BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Early Discovery of Anxiety/Depression in Teenagers Using Digital Tools

– MIRPR report –

Toth Alexandra-Melania
Vlădău Andra-Ioana
Mihalcea Ioan-Alexandu
Emails: alexandra.melania.toth@stud.ubbcluj.ro,
andra.vladau@stud.ubbcluj.ro, ioan.mihalcea@stud.ubbcluj.ro
Specialisation: Computer Science Romanian

2024-2025

Contents

List of Figures

Chapter 1

Introduction

1.1 Objective:

Anxiety/Depression for teenagers - To enhance mental health support for teenagers by developing digital tools that can proactively identify signs of anxiety and depression. These tools aim to engage adolescents in their digital environments, whether through chatbots, social media, video games, or other innovative platforms, to provide early intervention and emotional support.

1.2 Main Idea:

The Early Discovery of Anxiety/Depression in Teenagers solution uses digital platforms to identify and monitor mental health challenges among adolescents. The system combines multiple approaches, including AI-powered chatbots capable of conversational analysis, sentiment evaluation through social media interactions, and mental health assessments embedded in video game experiences. By engaging teenagers where they spend most of their time, be it online or in games, the system aims to provide real-time insights and early warnings of anxiety or depression. This integrated solution offers a holistic approach, blending digital engagement with predictive analytics and personalized intervention strategies.

1.3 Motivation:

Teen anxiety and depression are pressing issues, often going unnoticed due to social stigma, lack of awareness, or teens' reluctance to seek help. Traditional detection methods are limited in reach and timing, missing critical opportunities for early intervention. By embedding intelligent algorithms in digital spaces where teens naturally engage, like social media, chat apps, and video games, we can monitor signs of mental distress in real time.

AI-powered tools, especially those using natural language processing and machine learning, can analyze behavioural patterns, detect early signs of anxiety or depression, and respond immediately, offering instant, personalized support. These algorithms not only provide proactive intervention, but also reduce barriers to help seek by creating a safe, accessible way for adolescents to explore mental health support in familiar digital environments. This intelligent approach can help prevent more serious issues, offering timely and compassionate care to teens who need it most.

Chapter 2

The Scientific Problem Addressed

The prevalence of anxiety and depression among teenagers has reached alarming levels in recent years, with studies indicating a sharp increase in mental health issues within this demographic. According to the World Health Organization (WHO), nearly 10 - 20% of adolescents worldwide experience mental health conditions, with many cases undiagnosed and untreated. These issues, if unaddressed, can lead to long-term psychological, academic, and social consequences.

A significant challenge in addressing teenage mental health lies in early detection of symptoms. Adolescents often hesitate to seek help due to stigma, lack of awareness, or limited access to traditional healthcare services. Additionally, current diagnostic methods rely heavily on self-reporting or clinical observations, which may not capture the nuanced and early signs of anxiety and depression.

In the digital age, teenagers increasingly interact with technology, spending substantial amounts of time on social media platforms, messaging apps, and video games. These environments often capture behavioral patterns and emotional cues that can serve as indicators of mental health challenges. However, traditional mental health interventions have largely overlooked the potential of these digital spaces to provide proactive and non-intrusive support.

The scientific problem addressed by this project centers on bridging this gap by leveraging advanced digital tools to identify, monitor, and provide early interventions for anxiety and depression. Specifically, the project seeks to answer the following key questions:

- How can artificial intelligence and sentiment analysis be utilized to identify early signs of anxiety and depression in teenagers through their digital interactions?
- What methods can effectively integrate mental health assessments into engaging and non-invasive digital environments such as video games and chatbots?
- How can predictive analytics be used to ensure timely and personalized interventions for at-risk adolescents?

By addressing these questions, this project aims to create an innovative, evidence-based approach to teenage mental health care. It leverages the ubiquity of digital platforms to provide scalable, accessible, and proactive solutions to a growing global health challenge.

Chapter 3

Existing Problem-Solving Methods (Related Work)

This chapter provides an in-depth exploration of the algorithms applied to the problem of mental health classification, the methodologies for learning and optimization, and the metrics employed to evaluate the quality of solutions. The focus is on understanding the mechanisms, utility, limitations, and examples of the algorithms, as well as how their outputs are measured to ensure effective decision-making.

3.1 Algorithms Used

3.1.1 Support Vector Machine (SVM)

3.1.1.1 How it works

Support Vector Machine (SVM) is a supervised learning algorithm designed for binary and multiclass classification tasks. Its primary objective is to find the best possible hyperplane to separate data points belonging to different classes. For linearly separable data, SVM identifies the hyperplane that maximizes the margin between classes. When data is non-linear, SVM employs kernel functions such as Radial Basis Function (RBF) or polynomial kernels to map the data into a higher-dimensional space where separation becomes feasible.

3.1.1.2 Why it is useful

SVM is particularly effective in smaller datasets with distinct boundaries between classes, making it well-suited for the structured survey responses used in this study. It performs well in high-dimensional spaces and offers a robust mathematical foundation for classification. Additionally, its use of kernel functions allows for flexibility in handling non-linear data.

3.1.1.3 Limitations

Despite its strengths, SVM struggles with large datasets due to computational intensity, as its training time scales quadratically with the dataset size. Additionally, it is sensitive to outliers, which can significantly influence the decision boundary. Selecting the correct kernel and hyperparameters requires extensive experimentation, which can be time-consuming.

3.1.1.4 Examples

In the context of mental health classification, SVM was used to analyze text features extracted from survey data. For example, responses like *"I feel constantly worried"* may be classified as *Anxiety*,

while *"I feel hopeless"* may be categorized as *Depression*, using patterns in word frequency and semantics.

3.1.2 Linear Regression

3.1.2.1 How it works

Linear regression is a simple, interpretable model that assumes a linear relationship between input features and the target variable. In this study, it is used for multiclass classification by representing each sentence as a numerical vector of features (e.g., word frequencies or sentiment scores). The algorithm calculates weights for these features to predict the probability of a sentence belonging to a specific class.

3.1.2.2 Why it is useful

Linear regression offers simplicity and interpretability, making it a suitable baseline algorithm for understanding the data and relationships between features. It is computationally inexpensive, allowing for rapid prototyping and testing.

3.1.2.3 Limitations

Its reliance on linear assumptions limits its applicability to more complex datasets, where interactions between features or non-linear patterns are critical. Linear regression may also fail to perform well in cases of high feature dimensionality or multicollinearity.

3.1.2.4 Examples

For example, the model may classify sentences like *"I am happy"* or *"I cannot focus"* based on sentiment scores calculated from word frequencies. If the sentiment score is positive, the model might classify the sentence as *Normal*, while negative scores may correspond to *Depression* or *Anxiety*.

3.1.3 Decision Tree

3.1.3.1 How it works

Decision trees are a supervised learning algorithm used for classification and regression tasks. They function by splitting the dataset into subsets based on feature values, creating a tree-like structure of decisions. In this study, decision trees work alongside sentence embeddings generated by the *all-MiniLM-L6-v2* transformer model. These embeddings provide a rich, semantic representation of the input data, which the decision tree uses to make splits at each node.

