Project Report: Sentiment Analysis for Mental Health

Zein Elabedine Hamdan, Mohammad Yamout, Hassan Hobballah

Dr. Rida Assaf

Abstract— This project addresses sentiment analysis for mental health, aiming to accurately classify text statements into seven different mental health categories. To achieve this, we employed various machine learning and deep learning techniques; namely Logistic Regression, Random Forest, XGBoost, Neural Networks, and Transformer Based Neural Network (Light Weight). We preprocessed the data, tailoring it to each of these models. We generally performed data cleaning, synonym-based augmentation, stemming, and stopword removal, along with label encoding for classification. We conducted hyperparameter tuning for the models to optimize their performance. We reached promising accuracies with some models, especially with the Transformer Neural Network.

Contents

1 Introduction

This project focuses on the task of sentiment analysis for mental health, where the goal is to classify text statements into seven distinct mental health categories based on their linguistic and emotional content: Normal, Depression, Suicidal, Anxiety, Stress, Bi-Polar, and Personality Disorder. The complexity of mental health expression is characterized by many challenges some of which are diverse emotional tones and ambiguous language. The presented problems are hard to tackle using traditional classification methods, but the advancements reached in machine learning offer promising solutions to address the stated challenges and minimize the error in identifying the given mental health cases.

2 Dataset

2.1 Description of original Dataset

The dataset comprises user-submitted statements classified into seven categories: Normal, Depression, Suicidal, Anxiety, Bipolar, Stress, and Personality disorder. The classes are highly imbalanced, with majority categories like Normal and Depression containing significantly more instances than minority classes such as Personality disorder. This imbalance is problematic because it might not learn enough about the minority classes to accurately classify them and only learn patterns on the majority class. Moreover, while there are some missing instances, their limited number compared to the entire dataset is unlikely to substantially impact the overall model performance.

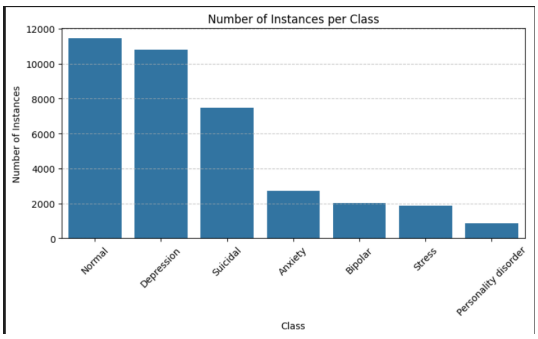

Figure 1: Class Imbalance Present

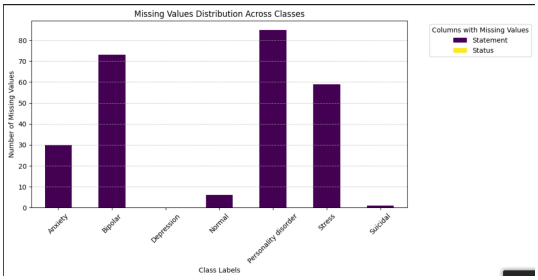

Figure 2: Missing values in each class

2.2 Data Preprocessing

To improve the overall data quality and balance, we dropped rows with missing statements. Dropping these missing instances is a reasonable approach in textual data since their absence does not convey useful sentiment information and only introduces noise. Beyond this, we handled the imbalance by undersampling the majority classes to 8,000 instances each. We found it a reasonable number to downsample to not create many instances in oversampling and not lose much data. Then, applying synonym-based data augmentation using WordNet to oversample the minority classes, which we did according to a defined function using the existing data and choosing whether to replace the word with its synonym or keep it with a fixed probability. The synonyms are obtained from WordNet, a large lexical database available through the NLTK library. This makes sure that we added "new" data to our dataset and not the normal SMOTE technique. Additionally, the text was preprocessed by cleaning non-alphabetic characters, converting to lowercase, removing stopwords, and applying stemming. This combined strategy resulted in a balanced and cleaned dataset (before about 37,000 instances, now 56,000 instances), ultimately saved as Adjusted_Sentiment_Dataset.csv.

