Words count: 2484 Formulas: 2 Figures: 5 Tables: 10

Quantifying Psycholinguistic and Mood Instability in Twitter

[redacted][1]

[1]University of Melbourne, Australia;

Abstract

The high prevalence of mental disorders has highlighted the importance of improving intervention strategies to early recognise the onset of mental disorders. As social media begins to emerge and people share their personal struggles online, researchers innovate in quantifying textual features to infer mental health conditions. This study aims to investigate the capabilities of textual analysis on social media to characterise people with mental disorders. The study utilised a large historical Twitter-STMHD dataset with eight disorders and a control group. We examined psycholinguistic with LIWC and temporal analysis with sentiment analysis to explore the distinguishing features. We also explored predictive analysis, achieving the best performance of 65% F1-score for binary classification and 19% in multi-class classification.

Introduction

Suicide as one of the leading causes of death among young people is often associated with the presence of mental disorders[1]. Approximately 970 million people globally suffered from at least one mental disorder before COVID-19. The COVID-19 pandemic further exacerbated mental health issues, particularly depression and anxiety. Despite the increase in mental health issue rates, more than 75% of individuals with mental disorders remain untreated[1]. One of the intervention problems stems from challenges in the early recognition of symptoms, including indeterminate symptoms, lack of awareness, and limited resources. There is indeed a need to improve the awareness of early symptoms for effective intervention.

Concurrently, social media enables people to share their personal experiences worldwide, including mental health struggles. Social media, with its nature as a historical, large-scale, and expressive dataset, offers valuable insights into mental health conditions. Previous research demonstrated the effectiveness of extracting textual features from social media to address mental health conditions. The research showed significant differences in linguistic style between individuals with and without mental disorders when expressing themselves online[2–6]. Further analysis revealed indications of mood instability and distinctive levels of affective in individuals with mental disorders[2,7,8]. Additionally, studies utilising social features, such as daily frequency and interactions among users, offered valuable insights[5,9]. By leveraging text processing, social media holds promise for mental health prevention and intervention efforts.

This study aims to explore the capabilities of text processing to distinguish people with and without mental disorders in social media context. We utilised the Twitter-STHMD dataset[2] and implemented the psycholinguistics analysis to infer linguistic style. We also extended the temporal analysis to explore mood instability. Finally, we combined the analysis into a set of features and inferred prediction with machine learning. However, it is important to note that the primary objective of this study was not to develop a diagnostic tool. Instead, our focus was on designing a supporting tool that could aid in characterising individuals experiencing the onset of mental disorders.

Related Work

Numerous studies have focused on quantifying linguistic analysis in social media to assess mental health conditions. Studies were conducted on platforms such as Reddit[3,4,10–13], Facebook[7], and Twitter[2,5,8,9]. Practically, Reddit dataset offered extensive textual analysis due to the presence of longer text. Analysing topics and linguistic style is naturally as advantageous in this platform[3,4,10]. Meanwhile, studies with Twitter datasets demonstrated the benefits of temporal analysis, such as mood instability[8] and daily activities pattern[5,8,9]. The datasets were typically large, consisting of 30K users classified into mental disorders sufferers and control group. While mood instability was often examined in the context of borderline personality and bipolar disorder [7,8], many studies encompassed a range of other disorders.

One notable dataset was the Social Media and Mental Health Dataset (SHMD) for Reddit [4], covering conversations from 2006 to 2017. The dataset consisted of conversations from 360K users in nine subreddits. Additionally, there were other well-known datasets covering older and shorter timelines [5,10]. Lastly, SelfReported Temporally-Contextual Mental Health Diagnosis Dataset (STMHD) for Twitter [2] offered high-quality assurance through hand annotation and validation. This dataset, spanning from 2017 to 2021, captured tweets both before and after the COVID-19 pandemic.

Psycholinguistic analysis, specifically vocabulary analysis, was utilised to examine the emotional, cognitive, and structural components in individuals' writing [2–4,7,9,10]. Researchers employed tools like Linguistic Inquiry and Word Count (LIWC) to examine linguistic patterns. Regarding emotions, the control group tended to exhibit less intense expressions of anger and sadness and more prominent positive affects [2]. Other studies showed higher presence of pronouns in people with mental disorders [2,3]. Researchers also discovered that individuals with mental disorders tended to display less proficiency in grammar and syntax expressions and talked less about life-related concerns, such as money and home, than control group [2,4,7]. These findings highlight the importance of psycholinguistic analysis in understanding mental health conditions.

Temporal analysis has gained popularity in textual analysis conducted on social media. The studies ranged from mood instability [3,8,11] as well as engagement and interaction [5,9]. Researchers employed sentiment polarity as a key feature to detect mood instability in bipolar and borderline personality [8]. They aligned sentiment timelines to demonstrate the periodic mental health status. However, the research itself provided no statement about the performance of this method. Additionally, other studies demonstrated sentiment analysis to predict mental health status in non-temporal settings [8,11]. These findings underscore the potential of sentiment analysis in capturing mood patterns.

Some studies have also employed classification techniques to identify mental health conditions in social media. Traditional classification methods achieved approximately 72% F1-score in detecting depression [9]. Meanwhile, multilabel classification to detect multiple disorders reached 40-50% accuracy [4–6,10]. Advanced models such as convolutional neural networks have shown improved performance with 70% accuracy [14].

Methods

In this section, we outline our approach to collect and analyse data. We provide the research methodology in detail, including information on the data sources, analysis, and model design. This section ensures transparency and reproducibility of the research process.

Data sources

We utilised the Social Media and Mental Health Dataset (STMHD) [2]. The data was collected from Twitter and focused on eight mental disorders. The dataset consisted of 26K users identified as having one of the disorders stated in Table 1. For convenience, we named these groups as the condition groups. Additionally, there were 8K control users who were least likely to have a disorder. Figure 1 yielded there were users belonging to more than one disorder. The dataset contained user profiles and their tweets spanning from January 2017 to May 2021, covering period before and after COVID-19. The dataset was gathered through self-reported diagnosis data anchor with specific patterns, e.g. "diagnosed with <disorder name>". False anchor tweets were removed through two-step processes. First, 60% of the data underwent hand annotation and were validated by clinical psychologists. The remaining 40% of the data was filtered through pattern-matching derived from hand annotation. Dataset statistics are available in Table 1.

Privacy and Ethic Statement

The dataset was publicly available on the Zenodo platform [15]. The study adhered to ethical considerations to ensure individuals' privacy [2]. The dataset consisted of public, anonymised, and de-identified information. This

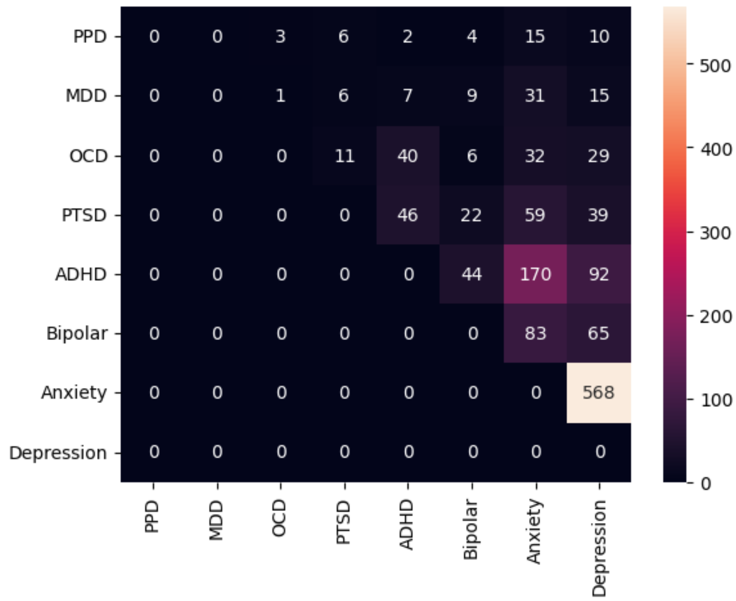

Figure 1. Commorbidity in dataset

Table 1. Statistics of dataset

Group	#Users	Status ratio	Favourites ratio	Followers ratio	Friends ratio
Neg (control group)	8199	25321 ± 66344	21763 ± 51291	818 ± 1006	844 ± 999
ADHD (attention deficit hyperactivity disorder)	8095	19060 ± 32750	34310 ± 60030	782 ± 1009	750 ± 930
Depression	6803	23177 ± 38986	35062 ± 57404	708 ± 936	774 ± 937
Anxiety	4843	20479 ± 32222	28832 ± 50205	810 ± 1029	905 ± 1085
PTSD (post traumatic stress disorder)	3414	21603 ± 43321	38871 ± 62137	648 ± 875	807 ± 922
Bipolar disorder	1651	20500 ± 33312	27995 ± 49069	757 ± 981	896 ± 1059
OCD (obsessive compulsive disorder)	1325	19180 ± 34714	33533 ± 52270	716 ± 918	800 ± 941
MDD (major depressive disorder)	325	27775 ± 57409	31266 ± 52082	732 ± 943	741 ± 881
PPD (postpartum depression)	247	21235 ± 37713	21683 ± 44739	743 ± 974	794 ± 991

study involved no direct interactions or interventions with human subjects. Therefore, obtaining informed consent was not applicable. The risk associated with this research was minimal.

