Configuration Manual

Research Project
MSc in Data Analytics

Naval Suvarna
X18183654

School of Computing
National College of Ireland

Submitted to: Dr. Vladimir Milosavljevic

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Naval Suvarna

Student ID: X18183654

Programme: MSc In Data Analytics **Year:** 2019-2020

Module: Research Project

Supervisor: Dr. Vladimir Milosavljevic

Submission Due Date: 17/08/2020

Project Title: Prediction of Mental Health among Twitter users.

Word Count: 1663 **Page Count:** 12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Naval Suvarna……

Date: ……16/08/2020……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Naval Suvarna
X18183654

1.Introduction

The current research aims to predict the mental health condition of online users through data gathered from Twitter. Its main focus is the implementation of machine learning models on the twitter dataset and to recognize the most optimum model for this research. This document has been created in order to give details regarding system specification, research implementation and code structure**.**

2.System Specification

2.1 Hardware Specification

Item	Value
OS Name	Microsoft Windows 10 Home Single Language
Version	10.0.18362 Build 18362
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	DESKTOP-DU1TUS3
System Manufacturer	LENOVO
System Model	81DE
System Type	x64-based PC
System SKU	LENOVO_MT_81DE_BU_idea_FM_ideapad 330-15IKB
Processor	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, 1800 Mhz, 4 C...
BIOS Version/Date	LENOVO 8TCN54WW, 11-11-2019
SMBIOS Version	3.0
Embedded Controller V...	1.54
BIOS Mode	UEFI
BaseBoard Manufacturer	LENOVO
BaseBoard Product	LNVNB161216
BaseBoard Version	SDK0Q55722 WIN
Platform Role	Mobile
Secure Boot State	On
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\WINDOWS
System Directory	C:\WINDOWS\system32
Boot Device	\Device\HarddiskVolume2
Locale	United States
Hardware Abstraction L...	Version = "10.0.18362.752"
User Name	DESKTOP-DU1TUS3\Lenovo
Time Zone	GMT Daylight Time

Fig 1. Local Desktop Configuration

2.2 Software Specification

Programming language Python has been used as the primary language for implementing this research. The version used is 3.7

- **Anaconda**
  The software used for python's distribution was Anaconda[1] which is an open-source platform consisting of all tools required for machine learning. Detailed steps for downloading the platform can be found on the website.

- **Jupyter Notebook**
  The code editor required for programming was jupyter notebook. It is in-built in Anaconda and is easy to use with separate cells for individual code segments. It can directly be accessed from Anaconda's dashboard.

- **Project Setup**

1. Once Anaconda is installed, the home dashboard opens up with all the different tools available for use. The tools need to be downloaded only during the first time from the dashboard by simply clicking on the icon named Install. After installing the icon chances to Launch and the code editor is ready to use.
2. After the launch, a webpage opens up with different folders in your local machine. You can either select any folder or create a new one to store your code. After creating a new folder, a new python code editor can be created by clicking the New icon on the upper right part of the screen which gives you an option to launch a new Python 3 notebook. After clicking the option, the notebook is ready and the project can be started.

- **Installing dependencies**

1. After the project environment has been set up different dependencies need to be installed in anaconda such as the stopword library, the imblearn library and the ntlk toolkit. For this, the anaconda command prompt is used. It can be launch from the search window on the desktop. When the prompt appears, the following command needs to be typed.
   - conda install -c conda-forge stopwords[2]
   - conda install -c conda-forge imabalanced-learn[3]
   - conda install -c anaconda nltk[4]

3.Code Structure

The code structure should be neat, easily interpretable and well commented. It also needs to be re-usable and well-structured. Hence, care has been taken to meet all the mentioned criteria.

- **Folder**
  The folder contains the dataset and the jupyter notebook.

1. Dataset:
   It contains a CSV file of 1.6 million tweets containing different columns such as time, usernames and tweets.

2. Jupyter notebook
   It contains the code implementation of the research project along with the graph plots extracted from the dataset.

[1] https://www.anaconda.com/products/individual
[2] https://anaconda.org/conda-forge/stopwords
[3] https://anaconda.org/conda-forge/imbalanced-learn
[4] https://anaconda.org/conda-forge/nltk

4.Data

4.1 Data Insights

A single dataset has been chosen for this research and has been downloaded from Kaggle[5] repository in a CSV format. The structure of the data is as follows:

	Target	Id	Time	Query	Username	Tweets
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

Fig 2. Structure of Twitter Dataset

4.2 Data Cleaning

- The data contains a lot of irrelevant columns such as Id , Query and Username. They have been dropped from the set.
- Null values have been dropped.
- A function is developed to clear out symbols, numbers, and to convert all the tweets to lower case (A. & Sonawane, 2016).
- Function was created to remove the stop-words from the tweets and also to remove the 100 most common words from the text. The cleaned tweets are stored in a new column labelled “tweets-without-stopwords”