3 Models Insights

3.1 Logistic Regression

Logistic Regression was chosen as our baseline model because it is simple, interpretable, and effective for initial evaluations. In our project, it estimates the probability of a statement belonging to a certain class, providing a clear starting point for comparing and improving more complex models.

3.1.1 Training and Testing

- We trained a Logistic Regression model using TF-IDF to convert text into numerical features. We split the dataset into 80% for training and 20% for testing, making sure each split kept a similar class distribution.

- After training, and trying to change learning rate and max iterations, the model achieved an accuracy of 86.14% on the training set and 81.02% on the test set (reported best one). This indicates good generalization performance across unseen data.

- A classification report showed strong performance across all classes, with higher precision and recall for dominant classes such as Bipolar, Personality disorder, and Normal, and relatively lower scores for Depression and Suicidal, which is expected as these classes had major augmentation (but performed way better than without augmentation where the recall and precision for these classes were poor before augmentation).

3.1.2 Rationale

To evaluate how the model's performance varies with different training sizes, a learning curve was plotted. This showed a convergence between training and validation accuracy as the training data increased, confirming that the model is neither overfitting nor underfitting and the model is actually learning with more data. This provided a good baseline performance to compare with the other models that we will test next.

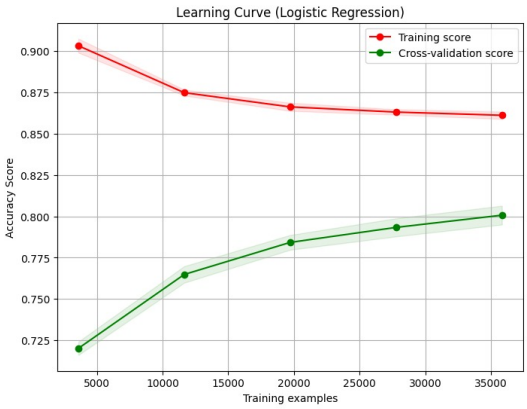

Figure 3: Logistic Regression Learning Curve

3.2 Random Forest

Next, we tried Random Forests—a model that builds multiple decision trees and averages their results. This approach helps capture complex patterns in the data, making it a strong alternative to our simpler baseline model and a new candidate.

3.2.1 Training, Tuning, and Testing

- We trained the Random Forest model using the preprocessed dataset with TF-IDF vectorization. The model was fine-tuned through a Randomized Search over hyperparameters such as the number of estimators, maximum depth, and minimum samples for splitting and leaf nodes. We couldn't increase the number of searches in randomized or perform grid search because of the massive training time and limited hardware (which was a limitation).

- The model achieved an accuracy of 99.05% on the training set and 82.22% on the test set (also using cross validation in this model), showing major overfitting but better testing accuracy than logistic regression. We reported the best model, which had the highest testing accuracy out of the randomized search after tuning the hyperparameters in the grid many times.

- A classification report indicated high precision and recall for most classes, particularly for Anxiety, Bipolar, and Personality disorders (which were an issue before, especially since they are the minority class), while Depression and Suicidal classes showed relatively lower performance, highlighting that the 2 classes can be tricky to distinguish between them (the model made the most mistakes in these classes shown in the confusion matrix).

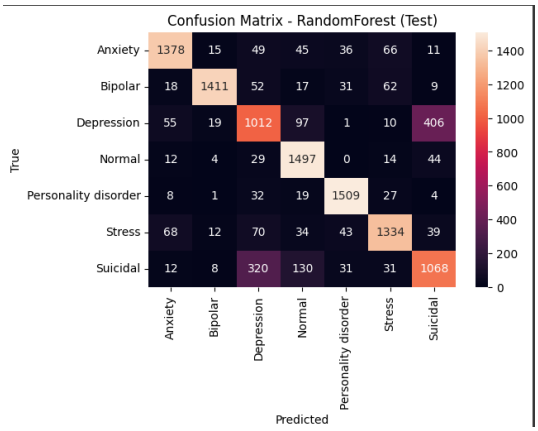

Figure 4: Random Forest Confusion Matrix

3.2.2 Rationale

Random Forests is a more complex model, so some overfitting was expected. We tuned hyperparameters such as tree depth and the number of estimators using a limited randomized search due to hardware constraints. Although tuning improved the training accuracy significantly, it also resulted in a considerable drop in testing accuracy. Consequently, we reported the highest testing accuracy observed during tuning before moving on to explore additional models. The graph below shows the major overfitting the model suffers.