Psycholinguistic Analysis

We conducted a closed-vocabulary analysis to examine the linguistic features between each condition group and the control group. We utilised the well-established lexicon, LIWC (version 15)[16], which categorises vocabularies into psychological and grammatical aspects. A range of aspects were explored, including grammatical, affective, cognitive, and more. Additionally, we included categories describing daily lifestyle and personal concerns, such as money, religion, and biological processes. Table 2 summarises the categories we used. To quantify the psycholinguistic feature, we calculated the proportion of words belonging to each category aggregated by each user. Through Formula 1, we obtained the empirical distribution of users in each group. We compared the distribution in each condition group and the control group through an independent sample t-test.

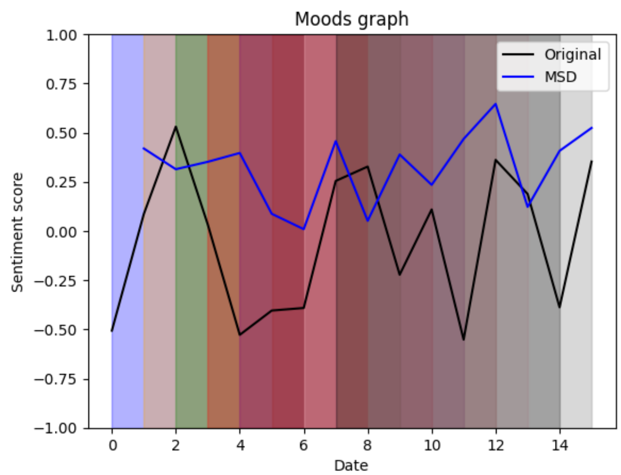

Figure 2. Illustration of moving standard deviation

$$F_{n_u}(x^c) = \frac{\Sigma_i^{n_u} 1_i^c}{n_u} \quad \forall c \in C, \forall u \in U \tag{1}$$

$$1_i^c : \text{indicator the } i\text{-th tokens is in lexicon category } c \tag{2}$$

$$n_u : \text{total number of tokens generated by user } u \tag{3}$$

Mood Instability

As the moods reflect the affective state, we attempted to quantify moods with sentiment polarity analysis. In this study, we defined good moods as positive sentiments while bad moods as negative sentiments. We aimed to capture the presence of mood instability in condition groups by assuming the control group has more stable moods. Additionally, we extended the analysis to further investigate decreased mental health conditions in the control group before and after COVID-19[17]. We performed sentiment analysis using Valence Aware Dictionary and sEntiment Reasoner (VADER) algorithm[18] which known for its effectiveness on social media.

For each tweet, we obtained a compound score of sentiment analysis. Daily moods were the average aggregation of scores for each user within a given day. We divided the time series into windows of dates with size $k = 2$. To capture instability variations, we utilised the value of moving standard deviation (MSD). For each window, we calculate its standard deviation and derived the maximum and average MSD in each user.

To further explore moods tendencies, we defined moods-ratio and moods-combo[8]. Moods-ratio, divided into positive and negative-ratio, is the proportion of days with good and bad moods respectively (See Formula 4 and 5). We also implemented the concept of moods-combo to capture the onset of hypomania/mania and depression. Moods-combo was divided into the positive-combo and negative-combo as illustrated in Figure 3. It refers to the frequency with which individuals experienced at least six consecutive days of good or bad moods. Our expectation was that condition groups would exhibit more negative tendencies and more negative-combo as indications of mental health struggles.

$$\text{positive-ratio} : F_{n_u}(x \geq 0) = \frac{\Sigma_i^{n_u} 1_{x_i \geq 0}}{n_u} \quad \forall c \in C, \forall u \in U \tag{4}$$

$$\text{negative-ratio:} F_{n_u}(x < 0) = \frac{\Sigma_i^{n_u} 1_{x_i < 0}}{n_u} \quad \forall c \in C, \forall u \in U \tag{5}$$

Table 2. LIWC 2015 chosen categories

Category	Examples
Grammars	
Personal pronouns	I, them, her
Impersonal pronouns	it, it's, those
Conjunctions	and, but, whereas
Comparisons	greater, best, after
Common adjectives	free, happy, long
Common adverbs	very, really
Affective prorcesses	
Positive emotion	love, nice, sweet
Negative emotion	hurt, ugly, nasty
Cognitive processes	
Insight	think, know
Causation	because, effect
Discrepancy	should, would
Tentative	maybe, perhaps
Certainty	always, never
Differentiation	hasn't, but, else
Perceptual processes	
See	view, saw, seen
Hear	listen, hearing
Feel	feels, touch
Time orientations	
Past focus	ago, did, talked
Present focus	today, is, now
Future focus	may, will, soon
Biological processes	
Body	cheek, hands, spit
Health	clinic, flu, pill
Sexual	horny, love, incest
Ingestion	dish, eat, pizza
Social processes	
Family	daughter, dad, aunt
Friends	buddy, neighbor
Personal concerns	
Work	job, majors, xerox
Leisure	cook, chat, movie
Home	kitchen, landlord
Money	audit, cash, owe
Drives	
Affiliation	ally, friend, social
Achievement	win, success, better
Power	superior, bully
Reward	take, prize, benefit
Risk	danger, doubt

Classification

We constructed two scenarios of classification. The first scenario involved binary classification, combining condition groups into one label ($y = 1$) while the control group has the opposite ($y = 0$). Secondly,

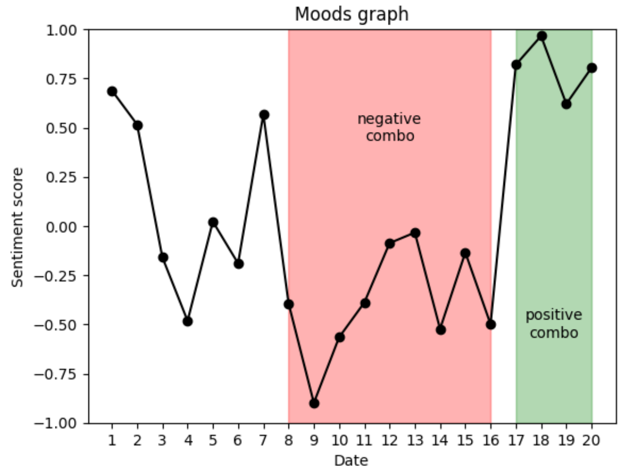

Figure 3. Illustration of moods-combo features

we performed multi-class classification, treating each condition group as separate labels. We utilised the calculation from previous sections as features. Additionally, we incorporated metadata aggregation, including average counts of likes, quotes, and more. We divided the data into training and testing sets in an 80:20 ratio, stratified on condition group labels. For classification, we employed traditional classifiers including *logistic regression*, *naïve bayes*, *support vector machine*, and *random forest*. Prior to the testing phase, we tuned the classifiers using cross-validation. Finally, we evaluated the performance of the classifiers using accuracy, precision, recall, and F1-score metrics (See Table 3).

Table 3. Confusion matrix

		Ground-truth		
		Positive	Negative	Total
Predicted	Positive	TP	FN	$TP+FN$
	Negative	FP	TN	$FP+TN$
	Total	$TP+FP$	$FN+TN$	N

1. accuracy: $\frac{TP+TN}{N}$
2. precision: $\frac{TP}{TP+FP}$
3. recall: $\frac{TP}{TP+FN}$
4. F1-score: $2\frac{precision \times recall}{precision+recall}$

Results

In this section, we present the result of our analysis. Our analysis revealed compelling evidence for the effectiveness of our textual extraction. Statistical significance is available in details.

