Faculty of Computing and Technology

University of Kelaniya

BSCS 23072 Group project

Final Report

Application of Deep Learning Algorithms for Bipolar Disorder Symptom Prediction

This final report is submitted in partial fulfilment of the requirements of the above module in the B.Sc. (Hons) in Computer Science degree

Under the supervision of

Dr. Anuradhi Welhenge

Senior Lecturer (Grade II)

on

August 4, 2022

Group 09

Members: -

O.A.A Kulathunge	- CS/2018/022
S.D Liyanage	- CS/2018/024
R.D.C Nethsara	- CS/2018/029
A.P.I.R Sankalpa	- CS/2018/040
G.D.V Shehan	- CS/2018/042
S.J.T Silva	- CS/2018/043

Abstract

Bipolar disorder also known as manic depression is a mental illness causes unusual shifts in mood, energy, activity levels, concentration, and the ability to carry out day-to-day tasks. People who have bipolar disorder can have periods in which they feel overly happy and energized and other periods of feeling very sad, hopeless, and sluggish. In between those periods, they usually feel normal. The idea was that both patients and psychiatrists can be supported by making a better diagnosis of this disease with the use of new technologies such as AI (Artificial Intelligence). Deep Learning is a subset of AI which makes artificial neural networks that mimic human brain.

This project springs from the Application of Machine Learning Algorithms for Bipolar Disorder Crisis Prediction [12] project. which studies the appearance of crisis in patients with bipolar disorder in order to predict them.

The main goal of this project is to apply different deep learning algorithms to symptom-based patient data in order to create a prediction model. The steps which have been followed when completing this project are data cleaning, data combining, building deep learning model, training deep learning model, performance evaluation, performance comparison and the conclusion.

Acknowledgement

we would like to express our heartfelt gratitude to our Project Coordinator and Advisor,Dr Anuradhi Welhenge,for their continued guidance, support and knowledge throughout the project.We would also like to express our heartfelt gratitude to our lecturers Dr. Chamli Pushpakumara and Mrs.Malsha Mahawatte for providing us with the guidance we need to achieve our goals.We would like to take this opportunity to thank you for your advice and encouragement.Finally, special thanks to the members of our team who have worked tirelessly to make this project a success so far.

Table of Content

Table of Figures

1. Introduction

1.1 Motivation

Artificial Intelligence (AI) is becoming rapid grown phase. In now a day's, most of the fields are using AI technology within their systems. Deep learning is a subset of AI. One of the uses of deep learning in the field of medicine is to diagnosis of different diseases.

Bipolar disorder is oscillation of the patient's mood between two states, mania and depression. Some patient's mood changes to another frequently. But some are not. The mood episode is not following a pattern. Therefore, accurately diagnosing of patient state is very important. The prediction model developed in this project will be help Psychiatrists to have a second opinion about patient state. Through this, it will be helpful to give accurate treatments to the patients.

1.2 Objectives

- Test different deep learning algorithms on patient data

  Prediction models are created by help of different deep learning algorithms. Models are trained on patient data. To find models performance they are tested.

- Find the best performing deep learning algorithm
  To find best performing deep learning algorithm had to compare performance of each prediction model. Highest performing algorithm was selected as the best deep learning algorithm.

1.3 The scope of the project

Aim of the project is to predict the episode in which a patient might be in or tend towards with the help of deep learning algorithms and find the best performing algorithm. If the prediction is not accurate it at least gives the psychiatrist a second opinion about the state a patient might be in or tend towards.

1.4 Limitations

- Team’s limited knowledge about machine learning subject.
- Data is scarce in the datasets. Small amount of data was available for the project.
- Only a few research has been carried out in this field, Because of that there are not good data sources.

2. Background

2.1 Description of the wider context of the problem

Doctors taking a multiple test to diagnose bipolar disorder. Because bipolar disorder shows distinct symptoms it cannot confirm by single test. Often doctors use a combination of method to make diagnosis. To diagnose bipolar disorder, a doctor may perform a physical examination, conduct an interview, and order lab tests. Bipolar disorder is most often misdiagnosed in its early stages. When it’s diagnosed as wrong, symptoms of bipolar disorder can get worse. This usually occurs because the wrong treatment is provided. These are some of the challenges when diagnosing bipolar disorder.

Episodes of mood swings may occur rarely or multiple times a year. Someone may feel the same mood (depressed or manic) several times before switching to the opposite mood. Therefore, accurate diagnosis of patient’s state is very important. Because proper diagnosis and treatment can help people with bipolar disorder lead healthy and active lives

2.2 Review

In recent years, the study of predictive patterns and patterns based on deep learning in the diagnosis of neurological diseases has come under intense scrutiny in biology. Some of them are similar to this research. They are discussed in here.

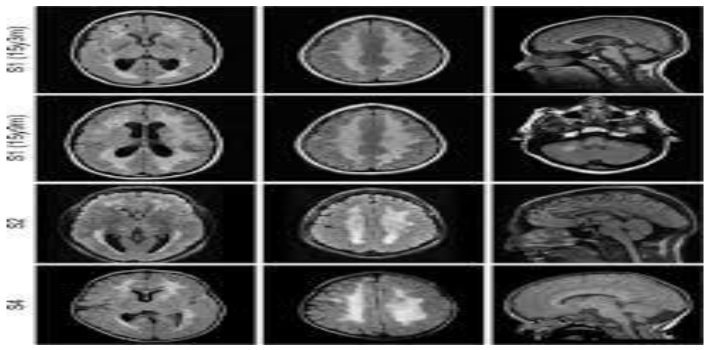


Figure 2.1 MRI image of brain

In many times the neurological diagnosis is made according to the changes in the image received during Magnetic Resonance Imaging (MRI). One of the research is CNN-based automatic diagnosis method to classify the grey matter density images of first-episode psychosis (FEP), bipolar disorder (BD) and healthy controls (HC)[4]. In this research Developed a prediction model based on Convolutional Neural Network(CNN) . The model was constructed using variations of images obtained from MRI. It aims to classify first episode psychology (FEP), bipolar disorder (BD) and healthy controls (HC). In this research reveals that abnormal gray matter volume in photographs obtained during magnetic resonance imaging is one of the key features for altering FEP, BD, and HC. The proposed CNN architecture achieves the best accuracy (99.24), recall (99.22), F1-score (99.23), and accuracy (99.15).

