LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CU6051NI Artificial Intelligence

75% Individual Coursework

Submission: Milestone 1

Academic Semester: Autumn Semester 2025

Credit: 15 credit semester long module

Student Name: Jeshmin Shrestha

London Met ID: 23048596

College ID: NP01AI4A230022

Assignment Due Date: 07/01/2026.

Assignment Submission Date: 07/01/2026

Submitted To: Mr. Alish KC

GitHub Link	*https://github.com/jeshmin-shrestha/ADHD-OCD-Reddit-Classifier*

I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Acknowledgement

I would like to express my sincere gratitude to the module leader of Artificial Intelligence, Er. Roshan Shrestha for his guidance and support throughout this coursework. I am equally thankful to our tutor Mr. Alish KC for his valuable advice, encouragement and assistance during the preparation and development of this project. I am deeply thankful for both of their constructive feedback and suggestions which significantly helped in shaping the quality and direction of my work.

I would also like to acknowledge the authors and researchers whose pioneering work on mental health, Natural Language Processing and computational psychiatry provided a solid foundation for this implementation. Finally, I am grateful to Islington College for providing the academic resources and learning environment that are necessary to complete this comprehensive AI application successfully.

Abstract

Mental Health conditions such as Attention Deficit/ Hyperactivity Disorder(ADHD) and Obsessive Compulsive Disorder (OCD) have a significant impact on individuals daily functioning and wellbeing, social media platforms particularly Reddit have become valuable sources for understanding mental health through user generated content. However, the vast and unstructured nature of these posts poses challenges for manual analysis. This coursework implements a functional AI solution for automated binary classification of Reddit posts into ADHD or OCD categories using NLP and supervised machine learning techniques.

The implemented system follows a comprehensive pipeline involving advanced text preprocessing with negation preservation, TF-IDF feature extraction with n-gram optimization and three individual classification models (Naïve Bayes, Logistic Regression and Support Vector Machine). To enhance predictive performance, three ensemble strategies were implemented: hard voting, soft voting and stacking meta-learning. The stacking ensemble achieved the highest accuracy of 85.40%, demonstrating the effectiveness of combining multiple models.

The solution offers a scalable framework for mental health discourse analysis with applications in early detection, research and digital monitoring. It demonstrates that traditional machine learning, enhanced with ensemble methods and advanced preprocessing can effectively significantly classify nuanced mental health text while maintaining interpretability and computational efficiency.

Keywords: Mental Health Classification, Natural Language Processing, Machine Learning, Ensemble Methods, ADHD, OCD, Text Classification, TF-IDF, Stacking Ensemble, Social Media Analysis

Table of Contents

1. Introduction 1

1.1. Mental Health Conditions : OCD and ADHD 1

1.2. Problem Domain 2

1.3. AI Concepts and Techniques 2

2. Background 4

2.1. Research on Related works in Mental Health Classification 4

2.1.1. Automatic Detection and Classification of Mental Illnesses from General Social Media Texts 4

2.1.2. NLP Reveals Vulnerable Mental Health Support Groups on Reddit During CoVID-19 6

2.1.3. A comparative Study of Traditional ML vs Deep Learning for Mental Health Text Analysis 7

2.1.4. Foundational Linguistic Analysis of Mental Health Subreddits 8

2.1.5. Not Just Depressed: Bipolar Disorder Prediction on Reddit 9

2.2. Summarized Analysis 10

2.3. Dataset Overview 11

2.3.1. Dataset Source 11

2.3.2. Dataset Description 11

2.3.3. Problem Relevance: 12

3. Solution 13

3.1. Explanation of the solution/used AI algorithm 13

3.1.1. Solution Techniques 13

3.1.2. Algorithms Used 17

3.2. Pseudocode of the overall system 35

3.3. Diagrammatical representations of the solution 40

3.3.1. State Transition Diagram 40
3.3.2. Flow chart 41
3.4. Explanation of the Development Process 42
3.4.1 Development Environment and Tools 42
3.4.2. Data Loading and Initial Inspection 43
3.4.3. Checking duplicate data and its distribution across the classes 46
3.4.4. Advanced Text Preprocessing 50
3.4.5. Feature Extraction with TF-IDF 52
3.4.6. Model Implementation and Training 54
3.4.7. Hyperparameter Optimization 59
3.4.8. Ensemble Methods 60
3.5. Achieved results 64
3.5.1. Performance of Baseline Models (5,000 features) 64
3.5.2. Performance of Optimized Models (10,000 features + Hyperparameter Tuning) 64
3.5.3. Comparison of baseline models and Optimized model accuracy 65
3.5.4. Ensemble Models Performance 65
3.5.5. Detailed Metrics Comparison 66
3.5.6. Graphical Comparison of Metric 67
3.5.7. ROC Curves Comparison 68
3.5.8. Sample Prediction 69
4. Conclusion 71
4.1. Analysis of the work done 71
4.2. How the application/solution addresses real world problems 72
4.3. Future Work 72

5. References 74

Table of Figures

Figure 1: Naïve Bayes Classifier Working Architecture 17
Figure 2: Flowchart of Naïve Bayes 22
Figure 3: Logistic Regression Working Architecture 23
Figure 4: Flowchart of Logistic Regression 28
Figure 5: SVM Algorithm Working Architecture 29
Figure 6: Flowchart of SVM 34
Figure 7: State Transitional Diagram of the overall System 40
Figure 8: Flowchart of the overall system 41
Figure 9: Load dataset 43
Figure 10: Basic data Information 44
Figure 11: Class Distribution Analysis 45
Figure 12: Visual representation of Class distribution 45
Figure 13: Checking null values 46
Figure 14: Checking duplicate values 46
Figure 15: Visualization of the number of duplicate posts per subreddit 47
Figure 16: Final Class distribution after cleaning 48
Figure 17: Text analysis after removing duplicate 49
Figure 18: Code of URL removal 50
Figure 19: Code of handing negation preservation 51
Figure 20: Comparison of original and clean text 51
Figure 21: Initial TF-IDF vectorization 52
Figure 22:Optimized TF-IDF vectorization 53
Figure 23: Baseline Naïve Bayes Performance Metric 54
Figure 24: ROC Curve of Naïve Bayes 55
Figure 25: Baseline Logistic Regression Performance Metric 56
Figure 26: ROC Curve of Logistic Regression 56
Figure 27: Baseline SVM Performance Metric 57
Figure 28:ROC Curve of SVM 58
Figure 29: Comparative analysis of Baseline Models 58
Figure 30: Code of Hyperparameter optimization 59

Figure 31: Training models after hyperparameter tuning..... 59
Figure 32:Hard Voting Ensemble Performance Metric 60
Figure 33: Soft Voting Ensemble Performance Metric..... 61
Figure 34: Stacking Ensemble Performance Metric 62
Figure 35: Comparative analysis of Baseline Models..... 64
Figure 36:Comparative analysis of Optimized Models 64
Figure 37: Comparison of baseline models and Optimized model accuracy 65
Figure 38: Comparative analysis of Ensemble Models..... 65
Figure 39: Detailed Metric Comparison of all the models 66
Figure 40: Graphical Comparison of Metric..... 67
Figure 41: ROC Curves Comparison..... 68
Figure 43: Sample 1 Prediction 69
Figure 44:Sample 2 Prediction 69
Figure 45: Sample 3 Prediction 69
Figure 46: Sample 4 Prediction 69
Figure 47:Sample 5 Prediction 70
Figure 48:Sample 6 Prediction 70

Table of Tables

Table 1: Summarized Analysis of Related work 10

1. Introduction

In today's digital age, mental health has become an increasingly critical concern worldwide. The rising levels of stress, anxiety, depression and other psychological challenges have profoundly affected their emotional, psychological and social well-being (Yangyan Fan, Ahui Fan, Zhiping Yang and Daiming Fan, 2025). In recent years, online and social media platforms have transformed into a significant arena for mental health discourse, with individuals increasingly turning online communities for support, information and shared experience (Parsons CE, Purves KL, Davies MR, Mundy J, Bristow S, Eley TC, Breen G, Hirsch CR, Young KS., 2023). Particularly on platforms like Reddit with specialized forums has provided spaces where people discuss symptoms, coping strategies and personal challenges related to various mental health conditions. These online expressions often contain early indicators of mental health issues, making them valuable for analysis (Robert Thorstad, Phillip Wolff, 2019). However, manually monitoring and analysing such vast amounts of unstructured textual data is quite impractical.

1.1. Mental Health Conditions : OCD and ADHD

Attention-Deficit/Hyperactivity Disorder (ADHD) and Obsessive-Compulsive Disorder (OCD) are two distinct mental health conditions that significantly impact daily functioning. ADHD is characterized by inattention, hyperactivity and impulsivity with subtypes including inattentive, hyperactive-impulsive and combined presentations. OCD involves persistent, intrusive thoughts and repetitive behaviours, commonly manifesting as contamination/washing, harm/checking, or symmetry/ordering subtypes (Nader Salari , Hooman Ghasemi , Nasrin Abdoli , Adibeh Rahmani , Mohammad Hossain Shiri , Amir Hossein Hashemian , Hakimeh Akbari , Masoud Mohammadi , 2023). Both conditions are prevalent worldwide, affecting a major population, highlighting their clinical and social significance (Sonja Cabarkapa,Joel A King,Nathan Dowling,Chee H Ng, 2019).

1.2. Problem Domain

This coursework addresses the specific problem domain of automate differential classification within mental health discourse. The challenge involves developing an AI system to distinguish between Reddit posts discussing Attention Deficit/Hyperactivity Disorder (ADHD) and Obsessive-Compulsive Disorder (OCD). Although both of these conditions may look like they share overlapping themes, yet it exhibits distinct linguistic patterns and narrative styles in online discussions (Ashwini Ashokkumar, James W Pennebaker , 2021).

Automated differentiation between ADHD and OCD related posts can assist researchers, platform moderators and mental health analysts in organizing content, monitoring community needs and identifying emerging concerns at scale. The primary challenge lies in handling noisy, informal language and overlapping symptom descriptions which require models capable of capturing contextual and semantic nuances rather than relying on surface level keywords.

1.3. AI Concepts and Techniques

The coursework focuses on applying Artificial Intelligence (AI) techniques to classify Reddit posts related to specific mental health conditions. AI provides the framework for creating systems capable of analysing and understanding human language at a specific scale. Unlike simple keyword matching or rule-based system, AI approaches in particular through Machine Learning (ML) enables models to identify complex patterns, contextual nuances and semantic relationship within the text (Stuart Russell and Peter Norvig, 2022). ML can be categorized into supervised, unsupervised learning and reinforcement learning but for this task binary classification within supervised learning was used since the dataset is labelled and allows the model to learn patterns that distinguish between posts related to two major categories.

A specialized subset of AI, Natural Language processing (NLP) is applied to process and interpret human language which transforms unstructured Reddit posts into structured representations suitable for computational analysis (Glen Coppersmith, Mark Dredze, Craig Harman, Kristy Hollingshead, 2015). This includes text preprocessing, feature extraction and capturing contextual and semantic relationship between words and

sentences. These steps ensures that the subtle, metaphorical and context dependent aspects of the language, common in mental health discourse are preserved and interpreted correctly by the models.

By applying AI, ML and NLP techniques to this problem domain, the coursework demonstrated how computational approach can provide actionable insights from the online textual data, potentially supporting early detection and understanding of the mental health challenges.

The dataset used in this coursework consists of 24,000 Reddit posts, labelled according to whether they discuss ADHD or OCD. It is unstructured text, containing informal language, abbreviations and diverse expressions common in social media discourse. It is suitable for supervised learning tasks because the labels allow models to learn patterns distinguishing the two categories.

The goal of this coursework is to develop a conceptual AI solution that classifies the posts effectively, compares the performance of multiple classification algorithms, explores the methods to improve the prediction accuracy and acquire the accurate classification to promote the mental health domain.

2. Background

This section provides a comprehensive review of existing research on the classification of mental health conditions using NLP and ML techniques applied to the mental health related social media data. Research in this domain has grown significantly due to the abundance of unstructured textual data on platforms like Reddit where users self-disclose their experiences in subreddit such as ADHD and OCD (Boettcher, 2021). The review drawn from key studies, selected for their relevance to supervised classification tasks on Reddit data. These works employ machine learning models variants to capture semantic nuances, achieving promising results but revealing gaps in binary differentiation between similar disorders like OCD and ADHD.

2.1. Research on Related works in Mental Health Classification

Existing studies demonstrate the feasibility of NLP driven classification but often treated disorders in multi class settings rather than targeted binary comparisons. The analysis of five representative research works, highlighting their approaches, finding and limitations are explained below:

2.1.1. Automatic Detection and Classification of Mental Illnesses from General Social Media Texts

This study experiments with the state-of-the-art deep learning models on the SMHD dataset from Reddit which compromises posting from various mental health sub reddits labelled for conditions including ADHD, OCD, depression, anxiety and eating disorders. The authors focus on post level classification using social media texts no just support groups to detect multiple illnesses (Anca Dinu, Andreea-Codrina Moldovan, 2021).

Methods:

Fine tuning transformer base models such as BERT, RoBERTa, XLNet with tokenization and handling informal language with models trained on binary and multi class tasks to identify linguistic patterns like symptoms descriptions.

Finding:

While achieving high F1 scores for explicit disorders like depression, models showed moderate performance (~0.75 -0.81 F1) in differentiating ADHD from OCD. The authors

attributed this to significant symptom overlap in user language that is shared with anxiety and intrusive thoughts.

Analysis and Relevance:

This stride is foundational for our work as it validates the use of transformer architectures as a state-of-the-art approach while explicitly identifying the ADHD/OCD distinction as a “harder” classification problem than separating more linguistically distinct conditions. It directly motivates our coursework’s focus and sets a realistic performance benchmark.

2.1.2. NLP Reveals Vulnerable Mental Health Support Groups on Reddit During CoVID-19

This large-scale observational study uses the Reddit Mental Health dataset that comprising 826,961 posts from 15 major mental health support groups and 11 control groups from 2018-2020. The research employees a comprehensive suite of NLP techniques to understand how language evolved during a major external stressor (COVID 19) (Daniel M Low ,Laurie Rumker,Tanya Talkar, John Torous,Guillermo Cecchi,Satrajit S Ghosh, 2020).

Methods:

The authors used regression analysis on 90+ linguistic features, supervised machine learning to classify posts and interpret key features and unsupervised methods like topic Modeling to uncover latent themes.

Finding:

The study found that groups for ADHD, anxiety and eating disorders exhibits the most negative semantic change during the pandemic. Crucially, it demonstrated a concerning linguistic convergence as COVID-19 discussions increased, language across all mental health sub reddit became more similar to Health Anxiety which indicates a pandemic driven homogenization of distress expression.

Analysis and Relevance:

This paper is critically important for two reasons. First, it proves that external macro events can significantly alter the linguistic signature of specific disorders, which leaves a vital consideration for building robust classifiers Second, it highlights the methodological value for applying different NLP techniques to fully characterize mental health language.

2.1.3. A comparative Study of Traditional ML vs Deep Learning for Mental Health Text Analysis

This paper provides a direct methodological comparison highly relevant to our proposed solution, evaluating the performance and interpretability trade-offs between different algorithmic paradigms on mental health text (Zhanyi Ding, Zhongyan Wang, Yeyubei Zhang, Yuchen Cao, Yunchong Liu, Xiaorui Shen, Yexin Tian & Jianglai Dai, 2025).

Methods:

The study conducted a head-to-head comparison on a dataset of Reddit posts which evaluated traditional machine learning models (Support Vector Machines, Random Forest) using engineered features like TF-IDF and LIWC against deep learning models like LSTM and CNNs with word embedding and a fin tuned BERT model.

Finding:

The Fine-tuned BERT model achieved the highest overall accuracy (~87-92%). However, traditional models offered significantly greater interpretability which clearly highlights which psychological constructs drove classifications, The performance gap was smallest on tasks valuing disorders with high semantic overlap.

Analysis and Relevance:

This study directly justifies the core comparative approach of the coursework. It validates our plan to implement SVM baseline and a fine-tuned BERT model allowing us to explicitly evaluate the trade-off between state-of-the-art performance and model interpretability as it is a crucial consideration in sensitive mental health applications.

2.1.4. Foundational Linguistic Analysis of Mental Health Subreddits

The proposed study is a widely used benchmark for computational mental health research using Reddit data from SMHD. The study identifies users who explicitly self-disclose a mental health diagnosis and analysis their language across multiple conditions including ADHD and OCD (Arman Cohan, Bart Desmet, Andrew Yates, Luca Soldaini, Sean MacAvaney, Nazli Goharian, 2018).

Methods:

A high-precision dataset was constructed using lexical patterns to detect self-reported diagnoses that was followed by validation. Traditional machine learning models, including Logistic Regression and Support Vector Machines (SVM) were evaluated using interpretable features such as TF-IDF word n-grams and LIWC categories. The experiments focused on binary classification tasks, comparing disorders against control users and against each other.

Findings:

The models achieved strong performance in distinguishing mental health conditions from control users. However, classification accuracy decreased for pairwise comparisons between linguistically similar disorders. In particular, ADHD and OCD showed high linguistic proximity due to overlapping symptom-related language.

Analysis and Relevance:

This study validates Reddit as a reliable and ethical data source for mental health NLP and establishes LR and SVM as effective and interpretable baseline models. By identifying ADHD and OCD as a challenging pair to distinguish, it directly motivates the focused binary classification task addressed in this coursework.

2.1.5. Not Just Depressed: Bipolar Disorder Prediction on Reddit

This study directly addresses the challenge of binary classification between closely related neuropsychiatric conditions using Reddit data. Although the focus is on Bipolar Disorder (BD), the methodological approach and findings are highly relevant to the task of distinguishing ADHD from OCD due to similar symptom overlap and linguistic ambiguity (Ivan Sekulic, Matej Gjurković, Jan Šnajder, 2018).