Features Analysis

Initially, we examined the metadata of the dataset, as presented in Table 4. The green-coloured cell indicated higher mean values within the condition group. From the analysis, we observed significant increases in the number of replies and likes for tweets in condition groups. Conversely, the number of mentioned users decreased, with the exception of PTSD group. There were as well fewer interactions in tweets from bipolar group and more media attached from ADHD group. Lastly, we duplicated findings of hourly posting frequency

Table 4. Statistical significance of metadata feature

	PPD	MDD	OCD	PTSD	ADHD	Bipolar	Anxiety	Depression
Likes count	-	***	***	***	***	-	***	***
Quote count	-	*	-	-	-	**	-	-
Reply count	***	***	***	***	***	***	***	***
Retweet count	-	*	-	-	-	.	-	-
Total mentioned users	-	***	***	***	*	**	***	***
Number of medias attached -	-	-	-	-	***	***	-	-

*** < 0.001   ** < 0.01   * < 0.05   . < 0.1

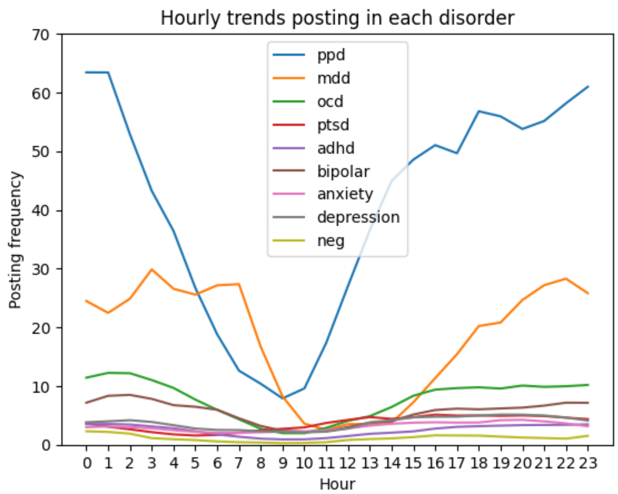

Figure 4. Hourly posting trends

from the original research as shown in Figure 4. It exhibited a pattern of sleep disturbance in condition groups which tended to post midnight.

The result of our psycholinguistic analysis is shown in Table 5. It indicated significant increases in most vocabulary categories for PPD, MDD, OCD, PTSD, and bipolar groups. Conversely, a decreasing trend was observed in the anxiety group for most categories. We also noticed that there were slight decreases in the vocabularies of depression groups. Surprisingly, the patterns of significance were relatively constant across categories and there were only slight differences between ADHD and control groups.

Table 6 present the analysis for mood instability features. As expected, the results yielded fluctuating moods in condition groups as evidenced by higher values of MSD. Additionally, there were significant negative tendencies in tweets made by depression, bipolar, PTSD, and ADHD groups. We also noticed a pattern of hypomania/mania in people with bipolar, anxiety, ADHD, PPD, and MDD groups. Interestingly, the positive-ratio increased across all condition groups and most of them showed fewer signs of negative-combo (depression signs), contrary to our initial expectations. Additionally, we discovered that tweets posted after the COVID-19 period exhibited a lower positive tendency.

Classification

Table 7 and 8 yielded the performance of our classification models. As past papers indicated, the performance of binary classification was higher than multi-class classification. In terms of binary classification, the performance yielded 70-80% accuracy and 45-65% F1-score. Meanwhile, multi-class performed at 25-39% accuracy and 15-19% F1-score. Overall, *random forest* stood out with the best performance of 65% for binary

Table 5. Statistical significance of psycholinguistic analysis

	PPD	MDD	OCD	PTSD	ADHD	Bipolar	Anxiety	Depression
Grammars								
Personal pronouns	***	***	***	***	-	***	**	-
i	***	***	***	***	*	***	-	-
we	***	***	***	*	-	-	**	*
you	***	***	***	***	-	***	***	-
shehe	***	***	***	**	-	.	**	-
they	***	***	***	***	-	*	**	-
Impersonal pronouns	***	***	***	***	-	**	**	-
Common adverbs	***	***	***	***	-	***	**	-
Conjunctions	***	***	***	***	-	***	*	-
Common Adjectives	***	***	***	***	-	**	***	-
Comparisons	***	***	***	***	-	*	***	-
Affective processes	***	***	***	***	-	***	**	-
Positive emotion	***	***	***	***	-	***	***	-
Negative emotion	***	***	***	***	-	**	***	-
Anxiety	***	***	***	***	-	***	*	-
Anger	***	***	***	***	-	**	*	-
Sad	***	***	***	***	-	***	*	-
Social processes	***	***	***	***	-	**	***	-
Family	***	***	***	***	-	**	***	-
Friends	***	***	***	***	-	**	***	-
Cognitive processes	***	***	***	***	-	**	**	-
Insight	***	***	***	***	-	**	**	-
Causation	***	***	***	***	-	**	**	-
Discrepancy	***	***	***	***	-	**	**	-
Tentative	***	***	***	***	-	**	**	-
Certainty	***	***	***	***	-	**	***	-
Differentiation	***	***	***	***	-	**	**	-
Perceptual processes	***	***	***	***	-	**	***	-
See	***	***	***	-	-	*	***	*
Hear	***	***	***	***	-	***	**	-
Feel	***	***	***	***	-	***	*	-
Biological processes	***	***	***	***	-	***	*	-
Body	***	***	***	***	-	***	**	-
Health	***	***	***	***	-	***	-	-
Sexual	***	***	***	***	*	***	-	.
Ingestion	***	***	***	-	-	-	*	-
Drives	***	***	***	**	-	*	***	.
Affiliation	***	***	***	***	-	*	***	.
Achievement	***	***	***	*	-	-	***	*
Power	***	***	***	**	-	-	***	.
Reward	***	***	***	*	-	.	***	*
Risk	***	***	***	***	-	*	**	-
Time orientation								
Past focus	***	***	***	***	-	**	**	-
Present focus	***	***	***	***	-	**	***	-
Future focus	***	***	***	***	-	**	***	-
Personal concerns								
Work	***	***	***	*	-	-	**	*
Leisure	***	***	***	.	-	-	***	**
Home	***	***	***	.	-	-	**	-
Money	***	***	***	-	.	-	**	.
Religion	***	***	***	*	-	**	*	-
Death	***	***	***	*	-	-	***	-

Table 6. Statistical significance of mood instability feature

	PPD	MDD	OCD	PTSD	ADHD	Bipolar	Anxiety	Depression	After covid
Max MSD	***	***	**	***	***	***	***	***	-
Average MSD	***	*	***	***	***	***	***	***	-
Positive ratio	***	*	***	***	***	***	***	***	***
Negative ratio	-	-	-	***	***	*	-	***	-
Positive combo	***	***	-	.	***	***	***	-	***
Negative combo	*	-	***	***	***	-	***	***	***

*** < 0.001 ** < 0.01 * < 0.05 . < 0.1

classification and 19% for multi-class classification. In multi-class scenario, we discovered that our model tended to classify individuals into PPD and Bipolar groups as evidenced by the higher predicted values in both true positives and false positives (See Figure 5).

Table 7. Binary classification result

	Acc	P	R	F1
Logistic Regression	0.77	0.67	0.51	0.46
Naive Bayes	0.72	0.55	0.53	0.53
Support Vector Machine	0.77	0.64	0.51	0.45
Random Forest	0.80	0.75	0.63	0.65

Table 8. Multi-class classification result

	Acc	P	R	F1
Logistic Regression	0.35	0.29	0.19	0.17
Naive Bayes	0.25	0.2	0.18	0.15
Support Vector Machine	0.36	0.26	0.2	0.18
Random Forest	0.39	0.26	0.21	0.19

Discussion

In most condition groups' interaction patterns, we noticed higher numbers of replies and likes, coupled with lower numbers of mentioned users. We suggested the tweets might generate significant engagement among users without direct interactions with the specific user. However, the finding was aligned with previous studies stating fewer interactions among users[5,9]. Conversely, the higher numbers of mentioned users and lower counts of followers in PTSD tweets might indicate intense engagement. It is possibly associated with specific attachment styles seen in individuals with PTSD, particularly those with Complex PTSD (C-PTSD) which have not been explored before.

For mood instability features, the depression-related groups exhibited decreasing trend in negative-combo values with higher negative-ratio. This suggested long exposure to depressive episodes with less frequent mood-shifting and denser episodes. This finding indicated the persistence and chronicity of depression in certain individuals. As for higher positive-ratio in condition groups, we found this insight interesting. We supposed individuals might present a curated version of themselves, including portraying positive images or as a way of coping mechanism. However, further sampling might be needed to explore the patterns. Additionally, the analysis of post COVID-19 tweets suggested there was an indication of worse mood stability in the control group as shown in previous studies[17]. Overall, the features extend the result of previous research[8], proving that the features showed valuable indicators.