3.1.3.2 Why it is useful

Decision trees are easy to interpret and can handle both numerical and categorical data. When paired with transformer-generated embeddings, they can effectively model complex relationships within the data. Their non-parametric nature allows them to capture non-linear patterns, making them versatile for text-based classification tasks.

3.1.3.3 Limitations

Decision trees are prone to overfitting, especially when the tree depth is not constrained. They can also be sensitive to noise in the data, which might lead to less robust predictions. Additionally, they may struggle with high-dimensional data unless used with dimensionality reduction techniques or well-structured embeddings like those from transformers.

3.1.3.4 Examples

For instance, responses encoded using *all-MiniLM-L6-v2*, such as *"I feel constantly tired and unmotivated,"* are processed by the decision tree to predict classes like *Depression* or *Fatigue*. The embeddings enable the decision tree to consider the semantic context of the response.

3.1.4 Neural Networks with Transformers (Sentence Transformers)

3.1.4.1 How it works

Transformer models such as *all-MiniLM-L6-v2* are based on the attention mechanism, enabling them to focus on important aspects of input data while processing sequences of text. These models use a neural network architecture that converts input sentences into numerical embeddings. These embeddings capture the semantic meaning of each sentence and its context within the overall dataset. By leveraging the power of pre-trained embeddings, the algorithm can quickly generalize to new tasks with minimal fine-tuning.

3.1.4.2 Why it is useful

Transformers are particularly well-suited for analyzing natural language data because they excel at capturing the subtleties of sentence meaning and relationships. Their ability to handle unstructured data makes them indispensable for tasks involving text classification or understanding complex linguistic patterns. They also allow for transfer learning, which reduces the computational cost of training from scratch.

3.1.4.3 Limitations

The primary limitation of transformers lies in their computational requirements, both in terms of memory and processing power. This makes them slower to train and deploy compared to traditional algorithms. Additionally, they can overfit on smaller datasets, necessitating careful fine-tuning or the use of regularization techniques.

3.1.4.4 Examples

For instance, a response like *"I feel overwhelmed by my responsibilities"* is encoded into a high-dimensional vector representing its semantic meaning. The transformer model identifies the closest match based on training data, categorizing this response as *Anxiety*.

3.1.5 Random Forest Classifier

3.1.5.1 How it works

Random Forest is an ensemble learning algorithm that builds multiple decision trees during training and merges their outputs to improve classification accuracy. In this study, the algorithm works with text data represented as TF-IDF vectors. TF-IDF (Term Frequency-Inverse Document Frequency) captures the importance of words in the dataset by quantifying their frequency and uniqueness. The random forest aggregates the predictions from individual trees to classify text responses.

3.1.5.2 Why it is useful

Random Forest is robust to overfitting and performs well on a wide range of classification problems. Its ability to handle high-dimensional data makes it suitable for text classification tasks. Additionally, by using TF-IDF features, the algorithm can focus on the most informative terms in the text data.

3.1.5.3 Limitations

The algorithm requires more computational resources compared to single decision trees. It can also become less interpretable as the number of trees increases. Additionally, the performance of Random Forest depends on the quality of the features; poorly chosen features may lead to suboptimal results.

3.1.5.4 Examples

For example, TF-IDF vectors representing responses like *"I am unable to concentrate and feel restless"* are fed into the Random Forest classifier. By analyzing the importance of terms like *"concentrate"* and *"restless,"* the algorithm predicts classes such as *Anxiety* or *ADHD*.

3.2 Learning/Optimization Process

The learning process involves the training and fine-tuning of the algorithms discussed above to maximize their performance on the classification task. Training data was divided into labeled classes, allowing supervised learning techniques to map input features to output classes. The optimization process involved selecting appropriate hyperparameters, such as:

- The choice of kernel for SVM (e.g., RBF kernel for non-linear data).
- Fine-tuning pre-trained transformer models for sentence classification.
- Adjusting regularization parameters for linear regression to avoid overfitting.
- Limiting the depth of decision trees to prevent overfitting.
- Optimizing the number of trees in the Random Forest to balance accuracy and computational cost.

The performance of each model was validated using techniques like cross-validation to ensure generalization to unseen data.

3.3 Metrics for Assessing Solution Quality

The quality of each algorithm's predictions was assessed using the following metrics:

- **Accuracy** – The proportion of correct predictions made by the model across all classes.
- **Precision, Recall, and F1-Score** – These metrics provide a detailed view of performance for imbalanced datasets:
  - **Precision** measures the accuracy of positive predictions.
  - **Recall** evaluates the algorithm's ability to capture all relevant instances.
- **Confusion Matrix** – A visualization of true vs. predicted labels for each class, highlighting errors and their distribution.

These metrics collectively ensured that the solutions were robust, accurate, and suitable for real-world application.

Chapter 4

Experimental results obtained

4.1 Description of the Datasets

This study leverages two publicly available datasets sourced from Kaggle to develop and evaluate the proposed digital tools for the early detection of anxiety and depression in teenagers. These datasets include the *Reddit Mental Health Dataset* [1] and the *Emotion Recognition Dataset* [2]. Below, we provide a detailed description of each dataset and the preprocessing steps undertaken.

4.1.1 The Datasets

4.1.1.1 Reddit Mental Health Dataset

The *Reddit Mental Health Dataset* contains posts from Reddit users, labeled based on various mental health conditions. For this project, we focused on two relevant categories: *Anxiety* and *Depression*. The dataset originally includes the following categories:

- 0: Stress
- 1: Depression
- 2: Bipolar Disorder
- 3: Personality Disorder
- 4: Anxiety

For our analysis, only posts labeled as *Depression* and *Anxiety* were retained. The *title* column was removed, and the data was concatenated with the *Emotion Recognition Dataset*. To ensure data consistency, null values were removed, and the dataset was balanced to prevent bias towards the larger class.

4.1.1.2 Emotion Recognition Dataset

The *Emotion Recognition Dataset* includes text samples labeled with six emotion categories:

- 0: Anger
- 1: Fear
- 2: Joy
- 3: Love
- 4: Sadness

- 5: Surprise

The labels were transformed into three broader categories to align with our focus on mental health:

- **Anxiety:** Includes posts labeled as *Fear* and *Surprise*.
- **Depression:** Includes posts labeled as *Sadness* and *Anger*.
- **Normal:** Includes posts labeled as *Joy* and *Love*.

The resulting dataset combines information from both sources, providing a diverse set of features for model training and evaluation. Oversampling was applied to the *Normal* class to balance its representation with *Anxiety* and *Depression*.

4.1.2 Quantitative Analysis of the Datasets

Small Dataset: For initial experiments and prototyping, we created a smaller subset by selecting the first 500 rows from each category (*Anxiety* and *Depression*), resulting in a balanced dataset of 1,000 rows.

Large Dataset: The larger dataset used for final model training and testing contains:

- **Anxiety:** 20,290 rows
- **Depression:** 21,796 rows

This dataset provides a robust foundation for training predictive models and evaluating their performance.

4.1.3 Related Work and Usage of Datasets

Both datasets have been used in prior research to analyze mental health and emotional patterns through text. For instance:

- The *Reddit Mental Health Dataset* has been utilized to study language patterns associated with mental health conditions and develop classification models for mental health prediction. Studies have achieved notable accuracy in identifying conditions like anxiety and depression using natural language processing (NLP) techniques.
- The *Emotion Recognition Dataset* has been widely used for emotion classification tasks, leveraging machine learning models to distinguish between different emotional states.

