ABSTRACT

Stigma, as described by the Cambridge Dictionary, refers to a strong feeling of disapproval held by the majority of individuals in a society towards something, particularly when it seems unfair. Within the context of mental health, individuals frequently face the full weight of societal stigma, which worsens the difficulties they already confront. This research explores the complex geography of mental health stigma by analysing Twitter data. The study aims to uncover the frequency and geographic variation of stigma-related information in tweets by employing a semi-supervised learning approach and machine learning techniques. The study evaluates the performance of various machine learning techniques, such as Support Vector Machines (SVM), Logistic Regression, Naive Bayes, and Random Forest, by manually labelling a subset of tweets and considering the metrics of accuracy, precision, recall, and F1 score. This evaluation is done with an understanding of stigma. The results indicate that Support Vector Machines (SVM) exhibit higher accuracy compared to other algorithms. In addition, a dashboard is created to visually represent mental health stigma distribution. This includes a choropleth map that displays the stigma for each country, a bar chart that illustrates the count of stigmas by country, and word clouds that highlight prevalent themes in negative and positive sentiment tweets. The dashboard improves comprehension of patterns of mental health stigma and enables stakeholders in mental health advocacy and policy-making to make well-informed decisions.

ABSTRAK

Stigma, seperti yang diterangkan oleh Kamus Cambridge, merujuk kepada perasaan tidak setuju yang kuat yang dipegang oleh majoriti individu dalam masyarakat terhadap sesuatu, terutamanya apabila ia kelihatan tidak adil. Dalam konteks kesihatan mental, individu sering menghadapi beban penuh stigma masyarakat, yang memburukkan lagi kesukaran yang telah mereka hadapi. Penyelidikan ini meneroka geografi kompleks stigma kesihatan mental dengan menganalisis data Twitter. Kajian ini bertujuan untuk mendedahkan kekerapan dan variasi geografi maklumat berkaitan stigma dalam tweet dengan menggunakan pendekatan pembelajaran separa penyeliaan dan teknik pembelajaran mesin. Kajian ini menilai prestasi pelbagai teknik pembelajaran mesin, seperti Mesin Vektor Sokongan (SVM), Regresi Logistik, Naive Bayes, dan Random Forest, dengan melabelkan subset tweet secara manual dan mempertimbangkan metrik ketepatan, ketepatan, ingatan semula dan F1. skor. Penilaian ini dilakukan dengan memahami stigma. Keputusan menunjukkan bahawa Mesin Vektor Sokongan (SVM) mempamerkan ketepatan yang lebih tinggi berbanding dengan algoritma lain. Di samping itu, papan pemuka dicipta untuk mewakili pengedaran stigma kesihatan mental secara visual. Ini termasuk peta choropleth yang memaparkan stigma bagi setiap negara, carta bar yang menggambarkan kiraan stigma mengikut negara dan awan perkataan yang menyerlahkan tema lazim dalam tweet sentimen negatif dan positif. Papan pemuka meningkatkan pemahaman corak stigma kesihatan mental dan membolehkan pihak berkepentingan dalam advokasi kesihatan mental dan penggubalan dasar membuat keputusan yang termaklum.

ACKNOWLEDGEMENT

All praise to Allah SWT for providing me with strength and blessings throughout my academic journey, allowing me to finally complete my final year project. I am grateful to Prof. Madya Ts. Dr. Fatimah Binti Sidi, my final-year project supervisor, for her encouragement, guidance, and advice. Without the help of my supervisor, I would be unable to complete my final year project successfully. Thank you for willing to spend your precious time for the meeting and follow up the progress of the project. Her sincerity and dedication have motivated me to keep contributing to society.

Finally, I want to thank my family members for their unconditional love and support throughout my entire life. Thank you for giving me the chance to study in university for further education.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Definition
NN	Neural Network
CRISP-DM	Cross Industry Standard Process for Data Mining
EDA	Exploratory Data Analysis
LR	Logistic Regression
SVM	Support Vector Machine
RF	Random Forest
SMOTE	Synthetic Minority Oversampling TEchnique
TF-IDF	Term Frequency-Inverse Document Frequency

CHAPTER 1

INTRODUCTION

1.1 Background

According to the American Psychiatric Association, stigma happens when people are negatively viewed for having a specific characteristic — whether that's mental, physical, or otherwise. "Mental health stigma" or "mental illness stigma" refers to the stigma attached to mental health conditions and the discrimination that can happen to people who are living with them, thereby reducing the likelihood of seeking help. The American Psychiatric Association identifies three dimensions of stigma, public stigma involves the negative or discriminatory attitudes that others have about mental illness. Self-stigma refers to the negative attitudes, including internalised shame, that people with mental illness have about their own condition. Institutional stigma is more systemic, involving policies of government and private organisations that intentionally or unintentionally limit opportunities for people with mental illness. Examples include lower funding for mental illness research or fewer mental health services relative to other health care. These insights, sourced from the American Psychiatric Association, underscore the need for a comprehensive understanding of mental health stigma, its prevalence, and regional patterns to effectively address and counteract its impact.

Twitter is a popular social media platform where users express their thoughts, opinions, and experiences in an open manner. It provides a large dataset of user-generated content, including mental health-related discussions and conversations. Analysing data from Twitter provides a unique opportunity to explore levels of mental health stigma in different geographic areas to identify areas that require more education and outreach efforts. By using data visualisation techniques, it uncover regional variations in mental health stigma. To enhance understanding, a dashboard is constructed to display maps highlighting countries with higher stigma levels. Predictive analysis is then conducted utilising machine learning algorithms including Naive Bayes, Support Vector Machines (SVM), Random Forest, and Logistic Regression. Prior to developing the prediction model, the dataset was thoroughly cleaned and processed to ensure its quality and relevance. The main objective of this

project is to identify areas where more education and outreach efforts may be required and highlight areas where more education or targeted interventions are needed to combat stigma effectively.

1.2 Problem Statement

The problem of mental health stigma continues to be a significant barrier for those seeking support and understanding. Understanding the scope and regional variations of mental health stigma is essential for developing targeted interventions, educational campaigns, and policies that effectively combat stigma. However, it can be difficult to collect timely and exhaustive data on mental health stigma because of the lack of comprehensive measurements and analysis of mental health stigma across diverse geographic regions using Twitter data.

In addition, there are difficulties in identifying and analysing stigma-related content in large volumes of tweets about mental health. Twitter is a dynamic platform where discussions about mental health can be both explicit and implicit, making it difficult to precisely capture and interpret stigma-related narratives. Lastly, the limited availability of data hinders the development of educational initiatives to address mental health stigma.

1.3 Objectives

The main objective of this project is to build automated stigma detection on twitter data, the sub-objectives are as follows:

I. To measure levels of mental health stigma across different geographic regions in Twitter data.

II. To develop efficient algorithms for identifying and analysing stigma-related content in tweets collected.

III. To analyse the effectiveness of initiatives addressing mental health stigma data gathered from Twitter.

1.4 Scope

The scope of this project revolves around a meticulous examination of mental health discourse, drawing insights from the MH_Campaigns1723.csv dataset obtained from Kaggle. This dataset contains a large number of tweets from various mental health campaigns, providing a diverse and comprehensive repository of user-generated content.

Geographical analysis is an important component of this study, which aims to delve into mental health discussions across a variety of global regions. This project covers the time period from 2017 to 2023, using historical data to identify long-term trends and patterns in mental health conversations. Machine learning algorithms such as Naive Bayes, Support Vector Machines (SVM), Random Forest and Logistic Regression are utilised to predict the stigma. Algorithms that yield higher accuracy are selected for implementation.

Flask is used to create a dynamic dashboard to present the findings in a visually clear approach. This dashboard uses the dataset to generate maps that illustrate the comprehensive view of stigma levels and patterns, allowing healthcare professionals, policymakers, and advocacy groups to tailor targeted interventions and education campaigns to combat mental health stigma.

1.4.1 Limitations

1. Focusing on analysing Twitter data in English language, which may not capture all conversations regarding mental health stigma.
2. The analysis only utilises Twitter data, which may not be representative of the entire population.
3. The analysis is based on the available Twitter data, which may have sample size, completeness, and replicability limitations.
4. The dashboard only produces results for countries that are included in the dataset.

1.5 Project Timeline

Gantt Chart Schedule for Final Year Project 1 is shown in Figure 1.0 and Gantt Chart Schedule for Final Year Project 2 is shown in Figure 1.2.

ID	Activity	Start Date	End Date	Days	2022															
					W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16
1	Final Year Project Briefing	20/03/2023	20/03/2023	1																
2	Project Title Submission	20/03/2023	07/06/2023	79																
3	Workshop 1: Project Planning	20/03/2023	20/03/2023	1																
4	Workshop 2: Literature Review	15/05/2023	15/05/2023	1																
5	Workshop 3: Methodology	15/05/2023	15/05/2023	1																
6	Project Proposal Presentation	16/06/2023	16/06/2023	1																
7	Proposal Correction	19/06/2023	30/06/2023	12																
8	Proposal Submission	09/07/2023	09/07/2023	1																

Figure 1.1: Gantt Chart Schedule for Final Year Project 1

ID	Activity	Start Date	End Date	Days	2023											2024				
					W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16
1	Project Planning	09/10/2023	22/10/2023	14																
2	Data Preparation	23/10/2023	05/11/2023	14																
3	Modeling	06/11/2023	03/12/2023	28																
4	Evaluation	27/11/2023	17/12/2023	21																
5	Deployment	18/12/2023	07/01/2024	21																
7	Project Presentation	-	-	1																
8	Project Report Submission	-	-	1																

Figure 1.2: Gantt Chart Schedule for Final Year Project 2

1.6 Thesis Structure

This thesis is comprised of six chapters:

Chapter 1 Introduction provides an overview of the research project, project objectives, motivation, approach used in this project, project contribution, scope of the project and thesis structure.

Chapter 2 Literature Review describes the literature relevant to the project, including previous research in analysis of mental health discussion using twitter data. Several machine learning models for stigma prediction are reviewed as well in this chapter.

Chapter 3 Methodology focuses on detailed description of the proposed methodology, workflow, and tools used for this project. Semi Supervised Learning is explained.

Chapter 4 System Analysis and Design explain the techniques and procedures that involve five important components: Data Pre-processing, Exploratory Data Analysis, Semi Supervised Learning for Stigma Prediction, Geolocation Integration Using OpenCage, and Dashboard Design for Stigma Visualisation.

Chapter 5 System Implementation and Discussion addresses the practical execution of the designs outlined in chapter 4 within the system. The system is specifically built to be in accordance with the project's objectives. An analysis is conducted on the hardware and software employed in the system's development. Moreover, this chapter encompasses an extensive analysis and analysis of the acquired results or findings.

Chapter 6 Conclusion aims to summarise the research findings, providing an overview of the results and suggesting ideas for further research. The sub-sections comprising this chapter include the conclusion, achieved goal, limitations, and future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter covers the study of the existing research that has utilised social media data, specifically Twitter, to analyse and visualise mental health stigma by conducting a thorough analysis of several journal papers, comparing and summarising the results in a table. This review analyses current and previous approaches, highlighting key points for debate and a thorough understanding of each study. Several studies have investigated the use of Twitter data to assess the stigmatisation of mental illness in various populations. Reviews on the related research are summarised at the end of this chapter.

2.2 Background Automated Stigma Detection

Automated stigma detection on social media plays a crucial role in comprehending and addressing mental health challenges. To achieve this, various machine learning algorithms are employed, including Naive Bayes, Support Vector Machines (SVM), Random Forest, and Logistic Regression. Additionally, these algorithms are adapted for semi-supervised learning, leveraging unlabeled data to enhance stigma prediction accuracy. During the training phase, machine learning models learn from both labelled and unlabeled examples. This enables the models to capture the underlying patterns and structures in the data more comprehensively. By utilising the information contained in the unlabeled data, the models can refine their predictions and improve their overall performance in detecting stigma-related content on Twitter. Python is utilised for model development, and a dashboard is implemented using Flask to facilitate data visualisation and analysis.

2.2.1 Naive Bayes

The Naive Bayesian Classification is both a method of supervised learning and a statistical classification technique. It is a probabilistic model that allows to determine probabilities to encapsulate uncertainty about the model in a principled manner. It assists in diagnosing and predicting issues. This facilitates the determination of precise probabilities for hypotheses and is also robust to disturbance in input data. Naive Bayes is a popular model because it is simple and allows all attributes to contribute equally to the final decision. The Bayes Theorem formula is the following (cited in Wibawa et al., 2019):

$$P(Q|X) = \frac{P(X|Q)P(Q)}{P(X)}$$

Where:

X Data with unknown class

Q The hypothesis X is a specific class

P(Q|X) Probability of the hypothesis Q in X

P(X|Q) Probability of the X in hypothesis Q

P(Q) Probability of the hypothesis Q

P(X) Probability of the X

2.2.2 Support Vector Machines (SVM)

Support Vector Machines (SVM) is a powerful supervised learning algorithm used for both classification and regression tasks. It operates by finding the optimal hyperplane that best separates data points of different classes in a high-dimensional space as shown in Figure 2.1. The primary objective is to maximise the margin, which is the distance between the hyperplane and the nearest data points of each class. SVM can handle linear and non-linear relationships through the use of different kernel functions. This theory was introduced by Vapnik and Corina (1995).

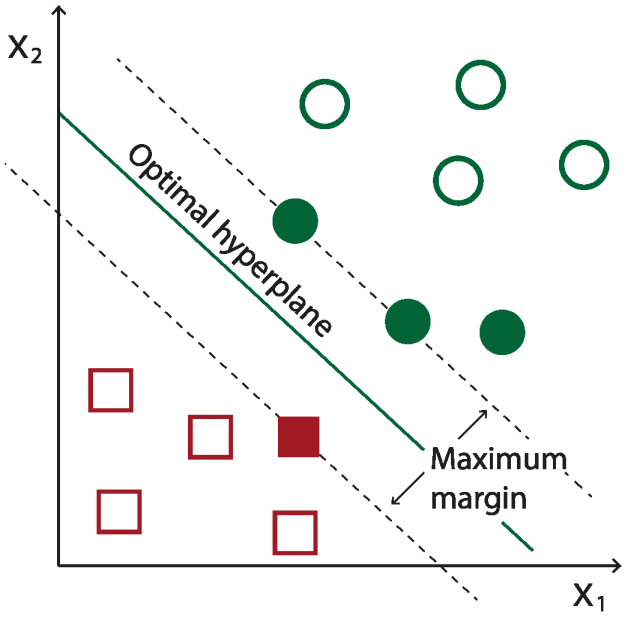


Figure 2.1: Support Vector Machine

2.2.3 Random Forest

Random Forest is a popular machine learning algorithm developed by Breiman (2001) that combines the outputs of multiple decision trees to produce a single result as shown in Figure 2.2. It functions as a group decision-making team in machine learning, aggregating the opinions of many individual models or "trees" to improve predictions. Random Forest uses the collective insights of these trees to create a more robust and accurate overall model, making it suitable for both classification and regression problems.

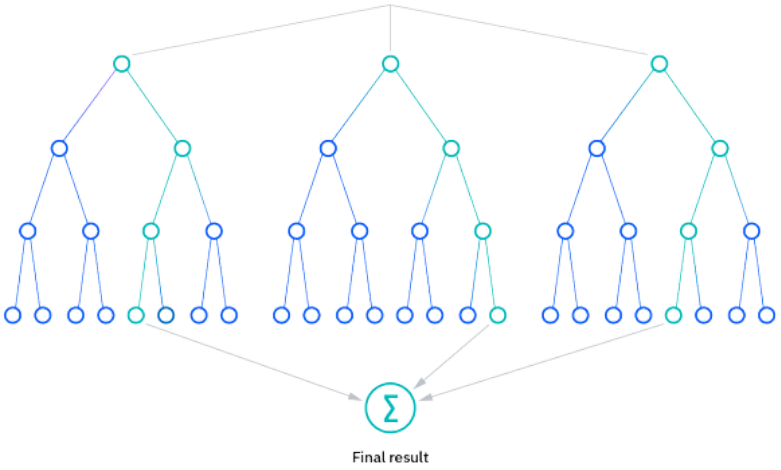


Figure 2.2: Random Forest

2.2.4 Logistic Regression

Logistic Regression is a classification and predicting algorithm. It employs the linear regression equation to produce discrete binary outputs, but, unlike linear regression, its cost function is the Sigmoid function. This function, which has an S-shaped trajectory, is also known as the logistic function. This algorithm's hypothesis tends to limit the logistic function between 0 and 1. The Logistic Regression formula is the following (cited in Couronné et al., 2018):

$$P(Y = 1|X1,\ldots, Xp) = \frac{exp(\beta 0+\beta 1X1+\cdots+\beta pXp)}{1+exp(\beta 0+\beta 1X1+\cdots+\beta pXp)}$$

Where:

Y	Binary response variable of interest
X1,...,Xp	Random variables considered as explanatory features
P(Y=1|X1,...,Xp)	Conditional probability of Y=1 given X1,...,Xp
β0,β1,…,βp	Regression coefficients
exp(β0+β1X1+···+βpXp)	Exponential term in the logistic function
P(Y=1)	Probability that Y=1 for a new instance
c	Threshold for class assignment (commonly c=0.5)
Bayes classifier	Y=1 if P(Y=1)>c, Y=0 otherwise

2.2.5 Dashboard

A dashboard, as defined by Tableau.com, is a centralised platform for presenting diverse visual data and combining various information in an easily understandable format. Typically, it conveys interconnected but distinct information, such as key performance indicators (KPIs) or critical business metrics, allowing stakeholders to grasp important insights quickly. Dashboards are highly adaptable and useful across a wide range of industries and sectors, accommodating a variety of data types and timeframes. Dashboards use visualisations such as tables, graphs, and charts to help stakeholders with varying levels of data familiarity quickly understand the narrative and derive actionable insights, such as understanding past events, their causes, potential future occurrences, and recommended actions.