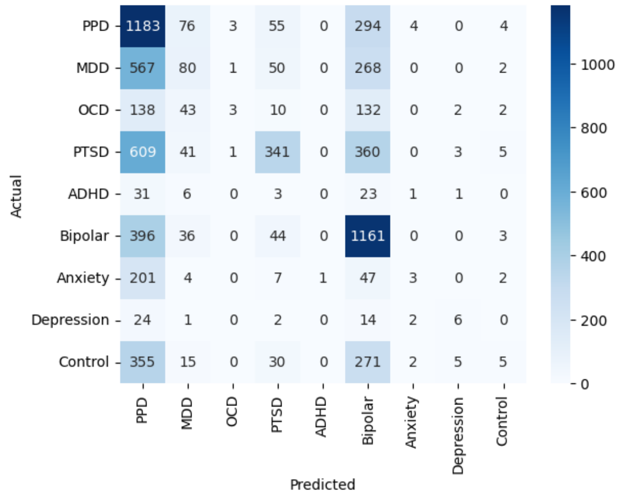

Figure 5. Confusion matrix of multi-class classification

Our analysis of psycholinguistics features yielded anomalous results. Further analysis indicated a limitation in our study, as unforeseen flaws in the code were discovered due to parallel data processing on ADHD, depression, and control groups. However, we could not revise prior to submitting the paper due to time constraints. To further validate the features and provide better results, we experimented classifications without psycholinguistic features. Table 9 and 10 show the result of the experiment. We discovered that the performance provided no significant differences. We supposed the psycholinguistic features gave slight contributions to the performance due to flaws.

Table 9. Binary classification result revised

	Acc	P	R	F1
Logistic Regression	0.77	0.68	0.51	0.46
Naive Bayes	0.72	0.55	0.53	0.53
Support Vector Machine	0.77	0.66	0.51	0.46
Random Forest	0.80	0.75	0.64	0.66

Table 10. Multi-class classification result revised

	Acc	P	R	F1
Logistic Regression	0.35	0.18	0.18	0.15
Naive Bayes	0.24	0.15	0.14	0.13
Support Vector Machine	0.36	0.17	0.18	0.15
Random Forest	0.40	0.21	0.2	0.17

Conclusion

Overall, we examined text processing techniques on the Twitter-STHMD dataset to distinguish individuals with mental disorders and control groups. We performed psycholinguistic and temporal analysis which were combined for predictive machine learning. Our predictive model employing random forest achieved the highest F1-score of 65% for binary classification and 19% for multi-class.

In future work, some areas warrant further exploration. Firstly, it is worth revisiting the psycholinguistic analysis and fixing associated flaws. Secondly, the prominence of time series moods would be valuable to investigate the aligning time series in each group. This would provide insights into temporal dynamics and patterns and further enable early recognition of symptoms based on mood fluctuations. Additionally, incorporating POS tagging and syntactic analysis tools might offer a more detailed and quantitative understanding of linguistic analysis. Furthermore, topical modeling techniques could be employed to uncover latent themes and topics within the dataset.

References

1. Organization WH. World mental health report: transforming mental health for all. World Health Organization; 2022.

2. Suhavi, Singh AK, Arora U, Shrivastava S, Singh A, Shah RR, et al. Twitter-STMHD: An Extensive User-Level Database of Multiple Mental Health Disorders. Proceedings of the International AAAI Conference on Web and Social Media. 2022 May;16:1182-91. Available from: https://ojs.aaai.org/index.php/ICWSM/article/view/19368.

3. Gkotsis G, Oellrich A, Hubbard T, Dobson R, Liakata M, Velupillai S, et al. The language of mental health problems in social media. In: Proceedings of the Third Workshop on Computational Lingusitics and Clinical Psychology. San Diego, CA, USA: Association for Computational Linguistics; 2016. p. 63-73. Available from: http://aclweb.org/anthology/W16-0307.

4. Cohan A, Desmet B, Yates A, Soldaini L, MacAvaney S, Goharian N. SMHD: A Large-Scale Resource for Exploring Online Language Usage for Multiple Mental Health Conditions. arXiv; 2018. ArXiv:1806.05258 [cs]. Available from: http://arxiv.org/abs/1806.05258.

5. Coppersmith G, Dredze M, Harman C. Quantifying Mental Health Signals in Twitter. In: Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality. Baltimore, Maryland, USA: Association for Computational Linguistics; 2014. p. 51-60. Available from: http://aclweb.org/anthology/W14-3207.

6. Gaur M, Kursuncu U, Alambo A, Sheth A, Daniulaityte R, Thirunarayan K, et al. "Let Me Tell You About Your Mental Health!": Contextualized Classification of Reddit Posts to DSM-5 for Web-based Intervention. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. Torino Italy: ACM; 2018. p. 753-62. Available from: https://dl.acm.org/doi/10.1145/3269206.3271732.

7. Saha K, Chan L, De Barbaro K, Abowd GD, De Choudhury M. Inferring Mood Instability on Social Media by Leveraging Ecological Momentary Assessments. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies. 2017 Sep;1(3):1-27. Available from: https://dl.acm.org/doi/10.1145/3130960.

8. Saravia E, Chang CH, De Lorenzo RJ, Chen YS. MIDAS: Mental illness detection and analysis via social media. In: 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). San Francisco, CA, USA: IEEE; 2016. p. 1418-21. Available from: http://ieeexplore.ieee.org/document/7752434/.

9. De Choudhury M, Gamon M, Counts S, Horvitz E. Predicting Depression via Social Media. Proceedings of the International AAAI Conference on Web and Social Media. 2021 Aug;7(1):128-37. Available from: https://ojs.aaai.org/index.php/ICWSM/article/view/14432.

10. Coppersmith G, Dredze M, Harman C, Hollingshead K. From ADHD to SAD: Analyzing the Language of Mental Health on Twitter through Self-Reported Diagnoses. In: Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality. Denver, Colorado: Association for Computational Linguistics; 2015. p. 1-10. Available from: http://aclweb.org/anthology/W15-1201.

11. Song H, You J. Feature Attention Network: Interpretable Depression Detection from Social Media. Information and Computation. 2018.

12. De Choudhury M, Kiciman E, Dredze M, Coppersmith G, Kumar M. Discovering Shifts to Suicidal Ideation from Mental Health Content in Social Media. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. San Jose California USA: ACM; 2016. p. 2098-110. Available from: https://dl.acm.org/doi/10.1145/2858036.2858207.

13. Gkotsis G, Oellrich A, Velupillai S, Liakata M, Hubbard TJP, Dobson RJB, et al. Characterisation of mental health conditions in social media using Informed Deep Learning. Scientific Reports. 2017 Mar;7(1):45141. Available from: https://www.nature.com/articles/srep45141.

14. Ive J, Gkotsis G, Dutta R, Stewart R, Velupillai S. Hierarchical neural model with attention mechanisms for the classification of social media text related to mental health. In: Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic. New Orleans, LA: Association for Computational Linguistics; 2018. p. 69-77. Available from: http://aclweb.org/anthology/W18-0607.

15. Suhavi, Singh A, Arora U, Shrivastava S, Singh A, Shah RR, et al.. Twitter Self-reported Temporally-contextual Mental Health Diagnosis (Twitter-STMHD) Dataset. Zenodo; 2022. Available from: https://doi.org/10.5281/zenodo.5854911.

16. Pennebaker J, Boyd R, Jordan K, Blackburn K. The development and psychometric properties of LIWC2015. University of Texas at Austin; 2015.

17. Xie Y, Xu E, Al-Aly Z. Risks of mental health outcomes in people with covid-19: cohort study. BMJ. 2022;376. Available from: https://www.bmj.com/content/376/bmj-2021-068993.

18. Hutto C, Gilbert E. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. Proceedings of the International AAAI Conference on Web and Social Media. 2014 May;8(1):216-25. Available from: https://ojs.aaai.org/index.php/ICWSM/article/view/14550.