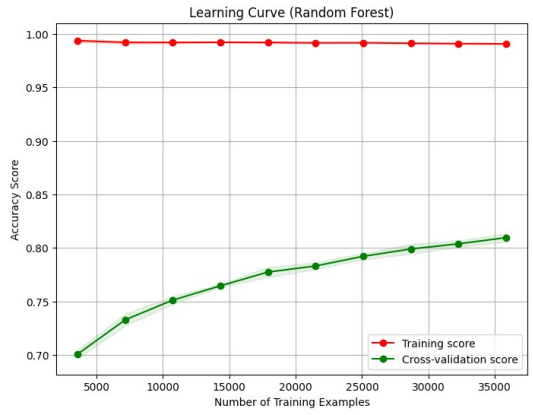

Figure 5: Random Forest Learning Curve

3.3 XGBoost

Next, we wanted to try XGBoost for penalizing incorrect answers at every iteration, as done in Random Forest. It is an implementation of gradient boosting decision trees. Making it a could candidate after random forest and expected to perform better.

3.3.1 Training, Tuning, and Testing

- We first tried to implement a randomized search similar to the ,random forest approach but the computation time was large, so we had to divert to using fixed hyperparameters and manual checking.

- The model was trained on the pre-processed and augmented dataset, which was split into an 80% training set and a 20% test set using a stratified split to preserve class distribution.

- We trained the XGBoost model with a fixed set of hyperparameters, including 350 estimators, a maximum depth of 6, and a learning rate of 0.1. This came after manually changing these hyperparameters multiple times to get a better model than random forest.

- The model achieved an accuracy of 92.07% on the training set, indicating strong performance on the training data. The test accuracy was 82.11%, which is similar to the test accuracy of the random forest, however the main difference is the overfitting here is much lower.

- Report about XGBoost showed better precision, recall, and F1-score values on the classes of major ambiguity (Depression and suicidal). But with the same testing accuracy across all classes.

- We wrote a section for using XGBoost with randomized search but couldn't increase the number of running iterations (6 fits) due to the limits of the hardware on the server (time-consuming more than Random Forest).

3.3.2 Rationale

We chose XGBoost as a follow-up to Random Forest due to its ability to penalize incorrect predictions more effectively through boosting, making it a strong candidate for improving performance. Despite hardware limitations that prevented extensive hyperparameter tuning, our manually tuned model outperformed Random Forest in key areas such as precision, recall, and F1-score—especially on ambiguous classes like Depression and Suicidal. This indicates that XGBoost provided a more balanced and generalizable model, with reduced overfitting and comparable overall accuracy.,

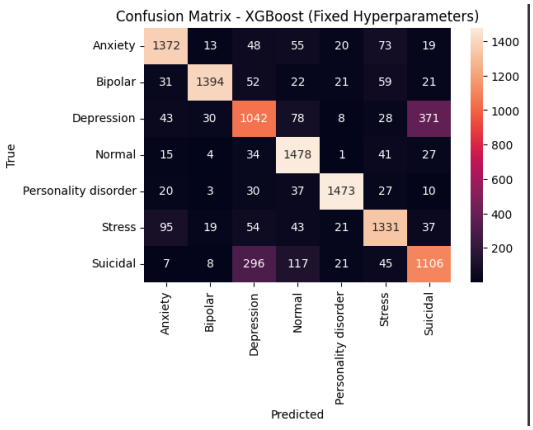

Figure 6: Confusion matrix of XGBoost

3.4 Neural Network from Scratch

Next, we wanted to try creating a neural network with a couple of layers and test it out using the TF-IDF features we generated earlier. Theoretically, a neural network should perform better by capturing non-linear relationships and complex feature interactions in the data, which classical models like Random Forest and XGBoost might miss. Given its flexibility and capacity to learn from high-dimensional data, we considered it a strong candidate to push performance further beyond tree-based models.