Methods:

The authors constructed a high precision dataset by identifying users with self-reported BD diagnoses and sampling a matched control group from Reddit. The study evaluated traditional machine learning classifiers such as Support Vector Machine (SVM), Logistic Regression and Random Forest (RF) within a supervised binary classification framework, using nested cross-validation.

Findings:

The Random Forest model was most effective with 86.9% accuracy and 86.3% F1-score. Linguistic analysis showed users with bipolar disorder used more first-person pronouns such as "I" and health-related words but fewer power associated terms. They also exhibited greater variance in emotional language over time.

Analysis and Relevance:

This study validates Reddit as a data source and interpretable ML models for mental health NLP, providing a strong methodological baseline. The detection of distinct linguistic patterns, including higher emotional variability proves that NLP can capture disorder-specific behaviours that directly supporting the feasibility of this coursework's goal to classify ADHD and OCD.

2.2. Summarized Analysis

The literature review confirms the viability of using Reddit data and NLP for mental health classification. A clear gap exists in focused, binary classification between linguistically proximate disorders like ADHD and OCD.

Table 1: Summarized Analysis of Related work

Concept	Key Insight	Related Work Sections	Relevance
Use of Reddit Data	Reddit is a suitable and widely used data source for mental health NLP due to large-scale self-disclosure.	2.1.2, 2.1.4, 2.1.5	This supports the use of Reddit data for ADHD vs OCD classification.
Problem Formulation	Most studies adopt multi-class or disorder vs control settings rather than targeted binary comparisons.	2.1.1, 2.1.2	This motivates the focused binary classification of ADHD vs OCD.
Linguistic Overlap	ADHD and OCD exhibit high semantic similarity, making them harder to distinguish than other disorders.	2.1.1, 2.1.4	This justifies the complexity of the task and realistic performance expectations.
Modeling Approaches	Traditional ML models provide interpretability, while deep learning models offer higher accuracy.	2.1.1, 2.1.3, 2.1.5	This supports the use of SVM, Logistic Regression and Naive Bayes as interpretable baselines.
Research Gap	Limited work explicitly focuses on ADHD vs OCD using interpretable ML models.	2.1.3, 2.1.4	This establishes the gap addressed by this coursework.

2.3. Dataset Overview

2.3.1. Dataset Source

The dataset used in this coursework is “Mental Health Reddit Posts” published on Hugging face by jsfactory (jsfactory, 2021). It consists of textual posts collected from Reddit post, which is a widely used social media platform, where individuals frequently discuss mental health experiences in dedicated communities. The dataset here has actually filtered all the personal information such as the account name for better analysis of the actual agenda, mental health analysis.

Link of dataset: Link

2.3.2. Dataset Description

The dataset contains 24,000 textual Reddit posts where each post is represented by two attributes:

1. **body:** The main textual content of the Reddit post that serves as the feature variable for natural language processing.
2. **subreddit**: A binary numerical label with 0 and 1 representing the mental health category, where:
    - 0 represents OCD related posts
    - 1 represents ADHD related posts

The dataset is actually balanced with each 12,000 posts labelled as 1 and 0 but the body columns need textual preprocessing. This balance ensures that the classification models are not biased towards a particular class and supports fair performance evaluation. However, the textual nature of the data in body column requires preprocessing including text cleaning, tokenization and normalization before model training.

2.3.3. Problem Relevance

The dataset is well suited for this coursework as it focuses on binary classification between ADHD and OCD which are two mental health conditions known to exhibit overlapping symptoms such as intrusive thoughts, compulsive behaviours and attention difficulties (Neff, 2022). These similarities make automated classification challenging and also academically meaningful for this coursework.

The textual nature of the dataset makes it for applying the NLP techniques such as text processing, tokenization, vectorization and transformer-based embeddings. Since each post is explicitly labelled with a class, either 0 or 1; the dataset is suitable for supervised machine learning classification.

Additionally, Reddit based mental health data set consists of anonymized Reddit posts with no personally identifiable information included. The use of anonymized and publicly available social media data aligns with ethical research practices and has been widely adopted in prior mental health NLP studies that strengthens the validity and ethical suitability of this dataset for academic research (Xueqin Lei,Hong Wu,Zhaohua Deng,Qing Ye, 2022).

3. Solution

3.1. Explanation of the solution/used AI algorithm

3.1.1. Solution Techniques

a. Dataset Preprocessing Pipeline:

Since the raw reddit posts contained highly informal language, slang, emojis and inconsistent grammar, to address these challenges, a comprehensive preprocessing pipeline was implemented to improve data quality and model robustness:

- **Text Cleaning:** All URLs, special characters, punctuation and unnecessary whitespaces are removed from the posts to ensure that only meaningful textual content remains (Steven Bird, Ewan Klein and Edward Loper, 2009).
- **Normalization:** All text was then converted into lowercase to maintain consistency and contractions are expanded to standardize the linguistic patterns.
- **Tokenization:** Each post was split into individual words or tokens which served as the basic units for analysis and feature extraction.
- **Stop word Removal:** Commonly used words that do not carry significant semantic meaning were removed to reduce noise and dimensionality.
- **Negation Preservation:** Although negation words such as “not” are typically treated as stop words in standard NLP preprocessing, removing them can reverse the meaning of mental health statements. Therefore, negation preservation was applied by combining negation terms with the following word (e.g., “not anxious” into “not_anxious”), ensuring that semantic polarity and psychological intent were retained during classification.
- **Lemmatization:** Words are reduced to their base or root forms to consolidate similar terms and improve model efficiency.

This enhanced preprocessing strategy significantly reduced noise while preserving psychologically relevant linguistic patterns present in ADHD and OCD-related text.

b. Feature Engineering:

Since machine learning models operate on numerical data, the cleaned textual data was transformed into numerical feature representation using established NLP techniques (Dan Jurafsky and James H. Martin, 2025).

i. **TF-IDF vectorization:** Term Frequency Inverse Document Vectorization was implemented to quantify the importance of words and word combinations across the dataset. Both unigrams and higher order n-grams were employed to capture contextual dependencies within the text.

ii. **Dimensionality Control:** To reduce computational complexity and mitigate overfitting, the feature space was restricted to the most informative features using parameters such as maximum feature limits, minimum document frequency and maximum document frequency thresholds (Géron, 2019).

This approach ensured a balance between expressive textual representation and computational efficiency.

c. Model Implementation and Training:

Multiple supervised machine learning models were implemented and trained to classify Reddit posts into ADHD or OCD categories.

i. **Baseline Models:**

Initially the following three traditional ML algorithms were trained using the initial TF-IDF with 5000 max features and bigrams.

- Multinomial Naïve Bayes
- Logistic Regression
- Linear Support Vector Machine (SVM)

ii. **Training process:** The TF-IDF feature vectors were used as input to train each classifier. Model parameters were optimized through iterative learning to minimize classification error between predicted and actual labels.

iii. **Model Optimization:**

- **Hyperparameter Tuning:** GridSearchCV with 5-fold cross-validation was used for the extracting the optimal parameters.

- **Performance Calibration:** SVM was wrapped in CalibratedClassifierCV for probability estimation.
- **Feature Selection Validation:** With the iterative refinement in TF-IDF, the new max features of 10000 was used to increase the performance of the models.

iv. Three Ensemble Strategy Implementation:

1. **Hard Voting Ensemble:**

   Hard Voting Ensemble was implemented using the Voting Classifier with voting='hard' parameter, which operates through majority voting on class labels from all three base models. This approach achieved 85.05% accuracy, providing a simple yet robust mechanism that is resilient to individual model errors by aggregating multiple independent predictions.

2. **Soft Voting Ensemble:**

   A Soft Voting Ensemble was implemented using Voting Classifier with voting='soft', which averages predicted probabilities rather than relying on majority voting. Since SVM does not natively provide probabilities, it was wrapped in CalibratedClassifierCV to enable predict_proba(). This approach achieved 85.13% accuracy by weighting predictions based on each model's confidence offering a more nuanced aggregation than hard voting.

3. **Stacking Ensemble (Meta-Learner):**

   The Stacking Ensemble implemented a two-layer system where a meta-learner (Logistic Regression) was trained on predictions from three base models using 5-fold cross-validation. This method learned the optimal combination strategy, achieving the highest accuracy of 85.40% by intelligently weighting each model's contribution to the final decision.

d. Evaluation:

A comprehensive performance assessment was conducted to measure the effectiveness of all the models.

i. **Primary Metrics**: Models were evaluated using Accuracy, Precision, Recall and F1 score to provide a balanced view of performance across both the classes (Marina Sokolova, Guy Lapalme, 2009).

ii. **Comparative analysis:** The performance of baseline machine learning models was compared against optimized and ensemble models to identify the trade-offs between interpretability and predictive power. Confusion matrices and performance visualizations were used to support this comparison.

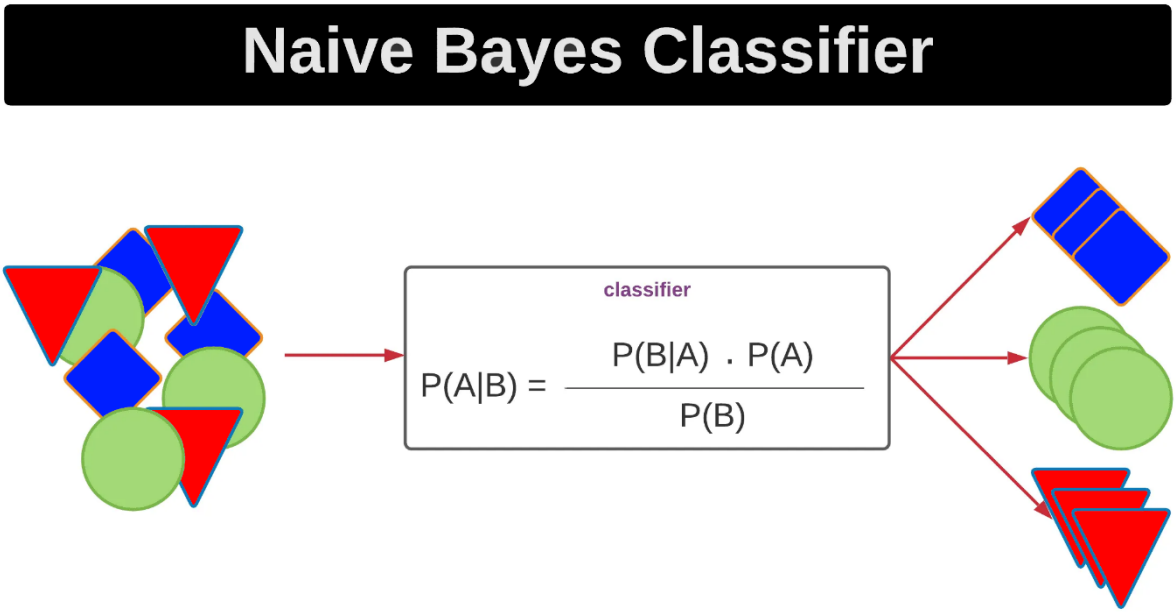
The evaluation demonstrated that ensemble-based approaches achieved superior performance compared to individual classifiers, confirming the effectiveness of the proposed solution.

3.1.2. Algorithms Used

The solution implements a classical machine learning pipeline for binary text classification, distinguishing between ADHD and OCD Reddit posts. Three distinct algorithms were employed to achieve optimal accuracy and predictive performance.

3.1.2.1. Naïve Bayes Classifier:

Naïve Bayes is a probabilistic classifier based on Bayes' theorem to calculate the likelihood of a document belonging to a class given the features(words). It assumes that words occur independently in posts which simplifies the computation need (Murphy, 2012). For our Reddit dataset, each post is represented as TF-IDF vector and the classifier computes probabilities for ADHD or OCD based on the word distributions.



Source:(Tech & Tales, 2023)

Figure 1: Naïve Bayes Classifier Working Architecture

Technical Suitability:

a. **Captures symptom specific word frequencies:** ADHD posts often include words like focus, distract or hyperactive, while OCD posts include obsessive compulsive or ritual which Naïve Bayes can efficiently captures these differences (Rish, 2001).
b. **Handles sparse, high dimension features:** TF-IDF vectors of Reddit posts are mostly zeros but Naïve Bayes naturally handles this sparsity without complex optimization (Andrew McCallum and Kamal Nigam, 1998).
c. **Manages overlapping vocabulary:** Common words like anxiety or stress appear in both the classes but the independence assumption in Naïve Bayes prevents overfitting and provides a stable baseline.
d. **Interpretable probabilistic output:** The model outputs the explicit probabilities, highlighting which words strongly influence classification which makes it valuable in sensitive mental health contexts (Steven Bird, Ewan Klein and Edward Loper, 2009).

Reason for choosing:

a. **Baseline establishment:** It provides a clear, understandable benchmark against which more complex models such as SVM which makes it comparable (Géron, 2019).
b. **Dataset specific advantage**: It handles noisy, informal and repetitive Reddit text efficiently (Izzah Maisarah Binti Ibrahim; Ruhaila Maskat; Azmi Bin Aminordin; Noor Hasimah Ibrahim Teo, 2024).
c. **Implementation simplicity:** It is simple to implement that demonstrates the core probabilistic reasoning for using in this project.
d. **Transparent and interpretable:** Unlike deep learning models, it clearly shows how individual words influence predictions, which is important for mental health analysis (Izzah Maisarah Binti Ibrahim; Ruhaila Maskat; Azmi Bin Aminordin; Noor Hasimah Ibrahim Teo, 2024).

1. Pseudocode of Naïve Bayes

The pseudocode below outlines the step-by-step procedure for implementing the Naïve Bayes classifier for this specific task. It begins with data loading and preprocessing, proceeds through feature extraction using TF-IDF and concludes with model training, evaluation and hyperparameter tuning.

```
START

  IMPORT pandas, NumPy

  IMPORT sklearn, nltk, re

  IMPORT evaluation metrics

  IMPORT matplotlib, seaborn

  IMPORT TfidfVectorizer FROM sklearn.feature_extraction.text

  IMPORT train_test_split, GridSearchCV FROM sklearn.model_selection

  IMPORT MultinomialNB FROM sklearn.naive_bayes

  IMPORT accuracy_score, classification_report, confusion_matrix

  IMPORT matplotlib.pyplot AS plt

  IMPORT seaborn AS sns

  DEFINE features and target

     SET X EQUAL TO df_clean['cleaned_text']

     SET y EQUAL TO df_clean['subreddit']


  SPLIT dataset into training and test sets

     USE stratified split

        SET test_size EQUAL TO 0.20

        SET random_state EQUAL TO 42
```

SET X_train, X_test, y_train, y_test

INITIALIZE TF-IDF Vectorizer with improved settings

SET max_features **EQUAL TO** 10,000

SET ngram_range **EQUAL TO** (1, 3)

SET min_df **EQUAL TO** 5

SET max_df **EQUAL TO** 0.9

SET stop_words **EQUAL TO** None

FIT vectorizer **ON** X_train

TRANSFORM X_train to obtain X_train_tfidf

TRANSFORM X_test to obtain X_test_tfidf

DISPLAY shape of X_train_tfidf

PERFORM hyperparameter tuning for Multinomial Naïve Bayes

DEFINE parameter grid

SET alpha values equal to {0.1, 0.5, 1.0, 1.5, 2.0}

INITIALIZE GridSearchCV

SET estimator **EQUAL TO** MultinomialNB()

SET cv **EQUAL TO** 5

SET scoring **EQUAL TO** 'accuracy'

FIT GridSearchCV **ON** X_train_tfidf and y_train

EXTRACT best_alpha **EQUAL TO** grid.best_params_['alpha']

RESULT: best_alpha **EQUAL TO** 1.5

INITIALIZE optimized Naïve Bayes classifier

SET nb_model **EQUAL TO** MultinomialNB(alpha **EQUAL TO** best_alpha)

TRAIN nb_model **ON** X_train_tfidf and y_train

PREDICT labels **ON** test set

SET y_pred equal to nb_model.predict(X_test_tfidf)

EVALUATE performance

COMPUTE accuracy

COMPUTE classification report (precision, recall, F1-score) **FOR** ADHD and OCD

COMPUTE confusion matrix

VISUALIZE results

PLOT confusion matrix USING seaborn heatmap

DISPLAY result

PRINT classification report

SHOW confusion matrix plot

SAVE trained model

DOCUMENT results, observations and key features

END

2. Flowchart of Naïve Bayes

The flowchart in below visualizes the sequential steps of the Naïve Bayes pipeline, highlighting key decision points such as text preprocessing, feature vectorization and model evaluation which clarifies the data flow and conditional logic involved in the classification process.