2.3 Related Work

Mental illness, including bipolar disorder (BD), is highly stigmatised in society, but its prevalence on social media platforms remains unclear. Budenz et al. (2019) conducted a study to characterise Twitter-based stigma and social support messaging related to mental health/illness (MH)/MI and BD while also investigating the tweet features associated with retweets. Using machine learning techniques, specifically logistic regression, a large volume of tweets found that the majority discussed MH/MI (94.7%), while a smaller percentage specifically addressed BD (5.3%). This research is not creating any dashboard for visualisation.

Zaydman (2017) conducted a study titled "Tweeting About Mental Health: Big Data Text Analysis of Twitter for Public Policy," aiming to demonstrate the value of social media, particularly Twitter, in understanding and influencing attitudes towards mental health. Machine learning algorithms were utilised for automated coding, with support vector machines (SVM) being used as supervised classification models. Sentiment analysis algorithms were employed to analyse the emotional tone of tweets, while network analysis techniques were utilised to study the social network structure of Twitter communication. Additionally, topic modelling algorithms were applied to identify and analyse the prominent topics and themes related to mental health in the Twitter data. This study does not develop any data visualisation using dashboard.

Jilka et al. (2019) proposed a study that aims to develop a machine learning pipeline that could reliably identify stigmatising tweets related to schizophrenia and determine the prevalence of public stigma on Twitter. They collected 13,313 public tweets on schizophrenia and manually identified stigma in 746 English tweets. Two machine learning models, a linear Support Vector Machine (SVM) algorithm and a random forest model, were trained and tested. The SVM algorithm, which produced the fewest false negatives and had higher service user validation, was chosen as the best-performing model. The results demonstrated that machine learning, particularly the SVM algorithm, could effectively identify stigmatising tweets at a large scale. This study also does not build any dashboard.

2.4 Summary

There are three papers being reviewed. Each paper discusses its research focus, the data analysis method, and machine learning algorithm being used. Budenz et al. (2019) focus on mental health/illness (MH)/MI and bipolar disorder (BD). Zaydman (2017) focuses on the value of Twitter for understanding and affecting attitudes towards mental health, while Jilka et al. (2022) focus on identifying schizophrenia. Although each paper addresses different aspects of mental health, all revolve around the overarching theme of mental health stigma. Budenz et al. (2019) utilised Logistic Regression, while Zaydman (2017) and Jilka et al. (2022) both employed Support Vector Machines (SVM), with Jilka et al. (2022) also utilising Random Forest. In contrast, this project aims to tackle all types of mental health stigma using semi-supervised learning, which includes Naive Bayes, Support Vector Machines (SVM), Random Forest, and Logistic Regression algorithms. Notably, none of the related works has developed a dashboard for data visualisation. Hence, this project fills the gap by incorporating a dashboard to visualise the results comprehensively.

A table comparison of related works is as follows:

Table 2.1: Comparison of Related Work

	Budenz et al. (2019)	Zaydman (2017)	Jilka et al. (2022)	Proposed project
Research Focus	Twitter-based stigma and social support messaging about mental health/illness (MH)/MI and bipolar disorder (BD)	Demonstrate the value of Twitter for understanding and affecting attitudes towards mental health	Identifying schizophrenia stigma on Twitter	Mental health stigma on Twitter
Data Analysis Method	Manual content analysis, machine learning (logistic regression)	Machine learning techniques, sentiment analysis, network analysis	Machine learning, supervised classification	Semi-supervised Learning
Naive Bayes				✔
Support Vector Machines		✔	✔	✔
Random Forests			✔	✔
Logistic Regression	✔			✔
Dashboard				✔

CHAPTER 3

METHODOLOGY

3.1 Introduction

In this chapter, the methodology used is presented. This project is developed using CRISP-DM (Cross-Industry Standard Process for Data Mining), which is a widely recognised and iterative framework. CRISP-DM provides a structured method for guiding the various phases of the data mining procedure.

3.2 CRISP-DM Model Phases

The CRISP-DM life cycle model consists of six phases: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation and lastly is Deployment, with arrows indicating key relationships between them. Projects are able to switch between phases as needed. Six phases comprise the cyclical CRISP-DM method, as shown in Figure 3.1.

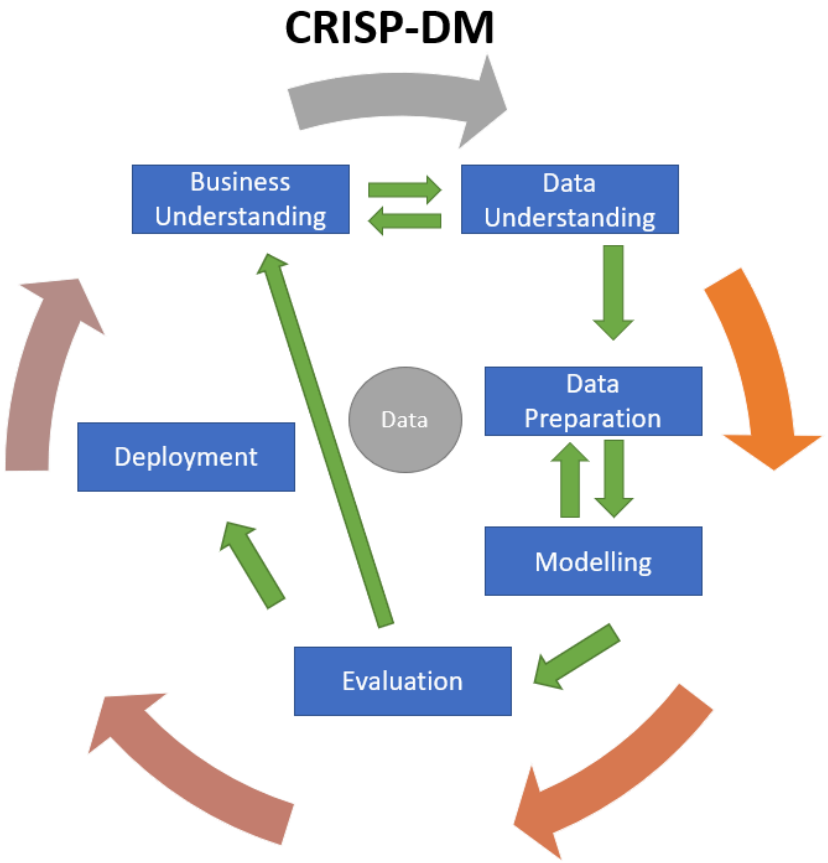


Figure 3.1: CRISP-DM Model

3.1.1 Business Understanding

In the Business Understanding phase, the problem statement, objectives, scope, target user, software used to develop and the machine learning algorithms of the project are defined. The primary objective of this project is to analyse mental health stigma on social media, particularly Twitter and understand regional variations through data visualisation. The project aims to support mental health organisations and government agencies in raising awareness and implementing targeted interventions to reduce stigma.

3.1.2 Data Understanding

The second phase in the CRISP-DM model is Data Understanding. The dataset used in this study, named MH_Campaigns_1723, was retrieved from Kaggle, a well-known platform for data science competitions, datasets, and projects. These datasets contain tweets from four mental health campaigns which are Eating Disorders Awareness Week, Mental Health Awareness Week, University Mental Health Day and OCD Awareness Week on Twitter, spanning the years 2017 to 2023. The dataset comprises various variables including Date, ID, URL, username, source, location, tweet content, likes, retweets, followers, replies, campaign names, likes per follower, replies per follower, retweets per follower, engagement, and engagement normalised. In total, the dataset consists of 724,756 rows and 17 columns, offering a comprehensive pool of data for analysis and exploration. An overview of the variables for the dataset is shown in Table 3.1 in next page:

Table 3.1: Overview of the Dataset

Variable	Data Type	Description
Date	String	Date of the tweet created
ID	Number	Unique identifier assigned to each tweet
url	String	Web address (URL) of the tweet
username	String	Twitter username associated with the tweet
source	String	Source or platform from which the tweet originated (e.g. Twitter for iPhone)
location	String	Location information provided by the user (e.g. city, country)
tweet	String	Text content of the tweet
likes	Number	Number of likes received by the tweet
rt	Number	Number of retweets received by the tweet
followers	Number	Number of followers of the Twitter user who posted the tweet
replies	Number	Number of replies received by the tweet
campaign	String	Mental health campaign associated with the tweet (e.g. Eating Disorders Awareness Week, Mental Health Awareness Week, University Mental Health Day, OCD Awareness Week)
likes_pf	Float	Ratio of likes to the number of followers
replies_pf	Float	Ratio of replies to the number of followers
rt_pf	Float	Ratio of retweets to the number of followers
engagement	Float	Overall engagement metric calculated for the tweet, including likes, retweets, and replies
engagement_0	Float	Another measure of engagement for the tweet

3.1.3 Data Preparation

The third phase of CRISP-DM is Data Preparation. This phase involved several key actions to refine the dataset and enhance its suitability for subsequent processing. Firstly, rows with unspecified locations “unknown” were eliminated to maintain data integrity. The Date attribute is also formatted to DD/MM/YYYY. Additionally, entries with missing values were systematically removed to avoid any potential distortions in the analysis. The dataset was then reduced to include only essential attributes such as Date, location, and tweet, discarding unnecessary information to improve focused analysis. A keyword-based filtering approach was used to distinguish between tweets containing stigma-related terms to target them to meet the study's scope. This process produced a subset of 4608 tweets that were ready for further analysis. Following that, extensive text cleaning procedures were used to standardise tweet formats and eliminate unnecessary elements like URLs, hashtags, and mentions, generating a cleaned dataset of 4347 distinct tweets after duplicate removal. In order to improve the dataset's utility for supervised learning, a random sample of 85 tweets were manually labelled. The labelled data was then seamlessly combined with the original dataset, resulting in a comprehensive dataset of 4432 instances. This carefully organised dataset serves as the foundation for future semi-supervised learning analysis with the purpose of predicting mental health stigma on social media platforms.

3.1.4 Modelling

Modelling is the fourth phase in the CRISP-DM model, where the focus shifts to constructing predictive models based on the prepared dataset. To address the challenge of label imbalance in the labelled data, where the '0' label comprises 62 instances, the '1' label has 17 instances, and the '2' label has 6 instance, the Synthetic Minority Over-sampling Technique (SMOTE) and Tomek links were applied. SMOTE generates synthetic samples from the minority class, effectively balancing the class distribution. Meanwhile, Tomek links identify and remove overlapping instances between classes, further refining the dataset. Leveraging the refined dataset, four machine learning algorithms which are Support Vector Machine (SVM), Logistic Regression, Naive Bayes, and Random Forest were implemented using Python for semi-supervised learning. With 4347 unlabeled instances and 85 labelled instances, these algorithms were trained on the labelled data and then applied to predict labels for the unlabeled data. SVM, a powerful classifier, utilises hyperplanes to separate data into different classes. Logistic Regression, on the other hand, models the probability of a binary outcome using a logistic function. Naive Bayes employs Bayes' theorem with strong independence assumptions between features to classify instances. Finally, Random Forest, a versatile ensemble learning method, combines the outputs of multiple decision trees to achieve robust classification performance.

3.1.5 Evaluation

The fifth phase is the Evaluation of the CRISP-DM model, which involves assessing the performance of the predictive models that have been built. Model performance is evaluated using a variety of metrics, including confusion matrices and classification reports, which reveal information about precision, recall, accuracy, and F1-score. These metrics provide a comprehensive understanding of the model's ability to correctly identify instances of stigma in social media data.

Confusion matrices provide a visual representation of true positive, false positive, true negative, and false negative predictions, allowing for a more in-depth examination of model performance. Classification reports provide in-depth information about the model's predictive capabilities. If the accuracy meets the project requirements and objectives, the best model with the highest accuracy can be used to create a system Accuracy is the ratio of correct predictions to total data points. Figure 3.2 is a confusion matrix which is useful in model evaluation performance.

		Predicted	
		Negative **(N)** -	Positive **(P)** +
Actual	Negative -	True Negative **(TN)**	**False Positive (FP) Type I Error**
	Positive +	**False Negative (FN) Type II Error**	True Positive **(TP)**

Figure 3.2: Confusion matrix

A true positive (TP) refers to the count of positive samples that have been accurately classified. For example, the number of tweets containing mental health stigma is correctly predicted as having stigma.

A true negative (TN) refers to the count of negative samples that have been accurately categorised. For instance, the accurate prediction of tweets without mental health stigma properly identifies them as without stigma.

A false positive (FP) refers to the count of samples that are falsely categorised as positive. For instance, the number of tweets not containing mental health stigma is incorrectly predicted as having stigma.

A false negative (FN) refers to the count of samples that have been inaccurately labelled as negative. For instance, the quantity of tweets that include mental health stigma is inaccurately predicted as without stigma.

3.1.6 Deployment

The final phase of the CRISP-DM is deployment, which involves implementing the developed predictive model for stigma prediction on unlabeled data. The model with the highest accuracy, based on SVM, is used for this project. In addition, a dashboard is developed to show regional variations in mental health stigma counts by country.

To accomplish this, Flask is being used to create a web application that allows for user-friendly interaction and visualisation of stigma prediction results as well as regional patterns in mental health stigma discourse. It provides a user-friendly interface for exploring the data and gaining insights into the patterns and trends. Through deployment, stakeholders have accessed to the developed predictive model and dashboard, allowing them to gain actionable insights and make informed decisions to effectively address mental health stigma.

3.2 Proposed Approach

This project aims to develop automated stigma detection on twitter data. The twitter data undergoes data cleaning and pre-processing. Then 85 random data is generated and manually labelled. SMOTETomek is applied to the combined data consist of labelled data and unlabeled data to handle imbalance follow by implementation of semi-supervised learning using four different machine learning algorithms: Support Vector Machine (SVM), Random Forest, Naive Bayes and Logistic Regression. The most optimum algorithm is chosen to predict the stigma on unlabeled data. Below is flowchart of proposed approach of this project depicted in Figure 3.3:

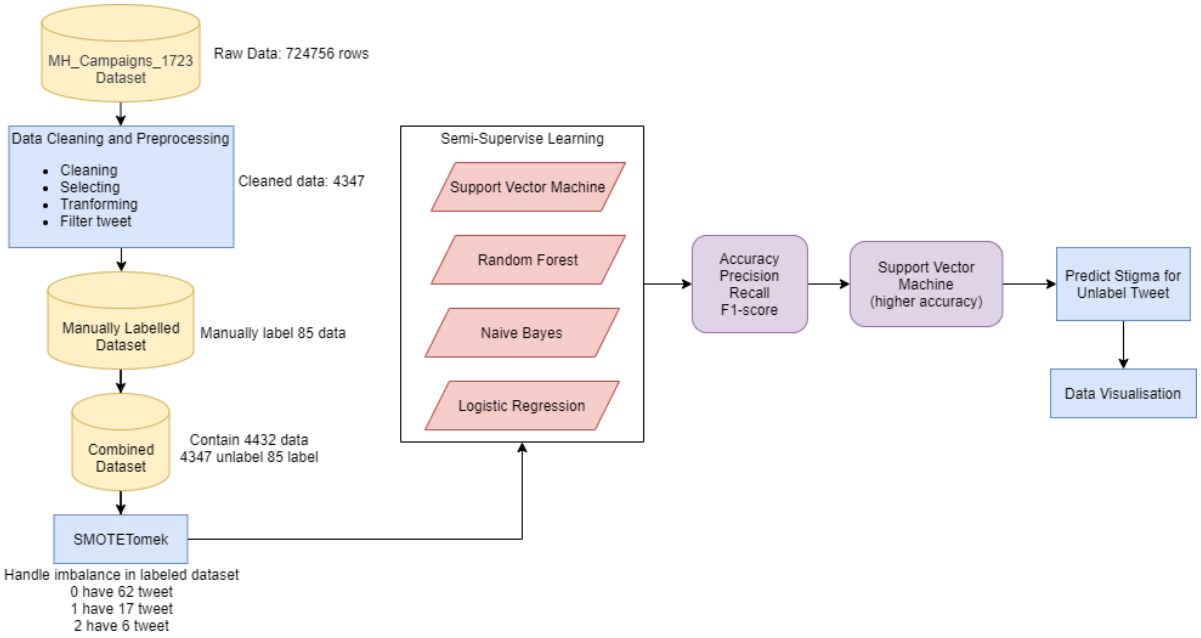


Figure 3.3: FlowChart of Proposed Approach

3.2.1 Semi-Supervised Learning

Semi-supervised learning holds a distinct position in the range of machine learning methods, combining elements from both supervised and unsupervised learning approaches. This strategy seeks to achieve a balance between the requirement for labelled data and the amount of unlabeled data in various real-world situations by utilising a small set of labelled data in conjunction with a large number of unlabeled instances. Semi-supervised learning aims to acquire knowledge from labelled instances while also utilising the inherent structure and patterns present in the unlabeled data. The consolidation of this information enables models to generalise more efficiently and generate more accurate predictions on unfamiliar inputs. Utilising unlabeled data allows algorithms to engage in self-supervision, leveraging the inherent data distribution to uncover significant representations. Furthermore, semi-supervised learning techniques provide practical benefits in scenarios where the process of labelling large quantities of data is difficult, time-consuming, or financially impractical. By utilising the extensive pool of unlabeled data that is frequently available in large quantities, these methods offer a cost-efficient approach to enhance the performance and scalability of models. As a result, semi-supervised learning has become a key technique in machine learning, providing a systematic strategy to extract knowledge from datasets that include both labelled and unlabeled data. Figure 3.4 below is example of flowchart for semi-surprised learning application:

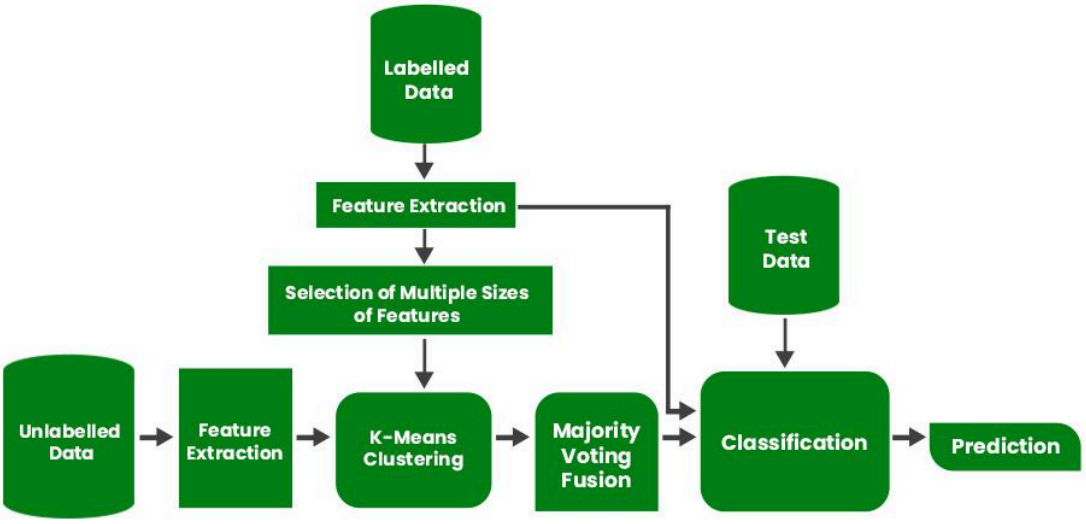


Figure 3.4: Semi-Supervised Learning FlowChart

Using only the very limited labelled data points available, semi-supervised learning can determine the problem's structure by establishing classes and clusters, and also offers further context that assists in comprehending the overall distribution of the data. An overview of semi-supervised learning is shown in Figure 3.5 below:

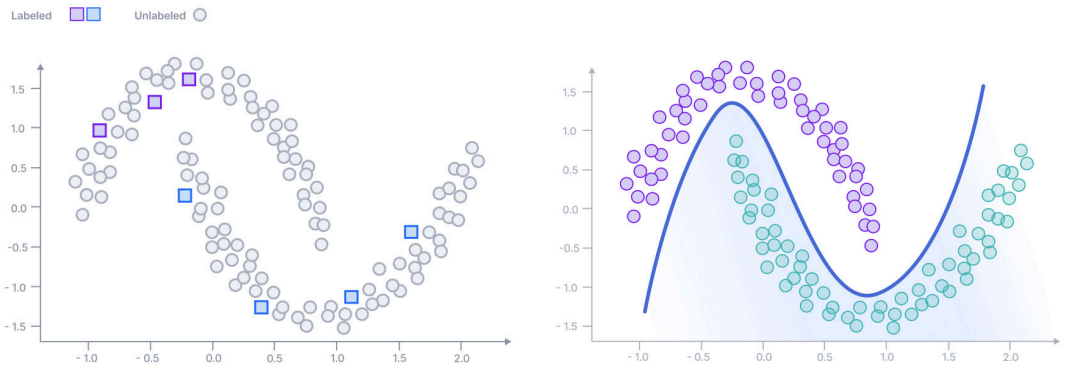


Figure 3.5: Semi-Supervised Learning Overview

3.3 Tools And Languages

Python is the primary language employed for the development and implementation of a dashboard. Python offers numerous libraries and tools that are well-suited for data analysis, machine learning, and visualisation. Specifically, libraries such as scikit-learn can be used to implement machine learning algorithms such as Naive Bayes, Support Vector Machines (SVM), Random Forest and Logistic Regression for algorithmic analysis. Additionally, Flask is utilised to construct an interactive interface, enabling user-friendly data visualisation and facilitating seamless navigation through the dashboard.

3.4 Expected Results

The results of this project are as follows:

First, the project aims to measure the levels of mental health stigma across different geographic regions using Twitter data. This has been achieved through data analysis techniques, resulting in a comprehensive assessment of stigma prevalence. The findings have been visualised through a regional variation map. This map provides a visual representation of stigma prevalence across different geographic areas. It helps healthcare professionals, policymakers, and organisations identify areas that may require targeted intervention and educational efforts.

Second, the project develops algorithms for efficiently identifying and analysing stigma-related tweet content. These algorithms detect and classify tweets containing stigma-related words. These algorithms contribute to a greater comprehension of mental health stigma by efficiently analysing large datasets. The results of the analysis integrate into an interactive Python dashboard. These dashboard serve as powerful tools for stakeholders to explore and gain insights from the visualised data.

Lastly, the project intends to analyse the efficacy of initiatives addressing mental health stigma based on Twitter data. This analysis provides insights on the effect of these initiatives. The findings incorporated into the interactive Python interfaces, allowing stakeholders to evaluate the efficacy of various initiatives and their impact on reducing stigma-related content.

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

4.1 Introduction

This chapter focuses on the system analysis and design phase of the project, specifically discussing the approaches and processes related to five key aspects: Data Pre-processing, Exploratory Data Analysis, Semi Supervised Learning for Stigma Prediction, Geolocation Integration Using OpenCage and Dashboard Design for Stigma Visualisation. The approach and strategies used for achieving the study's objectives are explained in full at every stage of these processes, offering valuable insight.

4.2 Data Pre-Processing

Data Pre-Processing is the first step before data analysis.

4.2.1 Data Cleaning

According to Ridzuan and Zainon (2019), Data cleaning is a process that is carried out on existing data to eliminate anomalies and generate a data set that is both accurate and distinct, representing the miniature universe. The process entails rectifying errors, resolving variations, and standardising the data into a consistent format. Due to the vast volume of data collected, manual data cleaning is nearly impracticable due to its time-consuming nature and susceptibility to errors. The data cleaning process is complicated and comprises multiple stages, such as defining quality norms, identifying data errors, and correcting them. The objective is to enhance the data's quality by discovering and eliminating errors and inconsistencies. Insufficient data results in uncertainty during the process of analysing the data, and it is crucial to address this issue during the data cleaning phase. Any errors or missing values in the dataset can lead to a distinct outcome and potentially impact the business decision-making process. Accurate data is essential to prevent financial losses, complications, and additional expenses resulting from data of inferior quality. Below is code used for data cleaning:

```
# Removes the rows with location unknown
df.drop(df[df['location'] == 'unknown'].index, inplace=True)

# Removes the rows that contains NULL values.
df=df.dropna()

def clean_text(tweet):
    # Remove mentions (@username)
    tweet = re.sub(r'@[\w_]+', '', tweet)

    # Remove hashtags
    tweet = re.sub(r'#\w+', '', tweet)

    # Remove URLs
    tweet = re.sub(r'http\S+|www\S+|https\S+', '', tweet,
flags=re.MULTILINE)

    # Remove emojis
    tweet = tweet.encode('ascii', 'ignore').decode('ascii')

    # Remove special characters and numbers
    tweet = re.sub(r'[^a-zA-Z\s]', '', tweet)

    # Remove extra whitespaces
    tweet = re.sub(' +', ' ', tweet)

    # Convert to lowercase
    tweet = tweet.lower()

    return tweet.strip()

# Clean the 'tweet' column
df['tweet'] = df['tweet'].apply(clean_text)
```

The df.drop() removes rows with 'unknown' location labels to ensure that only relevant data is used for analysis. The df.dropna() removes rows containing NULL values to ensure data integrity. The function clean_text() is used to clean the tweet. The function performs the following operations on the tweet:

1. The re.sub(r'@[\w_]+', '', tweet) method is used to remove mentions in the tweet. E.g. @username
2. The re.sub(r'#\w+', '', tweet) method is used to remove hashtags that exist in tweets collected. E.g. #mentalhealth
3. The re.sub(r'http\S+|www\S+|https\S+', '', tweet, flags=re.MULTILINE) method removes URLs in format http or www or https in the tweet. re.MULTILINE ensuring that URLs within multiline text are properly identified and removed.
4. The tweet.encode('ascii', 'ignore').decode('ascii') method is used to remove emoji in the tweet.
5. The re.sub(r'[^a-zA-Z\s]', '', tweet) method is to include tweets that contain text only. Number and special character is removed.
6. The re.sub(' +', ' ', tweet) method is to remove extra whitespaces.
7. The tweet.lower() method is used to make all tweets in lowerspace. Lowercasing tweets ensures consistency in text processing and analysis. It prevents issues where the same word might be treated differently due to differences in case.

4.2.2 Keyword Based Detection

Keyword based detection is being used to identify tweets containing stigma-related words or phrases. Stigma-related keywords are predefined terms that are indicative of mental health stigma or negative attitudes towards mental health. Then only tweets that contain those words are selected for further analysis.

```
# Stigma-related keywords
stigma_keywords = ['crazy', 'insane', 'psycho', 'lunatic',
'nutcase', 'schizo', 'bipolar', 'depressive', 'manic',
'neurotic',
                   'unstable', 'weak', 'broken', 'abnormal',
'dangerous', 'attention-seeking', 'faking it', 'imaginary',
                   'made up', 'handwashing', 'cleaning',
'neat', 'psychotic', 'just about', 'attention seeker']

# Function to check if the tweet contains stigma-related
keywords
def contains_stigma(tweet):

    # Check for the presence of stigma-related keywords in the
tweet
    return any(re.search(r'\b{}\b'.format(keyword), tweet) for
keyword in stigma_keywords)

# Filter the DataFrame to include only rows with
stigma-related keywords
df = df[df['tweet'].apply(contains_stigma)]
```

1. The stigma keyword is being defined. ['crazy', 'insane', 'psycho', 'lunatic', 'nutcase', 'schizo', 'bipolar', 'depressive', 'manic', 'neurotic', 'unstable', 'weak', 'broken', 'abnormal', 'dangerous', 'attention-seeking', 'faking it', 'imaginary', 'made up', 'handwashing', 'cleaning', 'neat', 'psychotic', 'just about', 'attention seeker']
2. The function contains_stigma(tweet) is used to filter the tweets using regular expression re.search(r'\b{}\b'.format(keyword), tweet). This function retains only the tweets that contain one or more of the selected keywords. This filtering process helps focus the analysis on relevant content that is associated with the concept being investigated.

4.2.3 Feature Selection

According to Li et al. (2017), feature selection is a data preprocessing technique that has been demonstrated to be successful and efficient in preparing data, particularly high-dimensional data, for a range of data-mining and machine-learning tasks. The goals of feature selection involve the creation of simplified and easily understandable models, enhancement of data-mining efficiency, and the preparation of unambiguous and comprehensible data. The increasingly widespread availability of large volumes of data has posed significant obstacles and opportunities for the process of selecting relevant features. In this project, TF-IDF is used. TF-IDF stands for Term Frequency-Inverse Document Frequency. It is a quantitative measure employed in natural language processing and information retrieval to assess the significance of a word within a document compared to a set of documents.

Term Frequency (TF) is a metric that quantifies the frequency of a term (word) in a document. The term frequency is determined by dividing the number of occurrences of a term in a document by the total number of terms in that document. Term frequency (TF) is directly proportional to the frequency of the term in the document.

The Inverse Document Frequency (IDF) quantifies the significance of a term within the entire corpus. The term frequency-inverse document frequency (TF-IDF) is computed by taking the logarithm of the ratio between the total number of documents and the number of documents that contain the specific word. Inverse Document Frequency (IDF) is directly proportional to the unavailability of a term within the corpus. Below is the code that included TF-IDF:

```
# Preprocessing and feature extraction
tfidf_vectorizer = TfidfVectorizer(max_features=1000)
X_tfidf =
tfidf_vectorizer.fit_transform(combined_data['tweet'])
```

The TfidfVectorizer is used to convert the text data (combined_data['tweet']) into a matrix of TF-IDF features (X_tfidf). The max_features parameter is configured to 1000, thus limiting the amount of features to the highest 1000 frequently occurring words, determined by their TF-IDF scores. This process efficiently identifies the most significant characteristics from the text data, hence functioning as a method of feature selection.

4.2.4 SMOTETomek Analysis

The SMOTETomek is used to solve the problem of unbalanced dataset in labelled data. SMOTETomek is a combination of two resampling techniques, SMOTE and Tomek links.

SMOTE, also known as Synthetic Minority Over-sampling Technique, is a method used to tackle the issue of class imbalance. It achieves this by creating artificial samples from the minority class in order to achieve a balanced distribution of classes. The process involves generating synthetic instances by interpolating between existing examples of the minority class.

Tomek links refer to pairs of cases from distinct classes that exhibit close proximity to one another. These occurrences are regarded as noise or borderline cases. Tomek links-based cleaning is a method that seeks to enhance the distinction between different classes by locating and eliminating instances that are noisy or near the boundary. These are the code snippet for doing SMOTETomek analysis:

```
# For imbalance label in labeled data
# Oversample the labeled data
smk = SMOTETomek(random_state=0)
X_labeled_resampled, y_labeled_resampled =
smk.fit_resample(X_labeled, labeled_data['stigma'])
```

SMOTETomek is used in the project because of the skewed distribution of the labelled dataset. The imbalance is caused by the uneven allocation of instances among various classes. For this particular scenario, the dataset is labelled and has three distinct classes(label), each with different frequencies:

Label 0: 62 occurrences

Label 1: 17 occurrences

Label 2: 6 occurrences

The distribution of classes in the dataset shows a notable imbalance, with the majority class (Label 0) being overrepresented in comparison to the minority classes (Label 1 and Label 2). Below is the Table 4.1 shown before and after undergo SMOTETomek:

Table 4.1: Before and After SMOTETomek

	Label 0 (no stigma)	Label 1 (stigma)	Label 2 (unrelated)
Before SMOTETomek	62	17	6
After SMOTETomek	62	62	62

4.2.5 Data Splitting

```
# Train-test split for oversampled labeled data
X_train, X_test, y_train, y_test =
train_test_split(X_labeled_resampled, y_labeled_resampled,
test_size=0.2, random_state=42)
```

The train_test_split function from Scikit-learn is employed to partition the oversampled labelled data into separate sets for training and testing purposes. The test_size option determines the fraction of the dataset that is allocated for the test split. In this case, it is set to 0.2, indicating that 20% of the data is utilised for testing. The random_state parameter guarantees the ability to duplicate the split by specifying the random seed.

4.3 Exploratory Data Analysis (EDA)

Komorowski et al. (2016) in their research explain that exploratory data analysis (EDA) is a crucial component of any research analysis. The main objective of exploratory analysis is to analyse the data for its distribution, outliers, and anomalies in order to guide targeted hypothesis testing. Additionally, it offers methods for generating hypotheses by visually representing and comprehending the data, typically through graphical means. EDA attempts to facilitate the analyst's recognition of natural patterns. Since Tukey's groundbreaking work in 1977, Exploratory Data Analysis (EDA) has become widely accepted as the definitive methodology for analysing a dataset.