3.4.1 Architecture and Implementation

- We built a simple feedforward neural network using two hidden layers with 64 and 32 neurons respectively, combined with ReLU activations and Dropout layers to reduce overfitting.
- The model was trained on TF-IDF vectorized features of the pre-processed and augmented dataset. A validation split of 10% from the training data was used during training to monitor performance.
- We used the Adam optimizer and sparse categorical cross-entropy loss, appropriate for multi-class classification.
- To avoid overfitting, we implemented early stopping with a patience of 6 epochs, monitoring the validation loss. We experimented with increasing the number of epochs, changing dropout rates, and adjusting the number of neurons and layers, but the current configuration provided the best balance between performance and training time.
- The model achieved a training accuracy of 89.4% and a test accuracy of 83.50%, slightly outperforming Random Forest and XGBoost in generalization while maintaining competitive performance.
- Notably, the neural network produced slightly better performance across all classes with balanced precision, recall, and F1-scores, particularly on difficult categories like Depression and Suicidal.
- The learning curves showed stable training without significant overfitting, indicating that dropout and early stopping helped improve generalization.

3.4.2 Rationale

The neural network was selected for its potential to learn complex, non-linear patterns in high-dimensional TF-IDF features. Although it required more experimentation with architecture and training parameters, the final model offered slightly better test accuracy and more stable predictions on ambiguous classes compared to tree-based models, confirming its value in our pipeline of getting better results of every model.

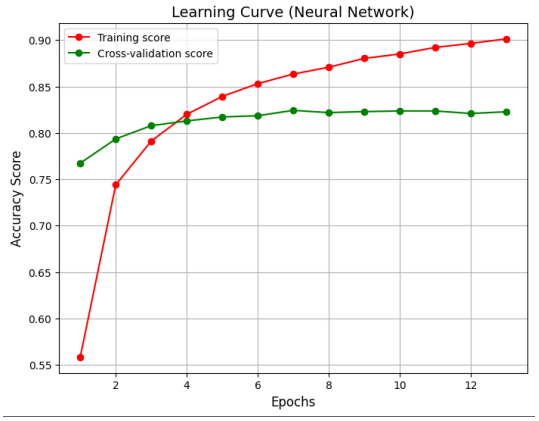

Figure 7: Learning Curve of Neural Network

3.5 Transformer-Based Neural Network

Transformer-based neural networks can track relationships between different parts of the input regardless of their position in the sentence. This motivated us to use this architecture for our classification task, as it captures context and dependencies between distant words much better than RNNs or LSTMs. So, the Transformer-based Neural Network helps us answer which part of the sentence we should focus on.

Our transformer model consists of seven core layers:

1. **Token Embedding Layer:** We convert each word in a sentence into a 300-dimensional vector using the pre-trained GloVe embeddings. This gives the model an initial understanding of word meanings.
2. **Positional Embedding Layer:** Since transformers do not have a natural sense of word order, we add positional encodings to the token embeddings to inform the model about the position of each word in the sentence.
3. **Multi-Head Self-Attention Layer:** Multiple attention heads analyze different aspects of the sentence simultaneously, learning contextual relationships between words.
4. **FeedForward Neural Network (FNN):** A small two-layer neural network is applied to each word vector independently to learn higher-level features. The first

layer increases dimensionality for learning complex relationships, while the second restores the original dimension (300).

5. **Attention Pooling Layer:** This layer summarizes the importance of each word into a single vector representation, giving higher weight to more influential words.

6. **Dense Layer (128 Neurons, ReLU):** This layer helps extract abstract features from the attention summary vector.

7. **Output Layer (SoftMax):** The final layer outputs probabilities for the seven classes. We select the class with the highest probability.

While training this model, we encountered overfitting due to its complexity. To mitigate this, we applied dropout and regularization techniques in several layers. Despite this challenge, the model showed promising accuracy and generalization, outperforming some classical models before on testing and less overfitting.