Data Cleaning

```
del df_clean["Query"]
for col in df_clean.columns:
    print(col)
```

```
Target
Time
Tweets
```

```
df_clean = df_clean[df_clean['Tweets'].notnull()]
```

```
df_clean.count()
```

```
Target    1599999
Time      1599999
Tweets    1599999
dtype: int64
```

```
def txt_clean(txt):
    t = txt
    t = re.sub(r'[^A-Za-z0-9\s]','',t) #removing symbols/special characters
    t = re.sub(r'[0-9]','',t) # removing numbers
    t = t.lower() # everything to lower case
    t = re.sub(r'[\n\t]',' ',t) # removing \n and \t
    t = " ".join(t.split()) # combine everything together
    return t
```

```
df_clean['Tweets']=df_clean['Tweets'].apply(txt_clean)
```

```
stop = stopwords.words('english')
```

```
# removing stop words from tweets
df_clean['tweet_without_stopwords'] = df_clean['Tweets'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
```

```
# removing the top 100 common words
freq = pd.Series(' '.join(df_clean['tweet_without_stopwords']).split()).value_counts()[:100]
freq = list(freq.index)
df_clean['tweet_without_stopwords'] = df_clean['tweet_without_stopwords'].apply(lambda x: " ".join(x for x in x.split() if x not
```

```
# removing the least 100 common words
freq = pd.Series(' '.join(df_clean['tweet_without_stopwords']).split()).value_counts()[-100:]
freq = list(freq.index)
df_clean['tweet_without_stopwords'] = df_clean['tweet_without_stopwords'].apply(lambda x: " ".join(x for x in x.split() if x not
```

Fig 3.Data Cleaning Steps

4.3 Data Pre-processing

- The data needs to be pre-processes in order to be interpretable by the machine learning models. Different features need to be clubbed together to form new variables. The steps carried out are detailed below.

[5] https://www.kaggle.com/

- The tweets in the new column are lemmatized. The hour and minute of each tweet are separated and stored separately.
- The time period from midnight to seven in the morning are labelled as “Critical” and the rest are labelled as “Regular” and stored in a new column “Time of day”.

Pre-processing

```
df_clean['tweet_without_stopwords'] = df_clean['tweet_without_stopwords'].apply(lambda x: " ".join([Word(word).lemmatize() for w
```

```
def cleanTime(text):
    text_time=text[11:19]


    return text_time

df_clean['OnlyTime']=df_clean['Time'].apply(cleanTime)
```

```
def timetotext(txt):
    if txt>='00:00:00' and txt<='07:00:00':
            txt="Critical"
            return txt


    else:
            txt="Regular"
            return txt
```

```
df_clean['Timeofday']=df_clean['OnlyTime'].apply(timetotext)
```

Fig 4.Data Pre-Processing Steps

- The data frame has been stored in a CSV on the local machine to create a checkpoint and to save time so that the entire steps are not repeated if the machine crashes or an incident is encountered.
- The dataset needs to be sampled as many models underperform on very large datasets. The data frame is first split into two data frame in which the “Target” corresponding to 0 are stored separately and 4 are stored separately. Hence a 10,000 row sample is selected randomly using four combination of columns “Target” and “Time of day”. 4 different data frames are created with combination such as where target is 0 and time of the day is regular, where target is 4 and time of day is critical, where target is 0 and time of day is critical and where target is 4 and time of day is regular.

```
rslt_df_0 = dataset.loc[dataset['Target'] == 0]
```

```
rslt_df_0_reg= rslt_df_0.loc[rslt_df_0["Timeofday"]=="Regular"]
rslt_df_0_reg
```

	Unnamed: 0	Target	Time	Tweets	tweet_without_stopwords	OnlyTime	Timeofday
0	0	0	Mon Apr 06 22:19:49 PDT 2009	is upset that he cant update his facebook by t...	upset update facebook texting might cry result...	22:19:49	Regular
1	1	0	Mon Apr 06 22:19:53 PDT 2009	kenichan i dived many times for the ball manag...	kenichan dived many time ball managed save res...	22:19:53	Regular
2	2	0	Mon Apr 06 22:19:57 PDT 2009	my whole body feels itchy and like its on fire ...	whole body feel itchy fire ...	22:19:57	Regular
3	3	0	Mon Apr 06 22:19:57 PDT 2009	nationwideclass no its not behaving at all im ...	nationwideclass behaving mad ...	22:19:57	Regular
4	4	0	Mon Apr 06 22:20:00 PDT 2009	kwesidei not the whole crew	kwesidei whole crew	22:20:00	Regular
...	...	...	...	...	...	...	...
799994	799994	0	Thu Jun 25 10:28:28 PDT 2009	sick spending my day laying in bed listening t...	spending laying listening taylorswift	10:28:28	Regular
799995	799995	0	Thu Jun 25 10:28:28 PDT 2009	gmail is down	gmail	10:28:28	Regular
799996	799996	0	Thu Jun 25 10:28:30 PDT 2009	rest in peace farrah so sad	rest peace farrah	10:28:30	Regular
799997	799997	0	Thu Jun 25 10:28:30 PDT 2009	ericurbane sounds like a rival is flagging you...	ericurbane sound rival flagging ad	10:28:30	Regular
799998	799998	0	Thu Jun 25 10:28:31 PDT 2009	has to resit exams over summer wishes he worke...	resit exam summer wish worked harder year uni	10:28:31	Regular