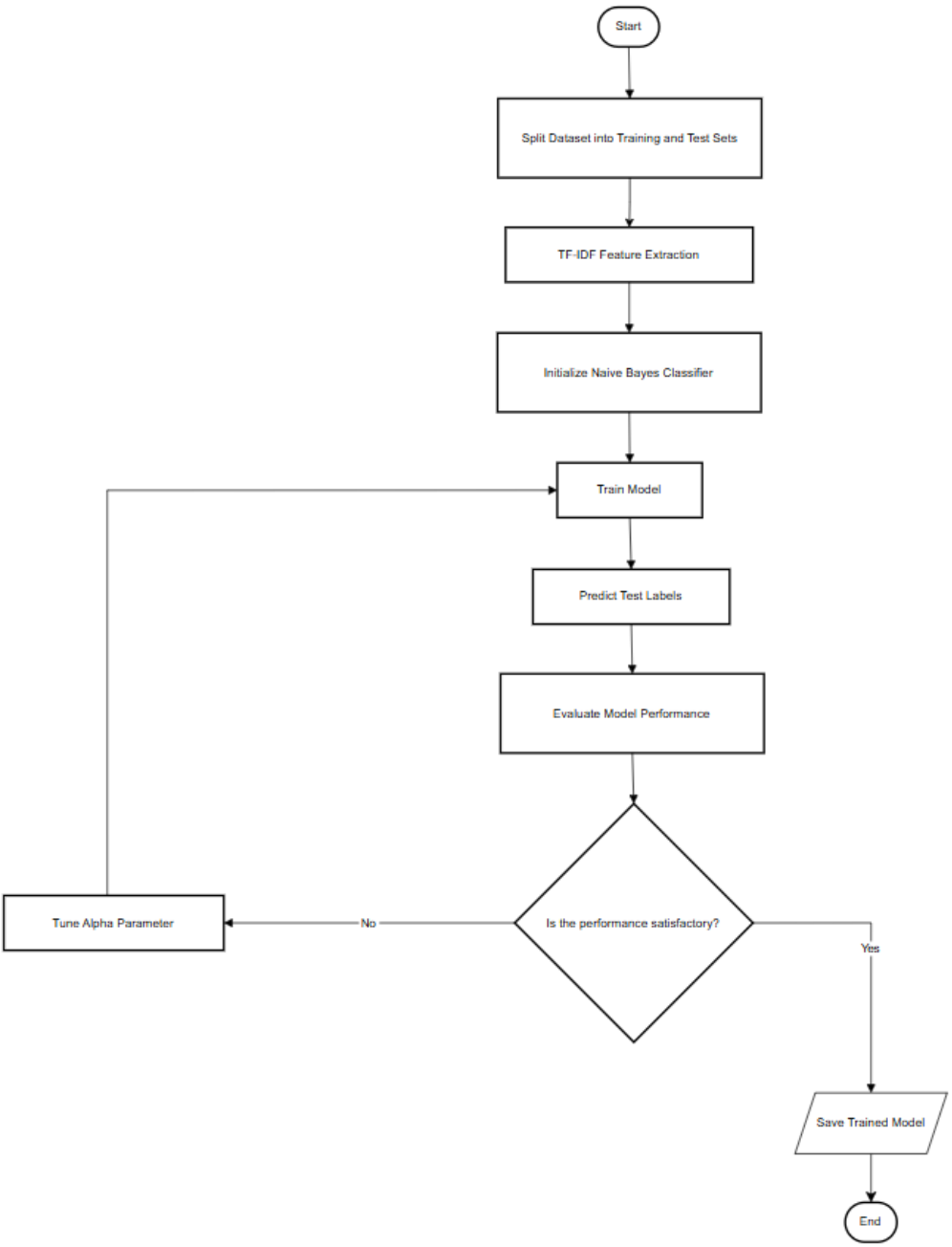
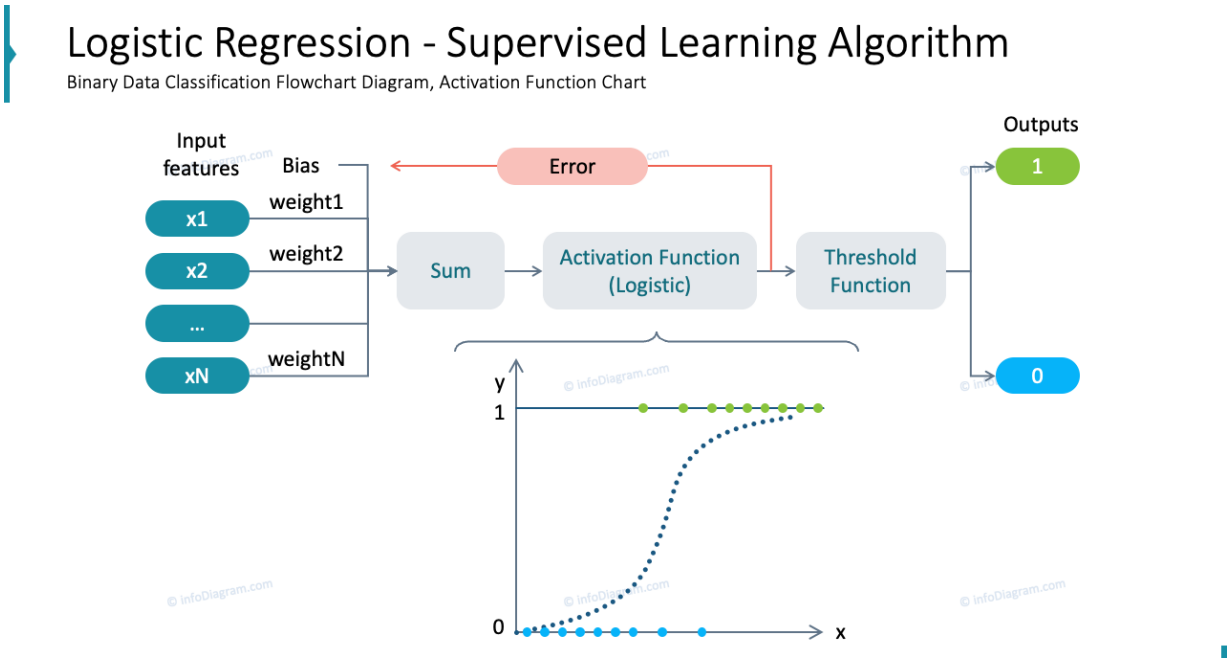


Figure 2: Flowchart of Naïve Bayes

3.1.2.2. Logistic Regression:

Logistic regression (LR) is a linear model used for binary classification that estimates the probability of a post belonging to ADHD or OCD using a logistic (known as sigmoid) function applied to a linear combination of feature weights. TF-IDF vectors serve as input features and the model learns weights that maximizes the likelihood of correct class predictions through gradient descent optimization (Bishop, 2006).



Source:(infoDiagram, 2023)

Figure 3: Logistic Regression Working Architecture

Technical Suitability:

a. **Specialized for binary classification:** It is perfectly suited for the ADHD vs OCD distinction that provides the calibrated probability scores (0-1) for classification decisions (Hosmer, D.W., Lemeshow, S. and Sturdivant, R.X. , 2013).
b. **Handles High dimensional sparse features:** It can efficiently process TF-IDF vectors which are mostly zeros without requiring dimensionality prediction, maintain computational efficiency while also handling the curse of dimensionality effectively (Géron, 2019).

c. **Interpretable Feature Analysis:** Each learned coefficient directly quantifies a word's contribution and the direction enabling a transparency identification of symptoms related vocabulary patterns for each condition (Bishop, 2006).
d. **Robust Regularization for Social Media text:** L1/L2 regularization prevents overfitting to sparse and noisy text which includes informal language, emojis and typos that are mostly found in Reddit posts.

Reason for choosing:

a. **Complementary to SVM and Naïve Bayes:** This completes the trio with Naïve Bayes with probabilistic and SVM with geometric that offer optimization-based contrast for comparative analysis (Rish, 2001).
b. **Text-Specific Efficiency:** LR is better suited than Random Forest for sparse and high-dimensional TF-IDF data that maintains both interpretability and computational performance.
c. **Clinical decision support alignment:** The probability estimates provide clinically relevant confidence assessments that aligns with mental health evaluation frameworks where decisions are often based on likelihood rather than binary predictions (Bishop, 2006).
d. **Practical experimental Value:** It can train quickly on 24,000 reddit allowing efficient iteration and hyperparameter tuning.
e. **Research and practical relevance:** LR is widely used in text classification and NLP for mental health analysis that ensures that results are interpretable and comparable with existing studies (Yang Lin, Xinyu Ma, Xu Chu, Yujie Jin, Zhibang Yang, Yasha Wang, Hong Mei, 2024).

1. Pseudocode of Logistic Regression

START

IMPORT required libraries

IMPORT TfidfVectorizer **FROM** sklearn.feature_extraction.text

IMPORT train_test_split, GridSearchCV **FROM** sklearn.model_selection

IMPORT LogisticRegression **FROM** sklearn.linear_model

IMPORT accuracy_score, classification_report, confusion_matrix FROM sklearn.metrics

IMPORT matplotlib.pyplot **AS** plt

IMPORT seaborn **AS** sns

DEFINE features and target using preprocessed data

SET X equal to df_clean['cleaned_text']

SET y equal to df_clean['subreddit']

SPLIT dataset into training and test sets

USE stratified split

SET test_size **EQUAL TO** 0.20

SET random_state **EQUAL TO** 42

INITIALIZE TF-IDF Vectorizer with optimized settings

SET max_features **EQUAL TO** 10,000

SET ngram_range **EQUAL TO** (1, 3)

SET min_df **EQUAL TO** 5

SET max_df **EQUAL TO** 0.9

SET stop_words **EQUAL TO** None

FIT vectorizer on X_train

TRANSFORM X_train to obtain X_train_tfidf

TRANSFORM X_test to obtain X_test_tfidf

PERFORM hyperparameter tuning **FOR** Logistic Regression

DEFINE parameter grid

SET C values **EQUAL TO** {0.01, 0.1, 1, 5, 10}

INITIALIZE GridSearchCV

SET estimator **EQUAL TO** LogisticRegression(max_iter equal to 2000)

SET cv **EQUAL TO** 5

SET scoring **EQUAL TO** 'accuracy'

FIT GridSearchCV **ON** X_train_tfidf and y_train

EXTRACT best_C **EQUAL TO** grid.best_params_['C']

RESULT: best_C **EQUAL TO** 1

INITIALIZE optimized Logistic Regression classifier

SET lr_model **EQUAL TO** LogisticRegression(C equal to best_C, max_iter equal to 2000)

TRAIN lr_model **ON** X_train_tfidf and y_train

PREDICT labels **ON** test set

SET y_pred equal to lr_model.predict(X_test_tfidf)

EVALUATE performance

COMPUTE accuracy

COMPUTE classification report (precision, recall, F1-score) for ADHD and OCD

COMPUTE confusion matrix

VISUALIZE results

PLOT confusion matrix using seaborn heatmap

DISPLAY results

PRINT accuracy and classification report

SHOW confusion matrix plot

SAVE trained model

DOCUMENT results, observations and key features

END

2. Flowchart of Logistic Regression

The flowchart below visualizes the sequential LR classification process, highlighting data transformation from raw text to TF-IDF features and iterative model optimization.

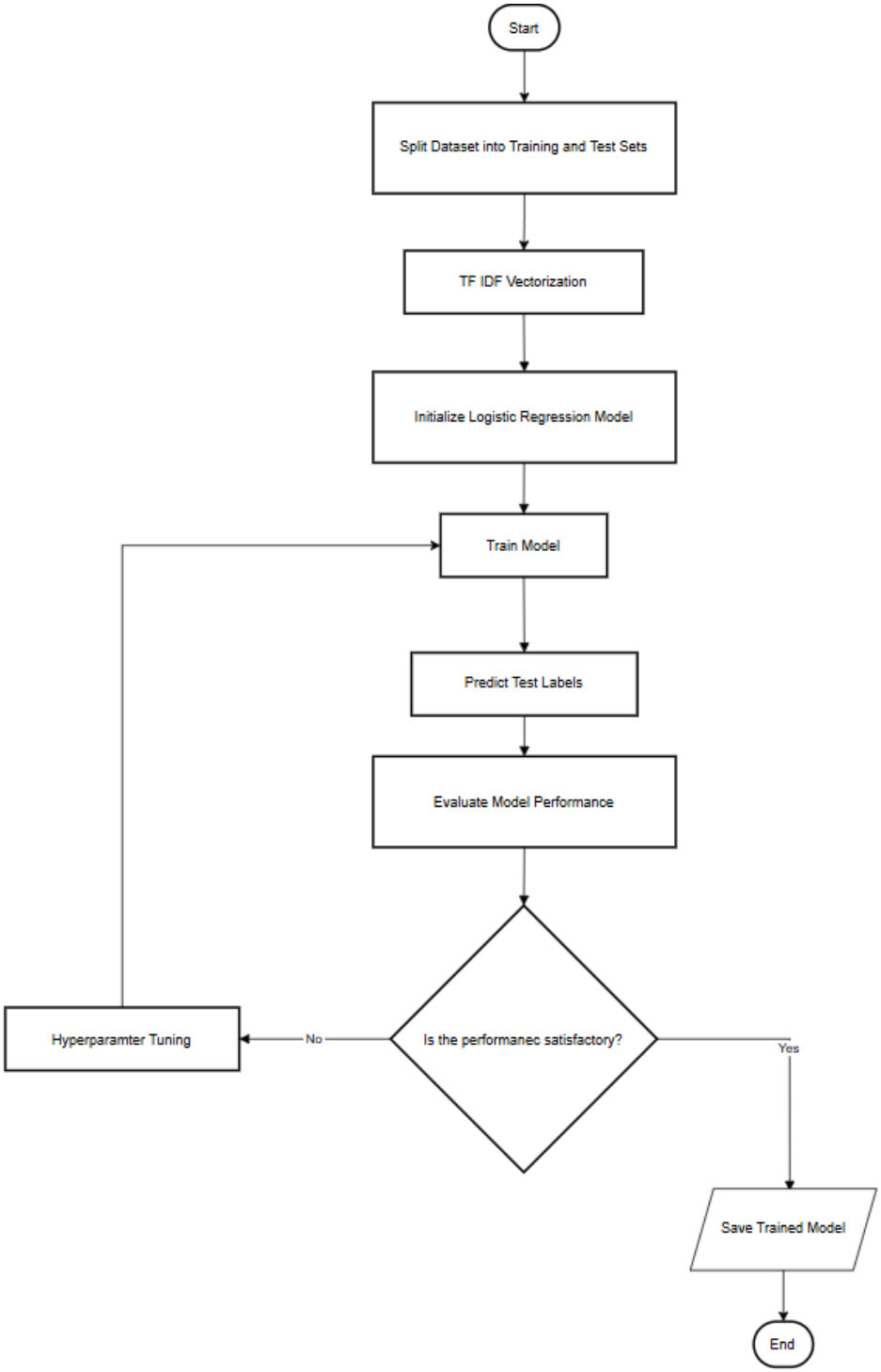
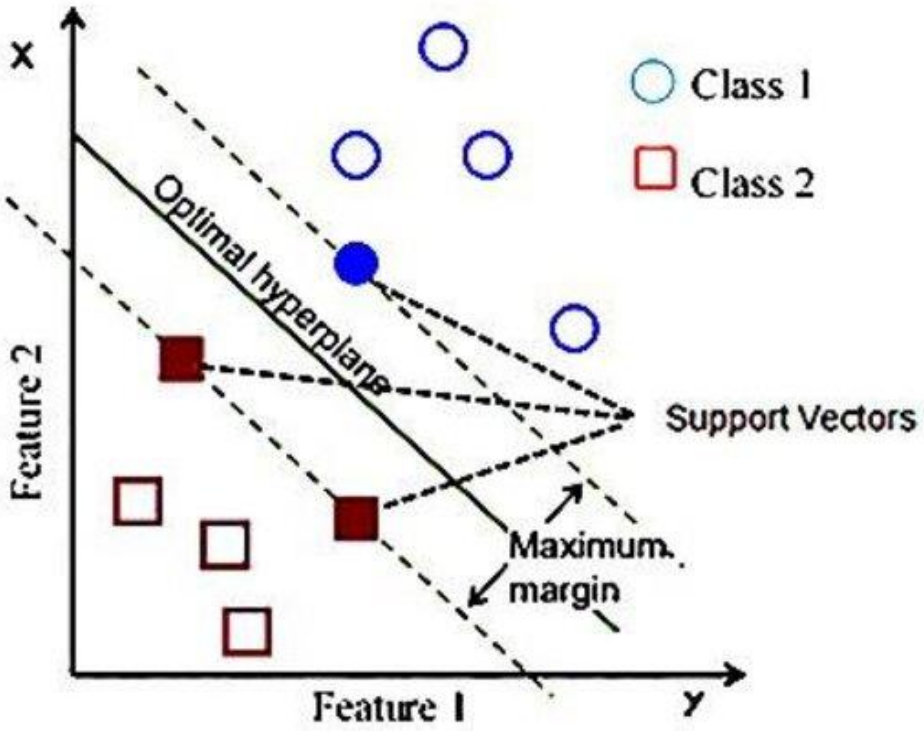


Figure 4: Flowchart of Logistic Regression

3.1.2.3. Support Vector Machine (SVM)

SVM is a margin-based supervised learning algorithm that aims to find the optimal hyperplane that best separates data points of different classes in high dimensional feature space by maximizing the margin between them. For text classification, it transforms TF-IDF and identifies the decision boundary that maximally distinguishes between ADHD and OCD posts (Corinna Cortes & Vladimir Vapnik, 1995).



Source:(Sundas Khan,Samra Urooj Khan,OSAMA SHAFIQUE,AMEER HAMZA, 2025)

Figure 5: SVM Algorithm Working Architecture

Technical Suitability:

a. **Optimizes margin for overlapping symptoms:** It creates robust separation when the posts contain common words occurring in both ADHD and OCD posts, it handles symptom overlap better than other probabilistic methods (Joachims, 1998).
b. **Excels with high dimensional text data:** With 5,000 + TF-IDF features, SVM finds optimal linear separation efficiently leveraging the curse of dimensionality as an advantage.

c. **Feature weight interpretability:** In SVM, learned weights reveals discriminative words providing clinically meaningful insights (Steven Bird, Ewan Klein and Edward Loper, 2009).
d. **Research validated performance:** Literature on mental health text classification consistently identifies SVM as a top performing traditional algorithm for this domain (Anon., 2017).