Appendices

Code: Transforming data from JSON to CSV

```python
import os
import sys
import json
import zipfile
import pandas as pd

# Get filename
file_arg = sys.argv[1]
dis = file_arg.split("/")[-1][:-4]
user_col = ["id","creation_timestamp","description","favourites_count","followers_count",
            "friends_count", "geo_tag","banner_link","display_image_link","status_count",
            "verified_check", "anchor_tweet", "anchor_tweet_date","disorder"]
tweet_col = ["disorder_flag", "text", "conversation_id", "tweet_id", "language",
             "likes_count", "quote_count", "reply_count", "retweet_count", "source_name",
             "timestamp_tweet", "mentionedUsers", "media", "user_id", "covid"]

# Database retrieve
if os.path.exists(f'users_{dis}.csv'):
    users_data = pd.read_csv(f'users_{dis}.csv', dtype=str, lineterminator=";")
    users = users_data[users_data.disorder == dis]['id'].values
else:
    users_data = pd.DataFrame(columns=user_col)
    users = []
print("Numbers of transformed users:", len(users))

# Filter directories (only the one which haven't processed)
covid_date = '2020-03-11'
archive = zipfile.ZipFile(f'{file_arg}', 'r')
listdirs = archive.namelist()
dirs = []
for s in listdirs:
    s_split = s.split("/")
    if len(s_split) == int(sys.argv[2]) and s.endswith("/") and s_split[-2] \
       not in users: dirs.append(s)
print("Total users left:", len(dirs))

# Function to save file, concat data only at the end
def save_file(users, tweets):
    users_data = pd.concat(users)
    tweets_data = pd.concat(tweets)
    print(len(tweets_data['user_id'][tweets_data['user_id'].isnull()]))

    if os.path.exists(f'users_{dis}.csv'):
        users_data.to_csv(f'users_{dis}.csv', mode='a', header=False,
                          index=False, lineterminator=";")
        tweets_data.to_csv(f'tweets_{dis}.csv', mode='a', header=False,
                           index=False, lineterminator=";")
    else:
        users_data.to_csv(f'users_{dis}.csv', index=False, lineterminator=";")
        tweets_data.to_csv(f'tweets_{dis}.csv', index=False, lineterminator=";")


# Run transformation
total_user = 0
while total_user != len(dirs):
    i = 0
    users_new = []
    tweets_new = []
    try:
        for dir in dirs[total_user:]:
            namespace = dir.split("/")
            user_id = namespace[-2]
            i += 1
```

```
            total_user += 1
            print("=", end="", flush=True)

            if dis == 'neg':
                anchor = {}
            else:
                anchor = json.load(archive.open(dir + 'anchor_tweet.json'))

            user = json.load(archive.open(dir + 'user.json'))
            tweets = json.load(archive.open(dir + 'tweets.json'))

            user_data = {**user, **anchor}
            user_df = pd.DataFrame(user_data, index=[0])
            user_df['disorder'] = namespace[5]

            tweet_temp = []
            for keydate, tweet in tweets.items():
                tweet_df = pd.DataFrame(tweet)
                tweet_df['user_id'] = user_id
                tweet_df['covid'] = 1 if keydate > covid_date else 0
                tweet_df['media'] = tweet_df['media'] \
                    .apply(lambda x: str([m["type"] for m in x]))
                tweet_df = tweet_df.reindex(columns=tweet_col).fillna("")
                tweet_temp.append(tweet_df)

            user_df = user_df.reindex(columns=user_col).fillna("")
            users_new.append(user_df)
            tweets_new += tweet_temp

            if i == 100:
                print()
                print("Taking a break...")
                break

        save_file(users_new, tweets_new)
        print("Let's do it again!", total_user, "out of", len(dirs))

    # Enabling intermittent transformation with keyboard interruption
    except KeyboardInterrupt:
        print("Interrupt execution...")
        save_file(users_new, tweets_new)
        break
```

Code: Scratches

```
# -*- coding: utf-8 -*-
"""Scratch.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/12M8xV3f0uf01sASpLj1JPVd2aLurkRs6
"""

import matplotlib.pyplot as plt
import random

# Generate 20 random x-values
x = range(1, 21)

# Generate random y-values between -1 and 1
random.seed(0)
y = [random.uniform(-1, 1) for _ in range(20)]

```

```
# Ensure at least 4 consecutive positive values
positive_start = random.randint(0, 17)
for i in range(positive_start, positive_start + 4):
    y[i] = random.uniform(0, 1)

# Ensure at least 7 consecutive negative values
negative_start = random.randint(0, 13)
for i in range(negative_start, negative_start + 7):
    y[i] = random.uniform(-1, 0)

# Plot the graph
plt.title('Moods graph')
plt.plot(x, y, marker='o', linestyle='-', color='black')

# Set y-axis limits between -1 and 1
plt.ylim(-1, 1)
plt.xticks(x)

# Add labels
plt.xlabel('Date')
plt.ylabel('Sentiment score')

plt.axvspan(8, 16, color='red', alpha=0.3)
plt.axvspan(17, 20, color='green', alpha=0.3)
plt.text(12, 0.5, 'negative\ncombo', color='black', ha='center',
         va='center', fontsize=10)
plt.text(18.5, -0.5, 'positive\ncombo', color='black', ha='center',
         va='center', fontsize=10)

# Show the plot
plt.show()

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Generate x values from 0 to 100
x = np.arange(0, 16)

# Generate random y values between -1 and 1
y = np.random.uniform(-0.6, 0.6, size=len(x))

# Set the sliding window size

# Define the colors for the sliding window
colors = ['blue', 'orange', 'green', 'red', 'purple', 'brown', 'pink', 'black', 'gray']

# Plot the original data
plt.plot(x, y, color='black', label='Original')

# Plot the sliding window averages with different colors
rolling_std = pd.Series(y).rolling(window=2).std()
plt.plot(x, y, color='black', label='Original')

# Set the y-axis limit
plt.ylim(-1, 1)

# Add legend and labels
plt.title('Moods graph')
plt.xlabel('Date')
plt.ylabel('Sentiment score')
k = 7
for i in range(len(colors)):
    plt.axvspan(i, i+k, color=colors[i], alpha=0.3)

# Show the plot
plt.show()
```

Code: Analysing user profiles

```python
# -*- coding: utf-8 -*-
"""User Analysis.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/18nDqWoHrTeIQ0EgApw3RE7kzczfOXm9m
"""

DIR = "/content/drive/MyDrive/College (Master)/Semester 3/ \
      COMP90090 - Natural Language Processing for Health/Assignment/Assignment 3/"

from google.colab import drive
drive.mount('/content/drive')

import pandas as pd

DISORDERS = ['ppd', 'mdd', 'ocd', 'ptsd', 'adhd',
             'bipolar', 'anxiety', 'depression']
LABEL = DISORDERS + ['neg']

users = []

for dis in LABEL:
    user = pd.read_csv(f"{DIR}/Dataset/users/users_{dis}.csv",
                       dtype=str, lineterminator=";")
    users.append(user)

df = pd.concat(users).reset_index(drop=True)
df = df[~df.disorder.isnull()]
df['favourites_count'] = df['favourites_count'].astype(int)
df['followers_count'] = df['followers_count'].astype(int)
df['friends_count'] = df['friends_count'].astype(int)
df.head()

"""# Disorder counts"""

# Statistics per user
df['disorder'].value_counts()

combined_df = df[['id', 'disorder']].groupby('id')['disorder'] \
                 .apply(lambda x: ','.join(map(str, x))).reset_index()
commorbid = combined_df[combined_df['disorder'].str.contains(',')]['disorder']
commorbid = commorbid.value_counts().reset_index(name="Frequency")
commorbid.head()

for i, disorder in enumerate(commorbid['index']):
    dsplit = disorder.split(",")
    if len(dsplit) > 2:
        for j in range(len(dsplit)):
            if j == (len(dsplit) - 1): break
            for d in dsplit[j+1:]:
                comm_id = commorbid[commorbid['index'] == f"{dsplit[j]},{d}"].index
                commorbid.loc[comm_id, 'Frequency'] += commorbid.loc[i, 'Frequency']
        commorbid.drop(i, inplace=True)

commorbid.head()

import matplotlib.pyplot as plt
import seaborn as sns

commorbid['Condition1'], commorbid['Condition2'] = commorbid['index'].str.split(',', 1).str
pivot_table = commorbid.pivot_table(index='Condition1', columns='Condition2',
```

```
                                    values='Frequency', fill_value=0)
pivot_table = pivot_table.reindex(index=DISORDERS, columns=DISORDERS) \
                .fillna(0).astype("int64")
ax = sns.heatmap(pivot_table, annot=True, fmt='g')

ax.set_xticklabels(["PPD", "MDD", "OCD", "PTSD", "ADHD",
                    "Bipolar", "Anxiety", "Depression"])
ax.set_yticklabels(["PPD", "MDD", "OCD", "PTSD", "ADHD",
                    "Bipolar", "Anxiety", "Depression"])
ax.set_xlabel(None)
ax.set_ylabel(None)
plt.show()

"""# Favorites, Followers, Friends"""

favourites = []
followers = []
friends = []
status = []

for l in LABEL:
    status.append(df[df.disorder == l].status_count.astype(int))
    favourites.append(df[df.disorder == l].favourites_count.astype(int))
    followers.append(df[df.disorder == l].followers_count.astype(int))
    friends.append(df[df.disorder == l].friends_count.astype(int))

stat = []

for i, l in enumerate(['neg', 'adhd', 'depression', 'anxiety', 'ptsd',
                       'bipolar', 'ocd', 'mdd', 'ppd']):
    print("=================== Statistics of group", l)
    stat.append([])
    print("Status count", round(status[i].mean()), "+-", round(status[i].std()))
    print("Favorites", round(favourites[i].mean()), "+-", round(favourites[i].std()))
    print("Followers", round(followers[i].mean()), "+-", round(followers[i].std()))
    print("Friends", round(friends[i].mean()), "+-", round(friends[i].std()))
```