In this project, we extend these works by combining both datasets, re-labeling their categories to focus specifically on anxiety, depression, and normal states, and balancing the data to enhance model performance. By addressing class imbalances and tailoring the datasets to the problem, we aim to improve the detection of mental health conditions in digital interactions.

4.1.4 Preprocessing and Data Transformation

- Extracted relevant labels and mapped them to *Depression*, *Anxiety*, and *Normal*.
- Removed unnecessary columns such as *title* from the Reddit dataset.
- Checked and ensured no null values in the combined dataset.
- Performed oversampling on the *Normal* class to address class imbalance.

The prepared datasets provide a strong foundation for training advanced AI models capable of identifying early signs of mental health challenges in adolescents.

4.2 First Approach - Small Data

4.2.1 Anxiety Data

We used a data set of 500 rows (small data).

- **Hyperparameters:** ID, Text, Target
- **Target:** Normal (246), Anxiety (254)
- **Models:** Support Vector Classification (Support Vector Machine) with encoding Word2Vec
- **Results:**
  - **Accuracy:** 0.64
  - **Precision (average):** 0.6481
  - **Precision for Anxiety Class:** 0.6481
  - **Precision for Normal Class:** 0.6304
  - **Recall:** 0.6731
  - **Confusion Matrix:**

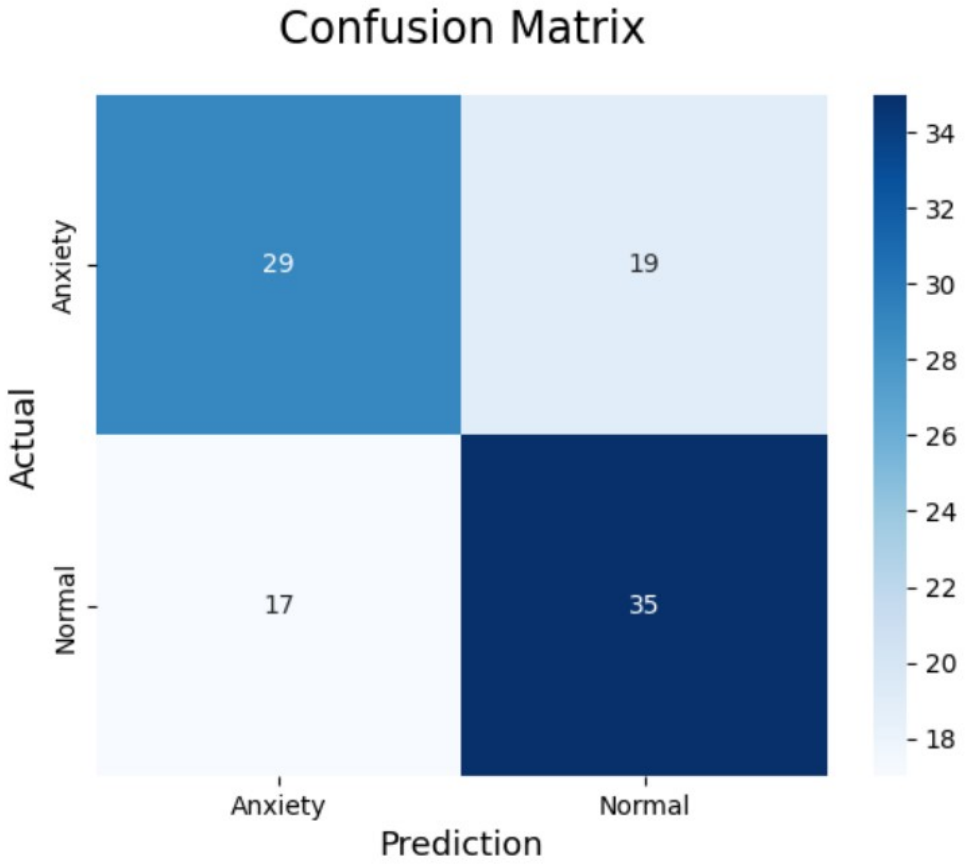

Figure 4.1: Confusion matrix for anxiety data (small dataset). This matrix shows the classification performance: 29 true positives and 17 false positives for normal cases, and 35 true positives and 19 false negatives for anxiety cases.

4.2.2 Depression Data

We used a data set of 500 rows (small data).

- **Hyperparameters:** ID, Text, Target
- **Target:** Normal (249), Depression (251)
- **Models:** Support Vector Classification (Support Vector Machine) with encoding Word2Vec
- **Results:**
  - **Accuracy:** 0.62
  - **Precision (average):** 0.6182
  - **Precision for Depression Class:** 0.6182
  - **Precision for Normal Class:** 0.6222
  - **Recall:** 0.6667
  - **Confusion Matrix:**

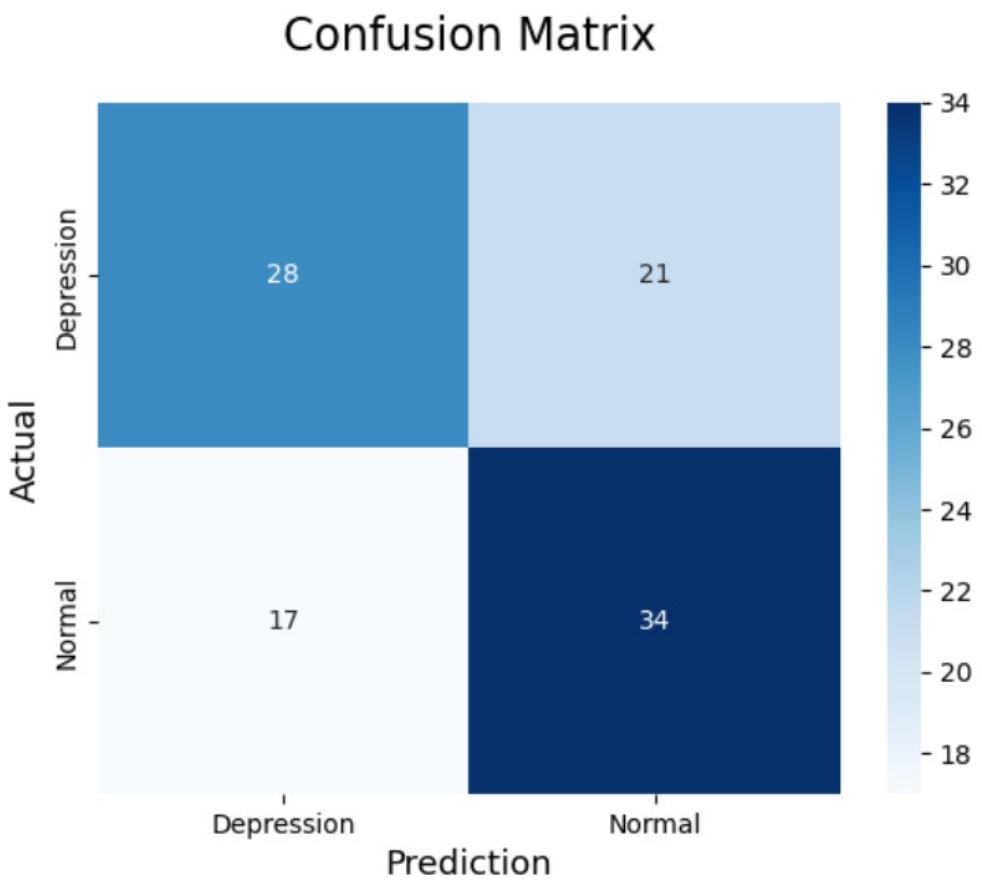

Figure 4.2: Confusion matrix for depression data (small dataset). This matrix illustrates the results with 28 true positives and 17 false positives for normal cases, and 34 true positives and 21 false negatives for depression cases.