A variety of exploratory data analysis (EDA) techniques were used for getting familiar with the dataset and gaining insights into its structure and contents. First, descriptive statistics were used to better understand the dataset's composition, followed by summary statistics. This involved using functions like info(), describe(), and head() to get an understanding of the dataset's structure, summary statistics, and initial rows. In addition, visualisation techniques were used. A word cloud was created to visualise the most frequently used words in the tweets, providing insight into common themes or topics discussed across campaigns. Furthermore, a time series analysis was performed by generating a bar chart to analyse the number of tweets per year, allowing for the observation of trends over time. Word clouds and bar charts were generated using the WordCloud library, seaborn and matplotlib in Python. These EDA techniques helped to gain a comprehensive understanding of the dataset, allowing for the identification of patterns, trends, and potential areas of interest for further analysis. The findings and graphs are discussed as below:

Figure 4.1: Most frequently words in the tweets

In Figure 4.1, a word cloud showcases the most frequent words found in the tweets within the dataset. This visualisation presents words in varying sizes, with larger words indicating higher occurrence and smaller words representing less frequent terms. By visually summarising the prevalent words, the word cloud offers insights into the key themes and topics discussed in the tweets. This enables a quick understanding of the dominant issues or sentiments expressed across the dataset, providing valuable context for further analysis and interpretation.

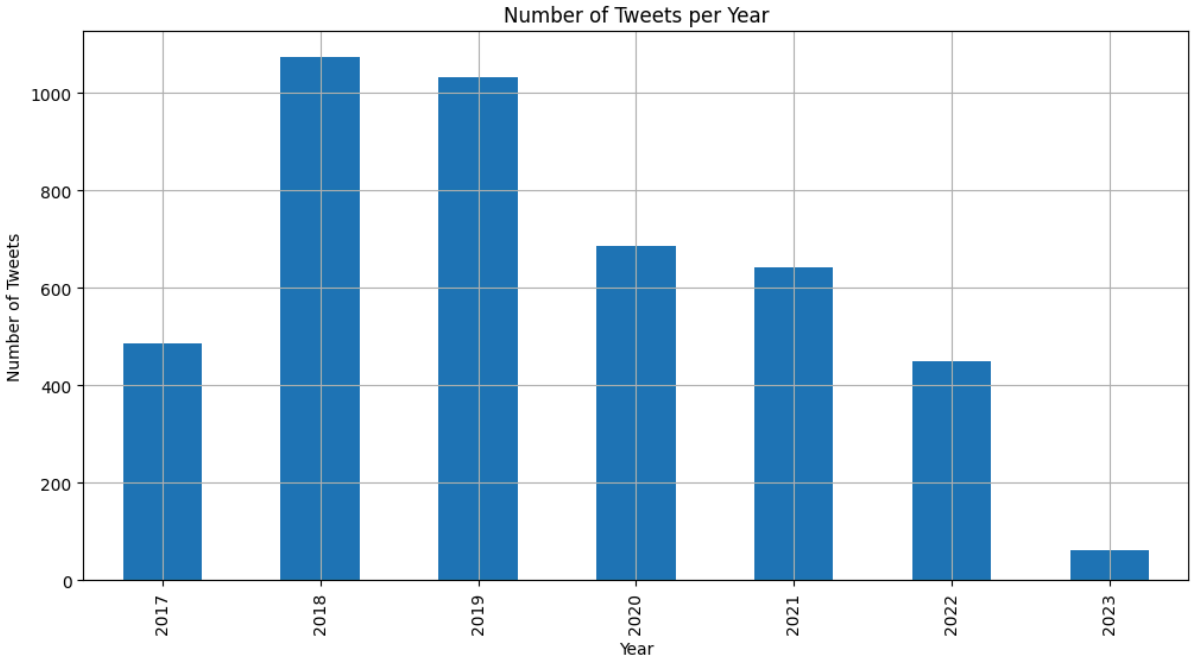


Figure 4.2: Number of tweets per year

In Figure 4.2, a bar chart illustrates the distribution of tweets across different years. The data reveals that the year 2018 witnessed the highest volume of tweets followed by 2019. Following these, the years 2020 and 2021 reflect a moderate number of tweets. Similarly, 2017 and 2022 both show a comparable count of tweets, demonstrating a lower count. Conversely, 2023 stands out with a notably lower count of tweets among the observed years.

4.4 Semi Supervised Learning for Stigma Prediction

Stigma prediction is developed using four machine learning algorithms which are Support Vector Machine (SVM), Logistic Regression, Naive Bayes, and Random Forest utilising semi-supervised learning. First, a function for plotting the confusion matrix is used.

```
# Function to plot confusion matrix
def plot_confusion_matrix(model_name, y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    accuracy = accuracy_score(y_true, y_pred)

    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
cbar=False)
    plt.title(f'Confusion Matrix - {model_name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.text(1.1, 0.5, f'Accuracy: {accuracy:.2f}',
transform=plt.gca().transAxes, verticalalignment='center')
    plt.show()
```

4.4.1 Support Vector Machine (SVM) Algorithm

Step 1: Initialise an SVM classifier model

```
svm_model = SVC(kernel='linear', random_state=42)
```

SVC is a class from the scikit-learn library (sklearn) that represents a Support Vector Classifier

- kernel='linear' indicates that the SVM is using a linear kernel, which means it attempts to find a linear decision boundary between the classes.
- The random_state parameter is used to set the random seed for reproducibility. By setting it to a 42, the random number generator produces the same sequence of random numbers each time the code is run. This ensures that the results of the SVM model training is consistent across different runs.

Step 2: Train the SVM model

```
svm_model.fit(X_train, y_train)
```

- X_train is a training data
- Y_train is its corresponding labels

Step 3: Use the trained SVM model to make predictions on the test data (X_test).

```
svm_predictions = svm_model.predict(X_test)
```

Step 4: Evaluate the accuracy of the SVM model

```
svm_accuracy = accuracy_score(y_test, svm_predictions)
```

- accuracy_score is a function from the scikit-learn library (sklearn) to calculate the accuracy of a classification model's predictions.
- svm_predictions is predicted labels.
- y_test is true labels.

Step 5: Plot a confusion matrix and generate a classification report

```
# Plot confusion matrix for SVM
plot_confusion_matrix('SVM', y_test, svm_predictions)
svm_report = classification_report(y_test, svm_predictions)
print(svm_report)
```

- Use the previous function plot_confusion_matrix and compare the actual (true) labels (y_test) with the predicted labels (svm_predictions) generated by the SVM model.
- classification_report is a function from the scikit-learn library (sklearn) that generates a precision, recall, F1-score, and support for each class, as well as the average metrics across all classes.
- y_test contains the true labels (ground truth) of the test dataset. It represents the actual classes of the samples in the test set.
- svm_predictions contains the predicted labels generated by the SVM model for the test dataset. These are the labels predicted by the model based on the input features (X_test).

4.4.2 Logistic Regression Algorithm

Step 1: Initialise an Logistic Regression model

```
lr_model = LogisticRegression(random_state=42)
```

The Logistic Regression model is initialised and assigned to the variable lr_model.

- The LogisticRegression function creates an instance of the logistic regression model.
- random_state=42 sets the random seed to ensure reproducibility. Each time the model is trained, it produces the same results, which is useful for debugging and comparing different runs.

Step 2: Train Logistic Regression Model

```
lr_model.fit(X_train, y_train)
```

- Fit the model to the training data (X_train, y_train) using the fit method.
- The X_train parameter represents the input features (independent variables) used for training the model.
- The y_train parameter represents the target labels (dependent variable) corresponding to the input features.

Step 3: Use the trained Logistic Regression model to make predictions on the test data (X_test).

```
lr_predictions = lr_model.predict(X_test)
```

Step 4: Evaluate the accuracy of the Logistic Regression model

```
lr_accuracy = accuracy_score(y_test, lr_predictions)
```

- Calculate the accuracy of the Logistic Regression model's predictions by comparing them to the y_test using the accuracy_score function.
- lr_predictions is predicted labels.
- y_test is true labels.

Step 5: Plot a confusion matrix and generate a classification report

```
# Plot confusion matrix for Logistic Regression
plot_confusion_matrix('Logistic Regression', y_test,
lr_predictions)
lr_report = classification_report(y_test, lr_predictions)
print(lr_report)
```

- Generate a confusion matrix for the Logistic Regression model's predictions using the plot_confusion_matrix function.
- Pass the model name ('Logistic Regression'), true labels y_test, and predicted labels lr_predictions to the function.
- Generate a classification report for the Logistic Regression model's predictions using the classification_report function.
- The y_test parameter represents the true labels from the test set.
- The lr_predictions parameter represents the predicted labels generated by the Logistic Regression model for the test set.

4.4.3 Naive Bayes Algorithm

Step 1: Initialise an Naive Bayes model

```
nb_model = MultinomialNB()
```

- The Multinomial Naive Bayes model is initialised and assigned to the variable nb_model.
- The MultinomialNB function creates an instance of the Multinomial Naive Bayes model.

Step 2: Train Naive Bayes Model

```
nb_model.fit(X_train, y_train)
```

- Fit the model to the training data (X_train, y_train) using the fit method.
- The X_train parameter represents the input features (independent variables) used for training the model.
- The y_train parameter represents the target labels (dependent variable) corresponding to the input features.

Step 3: Use the trained Naive Bayes model to make predictions on the test data (X_test).

```
nb_predictions = nb_model.predict(X_test)
```

Step 4: Evaluate the accuracy of the Naive Bayes model

```
nb_accuracy = accuracy_score(y_test, nb_predictions)
```

- Calculate the accuracy of the Naive Bayes model's predictions by comparing them to the y_test using the accuracy_score function.
- nb_predictions represent predicted labels.
- y_test represents true labels.

Step 5: Plot a confusion matrix and generate a classification report

```
# Plot confusion matrix for Naive Bayes
plot_confusion_matrix('Naive Bayes', y_test, nb_predictions)
nb_report = classification_report(y_test, nb_predictions)
print(nb_report)
```

- Generate a confusion matrix for the Naive Bayes model's predictions using the plot_confusion_matrix function.
- Pass the model name ('Naive Bayes'), true labels y_test, and predicted labels nb_predictions to the function.
- Generate a classification report for the Naive Bayes model's predictions using the classification_report function.
- The y_test parameter represents the true labels from the test set.
- The nb_predictions parameter represents the predicted labels generated by the Naive Bayes model for the test set.

4.4.4 Random Forest Algorithm

Step 1: Initialise an Random Forest model

```
rf_model = RandomForestClassifier(random_state=42)
```

- The Random Forest model is initialised and assigned to the variable rf_model.
- The RandomForestClassifier function creates an instance of the Random Forest classifier model.
- random_state=42 sets the random seed to ensure reproducibility. Each time the model is trained, it produces the same results, which is useful for debugging and comparing different runs.

Step 2: Train Random Forest Model

```
rf_model.fit(X_train, y_train)
```

- Fit the model to the training data (X_train, y_train) using the fit method.
- The X_train parameter represents the input features (independent variables) used for training the model.
- The y_train parameter represents the target labels (dependent variable) corresponding to the input features.

Step 3: Use the trained Random Forest model to make predictions on the test data (X_test).

```
rf_predictions = rf_model.predict(X_test)
```

Step 4: Evaluate the accuracy of the Random Forest model

```
rf_accuracy = accuracy_score(y_test, rf_predictions)
```

- Calculate the accuracy of the Random Forest model's predictions by comparing them to the y_test using the accuracy_score function.
- rf_predictions represent predicted labels.
- y_test represents true labels.

Step 5: Plot a confusion matrix and generate a classification report

```
# Plot confusion matrix for Random Forest
plot_confusion_matrix('Random Forest', y_test, rf_predictions)
rf_report = classification_report(y_test, rf_predictions)
print(rf_report)
```

- Generate a confusion matrix for the Random Forest model's predictions using the plot_confusion_matrix function.
- Pass the model name ('Random Forest'), true labels y_test, and predicted labels rf_predictions to the function.
- Generate a classification report for the Random Forest model's predictions using the classification_report function.
- The y_test parameter represents the true labels from the test set.
- The rf_predictions parameter represents the predicted labels generated by the Random Forest model for the test set.

4.5 Geolocation Integration Using OpenCage

The OpenCage Geocoding API is utilised to improve a dataset by retrieving specific geographical details, including latitude, longitude, and nation, that correlate to the given place in the dataset. By incorporating geolocation data, the dataset is enhanced, allowing for a deeper examination and visualisation of stigma-related tweets according to their geographic distribution.

Step 1: Install the necessary library for interfacing with the OpenCage Geocoding API. Next, import the OpenCageGeocode class from the opencage.geocoder module to facilitate interactions with the OpenCage Geocoding API.

```
!pip install opencage
from opencage.geocoder import OpenCageGeocode
```

Step 2: Initialise the API key provided by OpenCage.

```
api_key = '0d4c4d3b9cff4e09a705c5c53195e8b2'
geocoder = OpenCageGeocode(api_key)
```

Step 3: Extract the location information. Then, query the OpenCage Geocoding API to obtain latitude, longitude, and country details based on the provided location

```
for index, row in stigma_tweets.iterrows():
    location = row['location']

    # Query OpenCage Geocoding API to obtain location
information
    results = geocoder.geocode(location)

    # Process API results
    if not is_location_empty(results):
        latitude = results[0]['geometry']['lat']
        longitude = results[0]['geometry']['lng']
        country = results[0]['components']['country']

        # Add latitude, longitude, and country to DataFrame
        stigma_tweets.at[index, 'latitude'] = latitude
        stigma_tweets.at[index, 'longitude'] = longitude
        stigma_tweets.at[index, 'country'] = country
    else:
        # If location information is empty, remove the row
        stigma_tweets = stigma_tweets.drop(index)
```

Step 4: Handling empty or missing location information. Function is_location_empty() is used to check if the location information retrieved from the API is empty. If the location information is missing or invalid, discard the corresponding row from the dataset to ensure data integrity and accuracy.

```
# Function to check if location information is empty
def is_location_empty(result):
    return result is None or len(result) == 0 or 'geometry'
not in result[0]
```

4.6 Dashboard Design for Stigma Visualisation

4.6.1 Choropleth Map

Choropleth map visualisation to illustrate the distribution of stigma-related tweets across different countries.

```
# Choropleth Map
    fig_map = px.choropleth(
        stigma_counts,
        locations="country",
        locationmode="country names",
        color="predicted_stigma",
        color_continuous_scale=available_color_scales[1],  # 
Set the desired color scale
        labels={'predicted_stigma': 'Stigma Count'},
    )
```

- stigma_counts specifies the DataFrame containing the data to be visualised on the map. It includes columns such as "country" and "predicted_stigma" where "country" represents the geographic locations and "predicted_stigma" represents the count of stigma associated with each country.
- locations="country" indicates the column in the stigma_counts DataFrame that contains the names of the countries.
- locationmode="country names" specifies the mode used to determine the geographic location. In this case, it's set to "country names,".
- color="predicted_stigma" determines the variable used to colour the map.
- color_continuous_scale=available_color_scales[1] defines the colour scale used to represent the range of values for the variable specified in the colour parameter.
- The parameter labels={'predicted_stigma': 'Stigma Count'} enables the customisation of the colour legend that is shown next to the map. The "Stigma Count" is assigned to the variable represented by the colour, providing users with a clear understanding of the significance of the colour representation on the map.

4.6.2 Bar Chart

Bar chart illustrating the frequency of tweets associated with stigma, categorised by country, to provide users with a comparative analysis of stigma prevalence in various countries. Empowered users to get knowledge on geographical variations in stigma activities and identify countries with the highest and lowest levels of engagement in stigma.

```
# Bar Plot
    fig_bar = px.bar(
        stigma_counts,
        x="country",
        y="predicted_stigma",
        title="Stigma Count by Country",
        labels={'predicted_stigma': 'Stigma'},
        height=700,  # Set the desired height
        category_orders={"country":
stigma_counts.sort_values('predicted_stigma',
ascending=False)['country']},
        color_discrete_sequence=['#4B0082'] *
len(stigma_counts)  # Set the desired color for the bar chart
    )
```

- x="country" indicates the column in the stigma_counts DataFrame that contains the country names to be displayed on the x-axis of the bar chart.
- y="predicted_stigma" specifies the column in the stigma_counts DataFrame that contains the predicted count of stigma for each country to be displayed on the y-axis of the bar chart.
- title="Stigma Count by Country" sets the title of the bar chart, providing a descriptive title that indicates what the chart represents.
- labels={'predicted_stigma': 'Stigma'} allows custom labelling of the axes of the bar chart. It specifies the label to be used for the y-axis, "Stigma," providing clarity to users about the meaning of the numerical data represented on the chart.
- height=700 sets the height of the bar chart to 700 pixels, controlling the visual aspect of the chart.
- category_orders={"country":stigma_counts.sort_values('predicted_stigma', ascending=False)['country']} allows custom ordering of the categories on the x-axis of the bar chart. It sorts the countries based on their predicted stigma counts in

descending order, ensuring that the bars are displayed in order from highest to lowest stigma count.