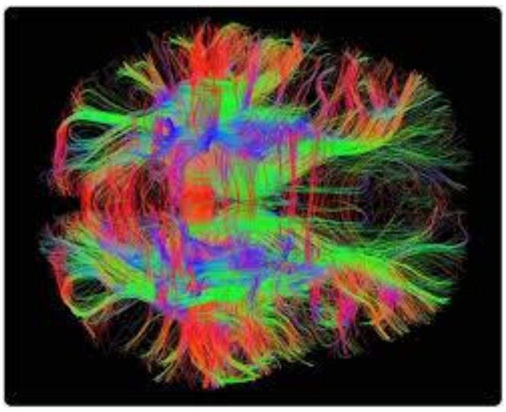


Figure 2.2 DTI image of brain

Clinically, behavioral disorders associated with Alzheimer's disease (AD) and bipolar disorder (BD) can be difficult to differentiate. Both diseases respond to certain common therapies. Another research is computer aided diagnostic systems on brain diffusion tensor imaging features. The aim of this study was the analyzed of white matter integrity, which may help to discriminate among BD and AD using diffusion tensor imaging (DTI). This new type of magnetic resonance imaging (MRI) is the only non-invasive technology that can provide information about axonal tracts integrity, allowing to investigate neuropathological mechanism underlying white matter (WM) changes. Here, training and testing of support vector machines (SVMs) on classification fractional anisotropy (FA) was performed. Fractional anisotropy (FA) is widely used as an MRI measurement of white matter microscopy in MRI studies of neurological disorders / disease. Accordingly, the potential for differential treatment between BD and AD patients and the diffusion tensor image (DTI) from HC are explored in multicenter analysis based on data features. Accuracy = 100%, Sensitivity = 100% and specificity = 100% The best rating results on feature vectors were obtained using 99.95% of the correlated distribution calculated from the FA data.

In here this two research were using neuroimaging data to diagnose bipolar disorder. Although these methods returned high accuracy, they require expensive equipments. It is little bit complex and costly method. Because of that these solutions cannot be used in everywhere. Therefore, the prediction model implemented in this project can be used as a simpler and more convenient solution.

2.3 Methods and Tools used

Deep learning

It is a field that is based on learning and improving on its own by examining computer algorithms. However, through this technology Big Data analytics have permitting to allowing computers to observe, learn, and react to complex situations faster than humans. Systems can build through this technology to based on data observed in the past, predict future outcomes.

Python

Python is a popular and simple to use computer language for deep learning. Python is a readable programming language that is used for data cleaning. Python has some libraries made for completing tasks of this kind. A good example is the pandas library. There are numerous techniques for handling data frames in Python.

Python is an open-source programming language that benefits from good worldwide support from numerous sites and high-quality documentation. Additionally, it features a huge and active developer community that offers support at any stage of creation. The majority of scientists use Python for Deep Learning projects, hence Python communities are home to the majority of the world's sharpest brains.

Google CoLab

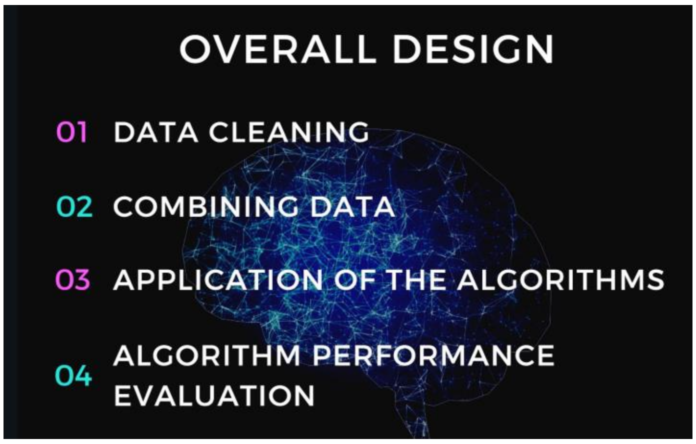
Google Colab is a runtime environment that allows for cloud-only code execution. Google Colab is a document that can be executed and allows for the writing, running, and sharing of code inside Google Drive. Each cell in a notebook page can include text, code, graphics, and other types of data. There is no setup necessary on your PC to run Python code. Text cells are utilized in Google Collaborate. Markdown, a format for plain text documents, is used to format text cells. The plain and flexible Markdown style enables the addition of headings, paragraphs, lists, and even mathematical formulas.

Google Colab is essential because it enables the training of massive ML and DL models without access to strong hardware or high-speed internet. Due to the processing limits of local workstations, Google Colab provides both Graphical Processing Unit (GPU) and Tensor Processing Unit (TPU) instances, making it a great tool for deep learning and data analytics aficionados. A Colab notebook is suitable for business use as well because it can be accessed remotely from any computer using a browser.

3. Design

3.1 Overall Design

This section describes about the data used for this project and the main steps followed in conducting the project. As shown in the below figure, the overall process has been divided to four major parts.



OVERALL DESIGN

01 DATA CLEANING

02 COMBINING DATA

03 APPLICATION OF THE ALGORITHMS

04 ALGORITHM PERFORMANCE EVALUATION

Figure 3.1 Overall Design

3.1.1 Data used for this project

The data sets used for this project was obtained from Application of Machine Learning Algorithms for Bipolar Disorder Crisis Prediction [12] project, which studies the appearance of crisis in patients with bipolar disorder. All the data which are used in this project has been gathered during medical appointments with four different patients that have bipolar disorder. Because of their privacy, those four patients have been mentioned using four different English language letters without mentioning their names.

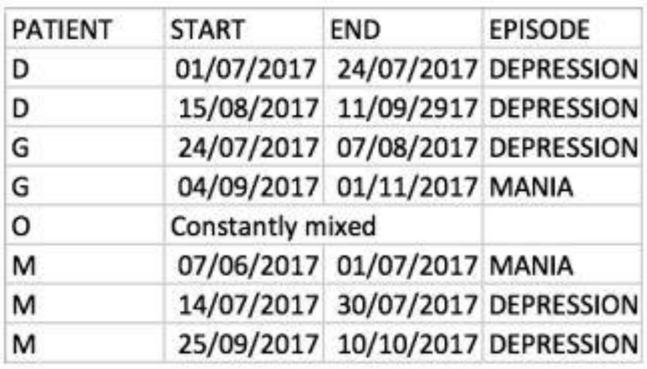
Data sets

1. Episode data set
2. YMRS data set
3. HDRS data set
4. Interview data set
5. Interventions data set

3.1.1.1 Episode data set

This is the first data set which is represented depression or mania as the two main different episode periods of the patients.

As shown in the following figure 3.2 the patient with English language letter “O” had regularly mixed episodes. As well as the time period which has been contained with the start and end dates has been represented the duration in which each patient has been suffering from depression or mania.