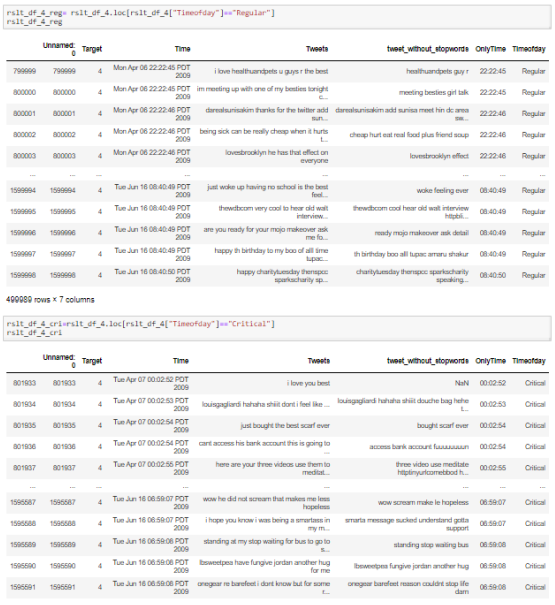
559139 rows × 7 columns

```
rslt_df_0_cri=rslt_df_0.loc[rslt_df_0["Timeofday"]=="Critical"]
rslt_df_0_cri
```

	Unnamed: 0	Target	Time	Tweets	tweet_without_stopwords	OnlyTime	Timeofday
1425	1425	0	Tue Apr 07 00:00:08 PDT 2009	japh i wish id known that there were more tick...	japh id known ticket earlier rang arvo left am...	00:00:08	Critical
1426	1426	0	Tue Apr 07 00:00:18 PDT 2009	woke up to find this coldfullnesstype thing ...	woke find coldfullnesstype thing isnt withou...	00:00:18	Critical
1427	1427	0	Tue Apr 07 00:00:18 PDT 2009	whf tscc killed derek reese brian austin green...	whf tscc killed derek reese brian austin green...	00:00:18	Critical
1428	1428	0	Tue Apr 07 00:00:22 PDT 2009	equinuxcom just crashed safari tks you	equinuxcom crashed safari tks	00:00:22	Critical
1429	1429	0	Tue Apr 07 00:00:41 PDT 2009	quotfire of anatoliaquot is fired costumes are...	quotfire anatoliaquot fired costume destroyed ...	00:00:41	Critical
...	...	...	...	...	...	...	...
792795	792795	0	Thu Jun 25 06:59:49 PDT 2009	early l and i put the same darn jeans on as ye...	early l put darn jean yesterday ughh morningbe...	06:59:49	Critical
792796	792796	0	Thu Jun 25 06:59:55 PDT 2009	aaaaaaaa its tooooooooo hot at work im melting	aaaaaaaa tooooooooo hot melting	06:59:55	Critical
792797	792797	0	Thu Jun 25 06:59:55 PDT 2009	last sweet was meant to be in reply to expsg fail	sweet meant reply expsg fail	06:59:55	Critical
792798	792798	0	Thu Jun 25 06:59:56 PDT 2009	tialaw morning trouble im having male problems	tialaw trouble male problem	06:59:56	Critical
792799	792799	0	Thu Jun 25 06:59:59 PDT 2009	in fact i hope i get my cut today arrials away t...	fact cut arrials away	06:59:59	Critical

240860 rows × 7 columns

Fig 5.Selecting negative comments



```
rslt_df_4_reg= rslt_df_4.loc[rslt_df_4["Timeofday"]=="Regular"]
rslt_df_4_reg
```

	Unnamed: 0	Target	Time	Tweets	tweet_without_stopwords	OnlyTime	Timeofday
799999	799999	4	Mon Apr 06 22:22:45 PDT 2009	i love healthuandpets u guys r the best	healthuandpets guy r	22:22:45	Regular
800000	800000	4	Mon Apr 06 22:22:45 PDT 2009	im meeting up with one of my besties tonight c...	meeting besties girl talk	22:22:45	Regular
800001	800001	4	Mon Apr 06 22:22:46 PDT 2009	darealsunisakim thanks for the twitter add sun...	darealsunisakim add sunisa meet hin dc area sw...	22:22:46	Regular
800002	800002	4	Mon Apr 06 22:22:46 PDT 2009	being sick can be really cheap when it hurts t...	cheap hurt eat real food plus friend soup	22:22:46	Regular
800003	800003	4	Mon Apr 06 22:22:46 PDT 2009	lovesbrooklyn he has that effect on everyone	lovesbrooklyn effect	22:22:46	Regular
...	...	...	...	...	...	...	...
1599994	1599994	4	Tue Jun 16 08:40:49 PDT 2009	just woke up having no school is the best feel...	woke feeling ever	08:40:49	Regular
1599995	1599995	4	Tue Jun 16 08:40:49 PDT 2009	thewdbcom very cool to hear old walt interview...	thewdbcom cool hear old walt interview httpbli...	08:40:49	Regular
1599996	1599996	4	Tue Jun 16 08:40:49 PDT 2009	are you ready for your mojo makeover ask me fo...	ready mojo makeover ask detail	08:40:49	Regular
1599997	1599997	4	Tue Jun 16 08:40:49 PDT 2009	happy th birthday to my boo of alll time tupac...	th birthday boo alll tupac amaru shakur	08:40:49	Regular
1599998	1599998	4	Tue Jun 16 08:40:50 PDT 2009	happy charilytuesday therapcc sparkscharity sp...	charilytuesday therapcc sparkscharity speaking...	08:40:50	Regular