4.3 Second Approach - Big Data

4.3.1 Anxiety Data

4.3.1.1 Linear Regression

- **Hyperparameters:** ID, Text, Target
- **Target:** Normal (246), Anxiety (254)
- **Models:** Linear Regression with transformer Sentence Transformer with the data set encoding 'all-MiniLM-L6-v2'
- **Results:**
  - **Accuracy:** 0.82
  - **Average Precision:** 0.82
  - **Precision for Anxiety Class:** 0.82
  - **Precision for Normal Class:** 0.82
  - **Recall for Anxiety Class:** 0.81
  - **Recall for Normal Class:** 0.83
  - **Confusion Matrix:**

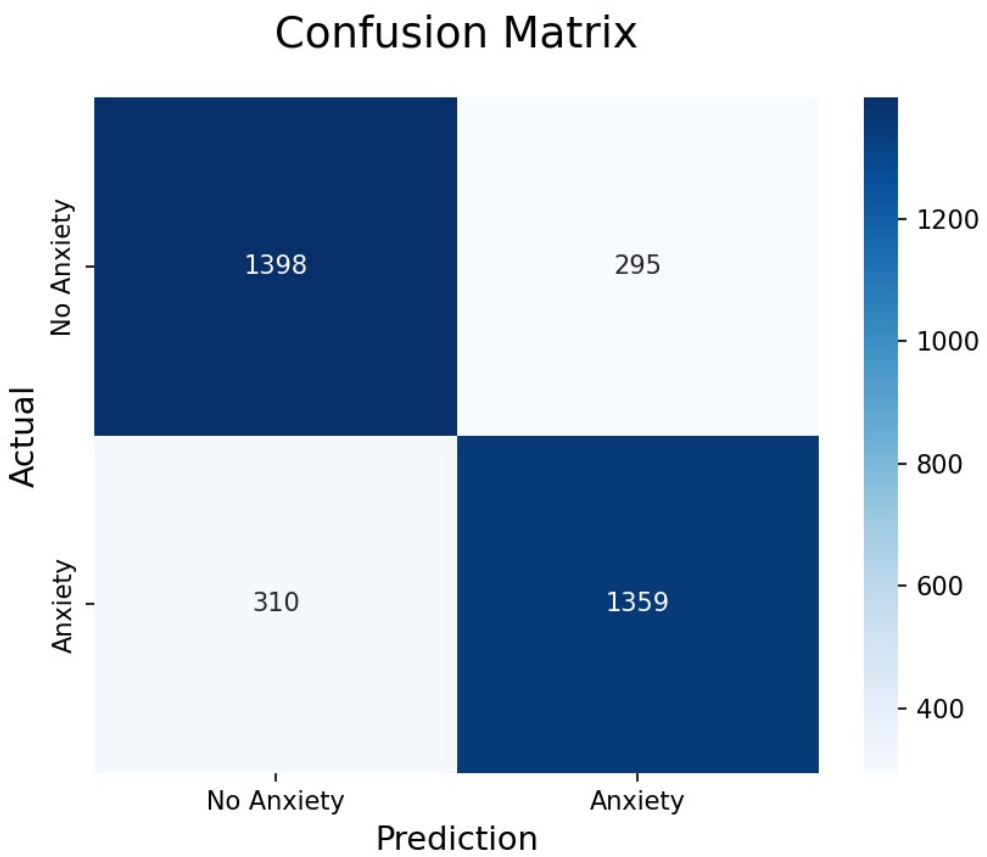

Figure 4.3: Confusion matrix for anxiety data using Linear Regression (large dataset). It details 1398 true positives and 310 false positives for normal cases, along with 1359 true positives and 295 false negatives for anxiety cases.

4.3.1.2 Decision Tree

- **Hyperparameters:** ID, Text, Target
- **Target:** Normal (246), Anxiety (254)
- **Models:** Decision Tree with transformer Sentence Transformer with the data set encoding 'all-MiniLM-L6-v2'
- **Results:**
  - **Accuracy:** 0.83
  - **Average Precision:** 0.83
  - **Precision for Anxiety Class:** 0.88
  - **Precision for Normal Class:** 0.79
  - **Recall for Anxiety Class:** 0.75
  - **Recall for Normal Class:** 0.90
  - **Confusion Matrix:**

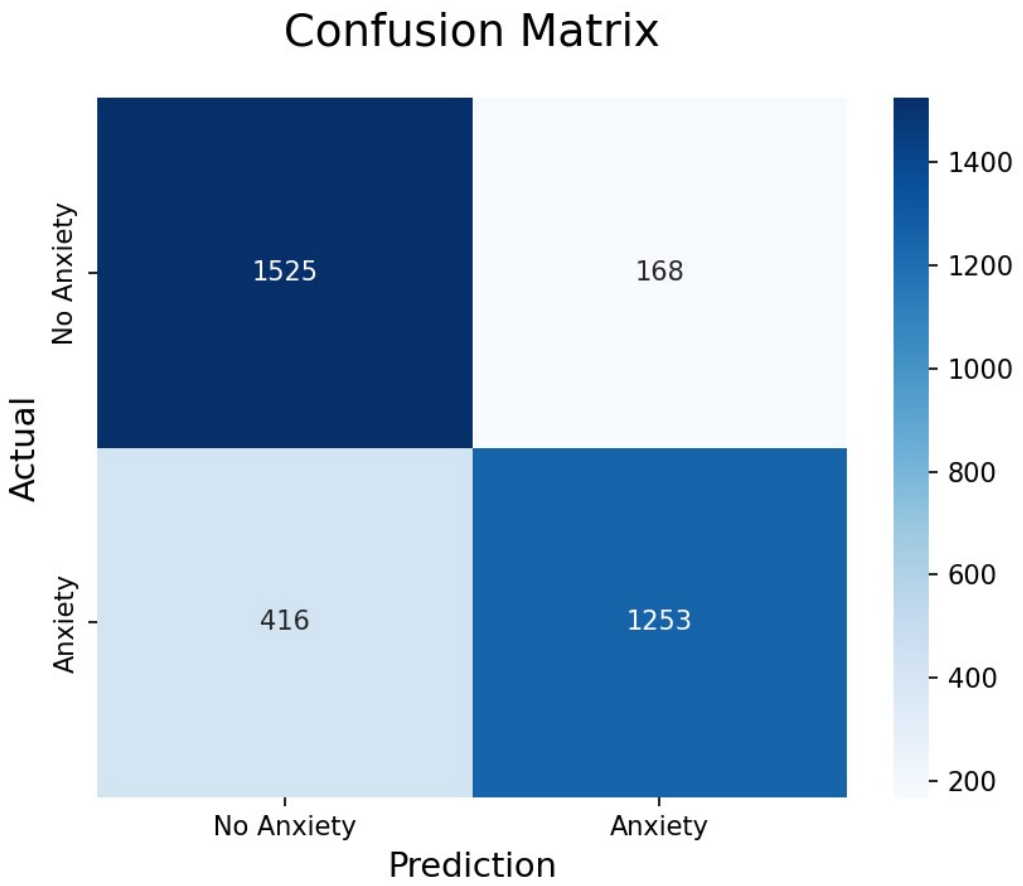

Figure 4.4: Confusion matrix for anxiety data using a Decision Tree (large dataset). The matrix indicates 1525 true positives and 416 false positives for normal cases, with 1253 true positives and 168 false negatives for anxiety cases.

