IJHSR
ijhighschoolresearch.org
RESEARCH ARTICLE

Identifying Mental Health Indicators from Social Media Activity Using AI

Riaan Kapoor

Redmond High School, 17272 NE 104th St, Redmond, Washington, 98052, USA; kapoorriaan@gmail.com

ABSTRACT: Mental health challenges among youth have become a pressing issue, with rising rates of stress, anxiety, and depression highlighting the need for early detection and support systems. The purpose of this study is to explore the use of AI in identifying mental health signals among youth based on data collected from social media posts. Accordingly, a mental health dataset from Kaggle is used to train a Naïve Bayes classifier that categorizes social media posts into various mental health conditions. The results showed that the Naïve Bayes classifier was able to predict various mental health conditions with an accuracy of 64%, which is an encouraging baseline.

KEYWORDS: Computer Science, Artificial Intelligence, Naive Bayes, Machine Learning, Mental Health.

Introduction

Mental health is an ongoing crisis in our world. Every year, over 49,000 people die from suicide, equivalent to one death every 11 minutes.[1] Factors such as stress from school, family pressure, and peer expectations often contribute to these tragedies. In addition, issues like depression and anxiety are becoming increasingly severe, especially among young people. A study by the Centers for Disease Control and Prevention found that from 2009 to 2019, the percentage of high school students who reported persistent feelings of sadness or hopelessness rose from 26% to 37%,[2] showing a clear increase in stress signals among adolescents.

Consequently, it is important to understand the categories used to classify youth sentiment. In this study, a dataset with seven mental health categories (normal, depression, anxiety, suicide, self-harm, stress, and bipolar) was used. These categories reflect different emotional states that can be identified from text-based social media posts. Fortunately, many open-source datasets, such as those available on Kaggle, contain real-world examples labeled for these conditions. These labeled datasets allow researchers to train Artificial Intelligence (AI) models with the labeled data and help us uncover if there is any pattern between social media posts and mental conditions.

As part of this research, AI and Machine Learning (ML) models are used to classify data from the datasets and recognize emotional patterns. In particular, a key technique called Natural Language Processing (NLP), which allows machines to analyze text from forums, social media, or survey responses written by youth, is used. Natural Language Processing (NLP) is a field of computer science and AI concerned with the interaction between computers and human language.[3] Its primary goal is to endow machines with human-like abilities to understand, interpret, and produce natural language.

NLP was primarily driven by a rule-based system when it came to life in the 1950s, where experts wrote detailed grammar rules to teach computers how language works.[3] However, rule-based systems have limitations (as highlighted in sections below), which paved the way for more sophisticated approaches. Over time, NLP evolved to use statistical methods and machine learning, allowing models to learn patterns from real-world data instead of relying only on fixed rules. Today, advanced NLP models can understand not just words, but also tone, context, and emotion, making them powerful tools for mental health analysis. As a result, it's critical to use technology like AI to detect sudden changes in sentiment, especially signs of self-harm, suicidal thoughts, or bullying, so that schools and mental health professionals can intervene effectively.

The motivation of this research comes from the fact that the author is part of and a witness to a competitive high school environment, where students often face pressure and are judged by academic performance, which impacts youth mental health. At the same time, the prevalence of AI/ML encourages the author to leverage this technology to improve the well-being of students in similar communities, while allowing building expertise in the area.

Methods

Given that this was the author's first foray into AI/ML, the primary focus was on learning the base concepts that are applicable in every AI/ML project before advancing to the more pointed research related to sentiment analysis. A key tip that also helped was implementing everything from scratch before using off-the-shelf libraries. Libraries are fast and easy to use, but building a model from scratch helps one understand the math and code that goes into making an ML model. Understanding the implementation under the hood is the best way to advance our learning in this field. This work leverages the NumPy library in Python,[4] for matrix calculations, and the Pandas library,[5] for handling datasets in the form of data frames.

DOI: 10.36838/IJHSR85.17

Understanding AI/ML Concepts:

To start, the focus was on understanding the core idea behind ML: training a model to recognize patterns in data and make predictions. The two ideas that underpin most of the classical ML approaches are:

• Feature extraction: identifying measurable properties (like color/shape in fruit, or words/sentiment in text).