Reason for choosing:

a. **Proven text classification excellence:** SVM is historically best-performing traditional algorithm for text data that offers superior accuracy to Naïve Bayes while also maintaining interpretability (Joachims, 1998).
b. **Geometric approach:** It provides a fundamentally different optimization-based approach compared to probability based Naïve Bayes that enables richer comparative analysis (Géron, 2019).
c. **Robustness to noise:** It handles Reddit's informal language, typos and irrelevant content through regularization better than other probability-based methods.
d. **Demonstrates Theoretical rigidness:** Implementing hinge loss and gradient optimization highlights comprehension of advanced machine learning concepts.

1. Pseudocode of SVM

```
START
    IMPORT required libraries
    IMPORT TfidfVectorizer FROM sklearn.feature_extraction.text
      IMPORT train_test_split, GridSearchCV FROM sklearn.model_selection
      IMPORT LinearSVC FROM sklearn.svm
      IMPORT accuracy_score, classification_report, confusion_matrix FROM
      sklearn.metrics
      IMPORT matplotlib.pyplot AS plt
      IMPORT seaborn AS sns

      DEFINE features and target using preprocessed data
        SET X EQUAL TO df_clean['cleaned_text']
        SET y EQUAL TO df_clean['subreddit']

      SPLIT dataset into training and test sets
        USE stratified split
          SET test_size EQUAL TO 0.20
          SET random_state EQUAL TO 42

      INITIALIZE TF-IDF Vectorizer with optimized settings
        SET max_features EQUAL TO 10,000
```

SET ngram_range **EQUAL TO** (1, 3)

SET min_df **EQUAL TO** 5

SET max_df **EQUAL TO** 0.9

SET stop_words **EQUAL TO** None

FIT vectorizer **ON** X_train

TRANSFORM X_train to obtain X_train_tfidf

TRANSFORM X_test to obtain X_test_tfidf

PERFORM hyperparameter tuning FOR Linear SVM

DEFINE parameter grid

SET C values **EQUAL TO** {0.01, 0.1, 1, 5, 10}

INITIALIZE GridSearchCV

SET estimator **EQUAL TO** LinearSVC(max_iter **EQUAL TO** 2000)

SET cv **EQUAL TO** 5

SET scoring **EQUAL TO** 'accuracy'

FIT GridSearchCV **ON** X_train_tfidf and y_train

EXTRACT best_C **EQUAL TO** grid.best_params_['C']

RESULT: best_C **EQUAL TO** 0.1

INITIALIZE optimized Linear SVM classifier

SET svm_model **EQUAL TO** LinearSVC(C **EQUAL TO** best_C, max_iter equal to 2000)

TRAIN svm_model **ON** X_train_tfidf and y_train

PREDICT labels **ON** test set

SET y_pred **EQUAL TO** svm_model.predict(X_test_tfidf)

EVALUATE performance

COMPUTE accuracy

COMPUTE classification report (precision, recall, F1-score) **FOR** ADHD and OCD

COMPUTE confusion matrix

VISUALIZE results

PLOT confusion matrix using seaborn heatmap

DISPLAY results

PRINT "Optimized Linear SVM Accuracy: 84.82%"

PRINT classification report

SHOW confusion matrix plot

SAVE trained model

DOCUMENT results, observations and key features

END

2. Flowchart of SVM

The flowchart in the below figure illustrates the SVM pipeline, emphasizing the creation of an optimal separating hyperplane in high-dimensional feature space.

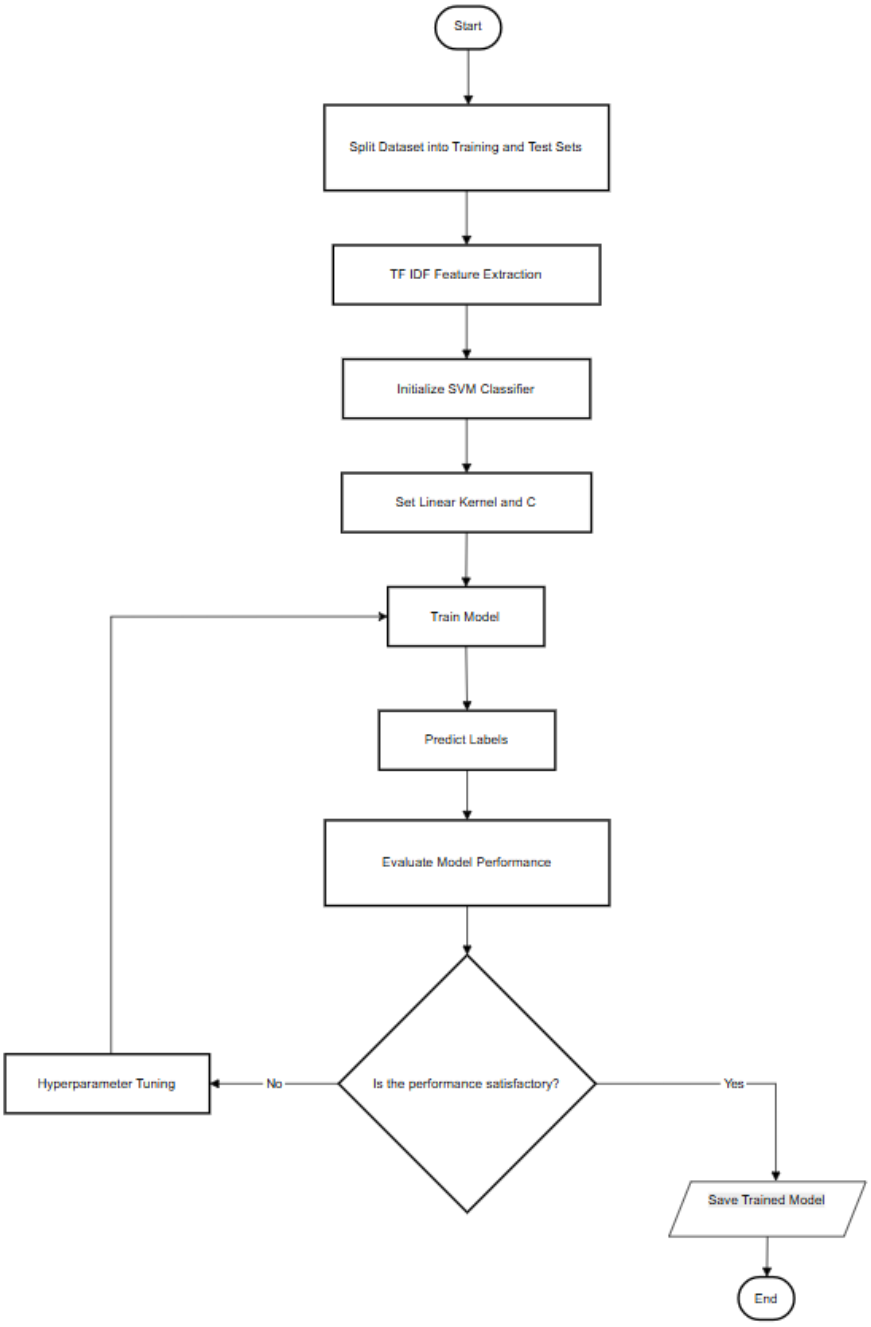


Figure 6: Flowchart of SVM

3.2. Pseudocode of the overall system

Pseudocode of the Mental Health Classifier Pipeline

START

IMPORT pandas, numpy

IMPORT matplotlib, seaborn

IMPORT nltk, re, emoji, contractions

IMPORT sklearn (train_test_split, TfidfVectorizer, classifiers, metrics)

IMPORT VotingClassifier, StackingClassifier, GridSearchCV

INITIALIZE WordNetLemmatizer

INITIALIZE stopwords set

INITIALIZE negation words set

LOAD dataset.csv **AS** df

DISPLAY df.shape, df.columns, class distribution

CHECK for missing values and duplicates

REMOVE duplicate posts

UPDATE class distribution

CALCULATE text_length, word_count per post

PLOT histograms for text_length and word_count by class

ANALYZE negation preservation in posts

FUNCTION preprocess(text):

IF text is empty **OR** not string:

RETURN empty string

LOWERCASE text

EXPAND contractions

REMOVE URLs, Reddit mentions

REMOVE emojis

REMOVE markdown/HTML entities

REMOVE unwanted characters

REMOVE standalone numbers

TOKENIZE text

HANDLE negations (merge negation + next token)

REMOVE stopwords except negated tokens

REMOVE very short tokens less than 3 characters

LEMMATIZE tokens preserve negation tokens

JOIN tokens **INTO** cleaned string

RETURN cleaned string

END FUNCTION

APPLY preprocess() to all rows in 'body' column

STORE result **AS** df['cleaned_text']

SET X **AS** df['cleaned_text']

SET y **AS** df['subreddit']

SPLIT X, y **INTO** training (80%) and test (20%) sets with stratification

INITIALIZE TfidfVectorizer with max_features, ngram_range, min_df, max_df

FIT vectorizer **ON** X_train

TRANSFORM X_train, X_test **INTO** TF-IDF matrices

INITIALIZE MultinomialNB(alpha)

IF hyperparameter tuning required:

  USE GridSearchCV for alpha

  SELECT best alpha

FIT Naïve Bayes **ON** X_train_tfidf, y_train

PREDICT ON X_test_tfidf

EVALUATE accuracy, precision, recall, F1-score, confusion matrix

INITIALIZE LogisticRegression(max_iter, random_state)

IF hyperparameter tuning required:

  USE GridSearchCV for C

  SELECT best C

FIT Logistic Regression **ON** X_train_tfidf, y_train

PREDICT ON X_test_tfidf

EVALUATE accuracy, precision, recall, F1-score, confusion matrix

INITIALIZE LinearSVC(max_iter, random_state)

IF hyperparameter tuning required:

USE GridSearchCV **FOR** C

SELECT best C

FIT Linear SVM **ON** X_train_tfidf, y_train

PREDICT ON X_test_tfidf

EVALUATE accuracy, precision, recall, F1-score, confusion matrix

INITIALIZE VotingClassifier **WITH** estimators **AS** [NB, LR, SVM and voting **AS** 'hard'

FIT ensemble **ON** X_train_tfidf, y_train

PREDICT ON X_test_tfidf

EVALUATE accuracy, precision, recall, F1-score

INITIALIZE SVM with probability **AS** True

INITIALIZE VotingClassifier with voting='soft'

FIT ensemble **ON** X_train_tfidf, y_train

PREDICT ON X_test_tfidf

EVALUATE accuracy, precision, recall, F1-score

INITIALIZE base estimators: NB, LR, SVM

INITIALIZE final estimator: LogisticRegression

FIT StackingClassifier **ON** X_train_tfidf, y_train

PREDICT ON X_test_tfidf

EVALUATE accuracy, precision, recall, F1-score

PLOT confusion matrix

SAVE trained models and TF-IDF vectorizer

COMPARE all model performances:

COLLECT accuracy scores: NB, LR, SVM, Hard Voting, Soft Voting, Stacking

PLOT comparative bar chart

SELECT best performing model

IF Stacking has highest accuracy:

SET final_model **AS** Stacking Ensemble

FUNCTION predict_new_post(text):

cleaned = equal preprocess(text)

vectorized = tfidf.transform([cleaned])

prediction = final_model.predict(vectorized)

confidence = final_model.predict_proba(vectorized)

RETURN prediction, confidence

END FUNCTION

TEST predict_new_post() with posts

ADD models and preprocessing scripts to Git repository

COMMIT changes with meaningful message

PUSH to GitHub

END

3.3. Diagrammatical representations of the solution

3.3.1. State Transition Diagram

The below figure presents the comprehensive system architecture that integrates all components of the proposed solution. This state transition diagram illustrates the end-to-end workflow from data acquisition to classification output, showing how the three algorithms operate within a unified framework. The architecture emphasizes the shared preprocessing and feature extraction stages, followed by parallel model training and evaluation pathways.

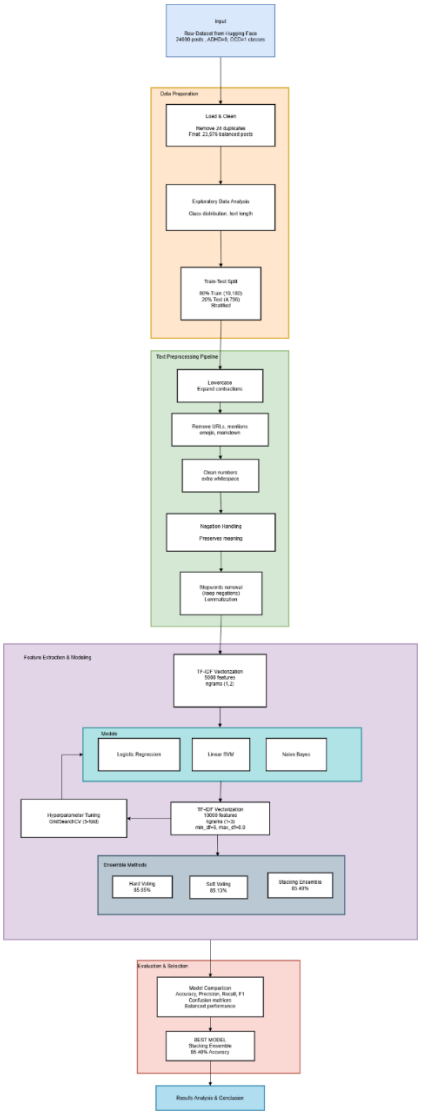


Figure 7: State Transitional Diagram of the overall System

3.3.2. Flow chart

The flowchart in the below figure illustrates the overall pipeline of the system, emphasizing the creation of an optimal separating hyperplane in high-dimensional feature space. This diagram illustrates the end-to-end workflow from data acquisition to classification output, showing how the three algorithms operate within a unified framework.

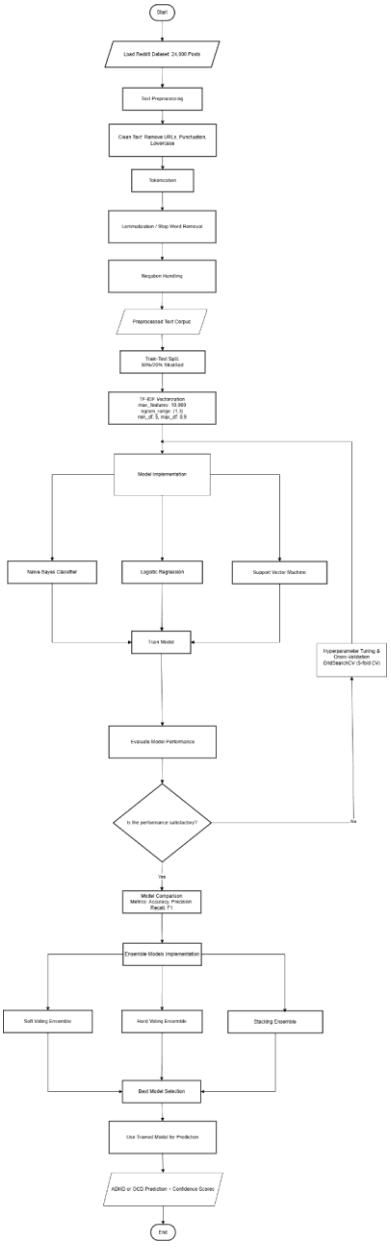


Figure 8: Flowchart of the overall system

3.4. Explanation of the Development Process

The development of the AI application followed a systematic and incremental process to ensure clarity correctness and reproducibility since the project is based on the classical artificial intelligence techniques, the entire pipeline was manually designed and implemented rather than relying on end to end automated or deep learning approaches. The development process was carried out using Python in Jupyter Notebook allowing step by step execution and close inspection of intermediate outputs.

3.4.1 Development Environment and Tools

The application was developed in a local environment using Jupyter Notebook which was selected for its ability to support interactive development and iterative testing. It enabled the execution of individual code cells, making it easier to verify each stage of the AI pipeline independently.

The following tools and libraries were used during development:

- **Python:** Python was used as the core programming language to implement all components of the application.
- **Jupyter Notebook:** It is used for development, experimentation and result visualization.
- **Pandas:** It is used for loading and managing structured datasets.
- **NumPy:** It is used for numerical operations and matrix-based computations.
- **NLTK:** It is used for natural language preprocessing tasks such as tokenization, stopwords removal and lemmatization.
- **Re and Contradiction:** It is used for natural language preprocessing the regular expression for text cleaning.
- **Matplotlib:** It is used for the plotting and graphical representation of the models.
- **Seaborn:** It is used for statistical data visualization in the project.
- **Scikit-learn:** It is used for TF-IDF feature extraction, model implementation and evaluation metrics.
- **GitHub:** The entire project is maintained on GitHub using Git for version control following modular organization for reproducibility and collaborative development.

All libraries used are open-source and widely adopted in academic and industrial AI applications.

3.4.2. Data Loading and Initial Inspection

The first stage of development involved loading the dataset into the application and performing an initial inspection of the data. This step was essential to understand the structure of the data, the number of instances and the distribution of class labels.