PATIENT	START	END	EPISODE
D	01/07/2017	24/07/2017	DEPRESSION
D	15/08/2017	11/09/2917	DEPRESSION
G	24/07/2017	07/08/2017	DEPRESSION
G	04/09/2017	01/11/2017	MANIA
O	Constantly mixed		
M	07/06/2017	01/07/2017	MANIA
M	14/07/2017	30/07/2017	DEPRESSION
M	25/09/2017	10/10/2017	DEPRESSION

Figure 3.2 Mania and Depression episodes in patients.

3.1.1.2 YMRS data set

This is the second data set which is represented Young Mania Rating Scale (YMRS) data of the same patients mentioned in the Episode data set. This scale is one of the most widely used scale to rate manic symptoms. It includes eleven items that are based on those four patients.

Those eleven items and their values are:

- Elevated Mood:
  - 0: absent
  - 1: mildly increased on questioning
  - 2: optimistic, self-confident, cheerful
  - 3: elevated, humorous
  - 4: Euphoric, singing, inappropriate laughter

- Increased Motor Activity-Energy:
  - 0: absent
  - 1: subjectively increased
  - 2: animated, gestures increased
  - 3: excessive energy, temporary hyperactivity
  - 4: motor excitement, continuous hyperactivity

- Sexual Interest:
  - 0: absent
  - 1: mildly increased
  - 2: subjective increase on questioning
  - 3: spontaneous sexual content, elaborates on sexual matters
  - 4: openly sexual towards other patients, staff or interviewer

- Sleep:
  - 0: no decrease in sleep
  - 1: sleeping less than normal amount by up to one hour
  - 2: sleeping less than normal amount by more than one hour
  - 3: decreased need for sleep
  - 4: denies need for sleep

- Irritability:
  - 0: absent
  - 2: subjectively increased
  - 4: irritable during times at interview
  - 6: frequently irritable during interview
  - 8: hostile, interview not possible

- Speech (verbal expressions):
  - 0: no increase
  - 1: talkative
  - 2; increased rate and amount at time

    - 6: consistently increased rate, different to interrupt
    - 8: uninterruptible, continuous speech

- Language-Thought Disorder:
    - 0: absent
    - 1: circumstantial
    - 2: distractible
    - 3: flight of ideas, difficult to follow
    - 4: incoherent, impossible to communicate with

- Content (activities):
    - 0: normal
    - 2: questionable plans, new interests
    - 4: special projects
    - 6: grandiose or paranoid ideas
    - 8: delusions, hallucinations

- Disruptive-Aggressive Behavior:
    - 0: absent
    - 2: subjectively increased
    - 4: animated, gestures increased
    - 6: excessive energy, temporary hyperactivity
    - 8: motor excitement, continuous hyperactivity

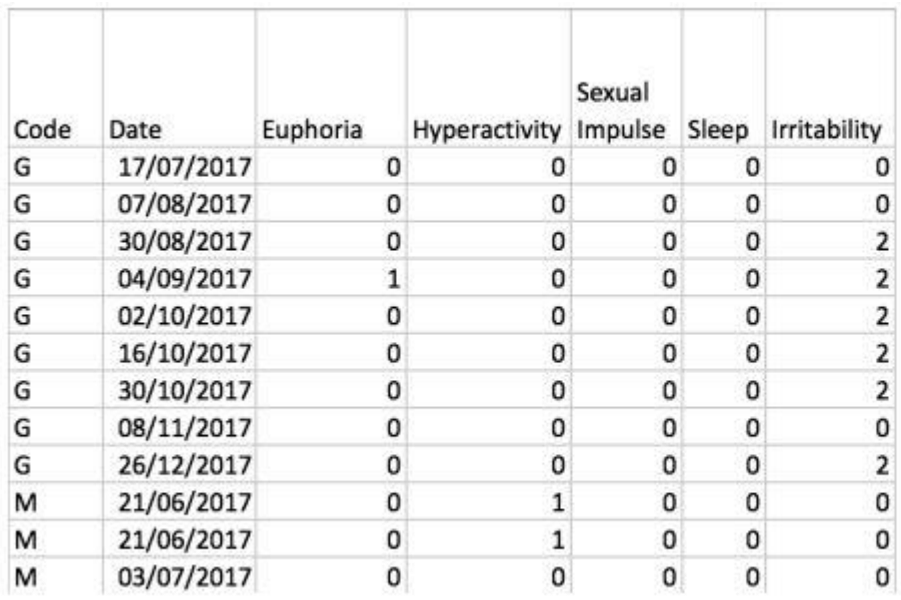
- Appearance:
    - 0: appropriate dress and grooming
    - 1: minimally unkempt
    - 2: poorly groomed
    - 3: disheveled, partly clothed
    - 4: completely unkempt, bizarre grab

- Insight (about the illness):
    - 0: present, admits illness
    - 1: possibly ill

- 2: admits behavior change but denies illness
- 3: admits possible behavior change but denies illness
- 4: denies any behavior change

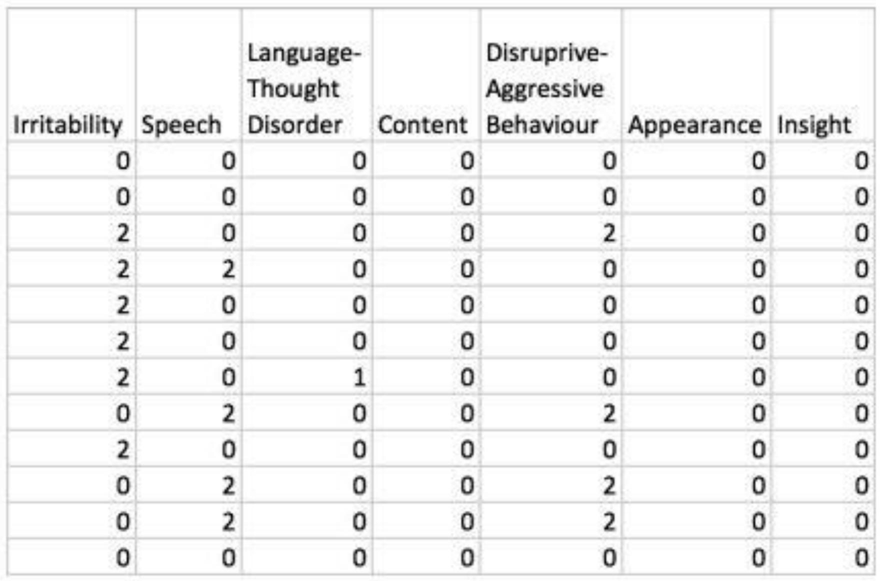
For more details about those eleven items and their values, follow this link. https://dcf.psychiatry.ufl.edu/files/2011/05/Young-Mania-Rating-Scale-Measure-withbackground.pdf

As shown in following figure 3.3 and 3.4, a set of YMRS data has been represented with eleven items with their values which have been mentioned above.