• Model training: using data to find a function $f(x)$ that best separates the categories.

The above foundational exercise helped visualize how an AI model learns and generalizes from examples. This video also helped in the process of learning.[6]

Probability and the Naive Bayes Approach:

Once the basics of classification were understood, a probabilistic algorithm, Naïve Bayes, which relies on conditional probability and the assumption that features are independent, was explored. This concept is important because the study would later dive into the use of Naive Bayes to classify mental health posts. To use the Naïve Bayes approach, the following terms need to be computed:

• Prior probabilities represent how frequently each mental health category, such as "anxiety" or "depression," occurs in the dataset.

• Likelihoods represent how often each word, such as "lonely" or "sad," appears in a mental health category.

• Posterior probabilities (prediction probabilities) based on Bayes' Theorem represent the final probability of how strongly a word falls under a certain mental health category after combining the likelihood and prior probability.

Understanding how to compute prior probabilities and likelihoods is the basis of the model, because the posterior probability (prior + likelihood) is the value that will be used to classify the sentence into one of the seven classes of the dataset.

This video cohesively explains how Naïve Bayes works.[7]

Learning Text Preprocessing & NLP Techniques:

Since the study involved analyzing text data, a concerted effort was made to dive into the details of Natural Language Processing (NLP). This included:

• Tokenization: breaking text into words.

• Stemming & Lemmatization: reducing words to their root forms.

• Count Vectorizer: converting words into numerical vectors for use in models.

This step was critical because computers cannot understand raw language—they need a numerical representation to process it. This process of creating a Bag-of-Words demonstrated how the conversion from the words in the dataset transformed into a numerical format, which is processable by a machine learning model.

Exploring the Dataset:

A labeled mental health dataset obtained from Kaggle was analyzed.[9] This dataset comprised seven emotional categories: normal, depression, anxiety, suicide, self-harm, stress, and bipolar. The following dataset analysis was done:

• Class distribution (how many examples per class),

• Biases in the data, and

• Label definitions to understand the kind of language used in each mental state.

Understanding the dataset made it possible to determine the most effective preprocessing and modeling steps. Additionally, it is important to observe any biases in the data (less representation of one class over another) to explain why the model may not provide the most accurate representation of that specific sentiment, since the dataset is skewed (more details in the Discussion section). For more information on datasets: visit.[10]

Building the Model – From Scratch:

Then, a custom Naive Bayes classifier was built from scratch, using NumPy to perform calculations. This hands-on implementation helped with understanding:

• How priors and likelihoods are stored and calculated,

• How predictions are made based on log probabilities,

• And how edge cases (like zero counts) are handled using techniques like Laplace smoothing.

This provided full control over every part of the model and facilitated a deeper understanding of how real algorithms operate under the hood. More information on Laplace smoothing.[11]

Implementing the Library Model:

To verify the correctness of the implementation, the scikit-learn library was used for comparison.[12] In addition, Scikit-Learn contains optimizations that provide a boost in the accuracy metrics. As is the good practice, the Scikit-Learn implementation of Naive Bayes was leveraged for the project. Though the library version was faster, building a custom model helped develop a deeper understanding of the model-building workflow.

Evaluating Performance:

To measure how well the models performed, the following metrics were computed:

• Confusion matrices to see where the model was getting predictions right or wrong,

• Accuracy, precision, recall, and F1-score to evaluate performance from different angles.

These metrics gave insight into which emotional states were harder to detect and where the model needed improvement. Understanding the relationship between these metrics helped with understanding where the model was performing better and where it was performing worse.

To get a better understanding of CMs (Confusion matrices) and metrics.[13]

Dataset Details:

Source:

The data set used in this study was obtained from Kaggle's Sentiment Analysis for Mental Health.[9] It contains labeled social media posts aimed at detecting and classifying mental

health conditions. Its variety of mental health labels indicates a multi-class classification problem, making it suitable for Naive Bayes classification. The dataset is publicly available and licensed for research purposes.