4.3.1.3 Multi-layer Perceptron Classifier

- **Hyperparameters:** ID, Text, Target
- **Target:** Normal (246), Anxiety (254)
- **Models:** Multi-layer Perceptron Classifier with transformer Sentence Transformer with the data set encoding 'all-MiniLM-L6-v2'
- **Results:**
  - **Accuracy:** 0.93
  - **Average Precision:** 0.93
  - **Precision for Anxiety Class:** 0.95
  - **Precision for Normal Class:** 0.90
  - **Recall for Anxiety Class:** 0.90
  - **Recall for Normal Class:** 0.95
  - **Confusion Matrix:**

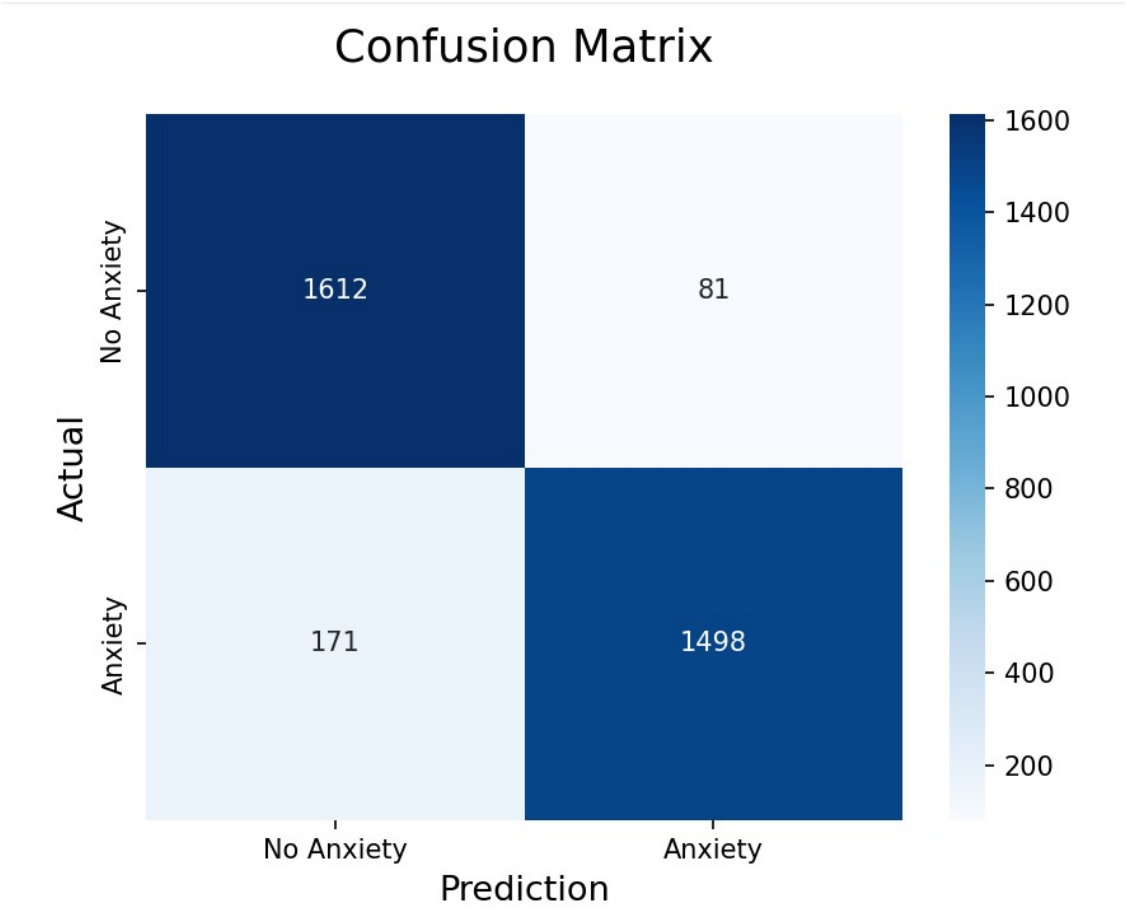

Figure 4.5: Confusion matrix for anxiety data using an Multi-layer Perceptron Classifier (large dataset). It shows 1612 true positives and 171 false positives for normal cases, and 1498 true positives with 81 false negatives for anxiety cases.

4.3.2 Depression Data

4.3.2.1 Linear Regression

- **Hyperparameters:** ID, Text, Target
- **Target:** Normal (249), Depression (251)
- **Models:** Linear Regression with transformer Sentence Transformer with the data set encoding 'all-MiniLM-L6-v2'
- **Results:**
  - **Accuracy:** 0.80
  - **Average Precision:** 0.81
  - **Precision for Depression Class:** 0.81
  - **Precision for Normal Class:** 0.80
  - **Recall for Depression Class:** 0.80
  - **Recall for Normal Class:** 0.81
  - **Confusion Matrix:**