Code	Date	Euphoria	Hyperactivity	Sexual Impulse	Sleep	Irritability
G	17/07/2017	0	0	0	0	0
G	07/08/2017	0	0	0	0	0
G	30/08/2017	0	0	0	0	2
G	04/09/2017	1	0	0	0	2
G	02/10/2017	0	0	0	0	2
G	16/10/2017	0	0	0	0	2
G	30/10/2017	0	0	0	0	2
G	08/11/2017	0	0	0	0	0
G	26/12/2017	0	0	0	0	2
M	21/06/2017	0	1	0	0	0
M	21/06/2017	0	1	0	0	0
M	03/07/2017	0	0	0	0	0

Figure 3.3 Fragment of the YMRS patient data (Part 1)



Irritability	Speech	Language-Thought Disorder	Content	Disruprive-Aggressive Behaviour	Appearance	Insight
0	0	0	0	0	0	0
0	0	0	0	0	0	0
2	0	0	0	2	0	0
2	2	0	0	0	0	0
2	0	0	0	0	0	0
2	0	0	0	0	0	0
2	0	1	0	0	0	0
0	2	0	0	2	0	0
2	0	0	0	0	0	0
0	2	0	0	2	0	0
0	2	0	0	2	0	0
0	0	0	0	0	0	0

Figure 3.4 Fragment of the YMRS patient data (Part 2)

3.1.1.3 HDRS data set

This is the third data set which is represented Hamilton Depression Rating Scale (HDRS) data of the same patients mentioned in the Episode data set. This is the most widely used scale for depression rating. It has different versions. For this project, $HDRS_{17}$ has been used. It includes 17 different items that are based on those four patients.

Those 17 items and their values are:

- Depressed Mood:
  - 0: absent
  - 1: feeling state indicated only on questioning
  - 2: feeling state spontaneously reported verbally
  - 3: communicates feeling state non-verbally (ie voice)
  - 4: reports feelings only on spontaneous verbal and non-verbal behavior
- Feelings of guilt:
  - 0: absent
  - 1: feels that he or she has let people down
  - 2: ideas of guilt
  - 3: delusions of guilt
  - 4: hears accusatory voices and experiences threatening hallucinations
- Suicide:
  - 0: absent
  - 1: feels life is not worth living
  - 2: wishes he or she was dead
  - 3: ideas of suicide
  - 4: suicide attempts
- Insomnia (early in the night):
  - 0: no difficulty falling asleep

▪ 1: occasional difficulty falling asleep

▪ 2: nightly difficulty falling asleep

• Insomnia (middle of the night):

▪ 0: no difficulty

▪ 1: restless and disturbed during night

▪ 2: waking up during night

• Insomnia (early hours of the morning):

▪ 0: no difficulty

▪ 1: waking in early hours of the morning, goes back to sleep

▪ 2: unable to fall asleep again if he / she wakes up

• Work and activities:

▪ 0: no difficulty

▪ 1: feelings of incapacity

▪ 2: loss of interest in activity

▪ 3: decrease in time spent (less than 3 hours) or productivity

▪ 4: stopped working because of illness

• Retardation:

▪ 0: normal speech and thought

▪ 1: slight retardation during interview

▪ 2: obvious retardation during interview

▪ 3: interview difficult

▪ 4: complete stupor

• Agitation:

▪ 0: none

` ▪ 1: fidgeting

▪ 2: playing with hands, hair, etc.

▪ 3: can't sit still

▪ 4: nail biting, hair-pulling, etc.

• Psychic anxiety:

▪ 0: no difficulty

▪ 1: subjective tension

▪ 2: worrying about minor matters

▪ 3: anxious attitude (apparent in face or speech)

▪ 4: fears expressed without questioning

• Somatic (physiological) anxiety (gastrointestinal, cardiovascular, respiratory, unary frequency, sweating):

▪ 0: absent

▪ 1: mild

▪ 2: moderate

▪ 3: severe

▪ 4: incapacitating

• Somatic symptoms (gastro-intestinal):

▪ 0: none

▪ 1: loss of appetite but eating.

▪ 2: difficulty eating without staff urging

• General somatic symptoms:

▪ 0: none

▪ 1: heaviness in limbs, back or head

▪ 2: clear-cut symptoms

• Genital symptoms (loss of libido, menstrual disturbances):

▪ 0: absent

▪ 1: mild

▪ 2: severe

• Hypochondriasis:

▪ 0: not present

▪ 1: self-absorption

▪ 2: preoccupation with health

▪ 3: frequent complaints

▪ 4: hypochondriacal delusions

• Loss of weight:

▪ According to patient:

or 0: no weight loss

or 1: probable weight loss associated with illness

or 2: definite weight loss

or 3: not estimated

▪ According to weekly measurements:

or 0: less than 1 lb (450g) loss in one week

or 1: more than 1 lb (450g) loss in one week

or 2: more than 2 lb (900g) loss in one week

or 3: not estimated

• Insight:

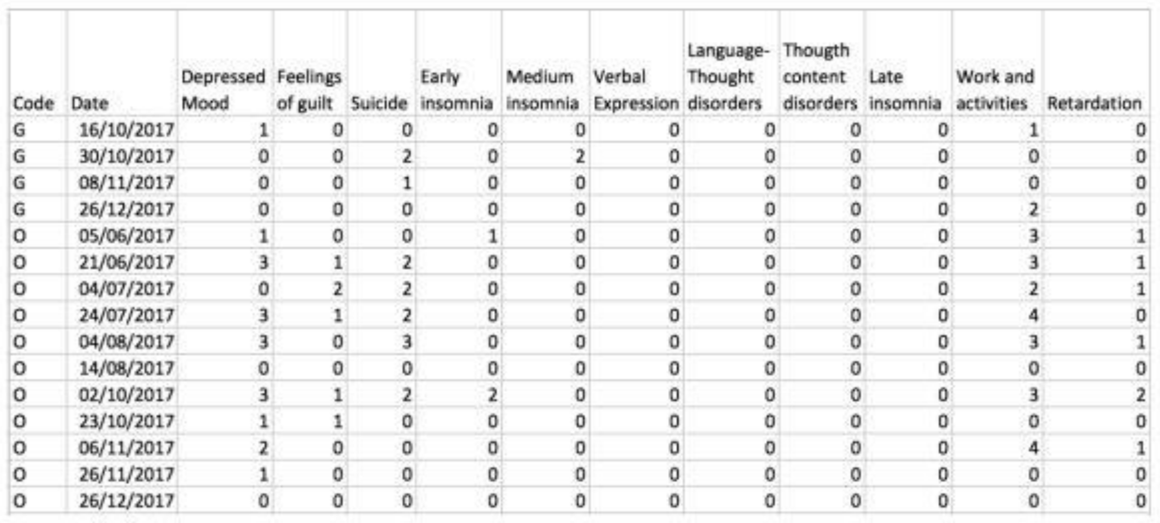
▪ 0: acknowledges being depressed and ill