Target Labels:

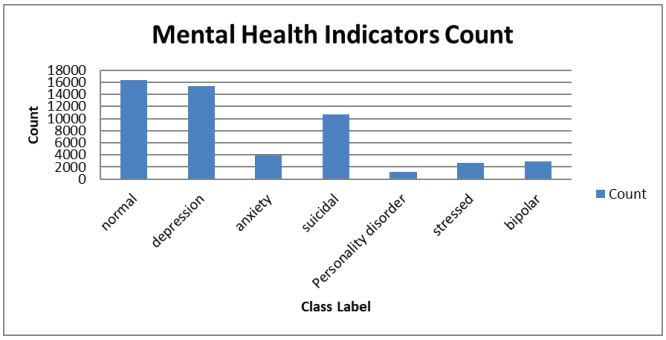

Figure 1: The bar chart shows the dataset is slightly imbalanced, with normal and depression being the most frequent classes.

Example Samples:

Table 1: Example mental health text samples.

Text Sample	Label
"Because this worries you."	Anxiety
"You're an addict that makes me doubt."	Normal

Preprocessing Steps:

All text was converted to lowercase, punctuation and special symbols were removed, and URLs were stripped out (Table 1). Tokenization was performed using Python's NLTK library, and stop words were removed to reduce noise. Lemmatization was applied to convert words to their root forms.

Custom Naive Bayes Implementation:

Priors/Likelihoods:

The custom Naive Bayes classifier was implemented from scratch to allow full control over preprocessing, feature extraction, probability computation, and learning purposes. The process began with preprocessing, which included lemmatization, stemming, stop word removal, punctuation removal, and conversion to lowercase. A Bag of Words representation was generated using a count vectorizer, transforming each text sample into a vector of word frequencies. These frequencies serve as the foundation for the Naive Bayes probability calculations. All this code can be found in the references section.[14]

Naive Bayes assumes independence of each word given the class label, allowing the likelihood of a text sample to be expressed as the product of the likelihoods (the probability of each word occurring given the class label) of all individual words. This likelihood is represented as $P(w/C)$. The class prior probability, $P(C)$, was computed by taking the ratio of the number of text samples in each class to the total number of text samples across all classes. This prior probability provides insight into how frequently each class occurs within the dataset. For example, the depression class occurs 29% of the time because 29% of the dataset is composed of depressive text samples.

The likelihood $P(w/C)$ was calculated using Laplace smoothing to address zero-frequency terms. When computing the probability of a certain word occurring in a class, it is possible that the word does not appear in that class at all, resulting in a probability of zero. Since the overall likelihood is calculated by multiplying the probabilities of all words, a single zero would cause the entire product to become zero. This would discard valuable information from other words in the sample. To prevent this, a small constant (in this case, 1) is added to each count, ensuring that no probability is ever exactly zero.

The priors are calculated as follows:

$$P(C) = \frac{\text{Number of Text Samples in Class } C}{\text{Total Number of Text Samples}}$$

Laplace smoothing is included in the calculation of likelihoods as shown:

$$p(w \mid C) = \frac{\text{count}(w, C) + 1}{\sum_{j=1}^{V} \text{count}(w_j, C) + V}$$

where,

w - a specific word we are calculating the probability for.

w_j - jth word in the entire vocabulary. Since vocabulary is a set, all the items are unique.

C - class (category)

count (w_j, C) - number of times the word (w_j) appears in class C, and we sum this for every word in the vocabulary count given by V. Since Laplace smoothing adds 1 to every word's count, the total word count for the class increases by V.

Predictions:

Predictions with the custom Naive Bayes implementation rely on the combination of prior/likelihood probabilities to make up the posterior probability. We can loop through each of the classes and compute a likelihood by either using $P(w/C)$ if the feature is present, or 1-$P(w/C)$ if the feature is absent. By using the prior probability for that specific class, in combination with the likelihood, the posterior probability is appended to the array. Finally, by appending the highest probability from the posterior array into the predictions array, we can classify the text sample into one of the seven classes.

Scikit-learn Implementation:

Overview:

To benchmark the custom Naive Bayes model, scikit-learn's built-in Multinomial Naive Bayes was used. This model is designed for text data represented as word counts by using a count vectorizer. The model already includes built-in smoothing, making it a strong and fast baseline for sentiment classification. Although the author implemented a Custom Naïve Bayes (for understanding), this work proceeded to leverage the Scikit-Learn library for the sake of robustness, as this is a well-tested and widely used implementation.