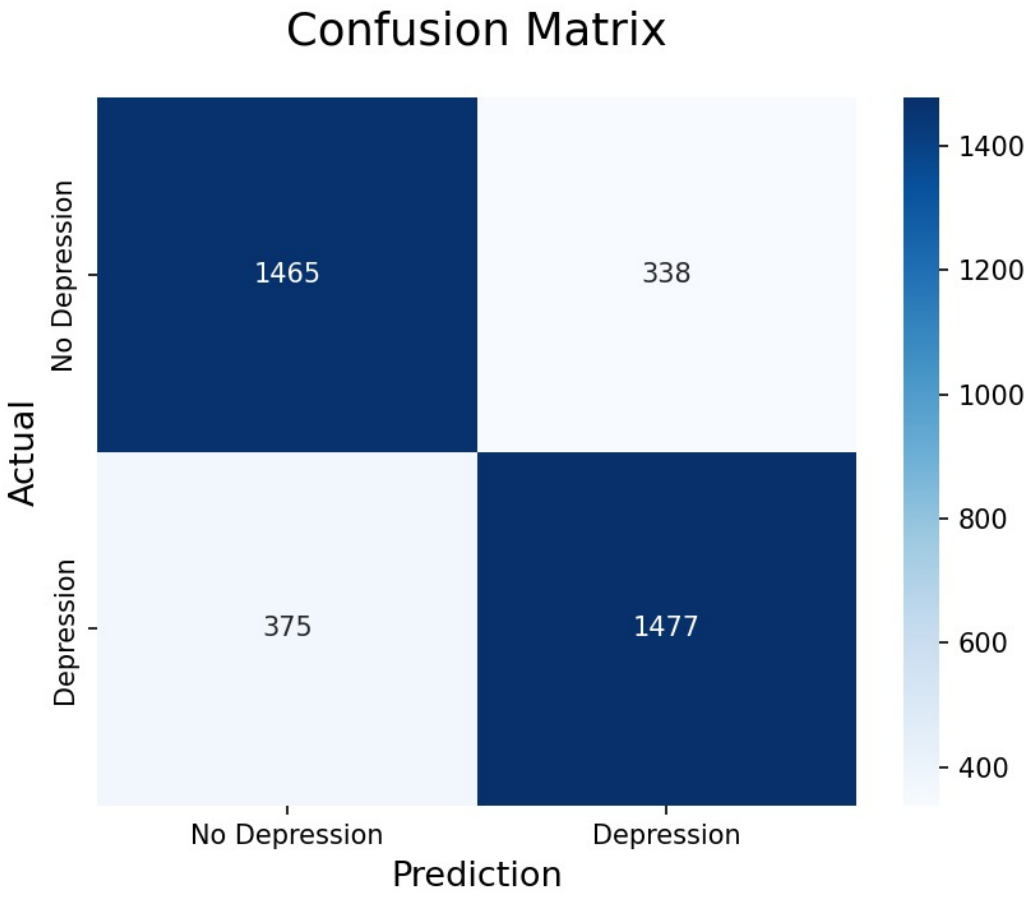

Figure 4.6: Confusion matrix for depression data using Linear Regression (large dataset). The results include 1465 true positives and 375 false positives for normal cases, as well as 1477 true positives and 338 false negatives for depression cases.

4.3.2.2 Decision Tree

- **Hyperparameters:** ID, Text, Target
- **Target:** Normal (249), Depression (251)
- **Models:** Decision Tree with transformer Sentence Transformer with the data set encoding 'all-MiniLM-L6-v2'
- **Results:**
  - **Accuracy:** 0.83
  - **Average Precision:** 0.84
  - **Precision for Depression Class:** 0.89
  - **Precision for Normal Class:** 0.78
  - **Recall for Depression Class:** 0.75
  - **Recall for Normal Class:** 0.91
  - **Confusion Matrix:**

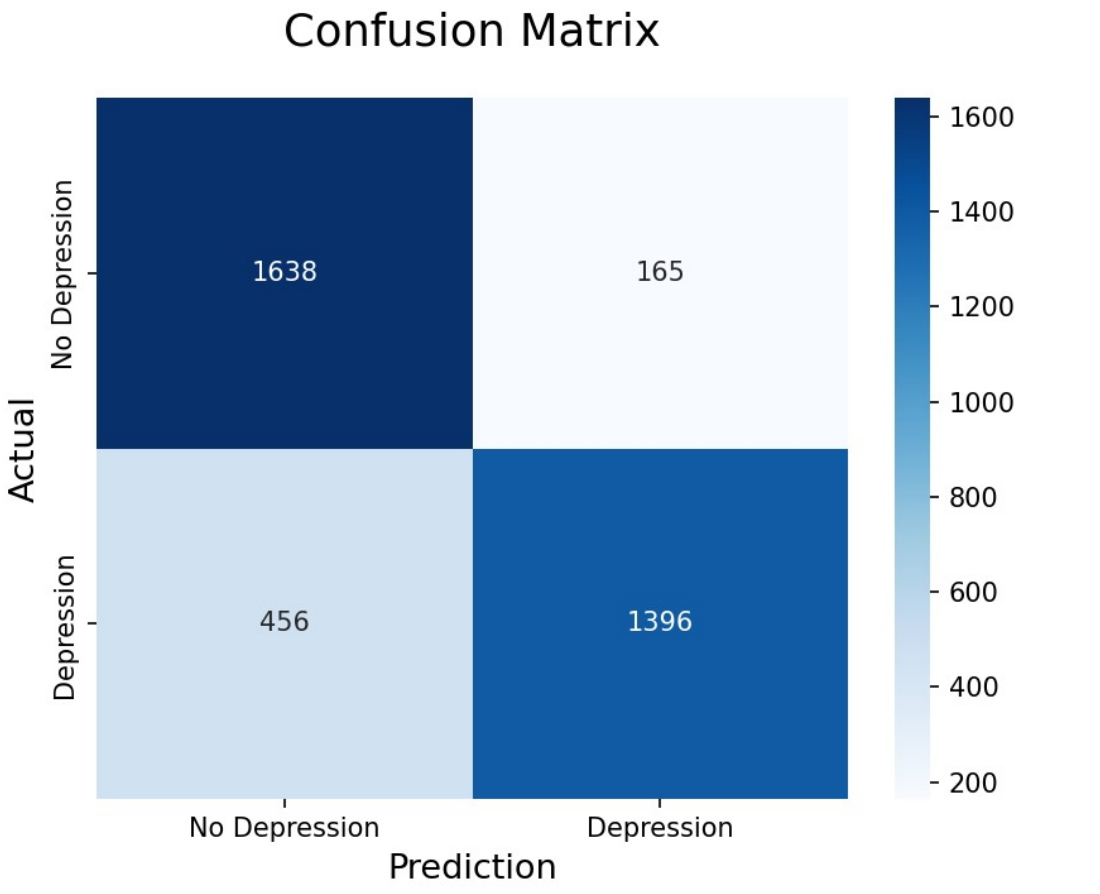

Figure 4.7: Confusion matrix for depression data using a Decision Tree (large dataset). It highlights 1638 true positives and 456 false positives for normal cases, with 1396 true positives and 165 false negatives for depression cases.

4.3.2.3 Multi-layer Perceptron Classifier

- **Hyperparameters:** ID, Text, Target
- **Target:** Normal (249), Depression (251)
- **Models:** Decision Tree with transformer Sentence Transformer with the data set encoding 'all-MiniLM-L6-v2'
- **Results:**
  - **Accuracy:** 0.93
  - **Average Precision:** 0.93
  - **Precision for Depression Class:** 0.96
  - **Precision for Normal Class:** 0.90
  - **Recall for Depression Class:** 0.89
  - **Recall for Normal Class:** 0.97
  - **Confusion Matrix:**