During this stage, the following steps were taken:

1. Load Dataset

```
[12]: #Load Dataset
      df= pd.read_csv('dataset.csv')
      df
```

[12]:

	body	subreddit
0	Thank you.\n\nMy son was recently diagnosed an...	1
1	Omg. Yes, I didn't realize how close this was ...	0
2	I love how our positive way is to laugh of our...	1
3	I am really, really struggling with the sudden...	0
4	i was just unconciously procrastinating doing ...	1
...	...	...
23995	I have to walk around in a particular pattern ...	0
23996	I handwash my water bottles and i may have had...	0
23997	Love how the same people who say "oh I'm so OC...	0
23998	i wish my brain would obsess over the tasks at...	0
23999	That's a good idea. I like to make things tang...	0

24000 rows × 2 columns

Figure 9: Load dataset

2. Basic Information

```
print(f"Dataset Shape: {df.shape}")
```

```
Dataset Shape: (24000, 2)
```

```
print(f"Columns: {df.columns.tolist()}")
```

```
Columns: ['body', 'subreddit']
```

```
print("\nFirst 5 rows:")
print(df.head())
```

```
First 5 rows:
                                                body  subreddit
0  Thank you.\n\nMy son was recently diagnosed an...          1
1  Omg. Yes, I didn't realize how close this was ...          0
2  I love how our positive way is to laugh of our...          1
3  I am really, really struggling with the sudden...          0
4  i was just unconciously procrastinating doing ...          1
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24000 entries, 0 to 23999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   body       24000 non-null  object
 1   subreddit  24000 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 375.1+ KB
```

Figure 10: Basic data Information

The initial dataset consisted of 24000 reddit posts with 2 columns including :

- body
- subreddit

The initial data consist of unnecessary words and punctuation which needed to be removed and cleaned.

3. Initial Class Distribution

```
# Analyze Class Distribution
class_dist = df['subreddit'].value_counts()
print("Class Distribution:")
print(f"ADHD posts (0): {class_dist[0]}")
print(f"OCD posts (1): {class_dist[1]}")
```

```
Class Distribution:
ADHD posts (0): 12000
OCD posts (1): 12000
```

Figure 11: Class Distribution Analysis

4. Visualization of the class Distribution

```
[22]: # Visualization
plt.figure(figsize=(8, 5))
colors = ['#FF6B6B', '#4ECDC4']
plt.bar(['ADHD (0)', 'OCD (1)'], [class_dist[0], class_dist[1]],
        color=colors, edgecolor='black')
plt.title('Class Distribution in Dataset', fontsize=14, fontweight='bold')
plt.xlabel('Mental Health Condition', fontsize=12)
plt.ylabel('Number of Posts', fontsize=12)
plt.text(0, class_dist[0]+200, str(class_dist[0]), ha='center', fontweight='bold')
plt.text(1, class_dist[1]+200, str(class_dist[1]), ha='center', fontweight='bold')
plt.grid(axis='y', alpha=0.3)
plt.show()
```

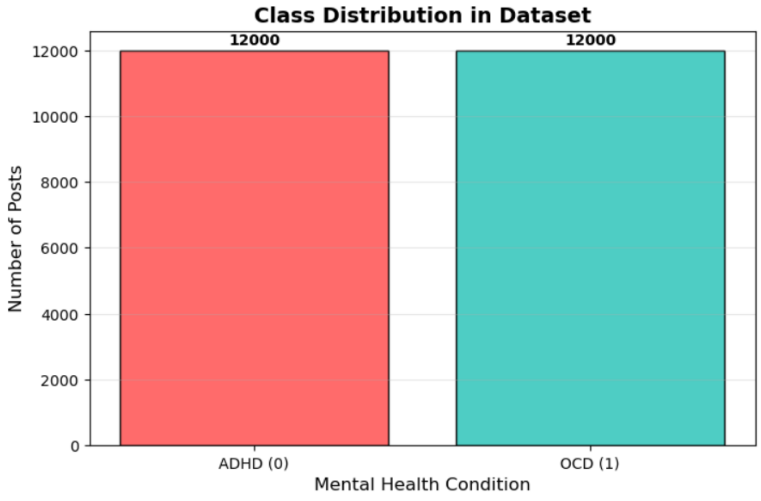


Figure 12: Visual representation of Class distribution

The dataset loading process began with importing the CSV file containing 24,000 Reddit posts evenly split between ADHD and OCD discussions. Initial inspection revealed a perfectly balanced dataset, 12,000 posts for each condition which is ideal for training classification models without needing class imbalance mitigation techniques.

5. Null or missing values

There were no null or missing values in the current dataset.

```
8]: df.isnull().sum()

8]: body         0
    subreddit    0
    dtype: int64
```

Figure 13: Checking null values

3.4.3. Checking duplicate data and its distribution across the classes

There were 24 duplicate values and each are shown below.

```
# Find all duplicates in the 'body' column
duplicates = df[df.duplicated(subset='body', keep=False)]

# Group by body text and subreddit (class) to see counts
duplicates_summary = duplicates.groupby(['body', 'subreddit']).size().reset_index(name='count')

# Show the duplicates with counts
print(duplicates_summary)
```

	body	subreddit	count
0	Call Of The Void, basically? COTV is one of th...	0	2
1	Honestly? This whole COVID thing has been rath...	0	2
2	I HAVE TO WRITE AN EMAIL TOO! Literally no dow...	1	2
3	I just came out of the shower and this reminde...	1	2
4	I love the way it looks, with the bit of shaki...	0	2
5	I struggle to clean because I tend more toward...	0	2
6	If you or someone you know is contemplating su...	0	2
7	It took me 4 months to return over $200 worth ...	1	2
8	I'd like to get back the ability to travel wit...	0	2
9	My life...\n\nLike, I know I'm going to be the...	1	2
10	My whole heart and soul are with you on this o...	1	2
11	Oh my God I haven't checked mine in months lik...	1	2
12	Ohh man. My tactic (my very inconsistent but e...	1	2
13	Omg as I sit here trying to will myself to do ...	1	2
14	Omg same!! I have an email to reply to and it'...	1	2
15	Poorly!\n\nMy semester was already 100% online...	1	2
16	Sorry but you can't just blame ADHD for this.....	1	2
17	This headline made me laugh so hard. And then ...	1	2
18	We need to make a 3,000 mile move in a couple ...	1	2
19	Yeah this works for minor obsessions, anything...	0	2
20	Yes! OMG thank you for this explanation. I hav...	1	2
21	You are living my life. Just stop it. I person...	1	2
22	You're not alone! I never really thought about...	0	2
23	"I would like to begin my ten-part apology by ...	1	2

[38]:

Figure 14: Checking duplicate values

1. Visualization of the number of duplicate posts per subreddit

The data quality checks were conducted which identified and removed 24 duplicate posts (16 from ADHD, 32 from OCD) to prevent data leakage and ensure each training example was unique.

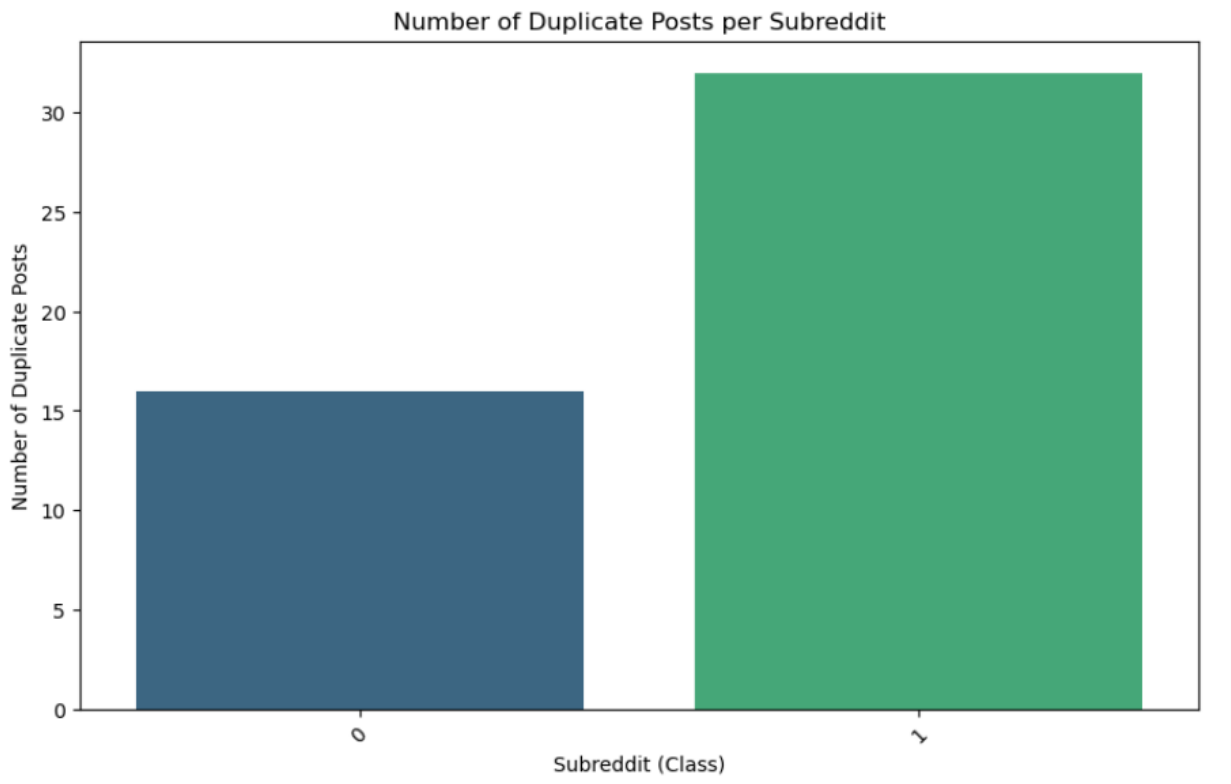


Figure 15: Visualization of the number of duplicate posts per subreddit

2. After removing the duplicate posts

Removing the duplicate posts to remove any bias or unwanted influence between the classes to improve the overall accuracy and prediction of labels.

```
FINAL CLASS DISTRIBUTION (After Cleaning)
Total unique posts: 23976
ADHD posts (subreddit=0): 11992
OCD posts (subreddit=1): 11984
Dataset balance: 1.00:1
```

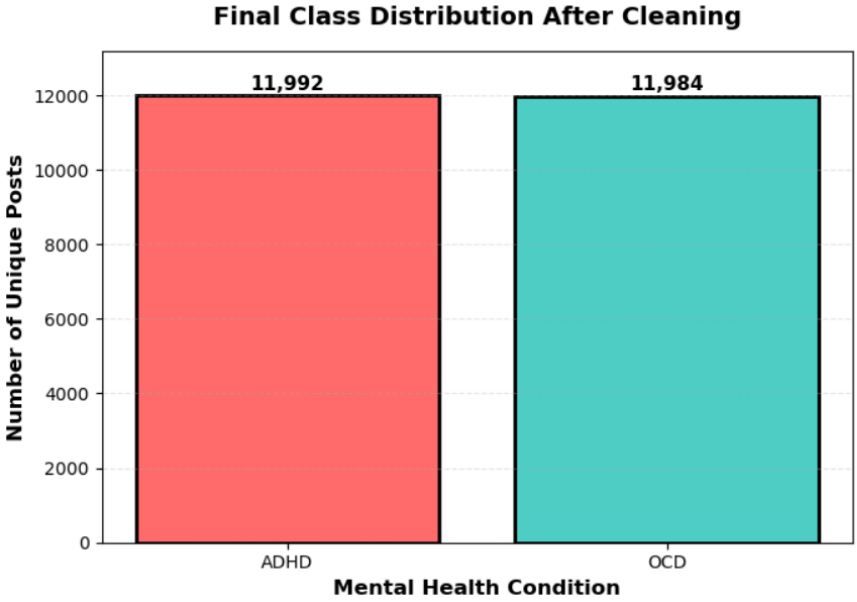


Figure 16: Final Class distribution after cleaning

3. After Removing duplicates

After the cleaning, the dataset the total post came down to 23,976 from 24,000, removing 24 unwanted elements. The total number of posts, average word count , minimum words and maximum words in each class are shown below.

```
TEXT ANALYSIS AFTER PREPROCESSING
============================================================
OVERALL STATISTICS:
Total posts: 23,976
Average words per post: 22.3
Average characters per post: 154.0
Minimum words: 0
Maximum words: 645
BY CONDITION:

ADHD:
 Posts: 11,992
 Avg words: 18.0
 Min words: 0
 Max words: 588

OCD:
 Posts: 11,984
 Avg words: 26.7
 Min words: 0
 Max words: 645
NEGATION PRESERVATION:
Posts containing negations: 11552
Percentage: 48.2%
TOP 10 MOST COMMON NEGATIONS:
 not_have: 895 times
 not_know: 865 times
 not_even: 487 times
 not_want: 418 times
 not_a: 384 times
 not_get: 331 times
 not_be: 326 times
 not_the: 296 times
 not_do: 275 times
 not_think: 244 times
```

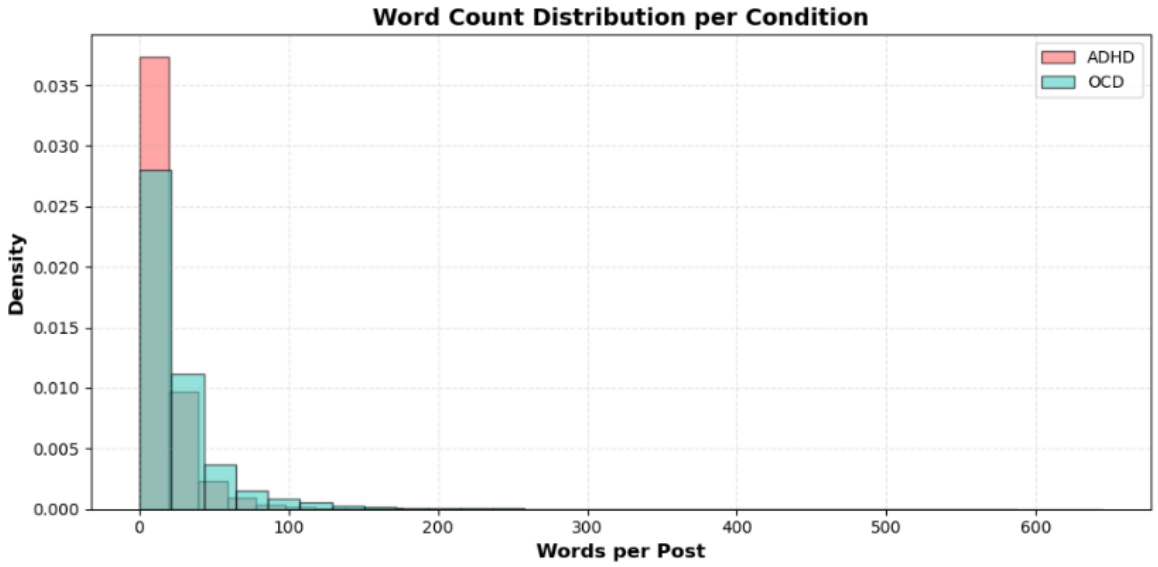


Figure 17: Text analysis after removing duplicate

After cleaning and removing duplicate posts, the dataset consisted of 23,976 unique posts, almost equally balanced between the two classes:

- ADHD (subreddit=0): 11,992 posts
- OCD (subreddit=1): 11,984 posts

This balanced distribution ensures that the models do not favour one class over the other.

3.4.4. Advanced Text Preprocessing

The text preprocessing pipeline was carefully designed to handle the unique challenges of social media text, particularly for mental health discussions. Basic preprocessing techniques such as tokenization, lemmatizing and URL removal applied along with some other reddit/ social media post specific preprocessing techniques. Each step addresses specific issues while preserving clinically relevant information.

Emoji removal was necessary since, the raw reddit post consist of emoji which cannot be consistently interpreted by the text classification models. Removing emojis focuses the model on linguistic patterns rather than visual symbols.

```
# Step 3: Remove URLs and Reddit mentions
text = re.sub(r'http\S+|www\S+|https\S+', '', text)
text = re.sub(r'r/\w+|u/\w+', '', text)
```

Figure 18: Code of URL removal

In mental health language classification, negation preservation was the most critical innovation which can fundamentally change the meaning: "I am not anxious" versus "I am anxious". Though they have completely opposite meaning, if negation preservation is not applied, the context can be lost when "not" is removed as a stopwords.

```
# Step 10: Handle negations carefully
processed_tokens = []
i = 0
while i < len(tokens):
    token = tokens[i]
    if token in negation_words and i + 1 < len(tokens):
        next_token = tokens[i + 1]
        # merge only if next token is a valid word
        if next_token.isalpha():
            processed_tokens.append(f"{token}_{next_token}")
            i += 2
            continue
    processed_tokens.append(token)
    i += 1

# Step 11: Remove stopwords but keep negated tokens
filtered_tokens = [t for t in processed_tokens if (t not in stop_words or '_' in t)]
```

Figure 19: Code of handing negation preservation

After the preprocessing function was applied, the new cleaned text consisted of only necessary words and characters sent for model training.

	body	clean_text
0	Thank you.\n\nMy son was recently diagnosed an...	thank you. son recently diagnosed want whateve...
1	Omg. Yes, I didn't realize how close this was ...	omg. yes, not_realize close obsession-compulsi...
2	I love how our positive way is to laugh of our...	love positive way laugh say fuck it. accept sh...
3	I am really, really struggling with the sudden...	really, really struggling sudden change right....
4	i was just unconciously procrastinating doing ...	unconciously procrastinating homework thank

Figure 20: Comparison of original and clean text

3.4.5. Feature Extraction with TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) was chosen to convert text into numerical features where words were weighted by their importance in individual documents relative to their commonness across the corpus.