DOI: 10.36838/IJHSR85.17

Implementation Details:

The data was normalized by cleaning and preparing the text, including tokenization, converting to lowercase, stemming, stop word removal, lemmatization, and punctuation removal.

The train/test split function splits the dataset into training data and testing data based on user-provided splitting criteria. It's important that the model gets to see enough samples, so around 70% of the data was used for training. To evaluate the model's performance, the other 30% of the data that the model had never seen before (test data) was used. A random fixed seed was used so that every time the model is tested, it operates the same, mainly for reproducibility.

The vocabulary (unique set of all words in the dataset) was learned from the training set using a fitting function that maps each word to a unique index and transforms the text samples into matrices of word counts.

Test data transformation was completed, ensuring that no new words were added to the features. Tests are meant to be performed on new data, but still the same dataset and vocabulary that the model was trained with.

Training was done by using the training data and the training labels to fit the model. Lastly, Laplace smoothing (default alpha=1.0) was incorporated to handle zero-frequency words.

The trained model predicted labels for the test data, producing outputs for each of the seven classes in the dataset.

Evaluation Metrics:

Accuracy, precision, recall, F1-score, and confusion matrix were chosen as the evaluation metrics for the proposed approach. By using sklearn.metrics to import classification_report, the full rundown of these metrics was obtained.

Accuracy. The share of correctly classified objects in the total number of objects (citation). In other words, it shows how often the model is right overall. In this mental health study, accuracy is calculated as the proportion of times the predicted labels (y_pred) match the true labels (y_test). Accuracy provides a quick baseline as it is easy to compute and understand; it tells you at a glance how well your model is performing overall. Accuracy works best with balanced datasets. However, in this Kaggle dataset, the data is slightly imbalanced, as shown in Section 3.2, where the "Structure" graph illustrates a skewed class distribution. Accuracy can also be quite misleading. For example, if 70% of the posts are classified as "normal" and the model predicts "normal" for everything, you'd get a 70% accuracy just by the model making a blind guess. For this reason, it's important to take into account other metrics such as F-1 score, precision, and recall to get a holistic understanding of the model's performance.

Precision. Measures how many of the items predicted as a certain class are actually correct. It answers the question: *When the model says a post belongs to this class, how often is it right?* High precision means fewer false positives. In the context of mental health detection, a high precision for "suicidal" means the model rarely mislabels non-suicidal posts as suicidal, reducing unnecessary alerts.

Recall- Measures how many of the actual items from a class the model successfully identifies. It answers the question: Of all the real examples of this class, how many did the model catch? High recall means fewer false negatives. For this study, high recall is crucial for severe conditions like "suicidal" or "depression," because missing these posts could have serious consequences.

F1-score. The harmonic mean of precision and recall balances the two metrics into a single score. It's particularly useful when the dataset is imbalanced, because it ensures that both precision and recall are considered. In this study, a strong F1-score for critical mental health classes means the model is identifying important posts and doing so accurately.

Results

This section compares the performance of the Scikit-learn Naive Bayes (Scikit-learn NB) implementation with the Custom Naive Bayes (Custom NB) model. Both models were evaluated using accuracy, precision, recall, F1-score, and confusion matrices.

Accuracy:

The accuracy of classification using the Naïve Bayes classifier was 64%, which indicates a promising baseline.

Class-wise Metrics:

Table 2: Shows Class-wise Metrics for Scikit-learn.

Class	Multinomial Naïve Bayes classification		
	Precision	Recall	F1
Anxiety	0.68	0.75	0.71
Bipolar	0.70	0.68	0.69
Depression	0.48	0.75	0.58
Normal	0.95	0.61	0.71
Personality Disorder	0.92	0.14	0.25
Stress	0.74	0.18	0.29
Suicidal	0.64	0.63	0.63

Visual Comparison of Metrics:

Model Precision for the different categories of mental health disorders:

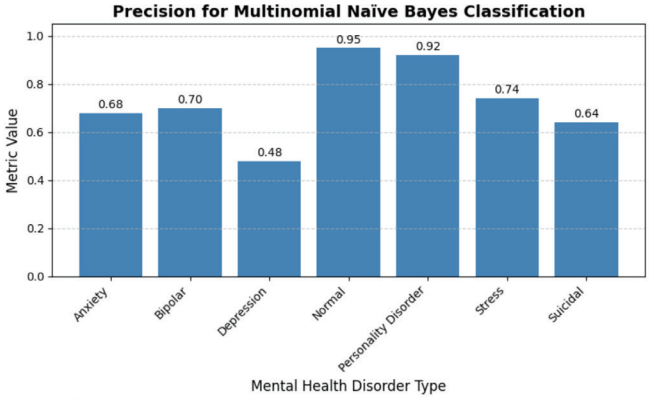

Figure 2: Bar chart showing the model Precision for various mental health disorders. Personality Disorder, Stress, and Bipolar Disorders have very less false positive results, as can be seen.

Model Recall for the different categories of mental health disorders:

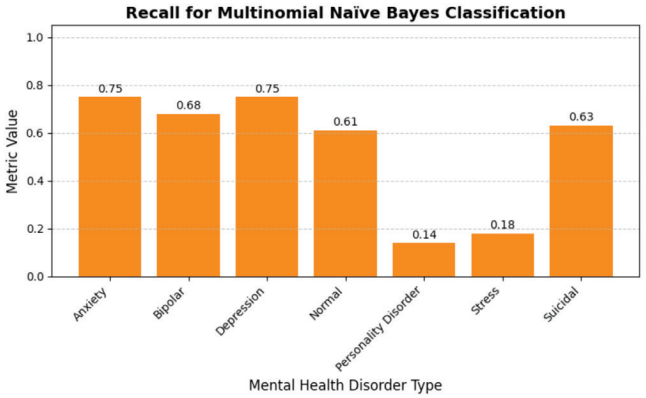

Figure 3: Bar Chart showing the model Recall for the various mental health disorders. Interesting to note that Personality Disorder & Stress have very poor Recall (unlike Precision), while Anxiety and Depression are detected with high Recall.

F1-score (Balance of Precision and Recall):

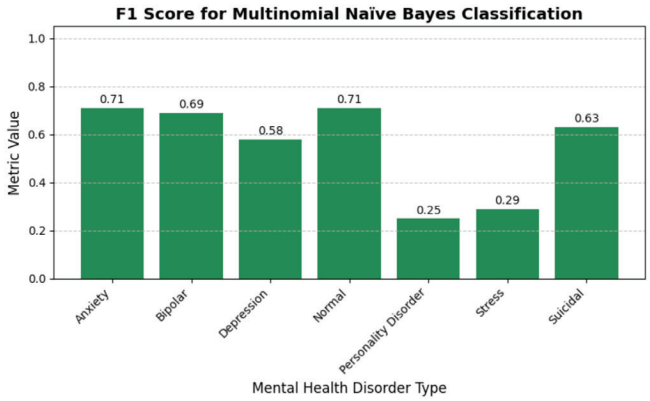

Figure 4: Bar Chart showing model F1-scores for the various mental health disorders. As can be seen, Anxiety and Bipolar disorders strike a good balance between Precision and Recall.

Confusion Matrices:

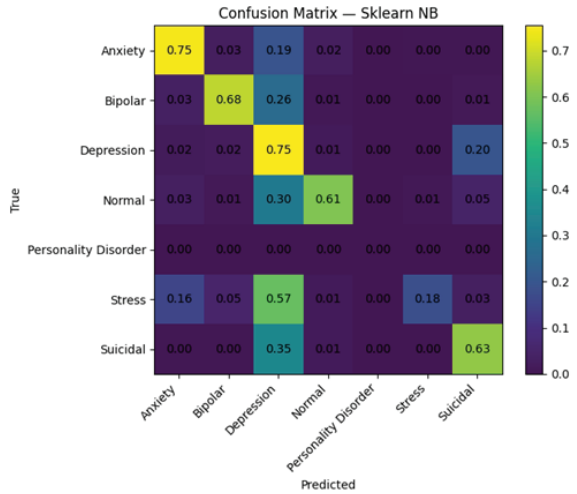

Figure 5: A Confusion Matrix that shows the correlation between the "True" Mental Health Disorders versus those predicted by Naïve Bayes. "Anxiety" and "Depression" are the best predicted categories, whereas "Stress" and "Personality" disorders are very hard to predict from the chosen dataset.