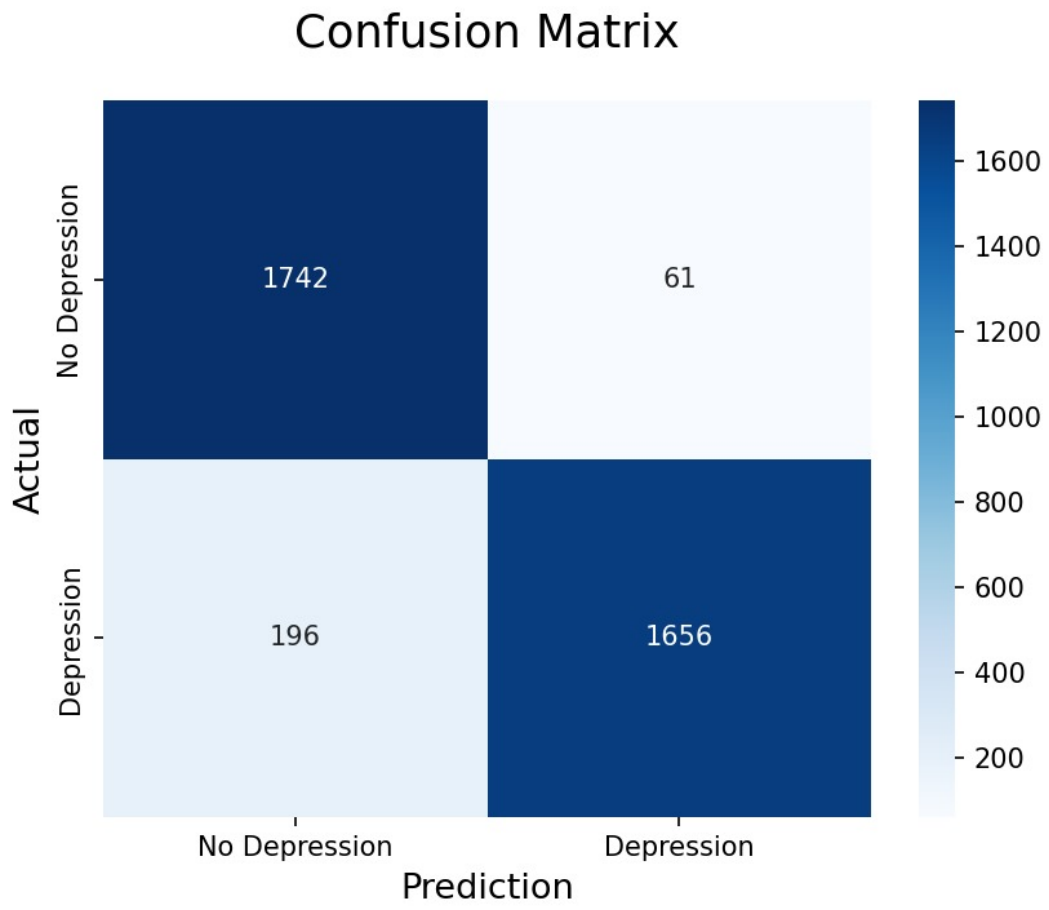

Figure 4.8: Confusion matrix for depression data using an Multi-layer Perceptron Classifier (large dataset). The matrix reports 1742 true positives and 196 false positives for normal cases, alongside 1656 true positives and 61 false negatives for depression cases.

4.3.3 Multi-class Classification Data

4.3.3.1 Random Forest Classifier

- **Hyperparameters:** ID, Text, Target
- **Target:** Normal (16543), Anxiety(16576), Depression (16456)
- **Models:** Random Forest Classifier with with the data set encoding TF-IDF Vectorizer
- **Results:**
  - **Accuracy:** 0.92
  - **Cross-Validation Accuracy:** 0.9202
  - **Average Precision:** 0.92
  - **Precision for Anxiety Class:** 0.95
  - **Precision for Depression Class:** 0.88
  - **Precision for Normal Class:** 0.95
  - **Recall for Anxiety Class:** 0.91
  - **Recall for Depression Class:** 0.91
  - **Recall for Normal Class:** 0.93
  - **Confusion Matrix:**

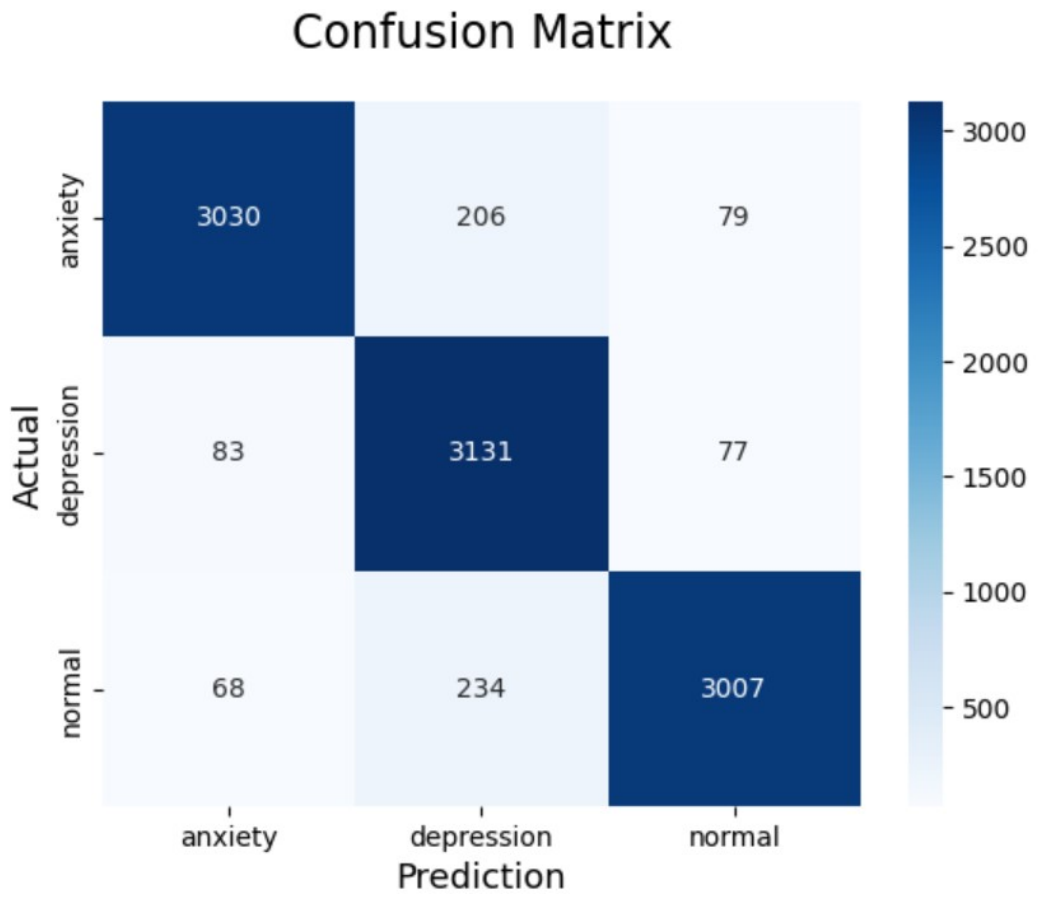

Figure 4.9: Confusion matrix for mental health classification using a Random Forest Classifier. The matrix indicates the model's performance with 3030 true positives and 285 misclassifications for anxiety, 3131 true positives and 160 misclassifications for depression, and 3007 true positives with 302 misclassifications for normal cases.

4.4 Spatial Complexity

4.4.1 What It Means

Spatial complexity refers to the memory required during the execution of the code. It depends on the size of data structures stored in memory, including intermediate and final outputs.

4.4.2 Code Analysis

4.4.2.1 TF-IDF Vectorizer

- **Details:** Converts text into a sparse matrix of size $n \times m$, where:
  - n: Number of documents in the dataset.
  - m: Maximum features defined (`max_features=5000` in your case).
- **Spatial Complexity:**

$$O(n \cdot m)$$

  The memory usage grows linearly with the number of documents and features. For sparse data (most elements being zero), storage efficiency improves.

4.4.2.2 Random Forest Classifier

- **Details:** The Random Forest classifier uses t trees, each with nodes storing:
  - Splitting features,
  - Threshold values,
  - Branches for splits until maximum tree depth (d).
- **Spatial Complexity:**

$$O(t \cdot d \cdot f)$$

  Here, t is the number of trees, d is the depth of each tree, and f is the features used per split. As t or d increases, memory usage scales proportionally.