Two configurations were tested:

- **Initial**: 5,000 features with unigrams and bigrams

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TF-IDF vectorizer
tfidf = TfidfVectorizer(
    max_features=5000,  # Limit vocabulary size
    ngram_range=(1,2),  # Unigrams + bigrams
    stop_words=None     # Already removed stopwords in preprocessing
)

# Fit on training data and transform
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

print("TF-IDF feature shape:")
print(X_train_tfidf.shape)
```

```
TF-IDF feature shape:
(19180, 5000)
```

Figure 21: Initial TF-IDF vectorization

Optimized: 10,000 features including trigrams

```
[62]: # trying to imporve accuracy
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TF-IDF vectorizer
tfidf = TfidfVectorizer(
    max_features=10000,  # Limit vocabulary size
    ngram_range=(1,3),  # Unigrams + bigrams
    min_df=5,    # Ignore words that appear in fewer than 5 documents
    max_df=0.9   # Ignore words that appear in >90% of documents

)

# Fit on training data and transform
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

print("TF-IDF feature shape:")
print(X_train_tfidf.shape)
```

```
TF-IDF feature shape:
(19180, 10000)
```

Figure 22:Optimized TF-IDF vectorization

The optimized configuration performed better because the three-word phrases captured important mental health patterns like “hard_to_focus” or “can’t_stop_thinking”. The min_df=5 parameter filtered rare words (often typos), while max_df=0.9 removed overly common words with little discriminative power.

3.4.6. Model Implementation and Training

The three machine learning algorithms were implemented each representing a different approach to classification.

1. Naïve Bayes

Naïve Bayes used the Multinomial variant with Laplace smoothing (alpha=1.0) with an accuracy of 83.99%. Its probabilistic approach and independence assumption made it fast but potentially limited in capturing word interactions.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
# Train Naïve Bayes
nb_model = MultinomialNB(alpha=1.0)
nb_model.fit(X_train_tfidf, y_train)

# Predict
y_pred_nb = nb_model.predict(X_test_tfidf)

# Accuracy
acc_nb = accuracy_score(y_test, y_pred_nb)
print(f"Naïve Bayes Accuracy: {acc_nb*100:.2f}%\n")

# Classification report
print(classification_report(y_test, y_pred_nb, target_names=['ADHD','OCD']))

# Confusion matrix
cm_nb = confusion_matrix(y_test, y_pred_nb)
plt.figure(figsize=(6,5))
sns.heatmap(cm_nb, annot=True, fmt='d', cmap='Oranges', xticklabels=['ADHD','OCD'], yticklabels=['ADHD','OCD'])
plt.title('Naïve Bayes Confusion Matrix')
plt.show()
```

```
Naïve Bayes Accuracy: 83.99%

              precision    recall  f1-score   support

        ADHD       0.84      0.84      0.84      2399
         OCD       0.84      0.84      0.84      2397

    accuracy                           0.84      4796
   macro avg       0.84      0.84      0.84      4796
weighted avg       0.84      0.84      0.84      4796
```

Figure 23: Baseline Naïve Bayes Performance Metric

ROC Curve

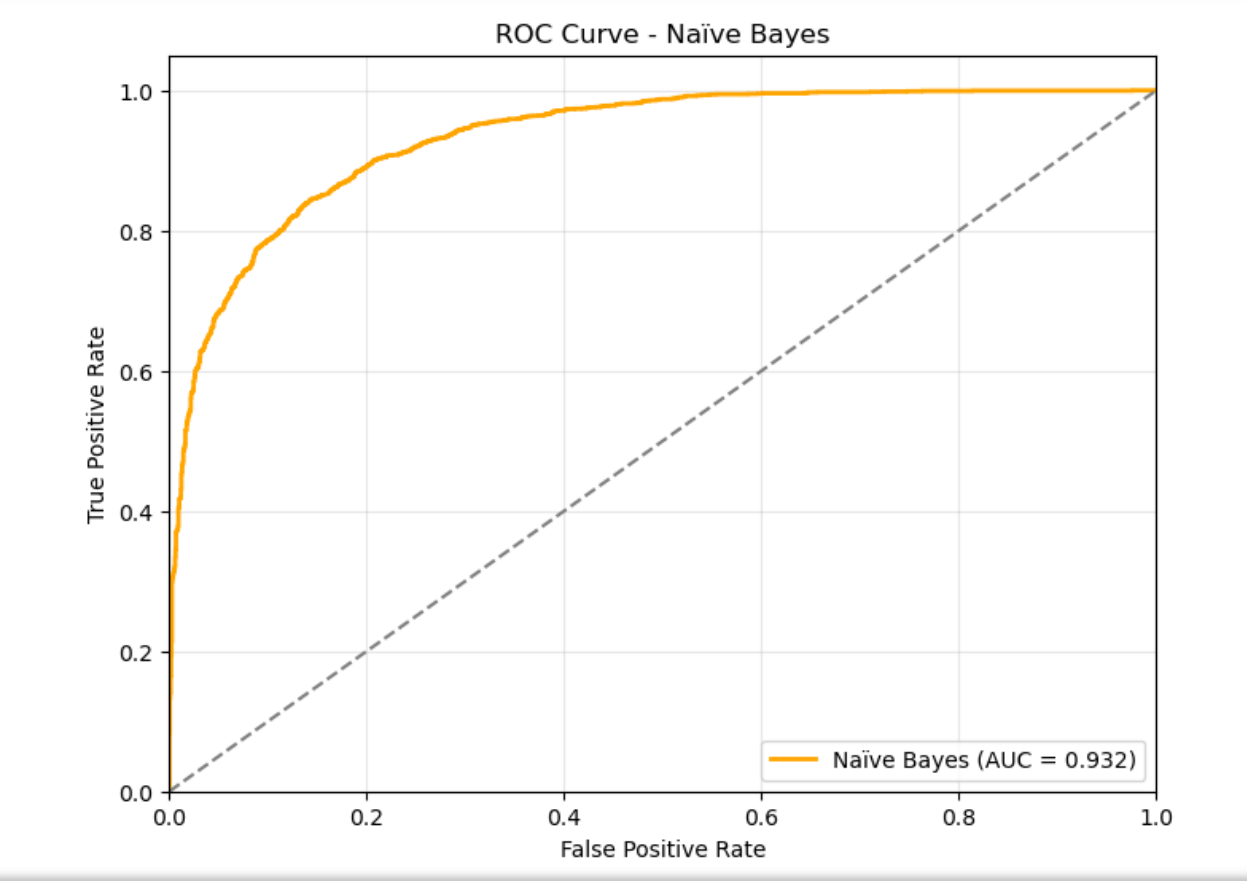


Figure 24: ROC Curve of Naïve Bayes

2. Logistic Regression

Logistic Regression with L2 regularization (C=1.0) provided excellent interpretability through feature coefficients. Its linear approach worked well with the high dimensional text features that made it the best individual performance with an accuracy of 84.57%.

```
# Train Logistic Regression
lr_model = LogisticRegression(max_iter=1000, random_state=42)
lr_model.fit(X_train_tfidf, y_train)

# Predict
y_pred_lr = lr_model.predict(X_test_tfidf)

# Accuracy
acc_lr = accuracy_score(y_test, y_pred_lr)
print(f"Logistic Regression Accuracy: {acc_lr*100:.2f}%\n")

# Classification report
print(classification_report(y_test, y_pred_lr, target_names=['ADHD','OCD']))

# Confusion matrix
cm_lr = confusion_matrix(y_test, y_pred_lr)
plt.figure(figsize=(6,5))
sns.heatmap(cm_lr, annot=True, fmt='d', cmap='Blues', xticklabels=['ADHD','OCD'], yticklabels=['ADHD','OCD'])
plt.title('Logistic Regression Confusion Matrix')
plt.show()
```

```
Logistic Regression Accuracy: 84.57%

              precision    recall  f1-score   support

        ADHD       0.85      0.83      0.84      2399
         OCD       0.84      0.86      0.85      2397

    accuracy                           0.85      4796
   macro avg       0.85      0.85      0.85      4796
weighted avg       0.85      0.85      0.85      4796
```

Figure 25: Baseline Logistic Regression Performance Metric

ROC Curve

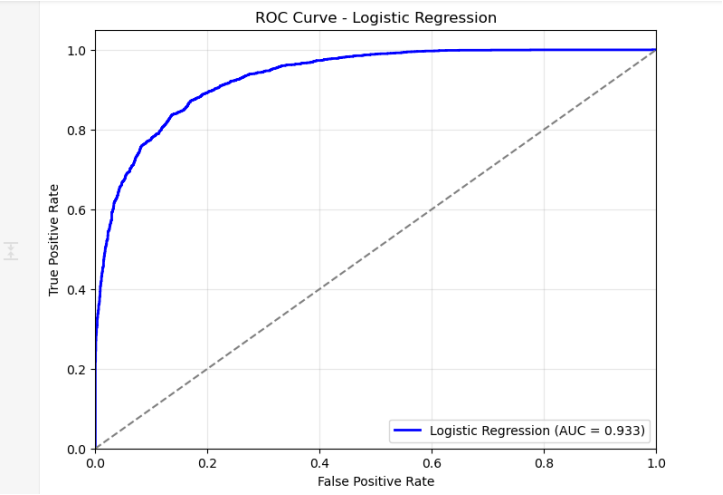


Figure 26: ROC Curve of Logistic Regression

3. SVM

Support Vector Machine with linear kernel (C=1.0) used margin maximization to find optimal separation between the two classes while slightly less accuracy of 83.34%, it showed strength in handling borderline cases.

[56]:
```
from sklearn.svm import LinearSVC

# Train Linear SVM
svm_model = LinearSVC(max_iter=1000, random_state=42)
svm_model.fit(X_train_tfidf, y_train)

# Predict
y_pred_svm = svm_model.predict(X_test_tfidf)

# Accuracy
acc_svm = accuracy_score(y_test, y_pred_svm)
print(f"Linear SVM Accuracy: {acc_svm*100:.2f}%\n")

# Classification report
print(classification_report(y_test, y_pred_svm, target_names=['ADHD','OCD']))

# Confusion matrix
cm_svm = confusion_matrix(y_test, y_pred_svm)
plt.figure(figsize=(6,5))
sns.heatmap(cm_svm, annot=True, fmt='d', cmap='Greens', xticklabels=['ADHD','OCD'], yticklabels=['ADHD','OCD'])
plt.title('Linear SVM Confusion Matrix')
plt.show()
```

```
Linear SVM Accuracy: 83.34%

              precision    recall  f1-score   support

        ADHD       0.84      0.83      0.83      2399
         OCD       0.83      0.84      0.83      2397

    accuracy                           0.83      4796
   macro avg       0.83      0.83      0.83      4796
weighted avg       0.83      0.83      0.83      4796
```

Figure 27: Baseline SVM Performance Metric

ROC Curve

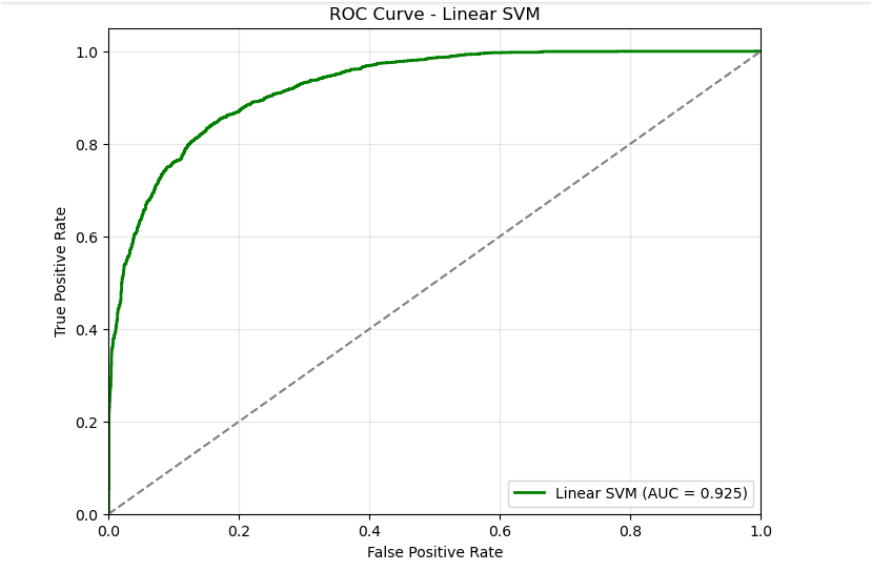


Figure 28:ROC Curve of SVM

Initial comparative analysis of the three model gave the honourable accuracy to Logistic Regression with 84.57% accuracy even though the win was through a tighter difference among the other two models.

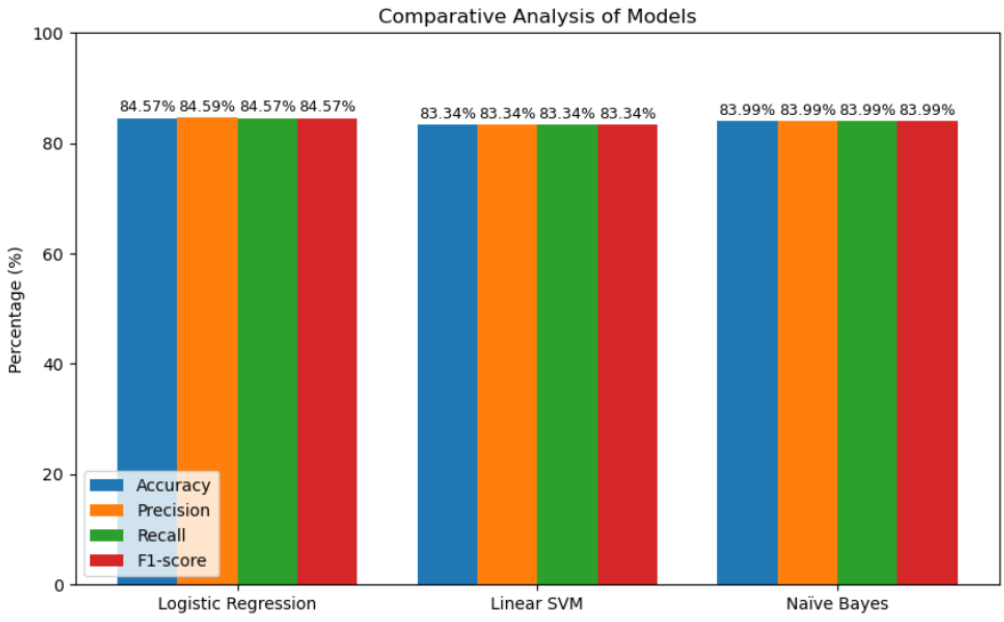


Figure 29: Comparative analysis of Baseline Models

3.4.7. Hyperparameter Optimization

GridSearchCV with 5-fold cross-validation was used to find optimal parameters for each model:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

# Logistic Regression
grid_lr = GridSearchCV(LogisticRegression(max_iter=2000), {'C': [0.01,0.1,1,5,10]}, cv=5, scoring='accuracy')
grid_lr.fit(X_train_tfidf, y_train)
print("Best LR params:", grid_lr.best_params_)
best_lr = grid_lr.best_estimator_

# Linear SVM
grid_svm = GridSearchCV(LinearSVC(max_iter=2000), {'C': [0.01,0.1,1,5,10]}, cv=5, scoring='accuracy')
grid_svm.fit(X_train_tfidf, y_train)
print("Best SVM params:", grid_svm.best_params_)
best_svm = grid_svm.best_estimator_

# Naïve Bayes
grid_nb = GridSearchCV(MultinomialNB(), {'alpha': [0.1,0.5,1.0,1.5,2.0]}, cv=5, scoring='accuracy')
grid_nb.fit(X_train_tfidf, y_train)
print("Best NB params:", grid_nb.best_params_)
best_nb = grid_nb.best_estimator_
```

```
Best LR params: {'C': 1}
Best SVM params: {'C': 0.1}
Best NB params: {'alpha': 1.5}
```

Figure 30: Code of Hyperparameter optimization

The optimal C=1.0, 0.1 and alpha=1.5 in logistic Regression, SVM and Naïve Bayes was found to be their balanced regularization for this dataset, respectively. Cross-validation ensured parameters generalized well beyond the training set.