Conclusions: The Scikit-learn Naive Bayes model achieves balanced performance across most classes, though misclassification occurs between related categories such as Anxiety and Depression (Figure 2). 19% of Anxious posts were classified as depressive, due to the similarity between the two categories.

Conclusions: The Custom NB model performed well in classifying Depression and Normal. However, the model struggled with classifying anxious posts, as 80% of the anxious posts were classified as depressive, similar to the similarities between the two categories. Additionally, more rare classes like bipolar got classified as depressive 96% of the time (Figure 3), highlighting that the Custom NB model struggles with rare classes.

Interpreting the Confusion Matrix:

These two matrices summarize how the two models' predicted labels (y_pred) against the true labels (y_true) line up for each class.

• Diagonal values indicate correct predictions (the model guessed the right class)

• Off-diagonal values indicate mistakes (the model guessed the wrong class)

Bigger numbers in the wrong spots mean the model consistently mislabeled one class as another (Figure 5)

Discussion

Remarks on Custom Implementation of Naive Bayes:

Pros of Naive Bayes (general):

In general, Naive Bayes is fast, easy to implement, and useful for balanced datasets. It can also work well with small amounts of training data. Text data usually has thousands of features, and NB works well in this setting because it doesn't get overwhelmed by large amounts of features.

Cons of Naive Bayes (general):

Naive Bayes assumes full independence of each word when computing probabilities. Usually, these mental health text samples are written in context, so assuming full independence of each word is not realistic. Additionally, classes that overlap each other (like stress vs anxiety) can be hard to distinguish. This is because these classes use similar words, so Naive Bayes won't be able to tell the difference between these classes for the most part, leading to a lower F1-score (Figure 4). Lastly, if excessive nuanced language like sarcasm or slang is used, NB doesn't adapt well.

Limitations of Rule-Based Systems (NLP):

Rule-based systems rely on fixed patterns, keywords, or handwritten rules to classify text. For this reason, these systems are rigid. They can work for specific cases, but they struggle when the language becomes complex, particularly when it involves slang, sarcasm, or new vocabulary. As shown in section 3.3's example statements from the dataset, these mental health posts often use creative or emotional language, so a strict set of rules may struggle with identifying the meanings, which can lead to misclassification. Rule-based systems also require constant updating, which is not scalable for large or evolving datasets.

DOI: 10.36838/IJHSR85.17

Limitations of the Mental Health Kaggle Dataset:

The dataset itself also has limits. Many posts are short, unstructured, or written with spelling mistakes and slang, making it hard for models to understand them correctly. Also, a clear imbalance is shown with *bipolar* and *personality disorder classes* having fewer examples (Figure 6). This means that the model is less trained towards identifying these classes, which results in a low recall. Lastly, labels may not always be perfect, since posts were tagged by humans who could make mistakes or interpret them differently.

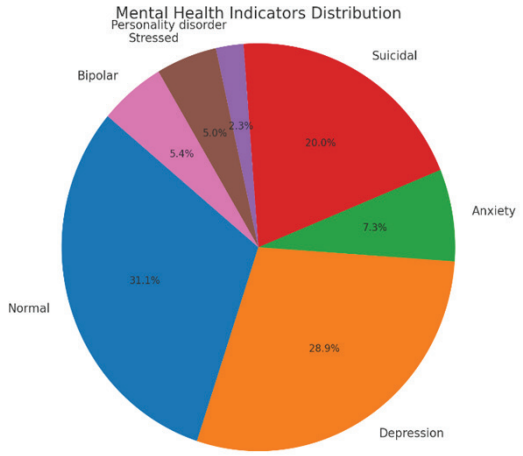

Figure 6: The pie chart illustrates the class imbalance present in the dataset. "Normal" and "Depression" are the majority classes, while "Personality Disorder" is a minority class, which is the reason why Naïve Bayes is unable to effectively model the probability distribution for this class.