499999 rows × 7 columns

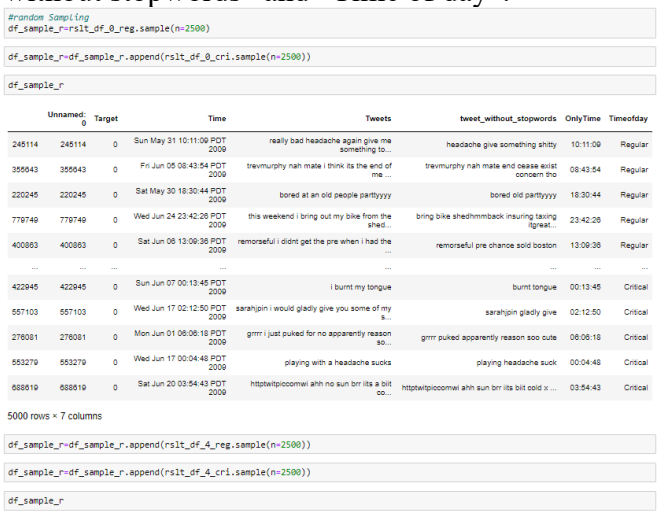
```
rslt_df_4_cri=rslt_df_4.loc[rslt_df_4["Timeofday"]=="Critical"]
rslt_df_4_cri
```

	Unnamed: 0	Target	Time	Tweets	tweet_without_stopwords	OnlyTime	Timeofday
801933	801933	4	Tue Apr 07 00:02:52 PDT 2009	i love you best	NaN	00:02:52	Critical
801934	801934	4	Tue Apr 07 00:02:53 PDT 2009	louisgagliardi hahaha shiiit dont i feel like ...	louisgagliardi hahaha shiiit douche bag hehe i...	00:02:53	Critical
801935	801935	4	Tue Apr 07 00:02:54 PDT 2009	just bought the best scarf ever	bought scarf ever	00:02:54	Critical
801936	801936	4	Tue Apr 07 00:02:54 PDT 2009	cant access his bank account this is going to ...	access bank account fuuuuuuuun	00:02:54	Critical
801937	801937	4	Tue Apr 07 00:02:55 PDT 2009	here are your three videos use them to meditat...	three video use meditate httptinyurlcomebbod h...	00:02:55	Critical
...	...	...	...	...	...	...	...
1595587	1595587	4	Tue Jun 16 06:59:07 PDT 2009	wow he did not scream that makes me less hopeless	wow scream make le hopeless	06:59:07	Critical
1595588	1595588	4	Tue Jun 16 06:59:07 PDT 2009	i hope you know i was being a smartass in my m...	smarta message sucked understand gotta support	06:59:07	Critical
1595589	1595589	4	Tue Jun 16 06:59:08 PDT 2009	standing at my stop waiting for bus to go to s...	standing stop waiting bus	06:59:08	Critical
1595590	1595590	4	Tue Jun 16 06:59:08 PDT 2009	ibsweetpea have fungive jordan another hug for me	ibsweetpea fungive jordan another hug	06:59:08	Critical
1595591	1595591	4	Tue Jun 16 06:59:08 PDT 2009	onegear re barefeet i dont know but for some r...	onegear barefeet reason couldnt stop life darn	06:59:08	Critical

300011 rows × 7 columns

Fig 6.Selecting positive comments

- A sample of 2500 rows are selected in random from the above mentioned four data frames to create a sample of 10000 rows. The only attributes in this sample are “Target”, Tweets without stopwords” and “Time of day”.



```
#random Sampling
df_sample_r=rslt_df_0_reg.sample(n=2500)
```

```
df_sample_r=df_sample_r.append(rslt_df_0_cri.sample(n=2500))
```

```
df_sample_r
```

	Unnamed: 0	Target	Time	Tweets	tweet_without_stopwords	OnlyTime	Timeofday
245114	245114	0	Sun May 31 10:11:09 PDT 2009	really bad headache again give me something to...	headache give something shitty	10:11:09	Regular
356643	356643	0	Fri Jun 05 08:43:54 PDT 2009	trevmurphy nah mate i think its the end of me ...	trevmurphy nah mate end cease exist concern tho	08:43:54	Regular
220245	220245	0	Sat May 30 18:30:44 PDT 2009	bored at an old people partyyyy	bored old partyyyy	18:30:44	Regular
779749	779749	0	Wed Jun 24 23:42:26 PDT 2009	this weekend i bring out my bike from the shed ...	bring bike shedhmmback insuring taxing itgreat...	23:42:26	Regular
400863	400863	0	Sat Jun 06 13:09:36 PDT 2009	remorseful i didnt get the pre when i had the ...	remorseful pre chance sold boston	13:09:36	Regular
...	...	...	...	...	...	...	...
422945	422945	0	Sun Jun 07 00:13:45 PDT 2009	i burnt my tongue	burnt tongue	00:13:45	Critical
557103	557103	0	Wed Jun 17 02:12:50 PDT 2009	sarahjpin i would gladly give you some of my s...	sarahjpin gladly give	02:12:50	Critical
276081	276081	0	Mon Jun 01 06:06:18 PDT 2009	grrrr i just puked for no apparently reason so...	grrrr puked apparently reason soo cute	06:06:18	Critical
553279	553279	0	Wed Jun 17 00:04:48 PDT 2009	playing with a headache sucks	playing headache suck	00:04:48	Critical
688619	688619	0	Sat Jun 20 03:54:43 PDT 2009	httptwitpiccomwi ahh no sun brr its a bit co...	httptwitpiccomwi ahh sun brr its bit cold x ...	03:54:43	Critical

