns carry meaning. The confusion matrix (comparing Figure 5 right with Figure 6 right) showed an improvement in the Normal class, but performance for other classes dropped. This suggests that removing too much context hurts the model's performance.

Data augmentation also didn't produce the desired improvement. The goal was to address class imbalance and help the model generalize better. We initially tried back translation, but it wasn't practical as it took more than 3 hours to process just 1000 samples. We then applied synonym replacement, which works well in general Natural Language Processing tasks. Nevertheless, in our case (mental health context), some synonyms can alter the meaning of a statement, especially when words closely relate to specific statuses - as we have seen in the EDA. The confusion matrix (Figure 8 right) shows that nearly all class performances dropped, likely due to the extra "noise" introduced.

On the other hand, embedding selection and hyperparameter tuning had a positive impact. By using GloVe embeddings, the model gained a better semantic understanding of the input data. Combined with grid search, where we fine-tuned the hidden size, batch size, and learning rate, we significantly reduced overfitting. The best configuration included a:

- Normal-sized hidden layer, which works well since mental health statements aren't usually long. So, the hidden layer size it's not too small to underfit nor too large to overfit.
- Larger batch size, which helped stabilize training caused by the diversity of social media texts. A bigger batch size has less noise during training.

Going onto the architectural change models, to further improve performance we introduced dropout. Our goal was to eliminate the small overfitting that remained after grid search (Figure 9 left and middle). This worked well, as seen in the improved loss and accuracy curves (Figure 10 left and middle). The dropout value that provided the best performance was 0.3, which is neither too high nor too low.

Regarding stacked LSTM, the results' table also shows a slight improvement in both accuracy and weighted F1-score. The plots and confusion matrices (Figure 11) look quite similar to the dropout ones, but by having a deeper architecture, the model could capture more difficult patterns in the data, such as irony.

However, the last two models did not improve performance. Contrary to our expectations, adding an attention layer did not work. It likely introduced too much complexity for the model to handle effectively, leading to worst generalization, as seen in the loss and accuracy curves (Figure 12).

Finally, we tested a GRU-based model to explore a different architecture, which also didn't perform well and showed signs of overfitting once again (Figure 13). Given that GRUs are a simpler architecture compared to LSTMs, it's not surprising that they couldn't match the performance of our tuned LSTM models. It's possible that the GRU would need additional architectural tuning or preprocessing adjustments to perform as better as the Stacked LSTM model.

Future work

We think that there are several directions we could take our work in from here. The most natural next step would be to try and test Bidirectional LSTMs (BiLSTMs) which work by processing the text in both forward and reverse, potentially capturing richer and deeper depth of context; they can "enhance context comprehension by considering past and future text simultaneously, significantly improving sentiment analysis" [5]. By trying this technique, we think we could improve upon our current LSTM model in more nuanced or ironic statements, since their context requires more examination.

Further, as mentioned previously in the state of the art section, we could look to implement our own transformer-based models (e.g. BERT, DistilBERT) which are shown to have high accuracy and bidirectional attention mechanisms. These could perform worse computationally but we believe that if we were to fine-tune them on mental health data, we could probably improve on our results. We would also be able to look further into a cost-benefit analysis of higher computational power vs accuracy.

Lastly for other models, we have also seen a popular architecture of Ensembling methods, which work by stacking different models together (e.g. CNN + BiGRU, seen in SotA). We have seen combinations like CNN + LSTM or averaging the outputs of multiple architectures to improve accuracy whilst reducing overfitting. To add to this, we could also try to implement attention layers, which may further help focus on emotionally strong words. By building these models, we would seek to gain the strengths of each individual model and reduce overfitting at the same time.

Pretraining is also an area we think is worth investigating, namely domain-specific pretraining. We used embeddings such as GloVe and Word2Vec, but we think we can expand on this by using mental health corpuses in order to start with a more relevant semantic representation. We could potentially use Reddit and other depression/label-specific forums for gathering relevant data. Or, by fine-tuning word embeddings specifically on text related to mental health, we could better capture the meaning behind slang words and disorder-specific terms that aren't so frequently used in general corpuses.

Aside from the underlying model, for more practical purposes, we would seek to integrate interpretability/explainability models. Giving clinicians the reasons for why a statement was labelled as it was, for example "depression" instead of "anxiety" , would greatly improve the models used for treating such cases professionally. We saw that tools like LIME and SHAP were capable of highlighting the important words and phrases during classification [6]. Adding tools like this would help build trust in the system and could also be used to provide insights into the language markers of mental health conditions.

Conclusions

Throughout the project, we have seen different models, architectures, and performance. Overall, we have arrived at the following conclusions:

1. **LSTM-based models can be effectively used for mental health classification:** Through our experiments, we showed that even the baseline LSTM architecture can perform well in identifying different mental health statuses from social media text.

2. **Data quality is just as important as model complexity:** We found that thoughtful data preprocessing (such as cleaning noisy inputs, using better embeddings, and tuning training parameters) had as big an impact on performance as adding architectural complexity to the model itself.

3. **Preprocessing must be adapted to the context of mental health:** Some common NLP techniques, like stopword removal and data augmentation, did not improve our model and even harmed performance. This is likely because the language used in mental health discussions is context-sensitive, and removing/altering certain words can change the meaning of a statement.

4. **Detecting mental health issues through text is challenging**: Our project shows that it's possible to detect mental health statuses from written statements with good performance, but even after several improvements, our best accuracy was 0.7820.

5. **Our final models outperform the baseline and some benchmark results:** After all preprocessing and tuning, our best model significantly improved over the baseline one. While we didn't outperform the benchmark transformer-based model or the ML-based one, our LSTM-based approach performs better than the benchmark LSTM.

Referring back to our original objectives, we effectively created an LSTM baseline model that classifies statements into mental health statuses, we improved this baseline by applying changes and, finally, we tried to outperform the benchmark results (although it hasn't been possible in some cases). It would be great to check if the future work proposed indeed improves the benchmark models.

References

[1] https://www.kaggle.com/datasets/suchintikasarkar/sentiment-analysis-for-mental-health

[2] https://www.kaggle.com/code/grantgonnerman/mental-health-sentiment-analysis-eda-modeling

[3] https://www.kaggle.com/code/rafaeldrago/lstm-sentiment-prediction/notebook

[4] https://www.kaggle.com/code/hnfrmdhni/klasifikasi-jenis-depresi

[5] https://link.springer.com/article/10.1007/s10462-023-10651-9

[6] https://www.nature.com/articles/s41746-023-00751-9

[7] Dash, R., Udgata, S., Mohapatra, R. K., Dash, V., & Das, A. (2025). A Deep Learning Approach to Unveil Types of Mental Illness by Analyzing Social Media Posts. Mathematical and Computational Applications, 30(3), 49. https://doi.org/10.3390/mca30030049

[8] A. Karamat, M. Imran, M. U. Yaseen, R. Bukhsh, S. Aslam and N. Ashraf, "A Hybrid Transformer Architecture for Multiclass Mental Illness Prediction Using Social Media Text," in IEEE Access, vol. 13, pp. 12148-12167, 2025, doi: 10.1109/ACCESS.2024.3519308. https://ieeexplore.ieee.org/abstract/document/10804794

[9] Bendebane, L., Laboudi, Z., Saighi, A., Al-Tarawneh, H., Ouannas, A., & Grassi, G. (2023). A Multi-Class Deep Learning Approach for Early Detection of Depressive and Anxiety Disorders Using Twitter Data. Algorithms, 16(12), 543. https://doi.org/10.3390/a16120543