```
# After grid search, evaluate each optimized model individually

# Evaluate Optimized Logistic Regression
y_pred_lr_opt = best_lr.predict(X_test_tfidf)
acc_lr_opt = accuracy_score(y_test, y_pred_lr_opt)
print(f"Optimized Logistic Regression Accuracy: {acc_lr_opt*100:.2f}%")
print(f"  Improvement over baseline: {acc_lr_opt*100 - 84.57:.2f}%")

# Evaluate Optimized SVM
y_pred_svm_opt = best_svm.predict(X_test_tfidf)
acc_svm_opt = accuracy_score(y_test, y_pred_svm_opt)
print(f"Optimized SVM Accuracy: {acc_svm_opt*100:.2f}%")
print(f"  Improvement over baseline: {acc_svm_opt*100 - 83.34:.2f}%")

# Evaluate Optimized Naïve Bayes
y_pred_nb_opt = best_nb.predict(X_test_tfidf)
acc_nb_opt = accuracy_score(y_test, y_pred_nb_opt)
print(f"Optimized Naïve Bayes Accuracy: {acc_nb_opt*100:.2f}%")
print(f"  Improvement over baseline: {acc_nb_opt*100 - 83.99:.2f}%")
```

```
Optimized Logistic Regression Accuracy: 84.84%
  Improvement over baseline: 0.27%
Optimized SVM Accuracy: 84.82%
  Improvement over baseline: 1.48%
Optimized Naïve Bayes Accuracy: 84.67%
  Improvement over baseline: 0.68%
```

Figure 31: Training models after hyperparameter tuning

After the Hyperparameter tuning and optimal TFIDF, there was a significant rise in accuracy but Logistic regression still was the top performer.

3.4.8. Ensemble Methods

Building upon the detailed implementation of the three ensemble strategies described in the Section 3.1.1.(c)., this section presents the empirical results and comparative analysis of these approaches As previously established, ensemble methods combine multiple base models to produce a single, superior prediction, leveraging the principle that a group of weak learners can form a strong learner (Dietterich, 2000).This approach is particularly effective for text classification, where different models may excel at recognizing different linguistic patterns.

1. Hard Voting Ensemble Results

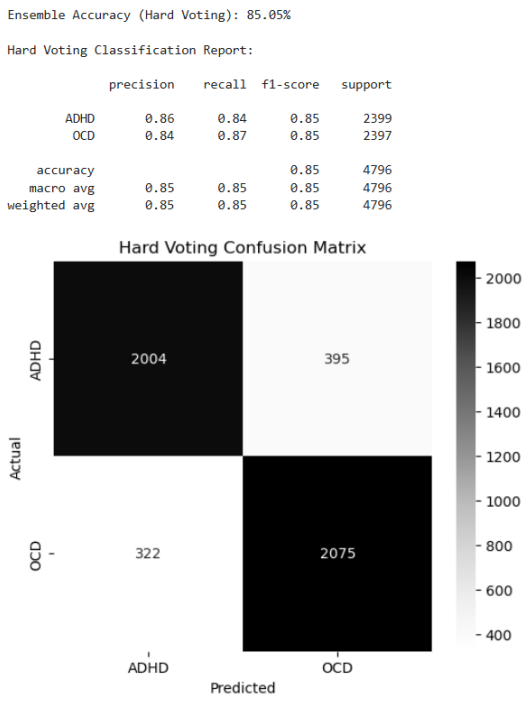


Figure 32:Hard Voting Ensemble Performance Metric

The Hard Ensemble, which operates on the majority voting principle achieved an accuracy of 85.05%. The classification report showed the precision (ADHD: 86%, OCD: 84%) and recall (ADHD: 84%, OCD: 87%) for both classes, with F1-scores of 0.85 for each. The confusion matrix reveals that the model correctly classified 2041 ADHD posts and 2048 OCD posts, while misclassifying 358 ADHD posts as OCD and 349 OCD posts as ADHD. This demonstrates the ensemble's effectiveness in maintaining class balance with symmetric error patterns.

2. Soft Voting Ensemble Results

```
Ensemble Accuracy (Soft Voting): 85.13%

Soft Voting Classification Report:

              precision    recall  f1-score   support

        ADHD       0.86      0.84      0.85      2399
         OCD       0.85      0.86      0.85      2397

    accuracy                           0.85      4796
   macro avg       0.85      0.85      0.85      4796
weighted avg       0.85      0.85      0.85      4796
```

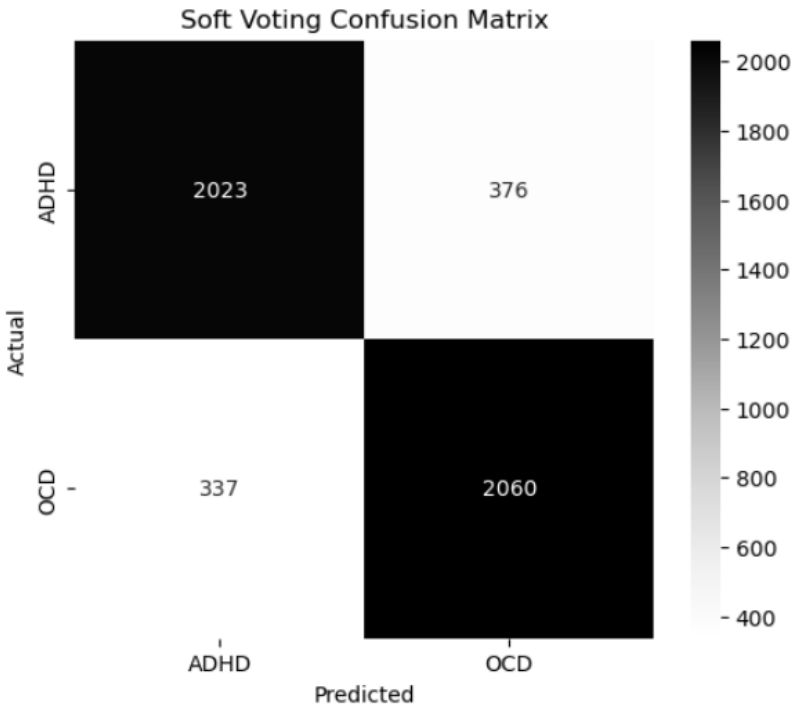


Figure 33: Soft Voting Ensemble Performance Metric

The Soft Voting ensemble which weights predictions based on model confidence through probability averaging achieved a slightly higher accuracy of 85.13%. This method showed improved classification with 2023 correct ADHD predictions and 2060 correct OCD predictions, reducing ADHD misclassifications to 375 and OCD misclassifications to 337. The probability-based aggregation allowed the ensemble to give more weight to confident predictions, with precision scores of 86% for ADHD and 85% for OCD and recall scores of 84% for ADHD and 86% for OCD.

3. Stacking Ensemble Results

```
Stacking Ensemble Accuracy: 85.40%

Classification Report:

              precision    recall  f1-score   support

        ADHD       0.85      0.86      0.85      2399
         OCD       0.86      0.85      0.85      2397

    accuracy                           0.85      4796
   macro avg       0.85      0.85      0.85      4796
weighted avg       0.85      0.85      0.85      4796
```

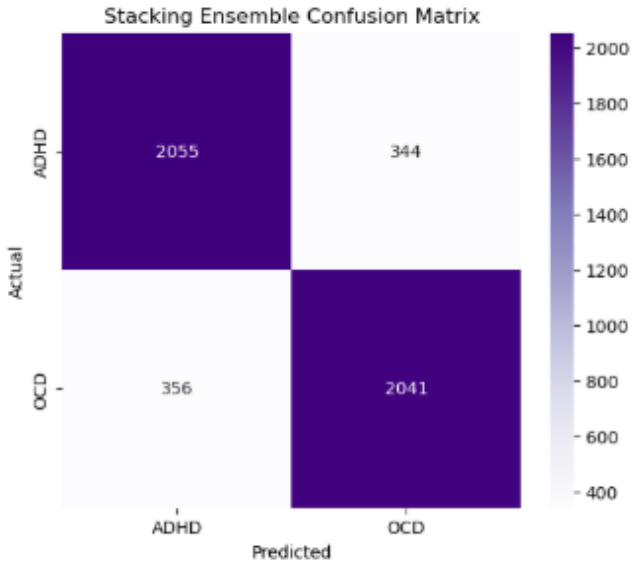
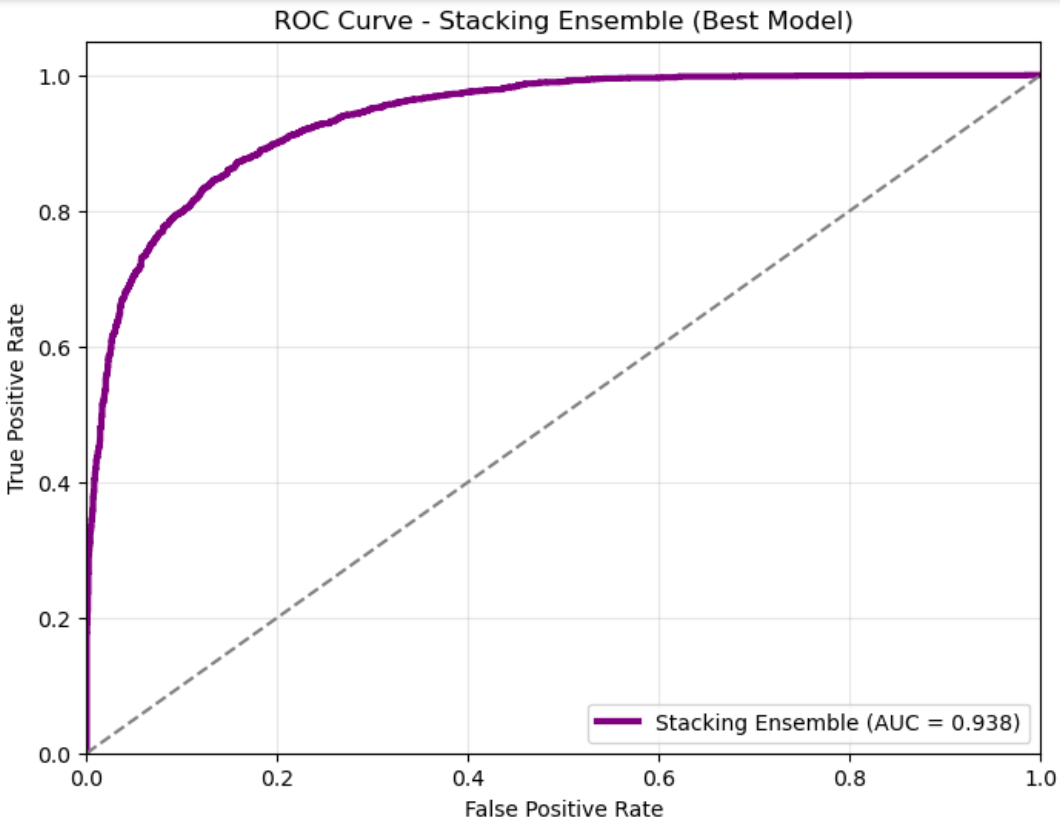


Figure 34: Stacking Ensemble Performance Metric

ROC Curve



The Stacking ensemble which employed a Logistic Regression meta-learner trained on base model predictions achieved the highest accuracy of 85.40%. This represents the most effective approach with 2055 correct ADHD classifications and 2041 correct OCD classifications, while reducing errors to 344 ADHD misclassifications and 356 OCD misclassifications. The meta-learner successfully learned to optimally combine the strengths of the three base models, achieving precision of 85% for ADHD and 86% for OCD with recall of 86% for ADHD and 85% for OCD.

3.5. Achieved results

After the implementation, the models and performing hyperparameter tuning and feature optimization the performance of the classifiers was evaluated using the standard metrics: Accuracy, Precision, Recall and F1-score. The results show the effectiveness of different approaches in classifying Reddit posts into ADHD or OCD categories.

3.5.1. Performance of Baseline Models (5,000 features)

Among the baseline models, Logistic performance slightly better than Naïve Bayes and SVM which indicates that a linear probabilistic approach effectively captures the relationship between TF-IDF features and class labels in the initial dataset configuration.

```
1. BASELINE MODELS (Original 5000 features):
   Naïve Bayes:     83.99%
   Logistic Reg:    84.57%
   SVM:             83.34%
```

Figure 35: Comparative analysis of Baseline Models

3.5.2. Performance of Optimized Models (10,000 features + Hyperparameter Tuning)

```
2. OPTIMIZED INDIVIDUAL MODELS (10000 features + GridSearch):
   Naïve Bayes:     84.67%
   Logistic Reg:    84.84%
   SVM:             84.82%
```

Figure 36:Comparative analysis of Optimized Models

After increasing the TF-IDF features to include trigrams and performing GridSearchCV for hyperparameter optimization, all the models show improved performance. Again, Logistic Regression maintained the highest accuracy among individual classifiers, demonstrating that additional feature granularity and parameter tuning improved model predictive power.

3.5.3. Comparison of baseline models and Optimized model accuracy

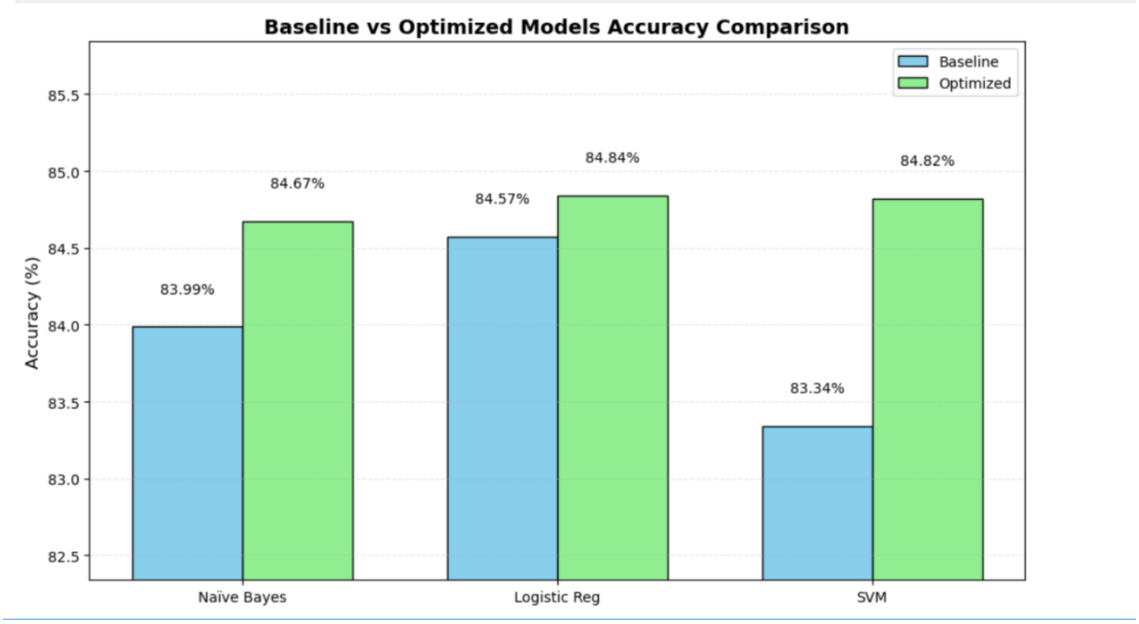


Figure 37: Comparison of baseline models and Optimized model accuracy

The bar chart above shows a direct comparison between the baseline and optimized versions of the three classification models. All models improved after hyperparameter tuning and increasing TF-IDF features from 5,000 to 10,000. Logistic Regression improved slightly from 84.57% to 84.84%, Naïve Bayes improved from 83.99% to 84.67%, and SVM improved the most from 83.34% to 84.82%. This demonstrates that optimization had a positive effect on model performance, with SVM showing the largest relative improvement.

3.5.4. Ensemble Models Performance

```
3. ENSEMBLE METHODS:
   Hard Voting:     85.05%
   Soft Voting:     85.13%
   Stacking:        85.40%
```

Figure 38: Comparative analysis of Ensemble Models

Ensemble methods further enhanced classification performance by leveraging the strengths of individual models. In particular, Stacking ensemble method achieved the highest accuracy of 85.40%, showing that combining model predictions through a meta-learner improves overall robustness and reduces individual model weaknesses.

3.5.5. Detailed Metrics Comparison

The table below shows a detailed comparison of Accuracy, Precision, Recall and F1-score for all models, including baseline, optimized and ensemble approaches.

```
======================================================================
FINAL MODEL PERFORMANCE SUMMARY
======================================================================

                   Model  Accuracy (%)  TF-IDF Features Parameters
  Naïve Bayes (Baseline)         83.99             5000    Default
 Logistic Reg (Baseline)         84.57             5000    Default
          SVM (Baseline)         83.34             5000    Default
 Naïve Bayes (Optimized)         84.67            10000 GridSearch
Logistic Reg (Optimized)         84.84            10000 GridSearch
         SVM (Optimized)         84.82            10000 GridSearch
    Hard Voting Ensemble         85.05            10000   Ensemble
    Soft Voting Ensemble         85.13            10000   Ensemble
       Stacking Ensemble         85.40            10000   Ensemble