5000 rows × 7 columns

```
df_sample_r=df_sample_r.append(rslt_df_4_reg.sample(n=2500))
```

```
df_sample_r=df_sample_r.append(rslt_df_4_cri.sample(n=2500))
```

```
df_sample_r
```

Fig 5.Creating a sample from four data frames

- The columns Target and Time of day has been selected to create a new variable called “Mental Health”. This is a crucial step as it classifies mental health as normal or abnormal based on the time and sentiment of the tweet (Coppersmith, et al., 2015).

```
# Classifying Mental Health
```

```
conditions=[(df_sample['Target'] == 0) & (df_sample['Timeofday'] == "Critical"),
            (df_sample['Target'] == 4) & (df_sample['Timeofday'] == "Critical"),
           (df_sample['Target'] == 0) & (df_sample['Timeofday'] == "Regular"),
           (df_sample['Target'] == 4) & (df_sample['Timeofday'] == "Regular")]
```

```
values=[1,0,0,0]
```

```
df_sample["Mental_Health"]=np.select(conditions,values)
```

```
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: 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: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

```
df_sample["Mental_Health"].unique()
```

```
array([0, 1], dtype=int64)
```

```
df_sample.head()
```

	Target	tweet_without_stopwords	Timeofday	Mental_Health
245114	0	headache give something shitty	Regular	0
355643	0	trevmurphy nah mate end cease exist concern tho	Regular	0
220245	0	bored old partyyyy	Regular	0
779749	0	bring bike shedhmmback insuring taxing itgreat...	Regular	0
400863	0	remorseful pre chance sold boston	Regular	0

Fig 6.Classifying mental health

- Common words are extracted from the normal and abnormal group to understand the words that are associated with the mental health condition.

```
unstablesentence=""
```

```
for i in range(len(df_mentalunstable)):
            #print(type(df_sample["tweet_without_stopwords"][i]))
            unstablesentence+=df_mentalunstable["tweet_without_stopwords"][i]+" "
```

```
# Create list of all the words in the string
word_list = unstablesentence.split()

# Get the count of each word.
word_count = Counter(word_list)

# Use most_common() method from Counter subclass
print(word_count.most_common(20))
```

```
[('hour', 60), ('hurt', 44), ('feeling', 43), ('suck', 42), ('thing', 41), ('look', 37), ('early', 36), ('year', 36), ('already', 34), ('exam', 34), ('another', 33), ('friend', 32), ('x', 32), ('doesnt', 32), ('man', 31), ('damn', 30), ('isnt', 29), ('yet', 29), ('find', 29), ('went', 29)]
```

```
#Cresting list of top 10 words from the list
height = [60, 44,43,42,41,37,36,36,34,34]
bars = ('hour', 'hurt', 'feeling', 'suck', 'thing','early','year','already','exam','another')
y_pos = np.arange(len(bars))

# Create horizontal bars
plt.barh(y_pos, height)

# Create names on the y-axis
plt.yticks(y_pos, bars)

# Show graphic
plt.show()
```

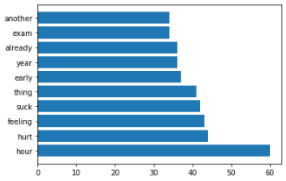


Fig 7.Common words displayed for abnormal group

```
stablesentence=""
for i in range(len(df_mentalstable)):
            #print(type(df_sample["tweet_without_stopwords"][i]))
            stablesentence+=df_mentalstable["tweet_without_stopwords"][i]+"|"
```

```
# Create list of all the words in the string
word_list_stable = stablesentence.split()

# Get the count of each word.
word_count_stable = Counter(word_list_stable)

# Use most_common() method from Counter subclass
print(word_count_stable.most_common(10))
```

```
[('friend', 137), ('thing', 118), ('look', 116), ('tweet', 99), ('guy', 91), ('let', 88), ('yay', 87), ('find', 84), ('hour', 84), ('year', 83)]
```

```
type(word_count_stable.most_common(20))
#word_count_stable.most_common(20).count().unstack(1).plot(kind="bar")
```

```
list
```

```
#Cresting list of top 10 words from the list
height = [137, 118, 116, 99, 91,88,87,84,84,83]
bars = ('friend', 'thing', 'look', 'tweet', 'guy','let','yay','find','hour','year')
y_pos = np.arange(len(bars))

# Create horizontal bars
plt.barh(y_pos, height)

# Create names on the y-axis
plt.yticks(y_pos, bars)

# Show graphic
plt.show()
```

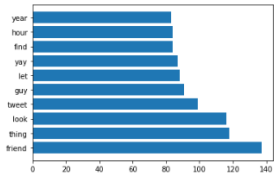


Fig 7.Common words displayed for normal group

- The data is now ready to be split into training and testing set as per 80-20 division. The target variable in the training set is the “Mental Health” and the “tweets without stopwords” will be used to train the data.
- The “X_train” and “y_train” have been concatenated together to form a train_set which is a data frame. Similar step has been performed with the “X_test” and “y_test” to form a data frame “test_set”
- Both the training and test data as vectorized using the Count Vectorization function which assigns tokens to the text in the column “tweets without stopwords”.