▪ 1: acknowledges illness but attributes it to other factors

▪ 2: denies being ill at all

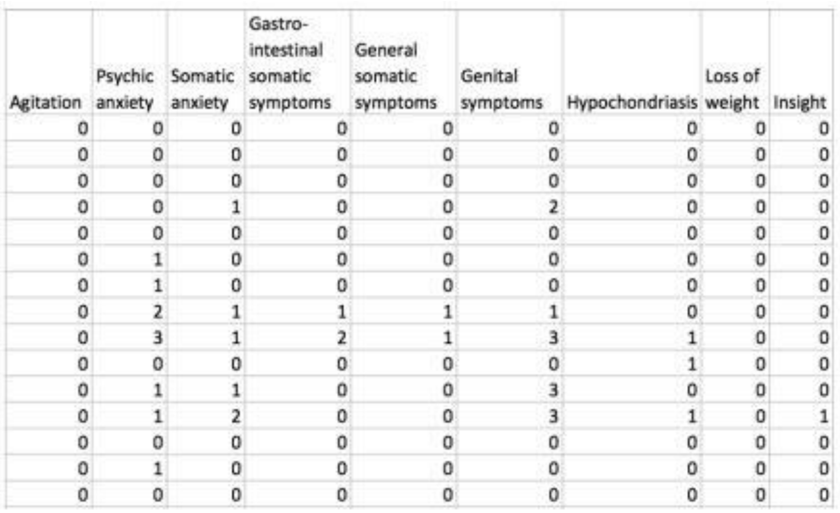
For more details about those eleven items and their values, follow this link. https://dcf.psychiatry.ufl.edu/files/2011/05/HAMILTON-DEPRESSION.pdf

As shown in following figures 3.5 and 3.6, a set of HDRS data has been represented with 17 items with their values which have been mentioned above.



Code	Date	Depressed Mood	Feelings of guilt	Suicide	Early insomnia	Medium insomnia	Verbal Expression	Language-Thought disorders	Thougth content disorders	Late insomnia	Work and activities	Retardation
G	16/10/2017	1	0	0	0	0	0	0	0	0	1	0
G	30/10/2017	0	0	2	0	2	0	0	0	0	0	0
G	08/11/2017	0	0	1	0	0	0	0	0	0	0	0
G	26/12/2017	0	0	0	0	0	0	0	0	0	2	0
O	05/06/2017	1	0	0	1	0	0	0	0	0	3	1
O	21/06/2017	3	1	2	0	0	0	0	0	0	3	1
O	04/07/2017	0	2	2	0	0	0	0	0	0	2	1
O	24/07/2017	3	1	2	0	0	0	0	0	0	4	0
O	04/08/2017	3	0	3	0	0	0	0	0	0	3	1
O	14/08/2017	0	0	0	0	0	0	0	0	0	0	0
O	02/10/2017	3	1	2	2	0	0	0	0	0	3	2
O	23/10/2017	1	1	0	0	0	0	0	0	0	0	0
O	06/11/2017	2	0	0	0	0	0	0	0	0	4	1
O	26/11/2017	1	0	0	0	0	0	0	0	0	0	0
O	26/12/2017	0	0	0	0	0	0	0	0	0	0	0

Figure 3.5 Fragment of the HDRS patient data (Part 1)



Agitation	Psychic anxiety	Somatic anxiety	Gastro-intestinal somatic symptoms	General somatic symptoms	Genital symptoms	Hypochondriasis	Loss of weight	Insight
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	2	0	0	0
0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	2	1	1	1	1	0	0	0
0	3	1	2	1	3	1	0	0
0	0	0	0	0	0	1	0	0
0	1	1	0	0	3	0	0	0
0	1	2	0	0	3	1	0	1
0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

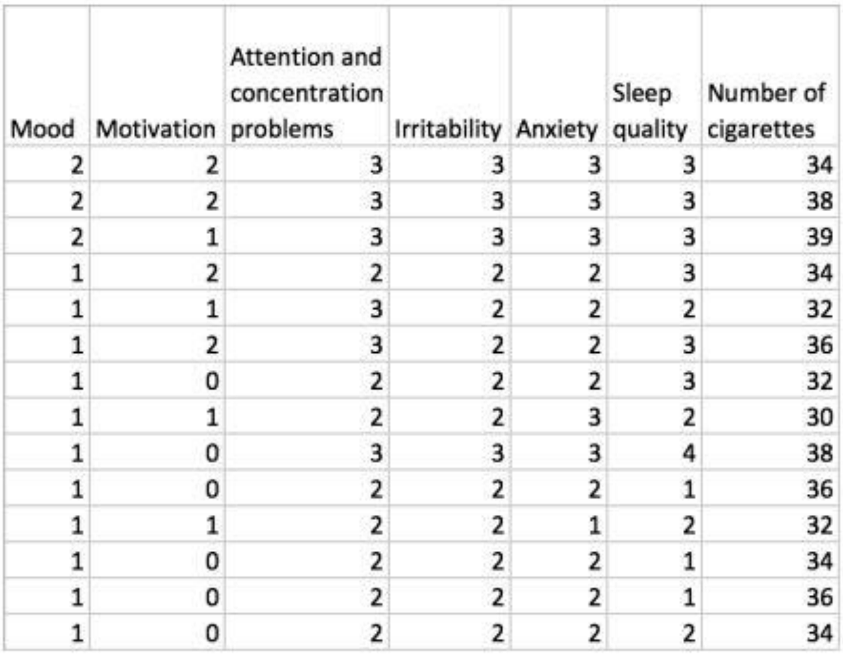
Figure 3.6 Fragment of the HDRS patient data (Part 2)

3.1.1.4 Interview data set

This is the fourth data set which is represented Interview data of the same patients mentioned in the Episode data set. It contains both physical and psychological items. The physical items represent more objective data and the psychological items represent more subjective data. Items which had been used in the interviews as follows:

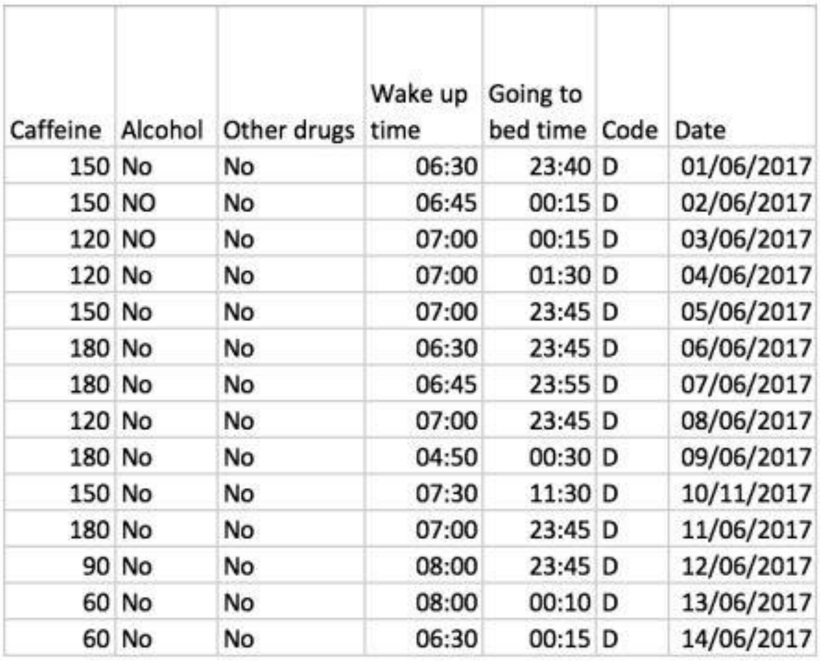
- Mood: mood level of the patient, ranging from -3 to 3.
- Motivation: motivation level of the patient, ranging from -3 to 3.
- Attention and concentration problems: level of attention and concentration problems of the patients, ranging from 0 to 4.
- Irritability: irritability level of the patient, ranging from 0 to 4.
- Anxiety: anxiety level of the patient, ranging from 0 to 4.
- Sleep quality: quality level of the patient's sleep, ranging from 0 to 4.
- Menstrual cycle: whether a female patient is in her menstrual cycle.
- Number of cigarettes: number of cigarettes smoked by the patient since he or she woke up.
- Caffeine: amount of caffeine ingested by the patient snice he or she woke up.
- Alcohol: whether the patient has consumed any alcohol.
- Other drugs: whether the patient has consumed any other drugs.
- Wake up time: when the patient woke up.
- Going to bed time: when the patient went to bed.
- Code: patient code (first letter of the patient's name).
- Date: when the questionnaire was completed.

As shown in following figures *3.7* and *3.8*, a set of Interview data has been represented with above those both physical and psychological items with their values.



Mood	Motivation	Attention and concentration problems	Irritability	Anxiety	Sleep quality	Number of cigarettes
2	2	3	3	3	3	34
2	2	3	3	3	3	38
2	1	3	3	3	3	39
1	2	2	2	2	3	34
1	1	3	2	2	2	32
1	2	3	2	2	3	36
1	0	2	2	2	3	32
1	1	2	2	3	2	30
1	0	3	3	3	4	38
1	0	2	2	2	1	36
1	1	2	2	1	2	32
1	0	2	2	2	1	34
1	0	2	2	2	1	36
1	0	2	2	2	2	34

Figure 3.7 Interview data (Part 1)



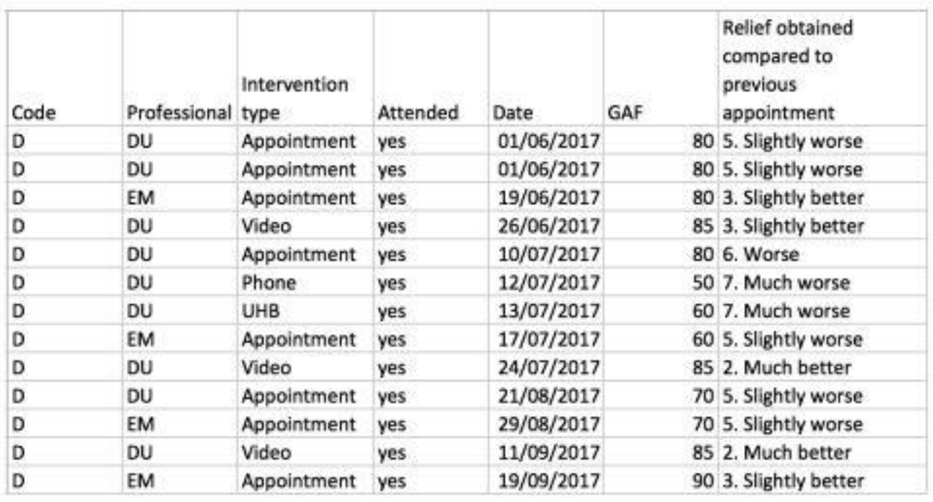
Caffeine	Alcohol	Other drugs	Wake up time	Going to bed time	Code	Date
150	No	No	06:30	23:40	D	01/06/2017
150	NO	No	06:45	00:15	D	02/06/2017
120	NO	No	07:00	00:15	D	03/06/2017
120	No	No	07:00	01:30	D	04/06/2017
150	No	No	07:00	23:45	D	05/06/2017
180	No	No	06:30	23:45	D	06/06/2017
180	No	No	06:45	23:55	D	07/06/2017
120	No	No	07:00	23:45	D	08/06/2017
180	No	No	04:50	00:30	D	09/06/2017
150	No	No	07:30	11:30	D	10/11/2017
180	No	No	07:00	23:45	D	11/06/2017
90	No	No	08:00	23:45	D	12/06/2017
60	No	No	08:00	00:10	D	13/06/2017
60	No	No	06:30	00:15	D	14/06/2017

Figure 3.8 Interview data (Part 2)

3.1.1.5 Intervention data set

This is the final data set which is represented intervention data of the same patients mentioned in the Episode data set.

These intervention data have been contained some videos, medical appointments, phone calls and interventions at Brief Hospitalization Units which are hospitalization units that provide a safe environment to disorder. It has also been contained a value for GAF (Global Assessment of Functioning) which evaluates symptoms, normal activities and interpersonal relationships of the patient.