Conclusion

The main goal of the study was to explore the use of AI to measure early signs of mental health signals on social media platforms. To achieve this goal, a Naïve Bayes classifier was trained on a Kaggle dataset to classify social media posts into seven mental health categories: *depression, normal, anxiety, suicide, personality disorder, stressed, and bipolar* (Figure 1). The results show that the Scikit-learn implementation of Naïve Bayes achieves an accuracy of 64%. The study also highlighted limitations. The Naïve Bayes assumption of word independence makes it less effective for overlapping categories like anxiety and stress (Table 2). When a post is showing signs of anxiety, the model may classify the text as stressed since the two classes are similar and the model doesn't use context clues to inform its decision. Additionally, the dataset itself was slightly imbalanced, with fewer examples for conditions such as personality disorder, which reduced recall for these classes. Short, informal, or slang-heavy social media posts also posed challenges for accurate classification. Future work could address these issues in several ways. First, collecting a higher-quality dataset with a more balanced representation across all mental health conditions would improve the overall metrics. Second, replacing simple count-based vectorization with TF-IDF could better capture the importance of words in context. Finally, expanding the label set to include more mental health categories could make the system more useful in real-world applications. Overall, this study demonstrates the potential of AI models in detecting mental health signals from social media. While Naïve Bayes has limitations, it serves as a strong foundation for beginners in machine learning and highlights both the positives and negatives of using AI for youth mental health support.

Acknowledgments

The author would like to sincerely thank Mr. Phani Srikanth (Principal Applied Scientist, NetApp) for mentoring and providing invaluable feedback.

References

1. Centers for Disease Control and Prevention. *Suicide Data and Statistics*. Centers for Disease Control and Prevention. https://www.cdc.gov/suicide/facts/data.html (accessed Aug 17, 2025).
2. Centers for Disease Control and Prevention. *Youth Mental Health: The Numbers*. Centers for Disease Control and Prevention. https://www.cdc.gov/healthy-youth/mental-health/mental-health-numbers.html (accessed Aug 17, 2025).
3. GeeksforGeeks. *History and Evolution of NLP*. https://www.geeksforgeeks.org/nlp/history-and-evolution-of-nlp/ (accessed Aug 17, 2025).
4. NumPy Developers. *NumPy Reference — NumPy v2.3 Manual*. https://numpy.org/doc/stable/ (accessed Aug 17, 2025).
5. Pandas Developers. *API Reference — pandas 2.3.2 Documentation*. https://pandas.pydata.org/docs/ (accessed Aug 17, 2025).
6. CodeBasics. *Python Machine Learning Tutorial (Data Science)*; YouTube, Jan 31, 2019. https://www.youtube.com/watch?v=7eh4d6sabA0 (accessed Aug 17, 2025).
7. StatQuest with Josh Starmer. *Naive Bayes*; YouTube, Jul 24, 2018. (accessed Aug 17, 2025). https://www.youtube.com/watch?v=O2L2Uv9pdDA (accessed Aug 17, 2025).
8. Data Science Tutorials. *Stemming*; YouTube, Nov 11, 2020. (accessed Aug 17, 2025). https://www.youtube.com/watch?v=HHAilAC-3cXw (accessed Aug 17, 2025).
9. Sarkar, S. T. *Sentiment Analysis for Mental Health*; Kaggle. https://www.kaggle.com/datasets/suchintikasarkar/sentiment-analysis-for-mental-health (accessed Aug 17, 2025).
10. Kaggle. *Kaggle: Your Home for Data Science*. https://www.kaggle.com (accessed Aug 17, 2025).
11. Analytics Vidhya. *Improve Naive Bayes Text Classifier Using Laplace Smoothing*, Apr 1, 2021. https://www.analyticsvidhya.com/blog/2021/04/improve-naive-bayes-text-classifier-using-laplace-smoothing/ (accessed Aug 17, 2025).
12. Scikit-learn Developers. *Scikit-learn: Machine Learning in Python — Official Documentation*. https://scikit-learn.org/stable/ (accessed Aug 17, 2025).
13. Evidently AI. *Interpreting Confusion Matrices*. https://evidentlyai.com/classification-metrics/confusion-matrix-Intpreting-CMS (accessed Aug 17, 2025).
14. Kapoorriaan-afk. *Research and Projects: Scikit-learn Naive Bayes*; GitHub Repository. https://github.com/kapoorriaan-afk/Research-and-Projects

Author

11th-grade high school student, Riaan Kapoor, is aspiring to solve real-world problems through the use of cutting-edge technology.