4.6.3 Word Clouds

Two word clouds are generated, one for negative sentiment tweets and another for positive sentiment tweets.

```
# Generate Word Clouds for negative and positive sentiment
words
wordcloud_img_negative =
generate_wordcloud(negative_tweets['tweet'], "Negative
Sentiment Tweets", '#FF9AA2')
wordcloud_img_positive =
generate_wordcloud(positive_tweets['tweet'], "Positive
Sentiment Tweets", '#FFD700')

# Function to generate a word cloud
def generate_wordcloud(text, title, color):
    wordcloud = WordCloud(width=700, height=400,
background_color='#faf8f9').generate(' '.join(text))
    plt.figure(figsize=(7, 4))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f"Frequent Word in {title}", fontsize=16)
    plt.tight_layout()

    # Save the word cloud image
    img_buf = BytesIO()
    plt.savefig(img_buf, format='png', bbox_inches='tight',
pad_inches=0.1)
    img_buf.seek(0)

    # Encode the image to base64
    img_encoded =
base64.b64encode(img_buf.read()).decode('utf-8')

    return f'{img_encoded}'
```

- The generate_wordcloud function is called twice, once for negative sentiment tweets and once for positive sentiment tweets. It generates a word cloud based on the provided text data. It takes three arguments:
    - text: The text data (tweets) for which the word cloud is generated.
    - title: The title of the word cloud.
    - colour: The colour theme of the word cloud.
- The WordCloud object is created using the WordCloud class from the wordcloud library. The generate method of the WordCloud object is called with the concatenated text data to generate the word cloud.
- The word cloud is then plotted using Matplotlib's imshow function. Various parameters such as interpolation, axis, title, and tight_layout are set to customise the appearance of the word cloud.

CHAPTER 5

SYSTEM IMPLEMENTATION AND DISCUSSION

5.1 Introduction

This chapter focuses on the implementation of the designs presented in chapter 4 into the system. The system is designed to align with the project's objectives. The hardware and software utilised in the development of the system is examined. Furthermore, this chapter includes a thorough analysis and examination of the obtained outcomes or findings.

5.2 System Hardware and Software

This section explains the hardware and software used to implement this project.

5.2.1 System Hardware

The hardware used to develop the system is a laptop with the specifications as below:

- Laptop Model : HP ENVY 13-inch
- Operating System : Windows 11 Home Single Language 22H2
- Processor : Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz
- Installed RAM : 16.0 GB
- System type : 64-bit operating system, x64-based processor

5.2.2 System Software

The framework used to build the dashboard is Flask. This framework needed to be installed first in Google Colab. The language used in Google Colab is Python. The software used to develop the system is listed below:

- Development Software: Google Colab, Visual Studio Code
    - Google Colab was used for coding and running Python scripts, taking advantage of its cloud-based Jupyter notebook environment.
    - VS Code served as the primary Integrated Development Environment (IDE) for writing and editing code, providing features such as syntax highlighting, code completion, and debugging tools.
- Framework: Flask
    - Flask is a lightweight web framework for Python, and was employed to develop the dashboard application.
- Spreadsheet Software: Microsoft Excel
    - Microsoft Excel was used for manual labelling of data. It provided a familiar interface for working with tabular data and performing basic data manipulations.
- Language: Python
    - Python was used as the main programming language for constructing both the stigma predictor and the dashboard. The platform provides an extensive selection of tools and frameworks for various activities including data analysis, machine learning, web development, and visualisation.

5.3 Manual Labelling

Manual labelling is an essential and crucial stage in both supervised and semi-supervised learning tasks. It acts as the foundation for training and evaluating models. In the field of mental health stigma prediction, this procedure involves classifying tweets based on their content in order to distinguish between those that include terms connected to mental health stigma and those that do not exhibit such characteristics.

Manual labelling in this particular implementation is performed according to an individual's understanding of mental health stigma and own understanding of the tweet's content. The categorization of tweets is based on this subjective comprehension, which involves giving labels to differentiate them into many groups, including:

- Label 0 (No Stigma): Tweets that do not contain any identifiable content that promotes or perpetuates mental health stigma.
- Label 1 (Stigma): Tweets using words or phrases that are linked to the negative perception or discrimination against mental health.
- Label 2 (Unrelated Tweet): Tweets that are not related to the stigma around mental health, including general questions or conversations that are not relevant.

```
# Randomly select 85 samples to manually label
random_sample = df.sample(n=85, random_state=42)  # Use a
fixed random_state for reproducibility

# open in csv
random_sample.to_csv('random_samples.csv', index=False)

# After manual labeling in Excel, read the labeled data back
into Python

file_path = '/content/drive/My Drive/FYP/labeled_data.csv'

labeled_data = pd.read_csv(file_path, encoding='ISO-8859-1')
```

- random_sample = df.sample(n=85, random_state=42): randomly selects 85 samples from the DataFrame df for manual labelling. The random_state parameter ensures reproducibility of the results.
- random_sample.to_csv('random_samples.csv', index=False): This line exports the randomly selected samples to a CSV file named 'random_samples.csv', which is then manually labelled in Excel based on own understanding and reading on mental health stigma.
- labeled_data = pd.read_csv(file_path, encoding='ISO-8859-1'): After manual labelling in Excel, the labelled data is read back into Python from the CSV file containing the manual annotations.

5.4 Confusion Matrix for Each Model

The confusion matrices for all algorithms indicate strong performance, with SVM achieving the highest accuracy. This suggests that SVM effectively distinguishes between tweets containing mental health stigma-related language and those that do not which are shown in Figure 5.1 to Figure 5.4.

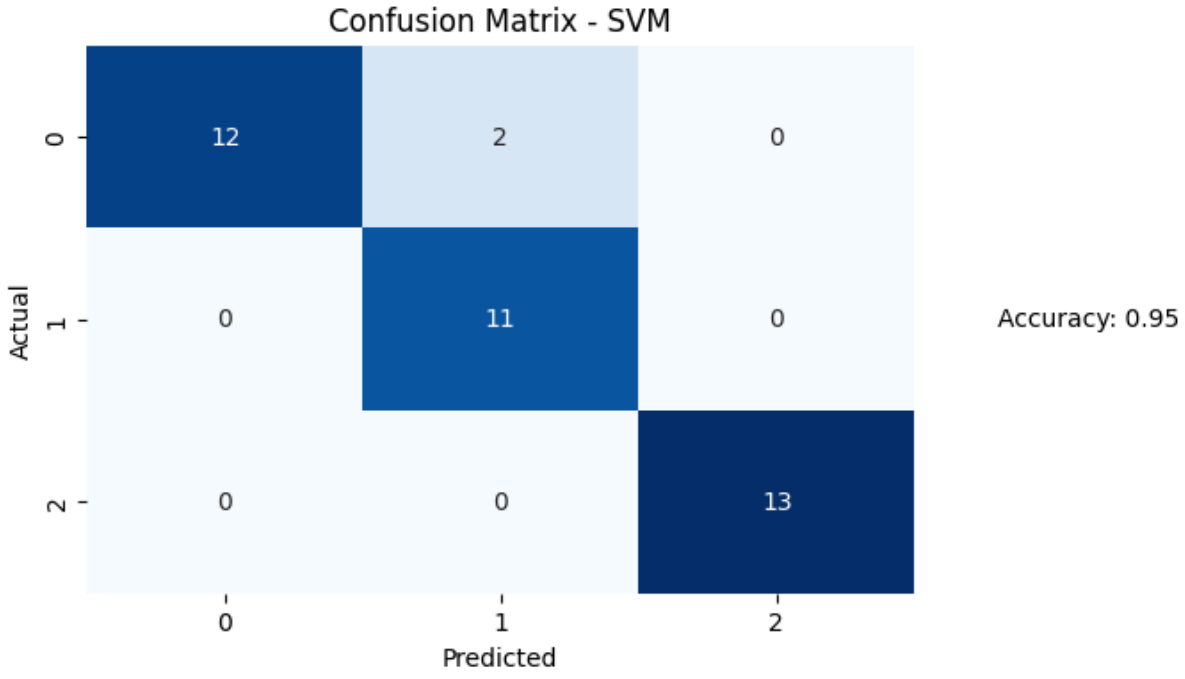


Figure 5.1: Confusion Matrix for SVM

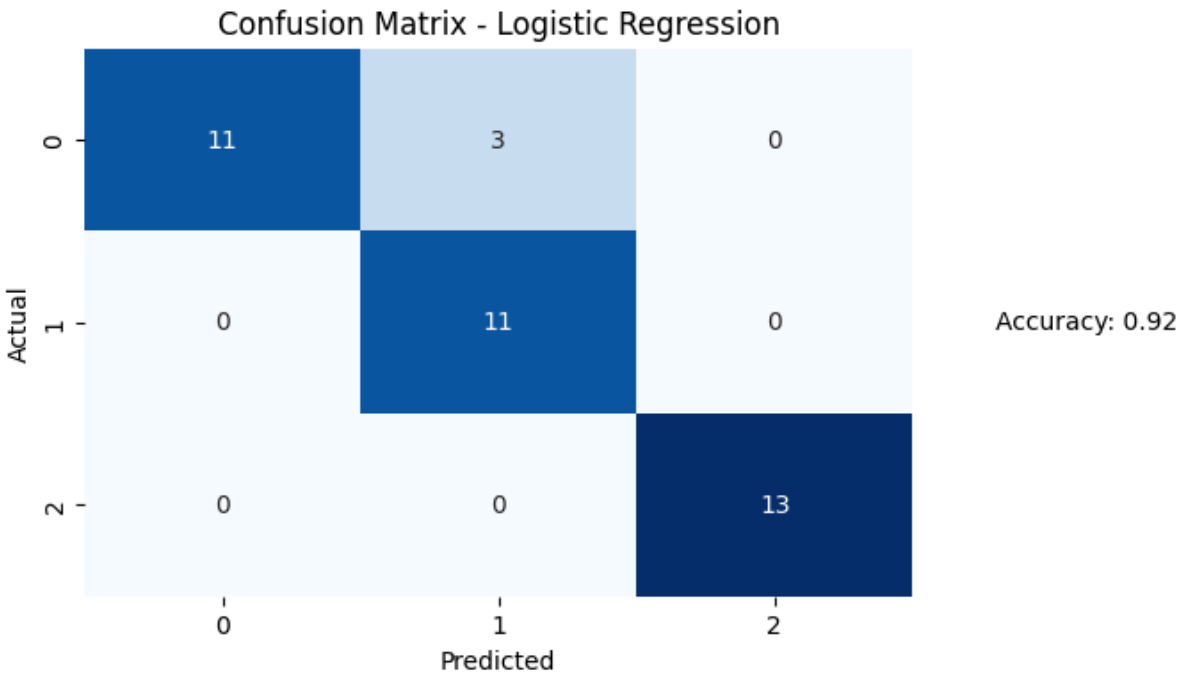


Figure 5.2: Confusion Matrix for Logistic Regression

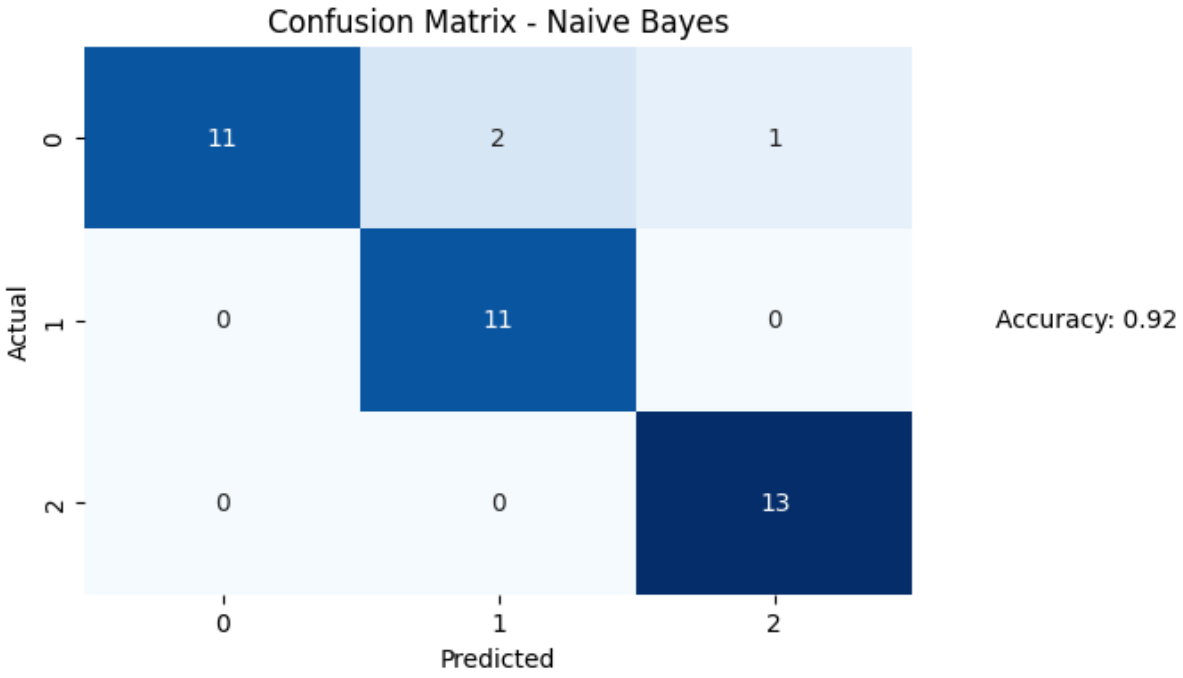


Figure 5.3: Confusion Matrix for Naive Bayes

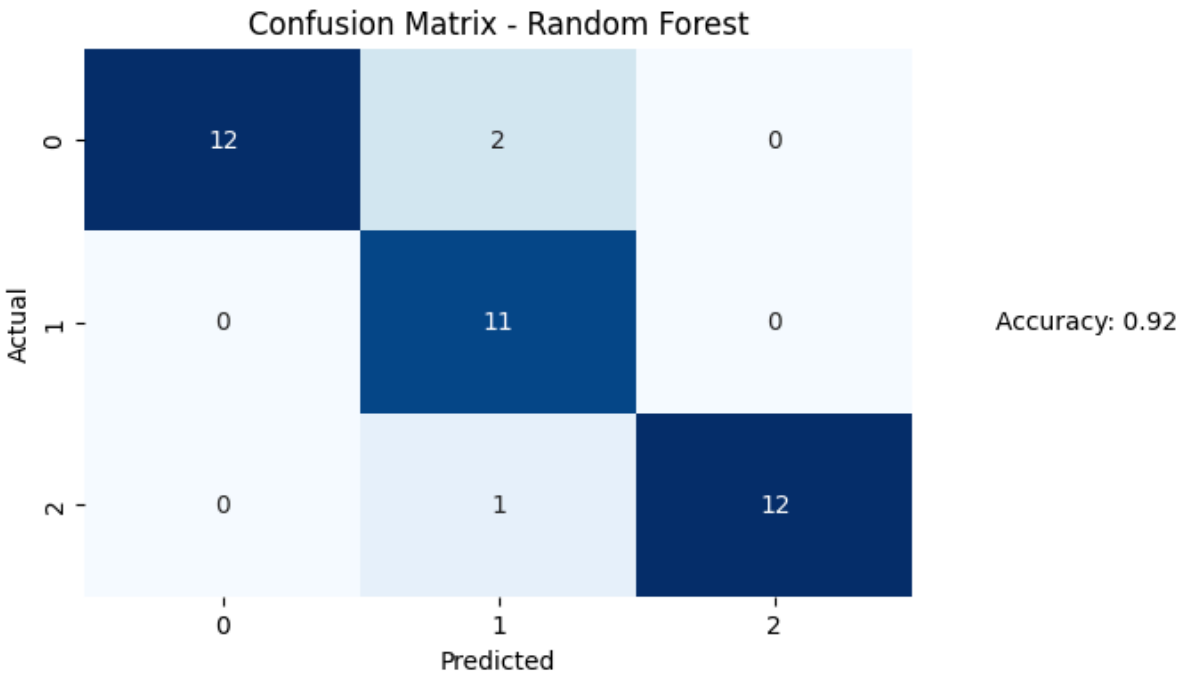


Figure 5.4: Confusion Matrix for Random Forest

5.5 Model Evaluation

The stigma prediction performance has been evaluated with several measures, including accuracy, precision, recall, and f1-score.