Code	Professional	Intervention type	Attended	Date	GAF	Relief obtained compared to previous appointment
D	DU	Appointment	yes	01/06/2017	80	5. Slightly worse
D	DU	Appointment	yes	01/06/2017	80	5. Slightly worse
D	EM	Appointment	yes	19/06/2017	80	3. Slightly better
D	DU	Video	yes	26/06/2017	85	3. Slightly better
D	DU	Appointment	yes	10/07/2017	80	6. Worse
D	DU	Phone	yes	12/07/2017	50	7. Much worse
D	DU	UHB	yes	13/07/2017	60	7. Much worse
D	EM	Appointment	yes	17/07/2017	60	5. Slightly worse
D	DU	Video	yes	24/07/2017	85	2. Much better
D	DU	Appointment	yes	21/08/2017	70	5. Slightly worse
D	EM	Appointment	yes	29/08/2017	70	5. Slightly worse
D	DU	Video	yes	11/09/2017	85	2. Much better
D	EM	Appointment	yes	19/09/2017	90	3. Slightly better

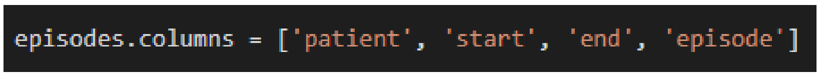
Figure 3.9 Medical Interventions

3.1.2 Data Cleaning

Data cleaning is a so important step in every machine learning project because, the ultimate result has completely based on the correctness of the data. Here, as shown in the following, the process of cleaning the dataset to obtain clean sets that can be used to train deep learning algorithms. Each data set has been cleaned separately because, they all had different sizes and structures.

3.1.2.1 Episode data set

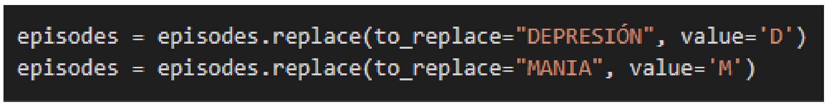
By giving proper names for each column in the data set, similar format could be obtained for each data set as shown in the following figure.



```
episodes.columns = ['patient', 'start', 'end', 'episode']
```

Figure 3.10 Episode data set columns

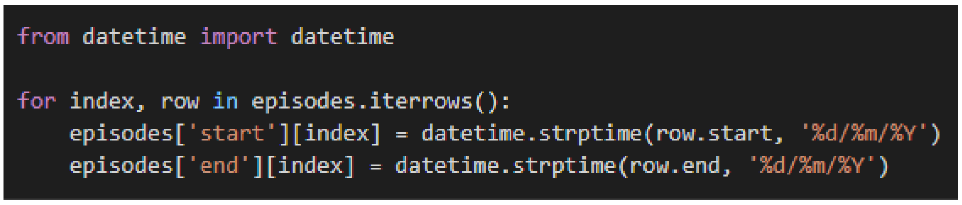
The episodes that the patients had gone through were recorded as DEPRESSION and MANIA, that means depression and mania respectively. In order to make the data set cleaner, the codes were changed to English letters ‘D’ and ‘M’ respectively for depression and mania.



```
episodes = episodes.replace(to_replace="DEPRESIÓN", value='D')
episodes = episodes.replace(to_replace="MANIA", value='M')
```

Figure 3.11 Replacing Depression with D and Mania with M

Originally, dates in the episode data set were in the String format. Therefore, they were converted into *datetime* format which is a Python module for manipulating dates as shown in the following figure.

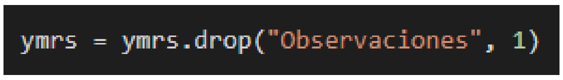


```
from datetime import datetime

for index, row in episodes.iterrows():
    episodes['start'][index] = datetime.strptime(row.start, '%d/%m/%Y')
    episodes['end'][index] = datetime.strptime(row.end, '%d/%m/%Y')
```

Figure 3.12 Changing data format

3.1.2.2 YMRS data set

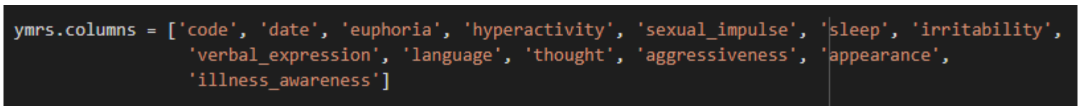
Originally YMRS data set consisted of fourteen columns. But the column named “*Observation*” that was filled with NAN (Not a Number) values. Since it did not add any value to the entire data set, it was dropped by using a Python method called *drop()* as shown in the following.



```
ymrs = ymrs.drop("Observaciones", 1)
```

Figure 3.13 Dropping observation column

As the next step, remained all thirteen columns were renamed in the same way as which is utilized with the Episode data set.



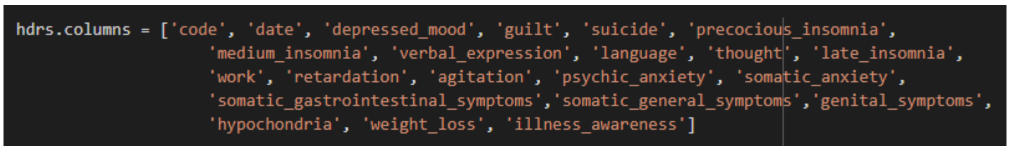
```
ymrs.columns = ['code', 'date', 'euphoria', 'hyperactivity', 'sexual_impulse', 'sleep', 'irritability',
                'verbal_expression', 'language', 'thought', 'aggressiveness', 'appearance',
                'illness_awareness']
```

Figure 3.14 YMRS data set columns

Then after that by using Pandas *isnull()* function, it was needed to check whether the data set was contained with any empty(null) values. After confirming that there were no empty values, the dates column in YMRS data set was converted into *datetime* format as in the Episode data set.