Code: Extracting metadata analysis

```
# -*- coding: utf-8 -*-
"""Metadata Analysis.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1Q8glz1mULTdqhp40OeGup1jz4fZn-1b2
"""

DIR = "/content/drive/MyDrive/College (Master)/Semester 3/ \
       COMP90090 - Natural Language Processing for Health/Assignment/Assignment 3/"

import os
import nltk
import pandas as pd
import scipy.stats as stats
from pandas.errors import ParserError
from nltk.sentiment.vader import SentimentIntensityAnalyzer

import warnings
warnings.filterwarnings("ignore")

nltk.download('vader_lexicon')

DISORDERS = ['ppd', 'mdd', 'ocd', 'ptsd', 'adhd',
```

```python
             'bipolar', 'anxiety', 'depression']
LABEL = DISORDERS + ['neg']

"""# Feature Extraction"""

# Store data about user_id, tweet_id, date, sentiment, disorder, covid
for l in LABEL[7:]:
    print(f"Working on label {l}")
    reader = pd.read_csv(f"{DIR}/Dataset/tweets/tweets_{l}.csv", dtype=str,
                         chunksize=100000, on_bad_lines='warn', lineterminator=";")

    i = 0
    for r in reader:
        i += 1
        df = r[['tweet_id', 'user_id', 'covid', 'disorder_flag',
                'likes_count', 'quote_count', 'reply_count', 'retweet_count']]
        df['mentioned_users'] = r['mentionedUsers']
                                    .apply(lambda x:
                                            0 if x == "[]" or
                                                x == ""
                                            else len(x.split(",")))
        df['medias'] = r['media']
                         .apply(lambda x:
                                0 if x == "[]" or
                                     x == ""
                                else len(x.split(",")))
        print(f"Chunk number {i}")

        if not os.path.exists(f"{DIR}/Dataset/MA/{l}.csv"):
            df.to_csv(f"{DIR}/Dataset/MA/{l}.csv")
        else:
            df.to_csv(f"{DIR}/Dataset/MA/{l}.csv", mode='a', header=False)

    print(" FINISHED")

"""# Analisis"""

columns = ['likes_count', 'quote_count', 'reply_count',
           'retweet_count', 'mentioned_users', 'medias']

dfs = []
hourly = []
types = dict(zip(columns,[int for i in range(len(columns))]))

for l in LABEL:
    print(f"Working on label {l}")

    reader = pd.read_csv(f"{DIR}/Dataset/MA/{l}.csv",
                         dtype=types, chunksize=1000000,
                         on_bad_lines='warn', index_col=0)

    temp = [[] for i in range(len(columns))]

    i = 0
    append_df = None
    for r in reader:
        i += 1
        # Moving chunks
        if i > 1: r = pd.concat([append_df, r], ignore_index=True, axis=0)

        # Get dataframe without last user
        last_user = r.user_id.unique()[-1]
        mask = (r.user_id == last_user)
        append_df = r[mask]
        r = r[~mask]

        # Aggregate values
        aggs = dict(zip(columns,['mean' for i in range(len(columns))]))
```

```
        result = r.groupby(['user_id']).agg(aggs)
        for j in range(len(columns)):
            temp[j] += result[columns[j]].tolist()

        print("Chunk number", i)

    df = pd.DataFrame([list(i) for i in zip(*temp)], columns=columns)
    dfs.append(df)
    print("FINISHED")

summ = pd.DataFrame(index=columns, columns=DISORDERS)

for j, col in enumerate(columns):
    for i in range(len(DISORDERS)):
        t_statistic1, p_value1 = stats.ttest_ind(dfs[i][col].values,
                                                 dfs[8][col].values)
        value = "***" if p_value1 < 0.001 else "**"
                      if p_value1 < 0.01 else "*"
                      if p_value1 <0.05 else "."
                      if p_value1 < 0.1 else ""
        color = "r" if round(t_statistic1,10) < 0 else "g"
                    if round(t_statistic1,10) > 0 else ""
        summ.at[col, DISORDERS[i]] = value + color

summ

summ.to_latex()

for i, df in enumerate(dfs):
    df.to_csv(f"{DIR}/Dataset/MA/{LABEL[i]}_extracted.csv")
```

Code: Extracting mood instability

```
# -*- coding: utf-8 -*-
"""Mood Instability.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1KO9pnqF3Qy4pM-kl6ITtBzYmNJ3z-jwQ
"""

DIR = "/content/drive/MyDrive/College (Master)/Semester 3/ \
       COMP90090 - Natural Language Processing for Health/Assignment/Assignment 3/"

import nltk
import numpy as np
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

from itertools import groupby
from pandas.errors import ParserError
from nltk.sentiment.vader import SentimentIntensityAnalyzer


import warnings
warnings.filterwarnings("ignore")

nltk.download('vader_lexicon')

DISORDERS = ['ppd', 'mdd', 'ocd', 'ptsd', 'adhd',
```

```
              'bipolar', 'anxiety', 'depression']
LABEL = DISORDERS + ['neg']

"""# Feature Extraction"""

import os

# Store data about user_id, tweet_id, date, sentiment, disorder, covid
sid = SentimentIntensityAnalyzer()

for l in LABEL[8:]:
    print(f"Working on label {l}")
    reader = pd.read_csv(f"{DIR}/Dataset/tweets/tweets_{l}.csv", dtype=str,
                         chunksize=500000, on_bad_lines='warn', lineterminator=";")

    i = 0
    for r in reader:
        i += 1
        sentiments = []
        for text in r['text'].values:
            try:
                sentiments.append(sid.polarity_scores(text))
            except AttributeError:
                sentiments.append({'neg':0, 'neu':0, 'pos':0, 'compound':0})

        sent_df = pd.DataFrame(sentiments)
        df = r[['tweet_id', 'user_id', 'disorder_flag', 'timestamp_tweet']]
        df.loc[:,'neg'] = sent_df['neg'].values
        df.loc[:,'neu'] = sent_df['neu'].values
        df.loc[:,'pos'] = sent_df['pos'].values
        df.loc[:,'compound'] = sent_df['compound'].values
        print(f"Chunk number {i}")

        if not os.path.exists(f"{DIR}/Dataset/NEW/{l}.csv"):
            print("masuk")
            df.to_csv(f"{DIR}/Dataset/NEW/{l}.csv")
        else:
            df.to_csv(f"{DIR}/Dataset/NEW/{l}.csv", mode='a', header=False)


    print(" FINISHED")

"""# Exploratory"""

mi = pd.read_csv(DIR + 'Dataset/MI/ppd.csv', index_col=0)
mi['date'] = mi['timestamp_tweet'].str.split(" ").str[0]
mi.drop(['timestamp_tweet'], axis=1, inplace=True)
print(mi.shape)

mi['date'] = pd.to_datetime(mi['date'])

result_date = mi.groupby(['user_id', 'date']).agg({'compound': 'mean',
                                                   'neg': 'mean',
                                                   'neu': 'mean',
                                                   'pos': 'mean'})

print(result_date)

plt.figure(figsize=(15, 6))
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Time Series Data with Smoothing')
plt.ylim(-1,1)
test = []

# Define the start and end dates of the period to highlight
highlight_start = '2020-03-01'  # Example start date