Accuracy is defined as a percentage of correctly categorised samples out of the total number of samples in the evaluation dataset. This metric is frequently used in Machine Learning and Data Analysis projects to assess the ability of a model to effectively predict outcomes or classify data. The accuracy can be calculated using the following formula:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

Precision is the metric used to measure the accuracy of a classifier or model in making predictions. The precision is determined by dividing the number of accurately categorised positive samples by the total number of samples classified as positive. In other words, precision refers to the degree of accuracy in a model's predictions, specifically the proportion of predicted positive cases that are actually positive. The precision can be calculated using the following formula:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall, also known as sensitivity or True Positive Rate (TPR), is a crucial measure in classification tasks that signifies the model's capacity to correctly identify instances that are positive. Essentially, it measures the classifier's ability to accurately identify the actual positive cases. This indicator is essential as it contributes in evaluating the classifier's effectiveness in accurately identifying positive instances within the dataset. Recall is mathematically determined by dividing the number of true positives by the total of true positives and false negatives. Recall, which takes into account both accurate positive predictions and missed positive examples, offers a thorough assessment of the classifier's ability to identify positive instances. This provides useful insights into the classifier's performance. The precision can be calculated using the following formula:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

The F1 score is a balanced evaluation metric that concurrently evaluates both precision and recall, as it is calculated as the harmonic mean of these two measures. The F1 score is a statistic that considers the true positives, false positives, and false negatives in imbalanced datasets, making it more suitable than accuracy for evaluating classifier performance. It serves as a beneficial instrument for situations when achieving an equilibrium between precision and recall is crucial, especially in instances with imbalanced class distributions. Moreover, the F1 score's sensitivity to extreme values of accuracy and recall guarantees that the metric effectively penalises classifiers that are biassed towards either class. This attribute renders it especially valuable in scenarios where the classifier's performance must be resilient across various class distributions, hence providing a more nuanced assessment of model efficacy. The F1 score can be calculated using the following formula:

$$F1\ score\ = \frac{2\ *(\ Precision\ *\ Recall)}{Precision + Recall}$$

Table 5.1 below is an evaluation metric for stigma prediction for each machine learning algorithms:

Table 5.1: Evaluation Metric of Stigma Prediction

Metrics	SVM	Logistic Regression	Naive Bayes	Random Forest
Accuracy	0.95	0.92	0.92	0.92
Precision	0.95	0.93	0.92	0.93
Recall	0.95	0.92	0.92	0.92
F1 score	0.95	0.92	0.92	0.92

The evaluation metrics table presents a thorough summary of the performance of different classification methods, with a particular emphasis on analysing the Support Vector Machine (SVM) technique. Support Vector Machines (SVM) provide consistent and strong performance across various measures. Firstly, it attains a remarkable precision of 0.95, signifying that 95% of the model's forecasts are accurate. This indicates a significant degree of accuracy in categorising tweets as either exhibiting mental health stigma or not. Furthermore, the Support Vector Machine (SVM) demonstrates high precision and recall scores, both of which are 0.95. This indicates that 95% of the tweets that the model predicts as having stigma actually do contain such content, and 95% of the tweets that do contain stigma are accurately detected by the algorithm. The F1 score, a metric that combines precision and recall using a harmonic mean, provides additional evidence of the effectiveness of SVM, with a score of 0.95. The balanced review highlights SVM's capacity to achieve a harmonious equilibrium between precision and recall. Although Logistic Regression, Naive Bayes, and Random Forest models also perform well, SVM stands out as the best choice because it consistently outperforms them in all evaluation criteria.

5.4 Stigma Prediction Dashboard

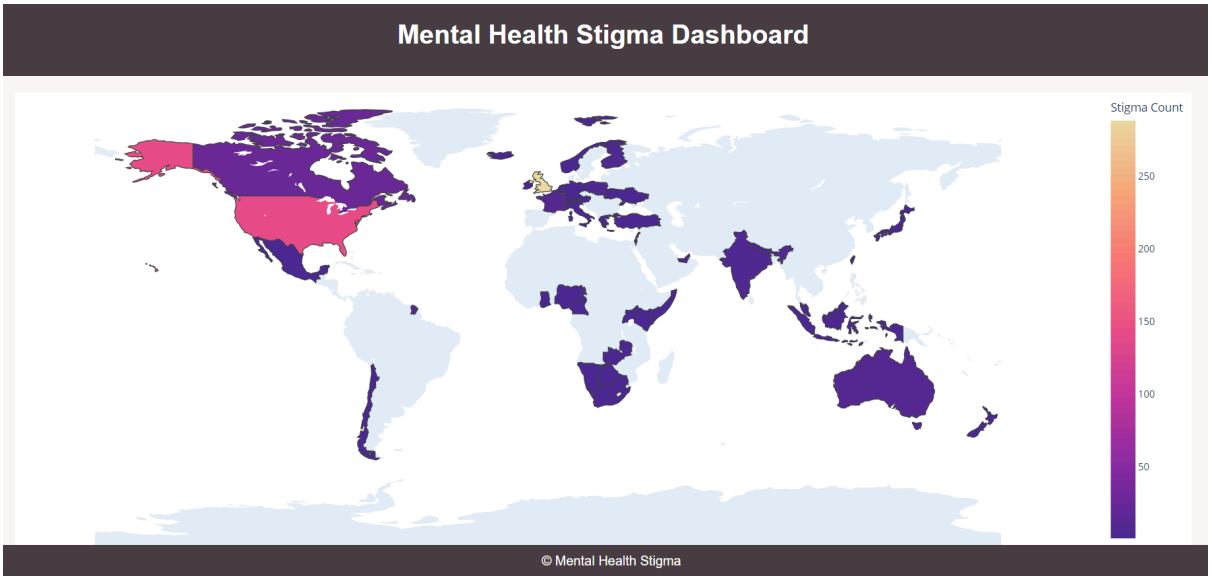


Figure 5.5: Map of Mental Health Stigma

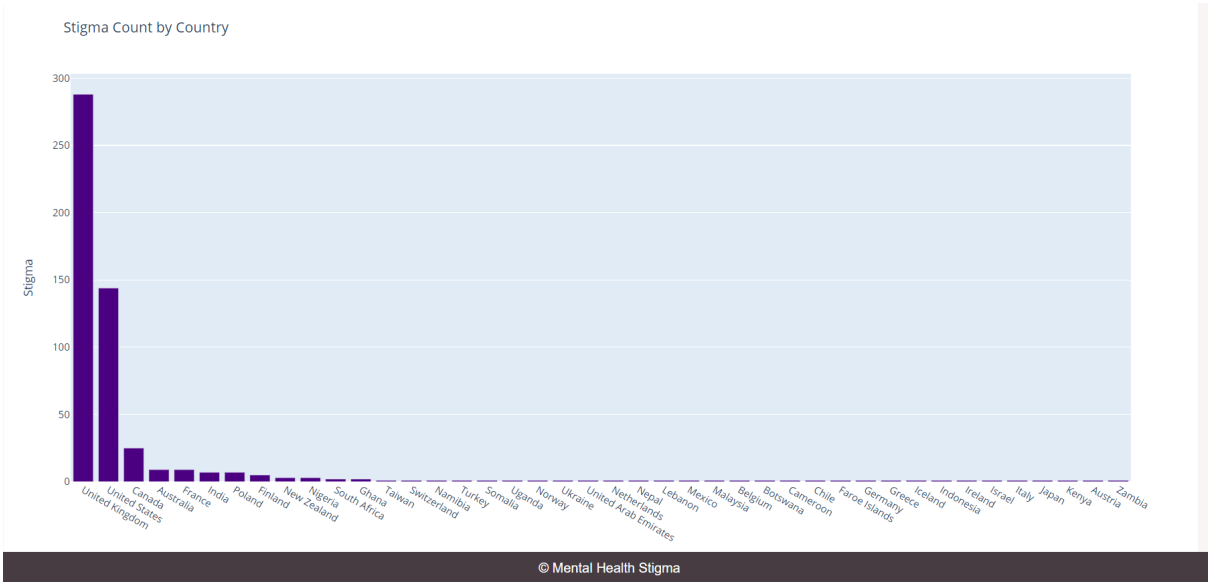


Figure 5.6: Bar Chart Stigma Count by Country

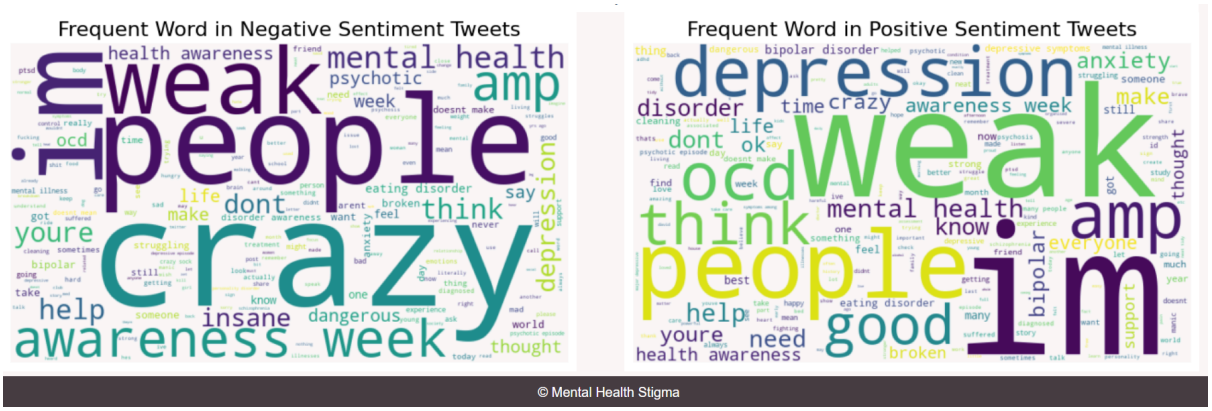


Figure 5.7: Word Cloud for Negative and Positive Sentiment Tweets

The dashboard functions as a potent instrument for visualising and comprehending the extent of mental health stigma in various countries. It utilises three essential visualisations to offer full insights:

1. The choropleth map in Figure 5.5 shows the geographic distribution of mental health stigma among different nations. The colour shading of each country corresponds to the degree of stigma, enabling users to promptly identify countries with greater or lesser levels of stigma. Users can obtain comprehensive information about each country, including the country name and the related count of stigmas, by simply hovering over it. This feature allows for a more in-depth understanding of certain geographic regions.

2. The bar chart in Figure 5.6 displays the number of stigmas for each country, providing a visual representation that complements the choropleth map. This visualisation facilitates the comparison of stigma prevalence across several countries in a more streamlined manner. Users may efficiently discern countries with the highest and lowest levels of stigma, enabling them to do comparison analysis and find trends.

3. Word clouds in Figure 5.7 provide a visual depiction of the sentiment expressed in discussions around mental health stigma. Different word clouds are produced for tweets with negative and positive sentiment, showcasing commonly appearing words in each classification. Through the analysis of word clouds, users can acquire valuable understanding of the language and recurring themes that are often linked to debates about stigma. This enables the identification of prevailing narratives and attitudes present in the discourse.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In conclusion, this project makes a significant contribution to the field of mental health analysis by utilising innovative data visualisation techniques and a wide range of machine learning algorithms to analyse Twitter data. The focus of this project is on dynamically illustrating variations in regional mental health stigma through attractive representations.

The primary goal is to provide healthcare and government agencies with actionable insights, providing them with the tools to create targeted awareness campaigns and data-driven interventions to reduce mental health stigma. By emphasising insightful data representation, the project aims to provide information and also foster understanding and support for mental health. Lastly, the goal of this project is to help create a broader societal shift towards a more empathetic and inclusive approach to mental health.

6.2 Limitation

This project acknowledges some limitations that need to be carefully evaluated. Primarily, this project relies solely on Twitter data which introduces a potential limitation in capturing the full range of mental health stigma variations. Twitter users are a specific demographic subset and may not fully represent the diversity of perspectives within the broader population, potentially biassing the study's findings.

A significant limitation is the dataset's representation of Malaysia, which is currently limited. The dataset's lack of Malaysian-specific tweets limits the project's ability to provide a comprehensive understanding of mental health stigma in Malaysia. To address this limitation and provide more targeted support to Malaysia's Ministry of Health, data collection efforts must be expanded within Malaysia, resulting in a more representative and varied dataset.

The manual labelling process, while necessary for training machine learning models, introduces an element of subjectivity. The current labelling effort, which includes 85 data points out of a total of 4347, provides opportunity for further development. A larger and more diverse set of labelled data points would improve the training process, improving the models' ability to accurately identify mental health-related content.

6.3 Future Work

The project's proposed future work outlines a comprehensive strategy for advancing and refining the stigma prediction. The expansion of data collection emerges as a critical step, advocating for increased dataset diversity by including data from various social media platforms such as Reddit, YouTube, and Facebook. This expansion aims to provide a more comprehensive understanding of mental health stigma across multiple social media environments.

Next, by enhancing the semi-supervised learning approach, it improves the model performance. The next phase could focus on implementing advanced techniques such as self-training or co-training. These methodologies aim to effectively leverage unlabeled data, thereby improving model performance, particularly in scenarios where labelled data is unlimited. By improving the semi-supervised learning approach, the project may be able to make more detailed and accurate predictions of stigma-related content.

Exploration of advanced machine learning techniques emerges as a new avenue for future research. Experimenting with cutting-edge methodologies outside the current scope, such as deep learning architectures or ensemble methods, has the potential to improve the accuracy of stigma-related content identification. Lastly, collaboration with mental health organisations improve the tweet labelling. The project can gain domain-specific insights and provides a deeper awareness of the content and ensures that the stigma prediction closely aligns with the complexities of mental health discussion.

REFERENCES

1. Budenz, A., Klassen, A., Purtle, J., Yom Tov, E., Yudell, M., & Massey, P. (2020). Mental illness and bipolar disorder on Twitter: Implications for stigma and social support. *Journal of Mental Health*, 29(2), 191-199. doi:10.1080/09638237.2019.1677878

2. Jilka, S., Odoi, C. M., Van Bilsen, J., Morris, D. H., Erturk, S., Cummins, N., Cella, M., & Wykes, T. (2022). Identifying schizophrenia stigma on Twitter: a proof of principle model using service user supervised machine learning. *Schizophrenia*, 8(1). https://doi.org/10.1038/s41537-021-00197-6

3. Ms, E. L. (2022). How We Can Change the Stigma Around Mental Health. Healthline. https://www.healthline.com/health/mental-health/mental-health-stigma-examples

4. Parveen, H., & Pandey, S. (2016). Sentiment analysis on Twitter Data-set using Naive Bayes algorithm. *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*. doi:10.1109/icatcct.2016.7912034

5. Reddy, P., Sri, D., Reddy, C., & Shaik, S. (2021). Sentimental Analysis using Logistic Regression. *International Journal of Engineering Research and Applications*. 11. 36-40. doi:10.9790/9622-1107023640.

6. Zaydman, M. (2017). Tweeting About Mental Health: Big Data Text Analysis of Twitter for Public Policy. *The Pardee RAND Graduate School*. https://www.rand.org/content/dam/rand/pubs/rgs_dissertations/RGSD300/RGSD391/RAND_RGSD391.pdf

7. Wibawa, Aji & Kurniawan, Ahmad & Murti, Della & Adiperkasa, Risky Perdana & Putra, Sandika & Kurniawan, Sulton & Nugraha, Youngga. (2019). Naïve Bayes Classifier for Journal Quartile Classification. *International Journal of Recent Contributions from Engineering, Science & IT (iJES)*. 7. 91. 10.3991/ijes.v7i2.10659.

8. Couronné, R., Probst, P., & Boulesteix, A. L. (2018). Random forest versus logistic regression: a large-scale benchmark experiment. *BMC bioinformatics*, 19(1), 270. https://doi.org/10.1186/s12859-018-2264-5

9. Tableau. (n.d.). *What is a dashboard? A complete overview*. https://www.tableau.com/learn/articles/dashboards/what-is#:~:text=Benefits-,Dashboard%20definition,easy-to-digest%20form.

10. IBM. (n.d.).*What is Random Forest?* https://www.ibm.com/topics/random-forest

11. Suresh, A. (2024). What is a confusion matrix? - Analytics Vidhya - Medium. *Medium*. https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5

12. Hady, M.F.A., Schwenker, F. (2013). Semi-supervised Learning. In: Bianchini, M., Maggini, M., Jain, L. (eds) *Handbook on Neural Information Processing. Intelligent Systems Reference Library*, vol 49. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-36657-4_7

13. Ridzuan, F., & Zainon, W. M. N. W. (2019). A review on data cleansing methods for big data. *Procedia Computer Science,* 161, 731-738.

14. Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM computing surveys (CSUR)*, 50(6), 1-45.

15. Komorowski, M., Marshall, D. C., Salciccioli, J. D., & Crutain, Y. (2016). Exploratory data analysis. *Secondary analysis of electronic health records*, 185-203.