```
count_vect = CountVectorizer()
```

```
X_train_counts = count_vect.fit_transform(train_set['tweet_without_stopwords'].values.astype('U'))
```

```
X_train_counts[0]
```

```
<1x15494 sparse matrix of type '<class 'numpy.int64'>'
        with 12 stored elements in Compressed Sparse Row format>
```

```
X_test_counts = count_vect.transform(test_set['tweet_without_stopwords'].values.astype('U'))
```

Fig 8.Count vectorization Function

- SMOTE: The training data is subjecting to smote which stands for synthetic minority oversampling technique where the imbalanced dataset is transformed into a balanced one with the minority portion being replicated.

```
Smote
```

```
print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train == 0)))

# import SMOTE module from imblearn library
# pip install imblearn

sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_sample(X_train_counts,train_set['Mental_Health'].ravel())

print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))
```

```
Before OverSampling, counts of label '1': 2002
Before OverSampling, counts of label '0': 5998

C:\Users\Lenovo\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:86: FutureWarning: Function safe_indexing is deprecated; safe_indexing is deprecated in version 0.22 and will be removed in version 0.24.
  warnings.warn(msg, category=FutureWarning)

After OverSampling, the shape of train_X: (11996, 15494)
After OverSampling, the shape of train_y: (11996,)

After OverSampling, counts of label '1': 5998
After OverSampling, counts of label '0': 5998
```

Fig 9.SMOTE

- The data is now ready for model implementation.

5.Model Implementation

The training data that has been prepared will be used to train the different machine learning models and the test data will be used to perform predictions. Different machine learning models are implemented to predict the mental health. The classification report function has been used to extract the evaluation parameters. Also, the cross validation function is used to see if the model overfits.

5.1 Naïve Bayes

```
# Model Implementation
```

```
#Naive Bayes
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(X_train_res, y_train_res)
```

```
MultinomialNB()
```

```
pred_NB =clf.predict(X_test_counts)
```

```
print(classification_report(test_set["Mental_Health"], pred_NB))
```

```
              precision    recall  f1-score   support

           0       0.80      0.78      0.79      1502
           1       0.38      0.41      0.39       498

    accuracy                           0.69      2000
   macro avg       0.59      0.59      0.59      2000
weighted avg       0.70      0.69      0.69      2000
```

```
precision_score(test_set["Mental_Health"], pred_NB)
recall_score(test_set["Mental_Health"], pred_NB)
```

```
0.40562248995983935
```

```
cv_results = cross_validate(clf,X_train_res, y_train_res, cv=5,return_train_score=True)
cv_results
```

```
{'fit_time': array([0.01495981, 0.00498819, 0.00299263, 0.00398898, 0.00398827]),
 'score_time': array([0.00598407, 0.        , 0.00099802, 0.00099754, 0.0009973 ]),
 'test_score': array([0.57166667, 0.6198416 , 0.74822843, 0.43434764, 0.44060025]),
 'train_score': array([0.65933722, 0.91049286, 0.91059706, 0.73793894, 0.73981453])}
```

Fig 10.Naive Bayes Model Implementation

5.2 Random Forest

```
#Random Forest
rdfrclf=RandomForestClassifier()
```

```
rdfrclf.fit(X_train_res, y_train_res)
```

```
RandomForestClassifier()
```

```
pred_rf =rdfrclf.predict(X_test_counts)
```

```
print(classification_report(test_set["Mental_Health"], pred_rf))
```

```
              precision    recall  f1-score   support

           0       0.79      0.62      0.69      1502
           1       0.30      0.49      0.37       498

    accuracy                           0.59      2000
   macro avg       0.54      0.55      0.53      2000
weighted avg       0.66      0.59      0.61      2000
```

```
recall_score(test_set["Mental_Health"], pred_rf)
```

```
0.4879518072289157
```

```
cv_results = cross_validate(rdfrclf,X_train_res, y_train_res, cv=5,return_train_score=True)
cv_results
```

```
{'fit_time': array([37.09855175, 39.2693572 , 41.67459536, 41.69230247, 41.08187294]),
 'score_time': array([0.41729522, 0.65624237, 1.51594472, 1.6266942 , 1.46807313]),
 'test_score': array([0.5525    , 0.61483952, 0.76573572, 0.77073781, 0.76781992]),
 'train_score': array([0.99135056, 0.99155986, 0.99135146, 0.99166406, 0.99083047])}
```

Fig 11. Random Forest Model Implementation

5.3 XGBoost

```
xg_reg = XGBClassifier()
xg_reg.fit(X_train_res, y_train_res)
```

```
XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method=None, validate_parameters=False, verbosity=None)
```