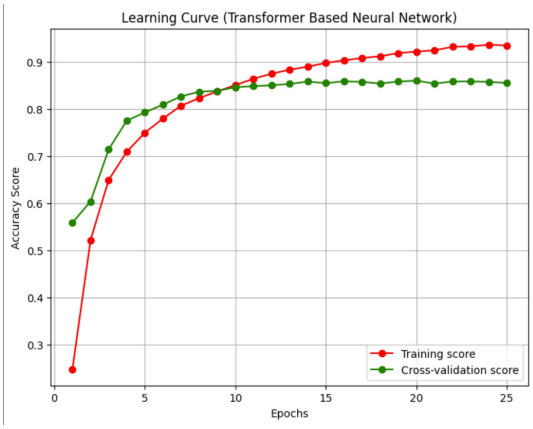

Figure 8: Learning Curve of Transformer Based Neural Network

4 Results

4.1 Findings

- **Logistic Regression:**
  - **Performance:** Achieved a training accuracy of 86.14% and a test accuracy of 81.02%.
  - **Observation:** Performed reliably on the majority classes but had difficulty with more nuanced or less-represented categories.(Good baseline)
- **Random Forest:**
  - **Performance:** Recorded an exceptionally high training accuracy of 99.05% coupled with a test accuracy of 82.22%.
  - **Observation:** The model tended to overfit the training data. Hyperparameter tuning helped reduce overfitting but majorly impacted testing accuracy.
- **XGBoost:**
  - **Performance:** Reached a training accuracy of 92.07% and a test accuracy of 82.11%.
  - **Observation:** Produced similar test accuracy to Random Forest but with reduced overfitting and better precision, recall, and F1-scores for more challenging mental health categories.
- **Neural Network from Scratch:**
  - **Performance:** Achieved a training accuracy of 89.4% and a test accuracy of 83.50%.
  - **Observation:** This model better captured nonlinear relationships in the TF-IDF features, offering a more balanced performance across classes with minimal overfitting(compared with the tree models) due to the use of dropout and early stopping.
- **Transformer-Based Neural Network:**
  - **Performance:** The transformer model achieved strong results, with an accuracy of 92% on the training set and 86.5% on the test set, indicating solid generalization even on ambiguous classes.
  - **Observation:** Its ability to capture long-range dependencies and context improved the classification of categories with subtle linguistic differences. The integration of pre-trained GloVe embeddings and positional encodings proved effective in enhancing its performance.

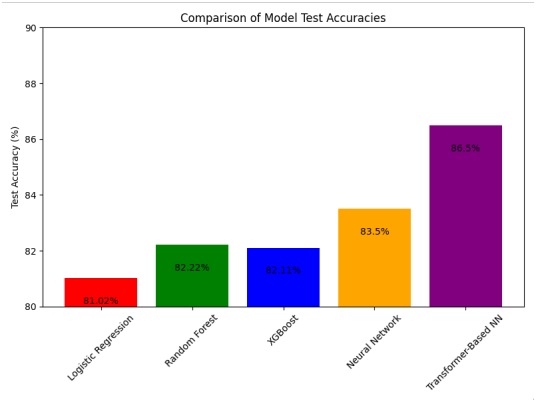

Figure 9: Comparison of Model Performances

4.2 Challenges

- **Class Imbalance:**
  - The original dataset featured a heavy imbalance with majority classes dominating the minority

ones. This necessitated undersampling the majority and applying synonym-based data augmentation to better balance the classes.

- **Hardware Limitations:**
  - The ability to perform extensive hyperparameter tuning, such as large-scale randomized searches or grid searches, was restricted by limited computational resources (on google collab). This particularly affected models with high complexity, like Random Forest and XGBoost.
- **Nuanced Language in Categories:**
  - Categories such as Depression and Suicidal contained ambiguous language and overlapping features, making them especially challenging to distinguish(usually they are similar in text). This affected all models, though improvements were noted when using advanced deep learning approaches.
- **Complexity vs. Performance Trade-off:**
  - While deep learning models like the transformer-based neural network demonstrated enhanced performance on nuanced classes, they also introduced additional complexity and training overhead compared to traditional machine learning approaches.

5 References

1. Neptune.ai Blog, *Data Augmentation for NLP*. Available at: https://neptune.ai/blog/data-augmentation-nlp.
2. Built In, *What Is a Transformer Neural Network?*. Available at: https://builtin.com/artificial-intelligence/transformer-neural-network.
3. Princeton WordNet (used for creating the new data). Available at: https://wordnet.princeton.edu/.
4. Your GitHub Repository. Available at: https://github.com/Zeinh5/CMPS261-Project.git.