16. Silwal, D. (2022). *Confusion Matrix, Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures*. https://www.linkedin.com/pulse/confusion-matrix-accuracy-precision-recall-f1-score-measures-silwal/

17. Hicks, S. A., Strümke, I., Thambawita, V., Hammou, M., Riegler, M. A., Halvorsen, P., & Parasa, S. (2022). On evaluation metrics for medical applications of artificial intelligence. Scientific reports, 12(1), 5979. https://doi.org/10.1038/s41598-022-09954-8

18. Shung, K. P. (2020). Accuracy, precision, recall or F1? - towards data science. Medium. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

19. Pudjihartono, N., Fadason, T., Kempa-Liehr, A. W., & O'Sullivan, J. M. (2022). A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction. *Frontiers in bioinformatics*, 2, 927312. https://doi.org/10.3389/fbinf.2022.927312

20. American Psychiatric Association. (n.d.). *Stigma, Prejudice and Discrimination Against People with Mental Illness*. https://www.psychiatry.org/patients-families/stigma-and-discrimination

21. Rössler W. (2016). The stigma of mental disorders: A millennia-long history of social exclusion and prejudices. *EMBO reports*, 17(9), 1250–1253. https://doi.org/10.15252/embr.201643041

22. Breiman, L. (2001) Random Forests. *Machine Learning* 45, 5–32. https://doi.org/10.1023/A:1010933404324

APPENDIX

1. Stigma Prediction Coding

```
pip install wordcloud matplotlib pandas
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-packages (1.9.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.10/dist-packages (from wordcloud) (1.23.5)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from wordcloud) (9.4.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.47.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

```
import pandas as pd
from google.colab import drive
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.utils import shuffle
from imblearn.combine import SMOTETomek

import seaborn as sns
import matplotlib.pyplot as plt

import re

from wordcloud import WordCloud

from scipy.sparse import csr_matrix, hstack, vstack, lil_matrix


# Mount Google Drive
drive.mount('/content/drive')

# Replace 'path_to_file' with the path to your file within Google Drive
file_path = '/content/drive/My Drive/FYP/MH_Campaigns1723.csv'

# Read the dataset using pandas
df = pd.read_csv(file_path)
```

```
Mounted at /content/drive
```

```
df.shape
```

```
(724756, 17)
```

```
df.shape
```

```
(724756, 17)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 724756 entries, 0 to 724755
Data columns (total 17 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   Date          724756 non-null  object 
 1   ID            724756 non-null  int64  
 2   url           724756 non-null  object 
 3   username      724756 non-null  object 
 4   source        724756 non-null  object 
 5   location      724756 non-null  object 
 6   tweet         724756 non-null  object 
 7   likes         724756 non-null  int64  
 8   rt            724756 non-null  int64  
 9   followers     724756 non-null  int64  
 10  replies       724756 non-null  int64  
 11  campaign      724756 non-null  object 
 12  likes_pf      724756 non-null  float64
 13  replies_pf    724756 non-null  float64
 14  rt_pf         724756 non-null  float64
 15  engagement    724756 non-null  float64
 16  engagement_0  724756 non-null  float64
dtypes: float64(5), int64(5), object(7)
memory usage: 94.0+ MB
```

```
df.describe()
```

	ID	likes	rt	followers	replies	lik
count	7.247560e+05	724756.000000	724756.000000	7.247560e+05	724756.000000	7.24756
mean	1.212629e+18	7.072277	2.342939	1.539511e+04	0.402076	
std	2.093917e+17	143.337525	31.214356	3.137420e+05	7.549365	
min	8.153481e+17	0.000000	0.000000	0.000000e+00	0.000000	0.00000
25%	1.000495e+18	0.000000	0.000000	4.330000e+02	0.000000	0.00000
50%	1.261968e+18	2.000000	0.000000	1.387000e+03	0.000000	6.50829
75%	1.392014e+18	5.000000	2.000000	4.364000e+03	0.000000	3.59582
max	1.636147e+18	74341.000000	11917.000000	8.483486e+07	2570.000000	

```
df.head()
```

	Date	ID	url	
0	2023-02-15 13:48:52+00:00	1625854658601418753	https://twitter.com/haringeyiapt/status/162585...	
1	2023-02-09 22:37:56+00:00	1623813475469344769	https://twitter.com/scrupulOCD_KC/status/16238...	s
2	2022-10-20 11:30:07+00:00	1583057973496406016	https://twitter.com/FootstepsCandC/status/1583...	Fo
3	2022-10-18 19:42:14+00:00	1582457044451627008	https://twitter.com/IOCDF/status/1582457044451...	
4	2022-10-18 19:42:13+00:00	1582457038969262080	https://twitter.com/IOCDF/status/1582457038969...	

```
df.tail()
```

	Date	ID	url
724751	2017-01-21 16:57:43+00:00	822850673314295808	https://twitter.com/ConorAllanIRL/status/82285..
724752	2017-01-21 16:30:49+00:00	822843904068321281	https://twitter.com/StudentMindsOrg/status/822..
724753	2017-01-11 18:19:02+00:00	819247260848553998	https://twitter.com/StudentMindsYSJ/status/819..
724754	2017-01-10 11:23:37+00:00	818780330983354368	https://twitter.com/SHUDisabledRep/status/8187..
724755	2017-01-05 14:37:25+00:00	817017163294867456	https://twitter.com/StudentMindsOrg/status/817..

```
# Removes the rows with location unknown
df.drop(df[df['location'] == 'unknown'].index, inplace=True)
```

```
# Removes the rows that contains NULL values.
df=df.dropna()
```

```
#check for null values
df.isnull().sum()
```

```
Date            0
ID              0
url             0
username        0
source          0
location        0
tweet           0
likes           0
rt              0
followers       0
replies         0
campaign        0
likes_pf        0
replies_pf      0
rt_pf           0
engagement      0
engagement_0    0
dtype: int64
```

```
# Convert 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Format 'Date' column as day/month/year
df['Date'] = df['Date'].dt.strftime('%d/%m/%Y')
```

```
df.head()
```

	Date	ID	url	
0	15/02/2023	1625854658601418753	https://twitter.com/haringeyiapt/status/162585...	h
2	20/10/2022	1583057973496406016	https://twitter.com/FootstepsCandC/status/1583...	Foots
3	18/10/2022	1582457044451627008	https://twitter.com/IOCDF/status/1582457044451...	
4	18/10/2022	1582457038969262080	https://twitter.com/IOCDF/status/1582457038969...	
5	17/10/2022	1582045780688637953	https://twitter.com/IOCDF/status/1582045780688...	

```
# Drop the column to exclude
df = df.drop(columns=['ID','url','username','source','likes', 'rt' , 'replies', 'followers','campaign','likes_pf','replies_pf','rt_pf',
df.head()
```

	Date	location	tweet
0	15/02/2023	Haringey	People with OCD experience intensely negative,...
2	20/10/2022	Gloucester	In light of last week's OCD Awareness week, we...
3	18/10/2022	Boston, MA	The GIFS now have over 1.4 million views on GI...
4	18/10/2022	Boston, MA	The 2022 #OCDweek events, activities, and live...
5	17/10/2022	Boston, MA	Did you catch last week's Ask The Experts live...

```
df.shape
```

```
(628364, 3)
```

Keyword-Based Detection

```
# Stigma-related keywords
stigma_keywords = ['crazy', 'insane', 'psycho', 'lunatic', 'nutcase', 'schizo', 'bipolar', 'depressive', 'manic', 'neurotic',
                   'unstable', 'weak', 'broken', 'abnormal', 'dangerous', 'attention-seeking', 'faking it', 'imaginary',
                   'made up', 'handwashing', 'cleaning', 'neat', 'psychotic', 'just about', 'attention seeker']

# Function to check if the tweet contains stigma-related keywords
def contains_stigma(tweet):

    # Check for the presence of stigma-related keywords in the tweet
    return any(re.search(r'\b{}\b'.format(keyword), tweet) for keyword in stigma_keywords)

# Filter the DataFrame to include only rows with stigma-related keywords
df = df[df['tweet'].apply(contains_stigma)]

# Display the filtered DataFrame
df.head(10)
```

	Date	location	tweet
65	12/10/2022	UK	as it's #OCDweek i'd urge anyone to try and be...
231	14/10/2021	1884 Market St San Francisco	This week is OCD Awareness Week. #OCD is not a...
233	14/10/2021	San Francisco, CA	This week is OCD Awareness Week. #OCD is not a...
279	12/10/2021	Warren, Ohio	You've probably heard someone say "I'm sooooo ...
350	11/10/2021	Everywhere	In honor of #worldmentalhealthday and the star...
386	20/10/2020	india	You are a human who gets thoughts just like ev...
387	20/10/2020	UK	THIS!!! I cannot clearly stress this enough!! ...
429	16/10/2020	gorseinon, wales, uk	OCD is horrific to live with. And while creati...
452	16/10/2020	Sacramento	I just learned it's #OCDWeek. My story is so s...

```
df.tail(10)
```

	Date	location	tweet
720887	01/03/2018	London/Essex	shouts to all the students struggling with the...
721029	01/03/2018	London, United Kingdom	It's #UniMentalHealthDay. Research shows that ...
721550	28/02/2018	London, United Kingdom	It's #UniMentalHealthDay tomorrow. Research sh...
721596	28/02/2018	United Kingdom	Tomorrow is #UniMentalHealthDay! Being healthy...
722607	02/03/2017	Peterborough, England	My love to all students on #UniMentalHealthDay...
722731	02/03/2017	Lincoln, England	'It is not weak to accept help, in fact it is ...
722791	02/03/2017	United Kingdom	Pressure on uni students is crazy. More awaren...
722925	02/03/2017	Auckland, New Zealand	#UniMentalHealthDay in a depressive episode I ...

```
df.shape
```

```
(4608, 3)
```

```
# # Randomly select 85 samples to manually label
# random_sample = df.sample(n=85, random_state=42)  # Use a fixed random_state for reproducibility

# # open in csv
# random_sample.to_csv('random_samples.csv', index=False)
```

```
# After manual labeling in Excel, read the labeled data back into Python

# Replace 'path_to_file' with the path to your file within Google Drive
file_path = '/content/drive/My Drive/FYP/labeled_data.csv'

labeled_data = pd.read_csv(file_path, encoding='ISO-8859-1')


print("the number of rated tweets by category:")

labeled_data.groupby('stigma')['tweet'].nunique()
```

```
the number of rated tweets by category:
stigma
0    62
1    17
2     6
Name: tweet, dtype: int64
```

```
# Function to clean unlabeled data
def clean_text(tweet):
    # Remove mentions (@username)
    tweet = re.sub(r'@[\w_]+', '', tweet)

    # Remove hashtags
    tweet = re.sub(r'#\w+', '', tweet)

    # Remove URLs
    tweet = re.sub(r'http\S+|www\S+|https\S+', '', tweet, flags=re.MULTILINE)

    # Remove emojis
    tweet = tweet.encode('ascii', 'ignore').decode('ascii')

    # Remove special characters and numbers
    tweet = re.sub(r'[^a-zA-Z\s]', '', tweet)

    # Remove extra whitespaces
    tweet = re.sub(' +', ' ', tweet)

    # Convert to lowercase
    tweet = tweet.lower()

    return tweet.strip()



# Clean the 'tweet' column
df['tweet'] = df['tweet'].apply(clean_text)

# Convert 'tweet' to lowercase before removing duplicates
df['tweet'] = df['tweet'].str.lower()

# Remove duplicates based on the 'tweet' column
df = df.drop_duplicates(subset='tweet')

# Reset index after removing duplicates
df.reset_index(drop=True, inplace=True)

df.shape
```

```
(4347, 3)
```

```
# Combine labeled and unlabeled data
combined_data = pd.concat([labeled_data, df])

# 4347 unlabeled data + 85 labeled data
combined_data.shape
```

```
(4432, 4)
```

```
# Function to clean combined data

# Clean the 'tweet' column
combined_data['tweet'] = combined_data['tweet'].apply(clean_text)

combined_data.head()
```

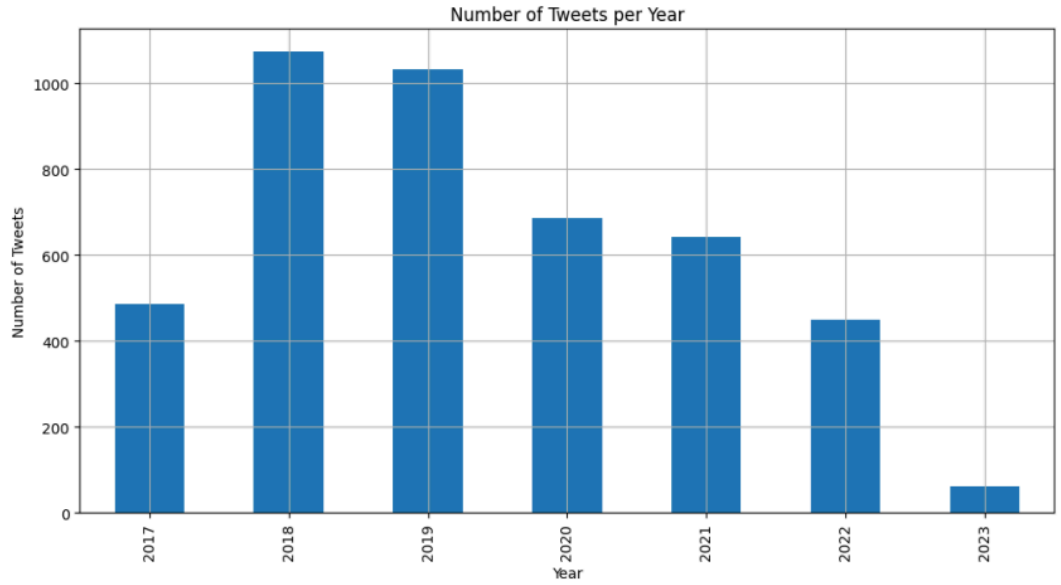
	Date	location	tweet	stigma
0	9/5/2017	London	i think i have ocd	1.0
1	4/10/2018	Courtenay, BC	mental health matters lets end the stigma and ...	0.0
2	8/5/2017	Warwickshire, UK	you are literally psychotic	1.0
3	9/10/2018	Calgary	please continue to join us in recognizing and ...	0.0
4	10/10/2018	Scotland	we are to be signatories of the charter for we...	0.0

Text Analysis to visualize the most frequent words in the tweets

```
# Combine all tweets into a single string
all_tweets_text = ' '.join(combined_data['tweet'])

# Generate word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(all_tweets_text)

# Display the word cloud using matplotlib
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Time Series Analysis number of tweets per year

```
# Convert 'Date' column to datetime format
combined_data['Date'] = pd.to_datetime(combined_data['Date'], format='%d/%m/%Y')

# Set the 'Date' column as the index
combined_data.set_index('Date', inplace=True)

# Resample the data to get the count of tweets per year
tweets_per_year = combined_data.resample('Y').size()

# Extract the year from the index
tweets_per_year.index = tweets_per_year.index.year

# Plot the time series
plt.figure(figsize=(12, 6))
tweets_per_year.plot(kind='bar', title='Number of Tweets per Year')
plt.xlabel('Year')
plt.ylabel('Number of Tweets')
plt.grid(True)
plt.show()
```

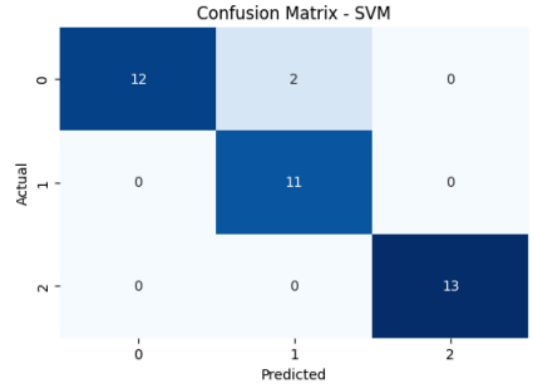


Semi supervised learning

```
# Preprocessing and feature extraction
tfidf_vectorizer = TfidfVectorizer(max_features=1000)
X_tfidf = tfidf_vectorizer.fit_transform(combined_data['tweet'])


# Split data back into labeled and unlabeled
X_labeled = X_tfidf[:len(labeled_data)]
X_unlabeled = X_tfidf[len(labeled_data):]


# For imbalance label in labeled data
# Oversample the labeled data
smk = SMOTETomek(random_state=0)
X_labeled_resampled, y_labeled_resampled = smk.fit_resample(X_labeled, labeled_data['stigma'])



# Print class distribution after oversampling
print("Class distribution after SMOTETomek:")
print(y_labeled_resampled.value_counts())

     Class distribution after SMOTETomek:
     1    62
     0    62
     2    62
     Name: stigma, dtype: int64


# Train-test split for oversampled labeled data
X_train, X_test, y_train, y_test = train_test_split(X_labeled_resampled, y_labeled_resampled, test_size=0.2, random_state=42)


# Function to plot confusion matrix
def plot_confusion_matrix(model_name, y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    accuracy = accuracy_score(y_true, y_pred)

    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
    plt.title(f'Confusion Matrix - {model_name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.text(1.1, 0.5, f'Accuracy: {accuracy:.2f}', transform=plt.gca().transAxes, verticalalignment='center')
    plt.show()
```

Support Vector Machine (SVM) Algorithm

```
# Train and evaluate SVM
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_predictions)

# Plot confusion matrix for SVM
plot_confusion_matrix('SVM', y_test, svm_predictions)
svm_report = classification_report(y_test, svm_predictions)
print(svm_report)
```