```
pred_cg =xg_reg.predict(X_test_counts)
```

```
print(classification_report(test_set["Mental_Health"], pred_cg))
```

```
              precision    recall  f1-score   support

           0       0.76      0.98      0.85      1502
           1       0.43      0.06      0.10       498

    accuracy                           0.75      2000
   macro avg       0.59      0.52      0.48      2000
weighted avg       0.68      0.75      0.66      2000
```

```
recall_score(test_set["Mental_Health"], pred_cg)
```

```
0.05622489959839357
```

```
cv_results = cross_validate(xg_reg,X_train_res, y_train_res, cv=5,return_train_score=True)
cv_results
```

```
{'fit_time': array([ 8.68197989,  9.73096323, 10.06404924, 10.27949786, 10.14984274]),
 'score_time': array([0.04687715, 0.03737307, 0.04771233, 0.02592969, 0.0239377 ]),
 'test_score': array([0.51916667, 0.65068779, 0.92205085, 0.92580242, 0.91162985]),
 'train_score': array([0.87484368, 0.85255809, 0.79285193, 0.79368553, 0.79378973])}
```

Fig 12. XGBoost Model Implementation

5.4 MLP Classifier

```
#MLP Classifier
classifier = MLPClassifier()
```

```
classifier.fit(X_train_res, y_train_res)
```

```
MLPClassifier()
```

```
pred_MLP =classifier.predict(X_test_counts)
```

```
print(classification_report(test_set["Mental_Health"], pred_MLP))
```

```
              precision    recall  f1-score   support

           0       0.79      0.62      0.70      1502
           1       0.31      0.51      0.38       498

    accuracy                           0.59      2000
   macro avg       0.55      0.56      0.54      2000
weighted avg       0.67      0.59      0.62      2000
```

```
cv_results = cross_validate(classifier,X_train_res, y_train_res, cv=5,return_train_score=True)
cv_results
```

```
{'fit_time': array([  872.96855998,  1034.90340424, 35694.02395344,   563.90407181,
         555.89586377]),
 'score_time': array([0.01596022, 0.01405168, 0.04276633, 0.00498652, 0.00797892]),
 'test_score': array([0.545     , 0.62442684, 0.77699041, 0.76907045, 0.76615256]),
 'train_score': array([0.99166319, 0.99197666, 0.99145566, 0.99103887, 0.99083047])}
```

Fig 13.MLP classsifer Model Implementation

5.5 Support Vector Machine

```
#Support Vector Machine
clf_svc = SVC()
```

```
clf_svc.fit(X_train_res,y_train_res)
```

```
SVC()
```

```
pred_SVC =clf_svc.predict(X_test_counts)
```

```
print(classification_report(test_set["Mental_Health"], pred_SVC))
```

```
              precision    recall  f1-score   support

           0       0.77      0.72      0.75      1502
           1       0.30      0.36      0.33       498

    accuracy                           0.63      2000
   macro avg       0.54      0.54      0.54      2000
weighted avg       0.65      0.63      0.64      2000
```

```
cv_results = cross_validate(clf_svc,X_train_res, y_train_res, cv=5,return_train_score=True)
cv_results
```

```
{'fit_time': array([ 4.25237727,  4.45505047, 11.53027034, 11.09597349, 11.07692051]),
 'score_time': array([0.60432029, 1.24699807, 2.07429624, 2.09815621, 2.07769465]),
 'test_score': array([0.54958333, 0.66819508, 0.88828679, 0.89328887, 0.88828679]),
 'train_score': array([0.94247603, 0.91695321, 0.93904345, 0.93925185, 0.93508388])}
```

Fig 14. Support Vector Machine Implementation

5.6 Neural Network

```
#create model
from keras import layers
input_vat = X_train_res.shape[1]
```

```
#add model layers
model = Sequential()
model.add(Dense(10,input_dim=input_vat,activation='relu'))
model.add(Dense(2, activation='softmax'))
```

```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
odel.fit(X_train_res, y_train_res,epochs=10,verbose=False,validation_data=(X_test_counts, test_set["Mental_Health"]),batch_size=1
```

```
pred_NN=model.predict(X_test_counts)
pred_NNnew=np.where(pred_NN > 0.5, 1, 0)
new_list=[]
for i in pred_NNnew:
    new_list.append(i[1])
print(classification_report(test_set['Mental_Health'],new_list))
```

```
              precision    recall  f1-score   support

           0       0.79      0.65      0.71      1502
           1       0.32      0.49      0.38       498

    accuracy                           0.61      2000
   macro avg       0.55      0.57      0.55      2000
weighted avg       0.67      0.61      0.63      2000
```

```
recall_score(test_set['Mental_Health'],new_list)
```

```
0.4899598393574297
```

Fig 15 Neural Network Implementation

6. Hyper Parameter Tuning

Models have been implemented with the default parameters and evaluated based on their performance. The parameters of all the models except for Naïve Bayes will be tuned in order to improve their performance. A detailed explanation of the parameters tuned has been included in the Research paper report.

6.1 Random Forest

```
param_grid = {
    'n_estimators': [200, 500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8],
    'criterion' :['gini', 'entropy'],
    'min_samples_leaf': [1, 2, 4],
    'min_samples_split': [2, 5, 10],
}
```

```
CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
CV_rfc.fit(X_train_res, y_train_res)
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [4, 5, 6, 7, 8],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [200, 500]})
```