```

```python
# Convert the highlight dates to matplotlib date format
highlight_start = mdates.datestr2num(highlight_start)

# Draw a rectangle to highlight the period

for user in mi['user_id'].unique()[:5]:
    comp = result_date.loc[user, 'compound']
    rolling_avg = comp.rolling(window=10).mean()
    rolling_std = comp.rolling(window=10).std()
    test.append(comp.values[~np.isnan(comp.values)])

    # plt.plot(comp.index, comp.values, color='blue')
    # plt.plot(comp.index, rolling_avg, color='green')
    plt.plot(comp.index, rolling_std)

plt.axvspan(highlight_start, plt.xlim()[1], facecolor='red', alpha=0.3, zorder=0)
# ccf = sm.tsa.stattools.ccf(test[0], test[1], adjusted=False)
# plt.plot(comp.index, ccf, label='Rolling Average (Window = 7)')
plt.show()

"""# Detailed Feature Extraction"""

features = ['max_msd', 'mean_msd',
            'positive_ratio', 'negative_ratio',
            'positive_combo', 'negative_combo']

columns = ['compound']

dfs = []
hourly_rate = np.zeros((len(LABEL), 24))

before_covid = []
after_covid = []
covid_date = pd.to_datetime('2020-03-11')

def extract_feature(compounds, temp):
    # Moving windows
    rolling_std = compounds.rolling(window=2).std()
    temp[0].append(np.max(rolling_std))
    temp[1].append(np.mean(rolling_std))

    goods = (compounds > 0)
    bads = (compounds < 0)
    # moods ratio
    positives = np.count_nonzero(goods)
    negatives = np.count_nonzero(bads)
    temp[2].append(positives / len(compounds))
    temp[3].append(negatives / len(compounds))

    # moods combo
    consecutive_positives = np.array([sum(1 for _ in group)
                                  for key, group in groupby(goods) if key == 1])
    consecutive_negatives = np.array([sum(1 for _ in group)
                                  for key, group in groupby(bads) if key == 1])
    temp[4].append((consecutive_positives >= 6).sum())
    temp[5].append((consecutive_negatives >= 6).sum())

    return temp

for l_i, l in enumerate(LABEL):
    print(f"Working on label {l}")
    reader = pd.read_csv(f"{DIR}/Dataset/MI/{l}.csv",
                         dtype=dict(zip(columns, [float for i in range(len(columns))])),
                         chunksize=500000, on_bad_lines='warn', index_col=0,
                         parse_dates=['timestamp_tweet'])

    if l == 'neg':
        temp_before_covid = [[] for i in range(len(features))]
```

```python
        temp_after_covid = [[] for i in range(len(features))]

    temp = [[] for i in range(len(features))]
    append_df = None
    i = 0
    total_user = 0
    for r in reader:
        i += 1
        # Moving chunks
        if i > 1: r = pd.concat([append_df, r], ignore_index=True, axis=0)

        # Get dataframe without last user
        users = r['user_id'].unique()
        last_user = users[-1]
        mask = (r.user_id == last_user)
        append_df = r[mask]
        r = r[~mask]
        users = users[:-1]

        # Convert timestamp
        r['date'] = r['timestamp_tweet'].dt.date
        r['hour'] = r['timestamp_tweet'].dt.hour
        r.drop(['timestamp_tweet'], axis=1, inplace=True)

        # Group by hours
        result_hour = r[['tweet_id', 'hour']].groupby(['hour']).count()
        if len(result_hour) == 0: continue
        hourly_rate[l_i,:] = result_hour.values.ravel()
        total_user += len(users)

        # Group by date
        aggs = dict(zip(columns,['mean' for i in range(len(columns))]))
        result_date = r.groupby(['user_id', 'date']).agg(aggs)
        df_before_covid = \
             result_date[result_date.index.get_level_values('date') < covid_date]
        df_after_covid =  \
             result_date[result_date.index.get_level_values('date') >= covid_date]

        # Extract features
        for u in users:
            compounds = result_date.loc[u, 'compound']
            temp = extract_feature(compounds, temp)

            # Extract data for covid analysis
            if l == 'neg' and
               u in df_before_covid.index.get_level_values('user_id') and
               u in df_after_covid.index.get_level_values('user_id'):
                temp_before_covid = extract_feature(df_before_covid.loc[u, 'compound'],
                                                    temp_before_covid)
                temp_after_covid = extract_feature(df_after_covid.loc[u, 'compound'],
                                                   temp_after_covid)

        print("Chunk number", i, total_user)

    hourly_rate[l_i,:] /= total_user
    df = pd.DataFrame([list(i) for i in zip(*temp)], columns=features)
    dfs.append(df)

    print("FINISHED")

for i, df in enumerate(dfs): df.to_csv(f"{DIR}/Dataset/MI/{LABEL[i]}_extracted.csv")

"""# Analysis

## Window-based feature, moods ratio, moods combo
"""

summ = pd.DataFrame(index=features, columns=DISORDERS)
```

```python

for col in features:
    for i in range(len(DISORDERS)):
        t_statistic1, p_value1 = stats.ttest_ind(dfs[i][col].values,
                                                 dfs[-1][col].values)
        value = "***" if p_value1 < 0.001 else "**"
                      if p_value1 < 0.01 else "*"
                      if p_value1 <0.05 else "."
                      if p_value1 < 0.1 else "-"
        color = "r" if round(t_statistic1,10) < 0 else "g"
                    if round(t_statistic1,10) > 0 else ""
        summ.at[col, DISORDERS[i]] = value + color

summ

# Covid analysis
df_before_covid = pd.DataFrame([list(i) for i in zip(*temp_before_covid)], columns=features)
df_after_covid = pd.DataFrame([list(i) for i in zip(*temp_after_covid)], columns=features)

for col in features:
    t_statistic1, p_value1 = stats.ttest_ind(df_after_covid[col].values,
                                             df_before_covid[col].values)
    value = "***" if p_value1 < 0.001 else "**"
                  if p_value1 < 0.01 else "*"
                  if p_value1 <0.05 else "."
                  if p_value1 < 0.1 else "-"
    color = "r" if round(t_statistic1,10) < 0 else "g"
                if round(t_statistic1,10) > 0 else ""
    summ.loc[col, 'after'] = value + color

summ

summ.to_latex()

"""## Hourly rate"""

for i in range(hourly_rate.shape[0]):
    plt.plot(hourly_rate[i], label=f'{LABEL[i]}')

# Set the plot title and labels
plt.title('Hourly trends posting in each disorder')
plt.ylim(0, 70)
plt.xticks(range(24))
plt.xlabel('Hour')
plt.ylabel('Posting frequency')

# Show the legend
plt.legend()

# Show the plot
plt.show()
```

Code: Extracting psycholinguistics analysis

```python
# -*- coding: utf-8 -*-
"""LIWC.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1aqAlaWpS-GNmXBszJ6acoqFSpAybdb5O
"""

!pip install liwc
```

```python

from google.colab import drive
drive.mount('/content/drive')

DIR = "/content/drive/MyDrive/College (Master)/Semester 3/ \
       COMP90090 - Natural Language Processing for Health/Assignment/Assignment 3/"

import os
import nltk
import liwc
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
from collections import Counter
from nltk.tokenize import word_tokenize

nltk.download('punkt')


import warnings
warnings.filterwarnings("ignore")

liwcPath = DIR + 'Code/LIWC2015_English_Flat.dic'
parse, category_names = liwc.load_token_parser(liwcPath)

category_used = ['function','ppron', 'i', 'we', 'you', 'shehe', 'they',
                 'ipron', 'adverb', 'conj', 'adj', 'compare',
                 'affect', 'posemo', 'negemo', 'anx', 'anger', 'sad',
                 'social', 'family', 'friend', 'female', 'male',
                 'cogproc', 'insight', 'cause', 'discrep', 'tentat', 'certain', 'differ',
                 'percept', 'see', 'hear', 'feel',
                 'bio', 'body', 'health', 'sexual', 'ingest',
                 'drives', 'affiliation', 'achieve', 'power', 'reward', 'risk',
                 'focuspast', 'focuspresent', 'focusfuture',
                 'work', 'leisure', 'home', 'money', 'relig', 'death']

DISORDERS = ['ppd', 'mdd', 'ocd', 'ptsd', 'adhd',
             'bipolar', 'anxiety', 'depression']
LABEL = DISORDERS + ['neg']

"""# Feature Extraction"""

# Store data about user_id, tweet_id, date, sentiment, disorder, covid
dfs = []

for l in ['adhd','neg']:
    print(f"Working on label {l}")
    reader = pd.read_csv(f"{DIR}/Dataset/tweets/tweets_{l}.csv", dtype=str,
                         chunksize=500000, on_bad_lines='warn', lineterminator=";")

    temp = [[] for i in range(len(category_used))]

    i = 0
    for r in reader:
        i += 1
        # if i < 101: continue
        # if i > 100: break
        df = r[['tweet_id', 'user_id']]

        counters = []
        r['text'] = r['text'].fillna("").str.lower()
        for j, text in enumerate(r['text']):
            try:
                tokens = word_tokenize(text)
                counter = Counter(category for token in tokens for category in parse(token))
                counters.append(counter)
            except:
                display(r.iloc[j,])
```

```
        counter_df = pd.DataFrame.from_dict(counters).fillna(0).astype(int)
        counter_df = counter_df.reindex(columns=category_used, fill_value=0)
        final_df = pd.concat([df, counter_df], axis=1)

        last_user = final_df.user_id.unique()[-1]
        mask = (final_df.user_id == last_user)
        append_df = final_df[mask]
        final_df = final_df[~mask]

        # Aggregate values
        result = final_df.groupby(['user_id']).agg(dict(zip(category_used,['mean' for i in range(len(catego
        for j in range(len(category_used)):
            temp[j] += result[category_used[j]].tolist()

        print("Chunk number", i)

    df = pd.DataFrame([list(i) for i in zip(*temp)], columns=category_used)
    dfs.append(df)
    print(" FINISHED")

"""# Analysis"""

columns = category_used
dfs = []

for l in LABEL:
    print(f"Working on label {l}")
    types = dict(zip(columns,['Int64' for i in range(len(columns))]))
    types['user_id'] = str
    reader = pd.read_csv(f"{DIR}/Dataset/LIWC/{l}.csv", dtype=types,
                         chunksize=500000, on_bad_lines='warn', index_col=0)

    temp = [[] for i in range(len(columns))]

    i = 0
    append_df = None
    for r in reader:
        i += 1
        # Moving chunks
        if i > 1: r = pd.concat([append_df, r], ignore_index=True, axis=0)
        # if i > 10: break

        # Get dataframe without last user
        r.dropna(subset=['user_id'], inplace=True)
        r.fillna(0, inplace=True)
        last_user = r.user_id.unique()[-1]
        mask = (r.user_id == last_user)
        append_df = r[mask]
        r = r[~mask]

        # Aggregate values
        result = r.groupby(['user_id'])
                  .agg(dict(zip(columns,['mean' for i in range(len(columns))])))
        for j in range(len(columns)):
            temp[j] += result[columns[j]].tolist()

        print("Chunk number", i)

    df = pd.DataFrame([list(i) for i in zip(*temp)], columns=columns)
    dfs.append(df)
    print("FINISHED")

for i, df in enumerate(dfs): df.to_csv(f"{DIR}/Dataset/LIWC/{LABEL[i]}_extracted.csv")

features = category_used
summ = pd.DataFrame(index=features, columns=DISORDERS)
```

```python
for col in features:
    for i in range(len(DISORDERS)):
        t_statistic1, p_value1 = stats.ttest_ind(dfs[i][col].values,
                                                 dfs[-1][col].values)
        value = "***" if p_value1 < 0.001 else "**"
                      if p_value1 < 0.01 else "*"
                      if p_value1 <0.05 else "."
                      if p_value1 < 0.1 else "-"
        color = "\cellcolor{red!25}"
                      if round(t_statistic1,10) < 0 else "\cellcolor{green!25}"
                      if round(t_statistic1,10) > 0 else ""
        summ.at[col, DISORDERS[i]] = color + value

summ

summ.to_latex()
```