4.4.2.3 Dataset Storage

- **Details:** The dataset is stored in a `pandas DataFrame` and requires memory for:
  - Raw text ($n \cdot$ average text length),
  - Labels (n),
  - Vectorized text ($n \cdot m$).
- **Spatial Complexity:**

$$O(n \cdot m + n)$$

  The majority of memory is consumed by the vectorized text.

4.4.2.4 Model Serialization

- **Details:** The model is saved using `joblib`, which includes:
  - The Random Forest classifier (including trees, features, and splits),
  - The TfidfVectorizer (vocabulary and IDF values).

- **Spatial Complexity:**
$$O(t \cdot d \cdot f + m)$$
This scales with the complexity of the classifier and vectorizer.

4.4.2.5 Confusion Matrix

- **Details:** The confusion matrix is stored as a $c \times c$ grid, where c is the number of classes.
- **Spatial Complexity:**
$$O(c^2)$$
Memory requirements here are negligible compared to other components.

4.4.3 Overall Spatial Complexity

Combining all components:
$$O(n \cdot m + t \cdot d \cdot f + c^2)$$

Where:

- n: Number of documents,
- m: Features in the vectorizer (5000 in this case),
- t: Trees in the Random Forest (200),
- d: Depth of each tree,
- f: Features considered per split,
- c: Number of classes.

4.4.4 Example Impact

For a dataset with:

- $n = 10,000$ (documents),
- $m = 5000$ (features),
- $t = 200$ (trees),
- $d = 10$ (depth),
- $f = 50$ (features per split),
- $c = 3$ (classes):

1. **Vectorized Data:** $10,000 \cdot 5000 = 50,000,000$ elements (approx. 190 MB in sparse format).
2. **Random Forest:** $200 \cdot 10 \cdot 50 = 100,000$ elements per tree, totaling $20,000,000$ elements (approx. 76 MB).
3. **Confusion Matrix:** $3^2 = 9$ elements (negligible).

Total Memory Impact: Around 266 MB for processing and storing the model.
Increasing t (trees) or d (depth) will significantly increase memory usage, making scalability an important factor to consider.

4.5 Temporal Complexity

4.5.1 What It Means

Temporal complexity refers to the runtime or the computational cost of executing the program, expressed in terms of how operations grow with input size.

4.5.2 Code Analysis

4.5.2.1 TF-IDF Vectorizer

- **What happens:**
  - The `TF-IDF Vectorizer` transforms text data into a numeric representation (a sparse matrix) by:
    1. Tokenizing the text into individual words.
    2. Computing term frequencies (TF) for each word in each document.
    3. Calculating the inverse document frequency (IDF) for each word across all documents.
    4. Combining TF and IDF values to create the final matrix.
- **Temporal Complexity:**

$$O(n \cdot \text{avg\_len} + n \cdot m)$$

  The first term accounts for tokenization, and the second term covers matrix computations.

4.5.2.2 Random Forest Classifier

- **What happens:**
  - The Random Forest classifier trains t decision trees by:
    1. Sampling data with replacement (Bootstrap sampling) from the training set.
    2. Iteratively splitting nodes based on f features (randomly selected) until a stopping condition is met (e.g., tree depth d).
    3. Predicting outcomes by aggregating the results from all trees.
- **Temporal Complexity:**

$$O(t \cdot n \cdot d \cdot f)$$

  Prediction involves evaluating all t trees for each document, with complexity:

$$O(t \cdot n)$$

4.5.2.3 Cross-Validation

- **What happens:**
  - Cross-validation splits the training set into k folds and trains the model k times, using $k - 1$ folds for training and 1 fold for testing.
- **Temporal Complexity:**

$$O(k \cdot t \cdot n \cdot d \cdot f)$$

4.5.3 Overall Temporal Complexity

Summing up:

$$O(n \cdot \text{avg\_len} + n \cdot m + k \cdot t \cdot n \cdot d \cdot f)$$

4.6 Portability Analysis

4.6.1 What It Means

Probability is a measure quantifying the likelihood that an event will occur. It is expressed as a number between 0 and 1, where 0 indicates impossibility and 1 indicates certainty. Mathematically, the probability $P(E)$ of an event E is defined as:

$$P(E) = \frac{\text{Number of favorable outcomes}}{\text{Total number of possible outcomes}}$$

Probability is a fundamental concept in statistics and machine learning, often used to predict outcomes and assess model confidence.

4.6.2 Code Analysis

4.6.2.1 File Path Handling

Current Situation: The code uses hardcoded, absolute paths such as:

```
data = pd.read_csv(
    "../data/anxiety_depression_data.csv"
)

joblib.dump(
    model,
    "../multiclass_classifiers/model_3_class_classifier_random_forests.joblib"
)
```

These paths make the code dependent on a specific directory structure, which is not portable across systems.

Solution: Replace absolute paths with relative paths or configurable paths using the `os` module:

```
import os

data_path = os.path.join(
    "app",
    "data",
    "anxiety_depression_data.csv"
)
data = pd.read_csv(data_path)

model_path = os.path.join(
    "app",
    "persistency",
    "model_3_class_classifier_random_forests.joblib"
)
joblib.dump(model, model_path)
```

4.6.2.2 Library Dependencies

Current Situation: The code does not specify exact versions of dependencies, leading to inconsistencies across environments.

Solution: Create a `requirements.txt` file to pin the library versions. Example:

```
pandas==1.5.3
scikit-learn==1.2.0
matplotlib==3.6.2
seaborn==0.12.2
joblib==1.2.0
```

Use `pip install -r requirements.txt` to set up dependencies.

4.6.2.3 Operating System Compatibility

Current Situation: The code uses Windows-style paths (`C:/...`), which are incompatible with Linux/macOS.

Solution: Use platform-agnostic paths via the `os` module:

```
os.path.join(
    "directory",
    "file.csv"
)
```

4.6.2.4 Deployment Across Languages/Platforms

Optional Improvement: If the model needs to be deployed in non-Python environments, consider exporting it to a universal format like ONNX:

```
from skl2onnx import convert_sklearn

onnx_model = convert_sklearn(
    model,
    initial_types=[('input', FloatTensorType([None, 5000]))]
)
```

4.6.2.5 Environment Configuration

Current Situation: Environment variables like file paths or runtime configurations are hardcoded.

Solution: Use environment variables or configuration files to store such settings.

4.6.3 Summary

To make the code portable:

1. Replace absolute file paths with relative ones.
2. Use a `requirements.txt` file for dependency management.
3. Avoid OS-specific code and use platform-agnostic methods for file handling.
4. *(Optional)* Export the model to a cross-platform format for deployment in non-Python environments.
5. Use environment variables or configuration files for runtime settings.

Chapter 5

Conclusions and possible improvements

Bibliography

[1] *Reddit Mental Health Dataset*. Neel Ghoshal (2020).
Available at: https://www.kaggle.com/datasets/neelghoshal/reddit-mental-health-data.

[2] *Emotion Dataset*. Parul Pandey (2020).
Available at: https://www.kaggle.com/datasets/parulpandey/emotion-dataset/data?select=training.csv.