```
CV_rfc.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 8,
 'max_features': 'auto',
 'n_estimators': 500}
```

```
#Random Forest Tuning
ran_tune=RandomForestClassifier(random_state=42, max_depth=50,max_features='auto',n_estimators=500,criterion='gini')
```

```
ran_tune.fit(X_train_res, y_train_res)
```

```
RandomForestClassifier(max_depth=50, n_estimators=500, random_state=42)
```

```
pred_ran_tune=ran_tune.predict(X_test_counts)
print(classification_report(test_set["Mental_Health"], pred_ran_tune))
```

```
              precision    recall  f1-score   support

           0       0.82      0.50      0.62      1502
           1       0.31      0.67      0.43       498

    accuracy                           0.55      2000
   macro avg       0.57      0.59      0.53      2000
weighted avg       0.70      0.55      0.58      2000
```

Fig 16.Hyper parameter tuning of Random Forest Model

6.2 XGBoost

```
#XGBoost Tuning
xg_reg_tune = XGBClassifier(learning_rate =0.4,n_estimators=5000,max_depth=2,min_child_weight=50,gamma=10,subsample=0.8,objecti
xg_reg_tune.fit(X_train_res, y_train_res)
```

```
XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=10, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.4, max_delta_step=0, max_depth=2,
              min_child_weight=50, missing=nan, monotone_constraints=None,
              n_estimators=5000, n_jobs=0, num_parallel_tree=1, random_state=3,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=3,
              subsample=0.8, tree_method=None, validate_parameters=False,
              verbosity=None)
```

```
pred_xg_tune =xg_reg_tune.predict(X_test_counts)
```

```
print(classification_report(test_set["Mental_Health"], pred_xg_tune))
```

```
              precision    recall  f1-score   support

           0       0.75      0.99      0.85      1502
           1       0.35      0.02      0.04       498

    accuracy                           0.75      2000
   macro avg       0.55      0.50      0.45      2000
weighted avg       0.65      0.75      0.65      2000
```

Fig 17. Hyper parameter tuning of XGBoost Model

6.3 MLP Classifier

```
#MLP Classifier Tuning
classifier_tune = MLPClassifier(max_iter=100,hidden_layer_sizes=(100,100,10),activation='relu',learning_rate='constant',solver='
```

```
classifier_tune.fit(X_train_res, y_train_res)
```

```
MLPClassifier(hidden_layer_sizes=(100, 100, 10), max_iter=100, random_state=1)
```

```
pred_MLP_tune =classifier_tune.predict(X_test_counts)
```

```
print(classification_report(test_set["Mental_Health"], pred_MLP_tune))
```

```
              precision    recall  f1-score   support

           0       0.80      0.61      0.69      1502
           1       0.31      0.53      0.39       498

    accuracy                           0.59      2000
   macro avg       0.56      0.57      0.54      2000
weighted avg       0.68      0.59      0.62      2000
```

Fig 18. Hyper parameter tuning of MLP classsifer Model

6.4 Support Vector Machine

```
from sklearn.svm import SVC
classifier_svc = SVC(kernel='rbf',C=1000, random_state = 2,gamma=100)
classifier_svc.fit(X_train_res,y_train_res)
```

```
SVC(C=1000, gamma=100, random_state=2)
```

```
pred_SVC =classifier_svc.predict(X_test_counts)
```

```
print(classification_report(test_set["Mental_Health"], pred_SVC))
```

```
              precision    recall  f1-score   support

           0       0.76      0.92      0.83      1502
           1       0.32      0.11      0.17       498

    accuracy                           0.72      2000
   macro avg       0.54      0.52      0.50      2000
weighted avg       0.65      0.72      0.67      2000
```

Fig 19.Hyper parameter tuning of Support Vector Machine

6.5 Neural Networks

```
#Tuned Neural Network
```

```
model_tune = Sequential()
model_tune.add(Dense(5,input_dim=input_vat,activation='relu'))
model_tune.add(Dense(10, activation='softmax'))
model_tune.compile(loss='sparse_categorical_crossentropy',optimizer='adam', metrics=['accuracy'])
```

```
model_tune.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
history = model_tune.fit(X_train_res, y_train_res,epochs=8,verbose=False,validation_data=(X_test_counts, test_set["Mental_Health"
```

```
pred_NN1=model_tune.predict(X_test_counts)
pred_NNnew1=np.where(pred_NN1 > 0.5, 1, 0)
new_list1=[]
for i in pred_NNnew1:
    new_list1.append(i[1])
print(classification_report(test_set['Mental_Health'],new_list1))
```

```
              precision    recall  f1-score   support

           0       0.80      0.66      0.72      1502
           1       0.33      0.51      0.40       498

    accuracy                           0.62      2000
   macro avg       0.56      0.58      0.56      2000
weighted avg       0.68      0.62      0.64      2000
```

Fig 20. Hyper parameter tuning Neural Network

This concludes the implementation of the research project.

7. References

A., V. & Sonawane, S., 2016. Sentiment Analysis of Twitter Data: A Survey of Techniques. *International Journal of Computer Applications*, 139(11), pp. 5-15

Coppersmith, G., Dredze, M. & Harman, C., 2015. *Quantifying Mental Health Signals in Twitter.* s.l., Association for Computational Linguistics (ACL)