----------------------------------------------------------------------
RANKING BY ACCURACY:
----------------------------------------------------------------------
1. Stacking Ensemble         85.40%
2. Soft Voting Ensemble      85.13%
3. Hard Voting Ensemble      85.05%
4. Logistic Reg (Optimized)  84.84%
5. SVM (Optimized)           84.82%
6. Naïve Bayes (Optimized)   84.67%
7. Logistic Reg (Baseline)   84.57%
8. Naïve Bayes (Baseline)    83.99%
9. SVM (Baseline)            83.34%
```

Figure 39: Detailed Metric Comparison of all the models

Observation:

After implementing and evaluating all models, several important insights were observed: the Stacking Ensemble achieved the highest accuracy of 85.40%, making it the best-performing model overall. Hyperparameter optimization had a positive effect on all individual models: Logistic Regression improved by +0.27%, Naïve Bayes by +0.69%, and SVM showed the largest gain at +1.48%, indicating that tuning parameters can meaningfully enhance model performance. Compared to the baseline Logistic Regression, the Stacking Ensemble provided an additional improvement of +0.83%, demonstrating the advantage of combining multiple models. Overall, SVM benefited the most from optimization, highlighting the importance of careful parameter selection for margin-based classifiers in text classification tasks. These results validate that ensemble methods, alongside proper feature extraction and optimization, lead to superior predictive performance for classifying ADHD and OCD posts.

3.5.6. Graphical Comparison of Metric

The below grouped bar chart visualizes the comparison of Accuracy, Precision, Recall and F1-score across all models.

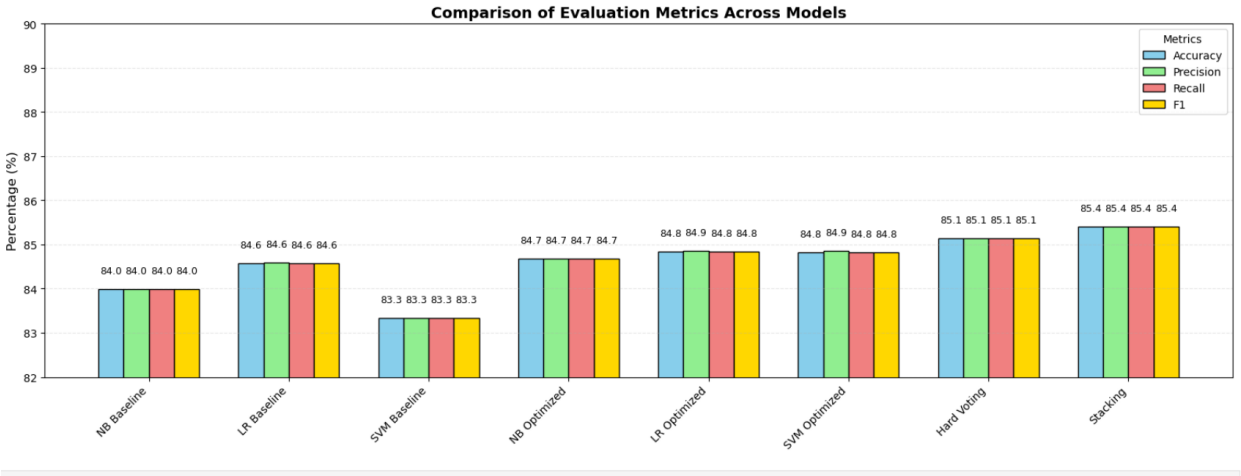
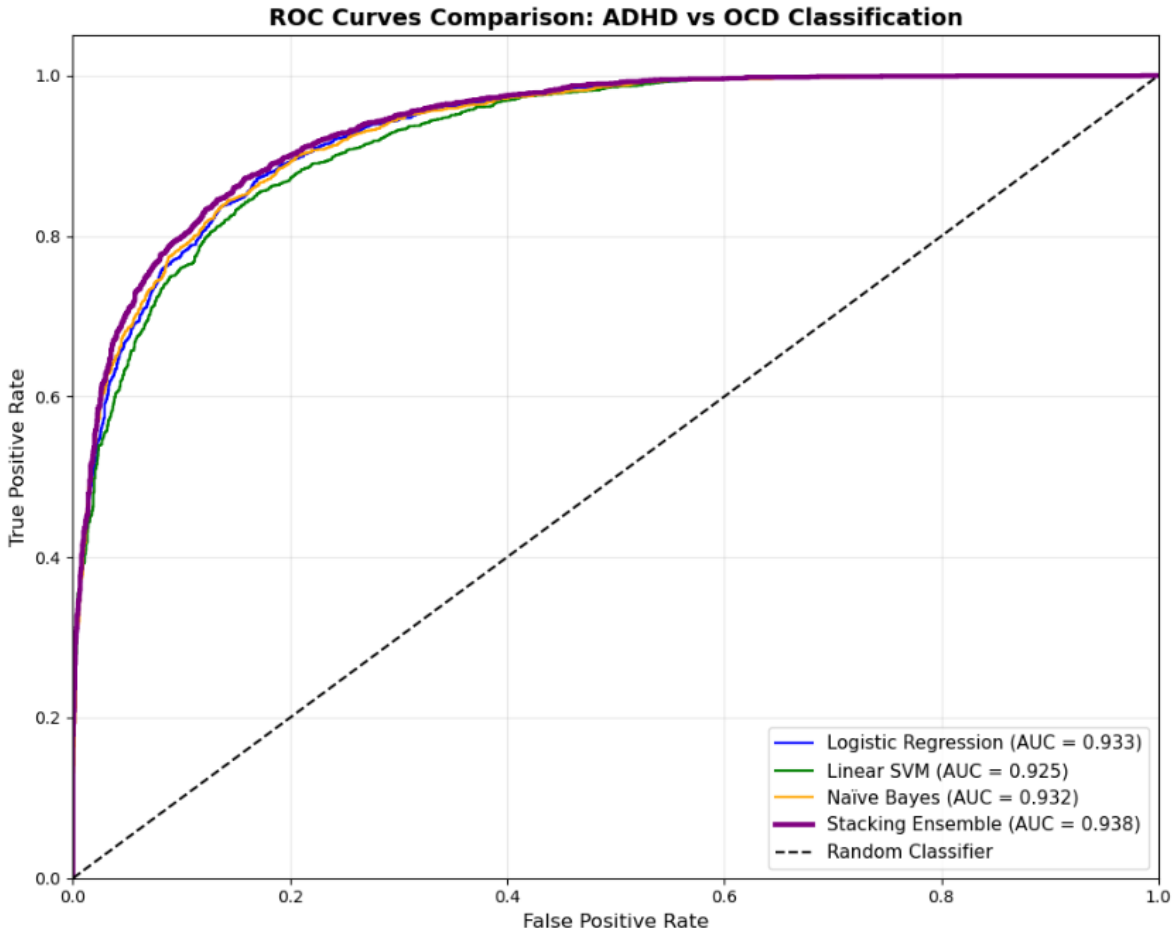


Figure 40: Graphical Comparison of Metric

Interpretation:

- Logistic Regression shows the strongest individual performance in all metrics.
- Stacking demonstrates a further improvement by combining model predictions.
- Naïve Bayes performs competitively, particularly in Recall, demonstrating its value as a simple yet effective baseline.
- The consistency of metrics across models indicates that the preprocessing, TF-IDF feature extraction and hyperparameter tuning pipeline was effective in capturing discriminative patterns in Reddit posts.

3.5.7. ROC Curves Comparison




```
ROC AUC Scores:
Logistic Regression: 0.933
Linear SVM: 0.925
Naïve Bayes: 0.932
Stacking Ensemble: 0.938
```

Figure 41: ROC Curves Comparison

Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) Scores were generated to access the discriminative ability of the classifiers across all the possible classification thresholds. The ROC curve plots True Positive Rate (sensitivity) against the False Positive Rate at various threshold settings. A model with strong discriminative power produces a curve that approaches the top-left corner, yielding a higher AUC value (maximum 1.0). An AUC of 0.5 represents random guessing, while values above 0.9 indicate excellent performance.

The ROC curves for the optimized individual models and the Stacking Ensemble are shown below. All models exhibit excellent discriminative capability (AUC > 0.92). The

Stacking Ensemble achieves the highest AUC of 0.938 which confirmed its superior ability to rank OCD posts higher than ADHD posts across varying probability thresholds. This result aligns with the accuracy findings and further validates the selection of the Stacking Ensemble as the final model.

3.5.8. Sample Prediction

To demonstrate the practical application and real-world performance of the final best model, six sample predictions were generated using the trained Stacking Ensemble classifier.

Sample 1:

```
Post: I have to wash my hands every time I touch the door handle, even if they're clean.
Prediction: OCD
Confidence: 99.38%
```

Figure 42: Sample 1 Prediction

Sample 2:

```
Post: I can't leave the house until I've double-checked that all appliances are off.
Prediction: OCD
Confidence: 73.47%
```

Figure 43:Sample 2 Prediction

Sample 3:

```
Post: I arrange my books in exact size order and feel anxious if they're even slightly misaligned.
Prediction: OCD
Confidence: 75.92%
```

Figure 44: Sample 3 Prediction

Sample 4:

```
Post: I started three different assignments today and forgot to finish any of them.
Prediction: ADHD
Confidence: 98.80%
```

Figure 45: Sample 4 Prediction

Sample 5:

```
Post: I can’t sit still in class and keep getting distracted by every little noise.
Prediction: ADHD
Confidence: 90.99%
```

Figure 46:Sample 5 Prediction

Sample 6:

```
Post: I keep losing my keys and phone even though I just had them a minute ago.
Prediction: ADHD
Confidence: 69.13%
```

Figure 47:Sample 6 Prediction

These sample predictions illustrate the ability of the model to distinguish between OCD related compulsive, ritualistic behaviour and ADHD related executive function challenges with varying confidence levels that reflect the clarity of the symptom expression in each statement. The high-confidence predictions (99.38%, 98.80%) correspond to classic, unambiguous symptom descriptions, while moderate confidence scores (69.13%-75.92%) likely reflect symptoms that could potentially overlap between conditions or be expressed with less specificity.

4. Conclusion

4.1. Analysis of the work done

This project successfully designed, implemented and evaluated a classical AI based application for classifying the Reddit posts into either ADHD(0) or OCD(1) categories. The solution follows a complete machine learning pipeline, starting with the extensive data preprocessing, advanced feature extraction using TF-IDF implementation of three supervised learning algorithms, i.e. Naïve Bayes, Logistic Regression and SVM, hyperparameter optimization and using ensemble learning techniques to improve the overall classification performance.

Among the baseline classifiers, Logistic Regression achieved the highest individual accuracy while SVM and Naïve Bayes also provide competitive results each offering unique advantages such as robustness to sparse data or probabilistic interpretability. The Hyperparameter tuning and optimized TF-IDF feature representation further improved the performance of all the models with SVM showing the largest relative gain through it. Ensemble methods in particular, Stacking achieved the highest overall accuracy of 85.40% which demonstrated that combining complementary models enhances the predictive capability, reduces individual weaknesses and ensures a more reliable classification of mental health text.

The development process emphasized interpretability, reproducibility and careful handling of sensitive textual data. Through TF-IDF vectorization strategy, incorporating unigrams, bigrams and trigrams, negation preservation and emoji removal techniques, distinguishing between OCD and ADHD health posts became more predictive and consistent.

4.2. How the application/solution addresses real world problems

This work directly addresses a real-world need for scalable tools to analyse mental health discourse by automatically distinguishing between symptom patterns associated with ADHD and OCD in online communities. The application serves as a potential support mechanism that could help identify emerging trends in public health, assist clinicians in preliminary screening and help online moderators identify posts that may require attention or human intervention.

Crucially, the system is designed as an aid to human judgment, not a replacement for it. The probabilistic outputs and interpretable features are intended to inform and support professional diagnosis, not to automate it. By leveraging AI and NLP, the approach enables scalable analysis, early detection of concerning content and informed decision making while also maintaining ethical use of anonymized data.

4.3. Future Work

Despite its success, the project has inherent limitations and its performance is tied to the specific language and demographics of Reddit users which may limit generalizability to other platforms or clinical settings. Furthermore, while TF-IDF effectively captures lexical importance, it cannot fully model semantic relationships between words, meaning subtle nuances and implicit meanings may be missed. Mental health also involves complex clinical evaluation, personal history and professional expertise that cannot be fully captured through text-based classification alone.

The logical next steps include integrating deep learning models like BERT for better contextual understanding and semantic representation. Expanding the system to include additional mental health conditions, multilingual support and more diverse dataset would further enhance its robustness. Most importantly, future development should prioritize developing robust ethical frameworks and clinician-facing interfaces to ensure responsible and effective real-world deployment.

In conclusion, this project demonstrates that classical AI and machine learning methods, when carefully designed, optimized and combined with thoughtful preprocessing and feature engineering, can provide accurate, interpretable and practical solutions for mental

health text classification. The findings highlight the effectiveness of combining probabilistic, linear and geometric approaches in a single framework, validating ensemble learning as a strategy for improving predictive performance and establish a solid foundation for future research in automated mental health analysis.

5. References

Anca Dinu, Andreea-Codrina Moldovan, 2021. *Automatic Detection and Classification of Mental Illnesses from General Social Media Texts.* s.l., INCOMA Ltd..

Andrew McCallum and Kamal Nigam, 1998. *A Comparison of Event Models for Naive Bayes Text Classification.* s.l., AAAI (Association for the Advancement of Artificial Intelligence).

Anon., 2017. Characterisation of mental health conditions in social media using Informed Deep Learning. *Scientific Reports,* 7(1), pp. 9-10.

Arman Cohan, Bart Desmet, Andrew Yates, Luca Soldaini, Sean MacAvaney, Nazli Goharian, 2018. *SMHD: a Large-Scale Resource for Exploring Online Language Usage for Multiple Mental Health Conditions.* New Mexico, Association for Computational Linguistics.

Ashwini Ashokkumar, James W Pennebaker , 2021. Social media conversations reveal large psychological shifts caused by COVID-19's onset across U.S. cities. *Science Advances,* 7(39), p. 10.1126/sciadv.abg7843.

Bishop, C. M., 2006. *Pattern Recognition and Machine Learning.* New York: Springer Science Business Media.

Boettcher, N., 2021. Studies of Depression and Anxiety Using Reddit as a Data Source: Scoping Review. *JMIR Mental Health,* 8(11), p. 10.2196/29487.

Corinna Cortes & Vladimir Vapnik, 1995. Support-vector networks. *Machine Learning,* Volume 20, pp. 273-297.

Dan Jurafsky and James H. Martin, 2025. *Speech and Language Processing.* 3rd ed. Upper Saddle River: Prentice Hall.

Daniel M Low ,Laurie Rumker,Tanya Talkar, John Torous,Guillermo Cecchi,Satrajit S Ghosh, 2020. Natural Language Processing Reveals Vulnerable Mental Health Support Groups and Heightened Health Anxiety on Reddit During COVID-19: Observational Study. *Journal of Medical Internet Research (JMIR),* 22(10), p. e22635.

Dietterich, T. G., 2000. *Ensemble Methods in Machine Learning. In: Multiple Classifier Systems.* Oregon, Lecture Notes in Computer Science.

Géron, A., 2019. *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.* 2nd ed. Sebastopol: O'Reilly Media.

Glen Coppersmith, Mark Dredze, Craig Harman, Kristy Hollingshead, 2015. *From ADHD to SAD: Analyzing the Language of Mental Health on Twitter through Self-Reported Diagnoses.* Denver, Association for Computational Linguistics, pp. 1-10.

Hosmer, D.W., Lemeshow, S. and Sturdivant, R.X. , 2013. *Applied Logistic Regression.* 3rd ed. Canada: John Wiley & Sons, Inc..

infoDiagram, 2023. *AI Algorithms, Neural Networks Diagrams, Machine Learning Presentation (PPT Template).* [Online] Available at: https://www.infodiagram.com/slides/logistic-regression-supervised-learning-algorithm/
[Accessed 16 December 2025].

Ivan Sekulic, Matej Gjurković, Jan Šnajder, 2018. *Not Just Depressed: Bipolar Disorder Prediction on Reddit.* Brussels, Association for Computational Linguistics (ACL), pp. 72-78.

Izzah Maisarah Binti Ibrahim; Ruhaila Maskat; Azmi Bin Aminordin; Noor Hasimah Ibrahim Teo, 2024. *Classification of Mental Health Conditions in Reddit Post using Multinomial Naïve Bayes Algorithm.* Selangor, IEEE.

Joachims, T., 1998. *Text Categorization with Support Vector.* s.l., Springer.

jsfactory, 2021. *Hugging Face.* [Online] Available at: https://huggingface.co/datasets/jsfactory/mental_health_reddit_posts/viewer
[Accessed 7 December 2025].

Marina Sokolova, Guy Lapalme, 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management,* 45(4), pp. 427-437.

Murphy, K. P., 2012. *Machine Learning A Probabilistic Perspective.* 1st ed. Cambridge: The MIT Press.

Nader Salari , Hooman Ghasemi , Nasrin Abdoli , Adibeh Rahmani , Mohammad Hossain Shiri , Amir Hossein Hashemian , Hakimeh Akbari , Masoud Mohammadi , 2023. The global prevalence of ADHD in children and adolescents: a systematic review and meta-analysis. *StatPearls,* 49(1), pp. 10.1186/s13052-023-01456-1.

Neff, D. M. A., 2022. *Neuro Divergent Insights.* [Online] Available at: https://neurodivergentinsights.com/adhd-vs-ocd/ [Accessed 2025 December 11].

Parsons CE, Purves KL, Davies MR, Mundy J, Bristow S, Eley TC, Breen G, Hirsch CR, Young KS., 2023. Seeking help for mental health during the COVID-19 pandemic: A longitudinal analysis of adults' experiences with digital technologies and services. *PLOS Digit Health,* 2(12), p. e0000402.

Rish, I., 2001. *An Empirical Study of the Naïve Bayes Classifier.* s.l., In IJCAI-01 Workshop on Empirical Methods in Artificial Intelligence.

Robert Thorstad, Phillip Wolff, 2019. Predicting future mental illness from social media: A big-data approach. *Behavior Research Methods ,* 51(4), pp. 1586-1600.

Sonja Cabarkapa,Joel A King,Nathan Dowling,Chee H Ng, 2019. Co-Morbid Obsessive–Compulsive Disorder and Attention Deficit Hyperactivity Disorder: Neurobiological Commonalities and Treatment Implications. *Frontiers in Psychiatry,* Volume 10, p. 557.

Steven Bird, Ewan Klein and Edward Loper, 2009. *Natural Language Processing with Python.* 1st ed. Sebastopol: O'Reilly Media.

Stuart Russell and Peter Norvig, 2022. *Artificial Intelligence.* 4th ed. Harlow: Pearson Education .

Sundas Khan,Samra Urooj Khan,OSAMA SHAFIQUE,AMEER HAMZA, 2025. COMPARATIVE ANALYSIS OF STOCK MARKET PRICE BEHAVIOR THROUGH MACHINE LEARNING APPROACHES. *Quantum Journal of Social Sciences and Humanities,* 6(2), pp. 29-41.

Tech & Tales, 2023. *Naïve Bayes: A Simple Yet Powerful Machine Learning Algorithm..* [Online]
Available at: https://techntales.medium.com/na%C3%AFve-bayes-a-simple-yet-powerful-machine-learning-algorithm-7b0dca78b483
[Accessed 13 December 2025].

Xueqin Lei,Hong Wu,Zhaohua Deng,Qing Ye, 2022. Self-disclosure, social support and postpartum depressive mood in online social networks: a social penetration theory perspective. *Information Technology and People,* 36(6), pp. 10.1108/ITP-12-2020-0825.

Yang Lin, Xinyu Ma, Xu Chu, Yujie Jin, Zhibang Yang, Yasha Wang, Hong Mei, 2024. *LoRA Dropout as a Sparsity Regularizer for Overfitting Control.* [Online] Available at: https://arxiv.org/abs/2404.09610
[Accessed 12 December 2025].

Yangyan Fan, Ahui Fan, Zhiping Yang and Daiming Fan, 2025. Global burden of mental disorders in 204 countries and territories, 1990-2021: results from the global burden of disease study 2021. *BMC Psychiatry,* 25(1), p. 486.

Zhanyi Ding, Zhongyan Wang, Yeyubei Zhang, Yuchen Cao, Yunchong Liu, Xiaorui Shen, Yexin Tian & Jianglai Dai, 2025. Trade-offs between machine learning and deep learning for mental illness detection on social media. *Scientific Reports,* Volume 15, p. 14497.