Code: Model classificatiion

```python
# -*- coding: utf-8 -*-
"""Prediction Model Full.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1XzNifgy3T_go-5QPmG4fqsRZZqlIC2eZ
"""

DIR = "/content/drive/MyDrive/College (Master)/Semester 3/ \
       COMP90090 - Natural Language Processing for Health/Assignment/Assignment 3/"

from google.colab import drive
drive.mount('/content/drive')

import nltk
import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats as stats
import statsmodels.api as sm
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, roc_auc_score, confusion_matrix

import warnings
warnings.filterwarnings("ignore")

"""# Read Data"""

FEATURES = ["MI", "MA", "LIWC"]
DISORDERS = ['ppd', 'mdd', 'ocd', 'ptsd', 'adhd',
             'bipolar', 'anxiety', 'depression']
LABEL = DISORDERS + ['neg']

full = []
mlabel, label = [], []

```

```
for l in LABEL:
    ea_disorder = []
    for f in FEATURES:
        df = pd.read_csv(f"{DIR}Dataset/{f}/{l}_extracted.csv", index_col=0)
        ea_disorder.append(df)
    feat = pd.concat(ea_disorder, axis=1)
    feat['mlabel'] = l
    feat['label'] = 0 if l == 'neg' else 1
    full.append(feat)

"""# Preprocess"""

data = pd.concat(full, axis=0, ignore_index=True)
data.fillna(0, inplace=True)
data[data.isna().any(axis=1)]

train, test = train_test_split(data, stratify=data['mlabel'],
                               test_size=0.2, random_state=42)

X_train = train.drop(['mlabel', 'label'], axis=1)
y_train = train['label']
my_train = train['mlabel']

X_test = test.drop(['mlabel', 'label'], axis=1)
y_test = test['label']
my_test = test['mlabel']

"""# Train"""

def tune(model, param_grid, X_train, y_train):
    grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
    grid_search.fit(X_train, y_train)
    print('* Best hyperparameters:', grid_search.best_params_)
    print('* Best accuracy score:', grid_search.best_score_)

    return grid_search.best_estimator_

"""## Binary Classification"""

nb = MultinomialNB()
svm = LinearSVC()
rf = RandomForestClassifier()
lg = LogisticRegression()
models = [lg, nb, svm, rf]

param_grids = [
    {'C': [0.001, 0.01, 0.1, 1, 10]},
    {'alpha': [0.001, 0.01, 0.1, 1, 10]},
    {'C': [0.001, 0.01, 0.1, 1, 10]},
    {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20]}
]

bgrids = []
# Perform hyperparameter tuning for each model
for i, model in enumerate(models):
    param_grid = param_grids[i]
    print(model.__class__.__name__)

    # For common training data
    bgrids.append(tune(model, param_grid, X_train, y_train))

"""## Multilabel classification"""

nb = MultinomialNB()
svm = LinearSVC()
rf = RandomForestClassifier()
lg = LogisticRegression()
models = [lg, nb, svm, rf]
```

```python

param_grids = [
    {'C': [0.001, 0.01, 0.1, 1, 10], 'multi_class': ['multinomial']},
    {'alpha': [0.001, 0.01, 0.1, 1, 10]},
    {'C': [0.001, 0.01, 0.1, 1, 10], 'multi_class': ['ovr']},
    {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20]}
]

mgrids = []
# Perform hyperparameter tuning for each model
for i, model in enumerate(models):
    param_grid = param_grids[i]
    print(model.__class__.__name__)

    # For common training data
    mgrids.append(tune(model, param_grid, X_train, my_train))

"""# Evaluate"""

def evaluate(y, y_pred, y_pred_proba=None):
    report = classification_report(y, y_pred)
    print('Classification Report:')
    print(report)

    if y_pred_proba is not None:
        auroc = roc_auc_score(y, y_pred_proba)
        print('AUROC:', round(auroc, 4))
    print()

    return classification_report(y, y_pred, output_dict=True)

breport = pd.DataFrame(columns=['Acc', 'P', 'R', 'F1'],
                       index=['Logistic Regression', 'Naive Bayes',
                              'Support Vector Machine', 'Random Forest'])
mreport = pd.DataFrame(columns=['Acc', 'P', 'R', 'F1'],
                       index=['Logistic Regression', 'Naive Bayes',
                              'Support Vector Machine', 'Random Forest'])

"""## Binary classification"""

for i, model in enumerate(bgrids):
    print("============", model.__class__.__name__, "============")
    y_pred = model.predict(X_test)
    if hasattr(model, "predict_proba"):
        report = evaluate(y_test, y_pred, model.predict_proba(X_test)[::,1])
    else:
        report = evaluate(y_test, y_pred)

    breport.iloc[i,:] = [round(report['accuracy'],2),
                         round(report['macro avg']['precision'],2),
                         round(report['macro avg']['recall'],2),
                         round(report['macro avg']['f1-score'],2)]

breport

breport.to_latex()

"""## Multilabel classifiction"""

for i, model in enumerate(mgrids):
    print("============", model.__class__.__name__, "============")
    my_pred = model.predict(X_test)
    report = evaluate(my_test, my_pred)

    mreport.iloc[i,:] = [round(report['accuracy'],2),
                         round(report['macro avg']['precision'],2),
                         round(report['macro avg']['recall'],2),
                         round(report['macro avg']['f1-score'],2)]
```

```

mreport.to_latex()

# Show confustion matrix for highest accuracy score
classification_matrix = confusion_matrix(my_test, my_pred)
ax = sns.heatmap(classification_matrix, annot=True, cmap='Blues', fmt='d')

ax.set_xlabel('Predicted')
ax.set_ylabel('Actual')
ax.set_xticklabels(['PPD', 'MDD', 'OCD', 'PTSD', 'ADHD',
                    'Bipolar', 'Anxiety', 'Depression', 'Control'], rotation=90)
ax.set_yticklabels(['PPD', 'MDD', 'OCD', 'PTSD', 'ADHD',
                    'Bipolar', 'Anxiety', 'Depression', 'Control'], rotation=0)
```