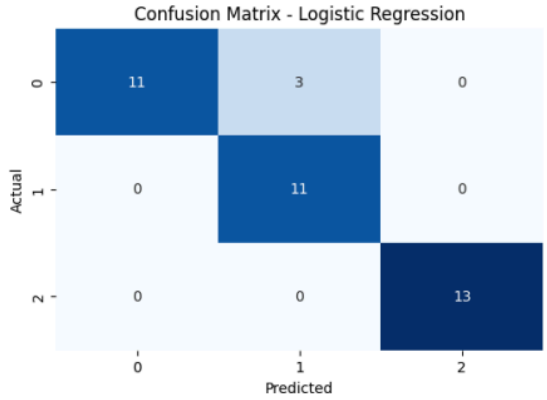


Accuracy: 0.95

```
              precision    recall  f1-score   support

           0       1.00      0.86      0.92        14
           1       0.85      1.00      0.92        11
           2       1.00      1.00      1.00        13

    accuracy                           0.95        38
   macro avg       0.95      0.95      0.95        38
weighted avg       0.96      0.95      0.95        38
```

Logistic Regression Algorithm

```
# Train and evaluate Logistic Regression
lr_model = LogisticRegression(random_state=42)
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)
lr_accuracy = accuracy_score(y_test, lr_predictions)

# Plot confusion matrix for Logistic Regression
plot_confusion_matrix('Logistic Regression', y_test, lr_predictions)
lr_report = classification_report(y_test, lr_predictions)
print(lr_report)
```

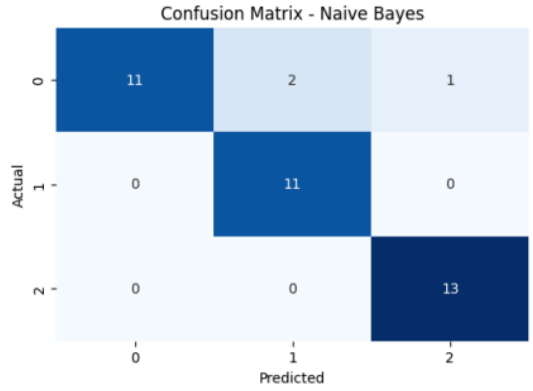


Accuracy: 0.9

	precision	recall	f1-score	support
0	1.00	0.79	0.88	14
1	0.79	1.00	0.88	11
2	1.00	1.00	1.00	13
accuracy			0.92	38
macro avg	0.93	0.93	0.92	38
weighted avg	0.94	0.92	0.92	38

Naive Bayes Algorithm

```
# Train and evaluate Naive Bayes
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
nb_predictions = nb_model.predict(X_test)
nb_accuracy = accuracy_score(y_test, nb_predictions)

# Plot confusion matrix for Naive Bayes
plot_confusion_matrix('Naive Bayes', y_test, nb_predictions)
nb_report = classification_report(y_test, nb_predictions)
print(nb_report)
```

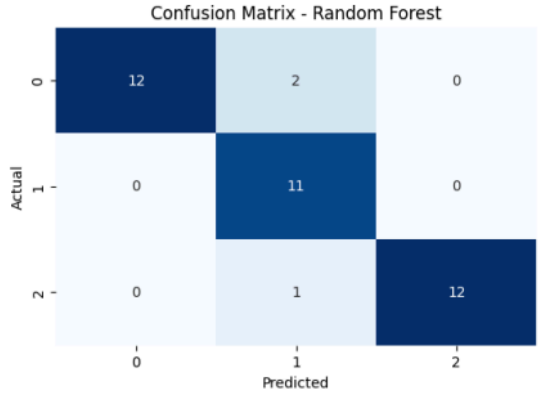


Accuracy: 0.92

	precision	recall	f1-score	support
0	1.00	0.79	0.88	14
1	0.85	1.00	0.92	11
2	0.93	1.00	0.96	13
accuracy			0.92	38
macro avg	0.92	0.93	0.92	38
weighted avg	0.93	0.92	0.92	38

Random Forest Algorithm

```
# Train and evaluate Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)

# Plot confusion matrix for Random Forest
plot_confusion_matrix('Random Forest', y_test, rf_predictions)
rf_report = classification_report(y_test, rf_predictions)
print(rf_report)
```



Accuracy: 0.92

```
              precision    recall  f1-score   support

           0       1.00      0.86      0.92        14
           1       0.79      1.00      0.88        11
           2       1.00      0.92      0.96        13

    accuracy                           0.92        38
   macro avg       0.93      0.93      0.92        38
weighted avg       0.94      0.92      0.92        38
```

```
# Create a DataFrame with algorithm names and accuracies
results_df = pd.DataFrame({
    'Algorithm': ['SVM', 'Logistic Regression', 'Naive Bayes', 'Random Forest'],
    'Accuracy': [svm_accuracy, lr_accuracy, nb_accuracy, rf_accuracy]
})

# Sort the DataFrame by Accuracy in ascending order
results_df.sort_values(by='Accuracy', ascending=False,ignore_index=True)
```

	Algorithm	Accuracy
0	SVM	0.947368
1	Logistic Regression	0.921053
2	Naive Bayes	0.921053
3	Random Forest	0.921053

Predict stigma using SVM

```
# df is unlabel data
svm_model.fit(X_labeled_resampled, y_labeled_resampled)

# Predict sentiment on unlabeled data
X_unlabeled_predictions = svm_model.predict(X_unlabeled)

# Add the predicted labels to the unlabeled data
df['predicted_stigma'] = X_unlabeled_predictions

# Combine predicted labels with location data
stigma_location_data = df[['Date', 'location', 'tweet', 'predicted_stigma']].copy()

# Filter tweets with stigma
stigma_tweets = stigma_location_data[stigma_location_data['predicted_stigma'] == 1]
```

```
stigma_tweets.shape
```

```
(573, 4)
```

```
stigma_tweets.head()
```

	Date	location	tweet	predicted_stigma
0	12/10/2022	UK	as its id urge anyone to try and be a bit more...	1
1	14/10/2021	1884 Market St San Francisco	this week is ocd awareness week is not all abo...	1
2	12/10/2021	Warren, Ohio	youve probably heard someone say im sooooo ocd...	1

Getting latitude and longitude

```
[ ] !pip install opencage
```

```
Requirement already satisfied: opencage in /usr/local/lib/python3.10/dist-packages (2.4.0)
Requirement already satisfied: Requests>=2.31.0 in /usr/local/lib/python3.10/dist-packages (from opencage) (2.31.0)
Requirement already satisfied: backoff>=2.2.1 in /usr/local/lib/python3.10/dist-packages (from opencage) (2.2.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from Requests>=2.31.0->opencage) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from Requests>=2.31.0->opencage) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from Requests>=2.31.0->opencage) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from Requests>=2.31.0->opencage) (2023.11.17)
```

```
[ ] from opencage.geocoder import OpenCageGeocode

    api_key = '0d4c4d3b9cff4e09a705c5c53195e8b2'
    geocoder = OpenCageGeocode(api_key)
```

```
[ ] # Function to check if location information is empty
    def is_location_empty(result):
        return result is None or len(result) == 0 or 'geometry' not in result[0]

    # Create empty columns for latitude, longitude, and country
    stigma_tweets['latitude'] = None
    stigma_tweets['longitude'] = None
    stigma_tweets['country'] = None

    # Iterate over DataFrame rows and add latitude, longitude, and country
    for index, row in stigma_tweets.iterrows():
        location = row['location']

        # Query OpenCage Geocoding API to obtain location information
        results = geocoder.geocode(location)

        if not is_location_empty(results):
            latitude = results[0]['geometry']['lat']
            longitude = results[0]['geometry']['lng']
            country = results[0]['components']['country']

            # Add latitude, longitude, and country to DataFrame
            stigma_tweets.at[index, 'latitude'] = latitude
            stigma_tweets.at[index, 'longitude'] = longitude
            stigma_tweets.at[index, 'country'] = country
        else:
            # If location information is empty, remove the row
            stigma_tweets = stigma_tweets.drop(index)
```

```
[ ] <ipython-input-43-18fd17d6508b>:6: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
      stigma_tweets['latitude'] = None
    <ipython-input-43-18fd17d6508b>:7: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
      stigma_tweets['longitude'] = None
    <ipython-input-43-18fd17d6508b>:8: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
      stigma_tweets['country'] = None
```

```
[ ] stigma_tweets.shape

    (534, 7)
```

```
[ ] stigma_tweets.head(10)
```

	Date	location	tweet	predicted_stigma	latitude	longitude	country
0	12/10/2022	UK	as its id urge anyone to try and be a bit more...	1	54.702354	-3.276575	United Kingdom
1	14/10/2021	1884 Market St San Francisco	this week is ocd awareness week is not all abo...	1	37.771059	-122.424836	United States
2	12/10/2021	Warren, Ohio	youve probably heard someone say im sooooo ocd...	1	39.420398	-84.180897	United States
3	11/10/2021	Everywhere	in honor of and the start of here is me the gu...	1	39.626226	19.903602	Greece
6	16/10/2020	gorseinon, wales, uk	ocd is horrific to live with and while creatin...	1	51.66938	-4.041592	United Kingdom
9	13/10/2020	Canada	this week is ocd awareness week among many oth...	1	61.066692	-107.991707	Canada
12	17/10/2019	Pennsylvania, USA	its ocd awareness week and my boy has had the ...	1	40.969989	-77.727883	United States
13	16/10/2019	United Kingdom	you cant have ocd you dont wash your hands an ...	1	54.702354	-3.276575	United Kingdom
14	16/10/2019	England	many people still think being ocd means being ...	1	52.531021	-1.264906	United Kingdom
15	15/10/2019	Los Angeles, CA	liking things to be neat is not ocd	1	34.053691	-118.242766	United States

```
[ ] # Save the DataFrame with latitude, longitude, and country
    stigma_tweets.to_csv('stigma_coords_with_country.csv', index=False)
```

2. Website Implementation for Dashboard Coding

```
from flask import Flask, render_template
import pandas as pd
import plotly.express as px
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from io import BytesIO
import base64
from textblob import TextBlob  # Import TextBlob for sentiment
analysis

app = Flask(__name__)

# Read your dataset
stigma_tweets = pd.read_csv('stigma_coords_with_country.csv')

# Handle variations in country names
stigma_tweets['country'] = stigma_tweets['country'].replace({
    'United States of America': 'United States',
    'USA': 'United States'
})
```

```
# Define a list of available Plotly colorscales
available_color_scales = ['aggrnyl', 'agsunset', 'algae', 'amp',
'armyrose', 'balance', 'blackbody', 'bluered', 'blues', 'blugrn',
'bluyl', 'brbg',
                          'brwnyl', 'bugn', 'bupu', 'burg',
'burgyl', 'cividis', 'curl', 'darkmint', 'deep', 'delta', 'dense',
'earth', 'edge', 'electric',
                          'emrld', 'fall', 'geyser', 'gnbu',
'gray', 'greens', 'greys', 'haline', 'hot', 'hsv', 'ice',
'icefire', 'inferno', 'jet',
                          'magenta', 'magma', 'matter', 'mint',
'mrybm', 'mygbm', 'oranges', 'orrd', 'oryel', 'oxy', 'peach',
'phase', 'picnic', 'pinkyl',
                          'piyg', 'plasma', 'plotly3',
'portland', 'prgn', 'pubu', 'pubugn', 'puor', 'purd', 'purp',
'purples', 'purpor', 'rainbow', 'rdbu',
                          'rdgy', 'rdpu', 'rdylbu', 'rdylgn',
'redor', 'reds', 'solar', 'spectral', 'speed', 'sunset',
'sunsetdark', 'teal', 'tealgrn',
                          'tealrose', 'tempo', 'temps',
'thermal', 'tropic', 'turbid', 'turbo', 'twilight', 'viridis',
'ylgn', 'ylgnbu', 'ylorbr', 'ylorrd']

# Route for the main dashboard
@app.route('/')
def dashboard():
    # Convert 'Date' column to datetime format
    stigma_tweets['Date'] = pd.to_datetime(stigma_tweets['Date'],
format='%d/%m/%Y')

    # Set the 'Date' column as the index
    stigma_tweets.set_index('Date', inplace=True)

    # Resample the data to get the count of tweets per year
    tweets_per_year = stigma_tweets.resample('Y').size()

    # Extract the year from the index
    tweets_per_year.index = tweets_per_year.index.year

    # Group by country and calculate stigma count
    stigma_counts =
```

```
stigma_tweets.groupby('country')['predicted_stigma'].sum().reset_i
ndex()

    # Perform sentiment analysis using TextBlob
    stigma_tweets['sentiment'] =
stigma_tweets['tweet'].apply(lambda x:
TextBlob(x).sentiment.polarity)

    # Filter tweets with negative sentiment
    negative_tweets = stigma_tweets[stigma_tweets['sentiment'] <
0]

    # Filter tweets with positive sentiment
    positive_tweets = stigma_tweets[stigma_tweets['sentiment'] >
0]

    # Generate Word Clouds for negative and positive sentiment
words
    wordcloud_img_negative =
generate_wordcloud(negative_tweets['tweet'], "Negative Sentiment
Tweets", '#FF9AA2')
    wordcloud_img_positive =
generate_wordcloud(positive_tweets['tweet'], "Positive Sentiment
Tweets", '#FFD700')

    # Choropleth Map
    fig_map = px.choropleth(
        stigma_counts,
        locations="country",
        locationmode="country names",
        color="predicted_stigma",
        color_continuous_scale=available_color_scales[1],  # Set
the desired color scale
        labels={'predicted_stigma': 'Stigma Count'},
    )

    # Set the size of the choropleth map
    fig_map.update_layout(
        geo=dict(
            showframe=False,
            showcoastlines=False,
            projection_type='equirectangular',
```

```
            bgcolor='rgba(255, 255, 255, 0.0)',
        ),
        margin=dict(l=0, r=0, t=0, b=0),
        height=550  # Set the desired height
    )

    # Bar Plot
    fig_bar = px.bar(
        stigma_counts,
        x="country",
        y="predicted_stigma",
        title="Stigma Count by Country",
        labels={'predicted_stigma': 'Stigma'},
        height=700,  # Set the desired height
        category_orders={"country":
stigma_counts.sort_values('predicted_stigma',
ascending=False)['country']},
        color_discrete_sequence=['#4B0082'] * len(stigma_counts)
# Set the desired color for the bar chart
    )

    # Convert the plots to HTML
    plot_divs = [graph.to_html(full_html=False) for graph in
[fig_map, fig_bar]]

    return render_template('index.html', plot_divs=plot_divs,
wordcloud_img_negative=wordcloud_img_negative,
wordcloud_img_positive=wordcloud_img_positive)

def generate_wordcloud(text, title, color):
    wordcloud = WordCloud(width=700, height=400,
background_color='#faf8f9').generate(' '.join(text))
    plt.figure(figsize=(7, 4))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f"Frequent Word in {title}", fontsize=16)
    plt.tight_layout()

    # Save the word cloud image
    img_buf = BytesIO()
    plt.savefig(img_buf, format='png', bbox_inches='tight',
pad_inches=0.1)
```

```
    img_buf.seek(0)

    # Encode the image to base64
    img_encoded = base64.b64encode(img_buf.read()).decode('utf-8')

    return f'{img_encoded}'

if __name__ == '__main__':
    app.run(debug=True)
```

3. **Index.html code**

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Mental Health Stigma Dashboard</title>
    <script
src="https://cdn.plot.ly/plotly-latest.min.js"></script>

    <!-- Your custom styles -->
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #faf8f9;
            margin: 0;
            padding: 0;
        }

        header {
            background-color: #483d41;
            color: #ffffff;
            padding: 10px;
            text-align: center;
        }

        main {
```

```
            padding: 20px;
        }

        #choropleth-plot {
            width: 100%;
            height: 400px;
            margin-bottom: 20px;
        }

        #wordcloud-tweet-container {
            display: flex;
            justify-content: space-between;
            width: 100%;
            margin-bottom: 20px;
        }

        #wordcloud{
            width: 48%; /* Adjust the width as needed */
        }

        #wordcloud img {
            width: 100%;
            height: auto;
        }

        /* #tweet-per-year {
            height: 400px;
        } */

        footer {
            background-color: #483d41;
            color: #ffffff;
            text-align: center;
            padding: 10px;
            position: fixed;
            bottom: 0;
            width: 100%;
        }
    </style>
</head>

<body>
```

```
    <header>
        <h1>Mental Health Stigma Dashboard</h1>
    </header>

    <main>
        {% for plot_div in plot_divs %}
        {{ plot_div | safe }}
        {% endfor %}

        <!-- Word Cloud Container -->
        <div id="wordcloud-tweet-container">
            <div id="wordcloud">
                <img src="{{ wordcloud_img_negative }}" alt="Word
Cloud for Negative Sentiment">
            </div>

            <div id="wordcloud">
                <img src="{{ wordcloud_img_positive }}" alt="Word
Cloud for Positive Sentiment">
            </div>
        </div>
    </main>

    <footer>
        &copy; Mental Health Stigma
    </footer>

</body>

</html>
```