3.1.2.3 HDRS data set

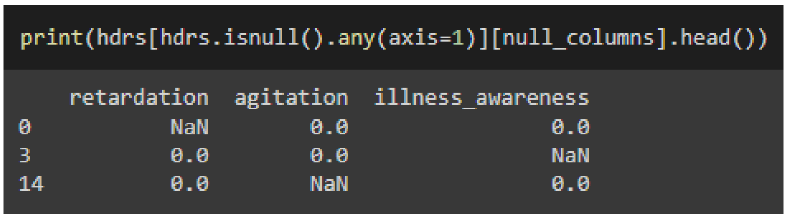
As done in the YMRS data set, initially started by dropping the column named “*Type of intevention*” as the same way used in the YMRS data set because it only had NaN values. An also renamed all remained columns as done in the earlier data sets.



```
hdrs.columns = ['code', 'date', 'depressed_mood', 'guilt', 'suicide', 'precocious_insomnia',
                'medium_insomnia', 'verbal_expression', 'language', 'thought', 'late_insomnia',
                'work', 'retardation', 'agitation', 'psychic_anxiety', 'somatic_anxiety',
                'somatic_gastrointestinal_symptoms','somatic_general_symptoms','genital_symptoms',
                'hypochondria', 'weight_loss', 'illness_awareness']
```

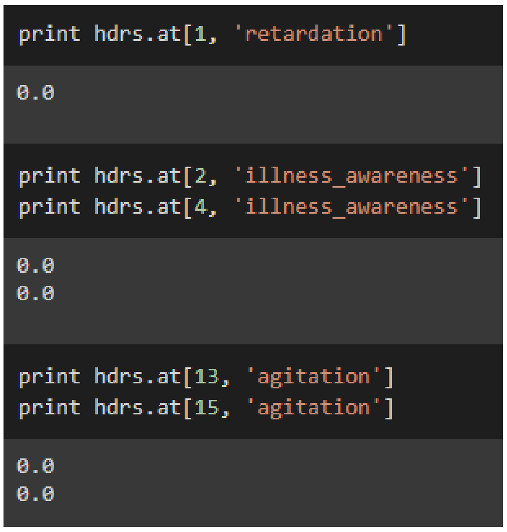
Figure 3.15 HDRS data set columns

Then after that, it could be proved that the HDRS data set was contained with empty values with the help of *isnull()* function. Then all null values were proceeded as null. By printing the values of the rows before and after these empty values, it could be get an idea of which values that could be filled them with, which in this case was the same for all.



```
print(hdrs[hdrs.isnull().any(axis=1)][null_columns].head())

    retardation  agitation  illness_awareness
0           NaN        0.0                0.0
3           0.0        0.0                NaN
14          0.0        NaN                0.0
```

Figure 3.16 Checking the rows where the previous columns are null



```
print hdrs.at[1, 'retardation']
```

```
0.0
```

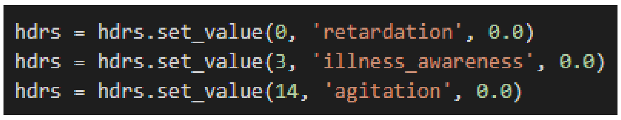
```
print hdrs.at[2, 'illness_awareness']
print hdrs.at[4, 'illness_awareness']
```

```
0.0
0.0
```

```
print hdrs.at[13, 'agitation']
print hdrs.at[15, 'agitation']
```

```
0.0
0.0
```

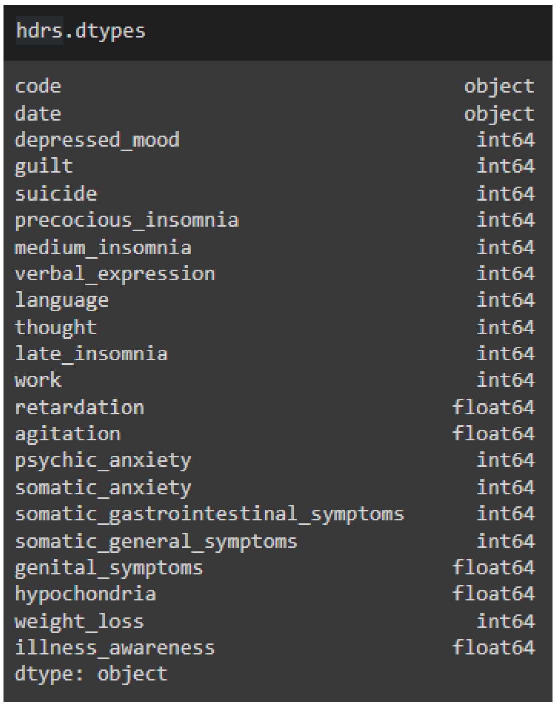
Figure 3.17 Replacing null values with values which are similar as previous and next row values



```
hdrs = hdrs.set_value(0, 'retardation', 0.0)
hdrs = hdrs.set_value(3, 'illness_awareness', 0.0)
hdrs = hdrs.set_value(14, 'agitation', 0.0)
```

Figure 3.18 Filling the missing values with the replacement

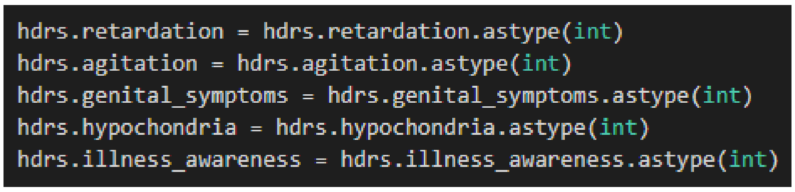
As shown in the above figures, all empty values were filled with the ones which were obtained from those rows. The next step was to format all the values in the data set so that they had the same type, because some values had Floating Point type and others had Integer type. So, all were changed into Integer.



```
hdrs.dtypes
```

```
code                                 object
date                                 object
depressed_mood                        int64
guilt                                 int64
suicide                               int64
precocious_insomnia                   int64
medium_insomnia                       int64
verbal_expression                     int64
language                              int64
thought                               int64
late_insomnia                         int64
work                                  int64
retardation                         float64
agitation                           float64
psychic_anxiety                       int64
somatic_anxiety                       int64
somatic_gastrointestinal_symptoms     int64
somatic_general_symptoms              int64
genital_symptoms                    float64
hypochondria                        float64
weight_loss                           int64
illness_awareness                   float64
dtype: object
```

Figure 3.19 HGRS data columns with data types



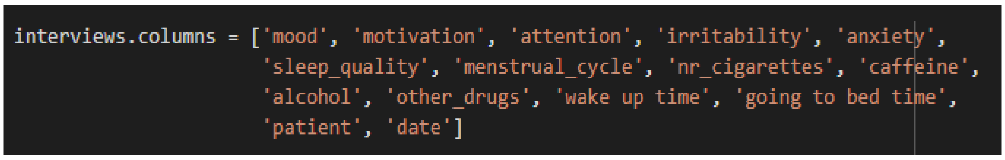
```
hdrs.retardation = hdrs.retardation.astype(int)
hdrs.agitation = hdrs.agitation.astype(int)
hdrs.genital_symptoms = hdrs.genital_symptoms.astype(int)
hdrs.hypochondria = hdrs.hypochondria.astype(int)
hdrs.illness_awareness = hdrs.illness_awareness.astype(int)
```

Figure 3.20 Converting all Floating Points into Integers

After that, just done in both Episode and YMRS data sets, the format of the date column was changed into *datetime*.

3.1.2.4. Interview data set

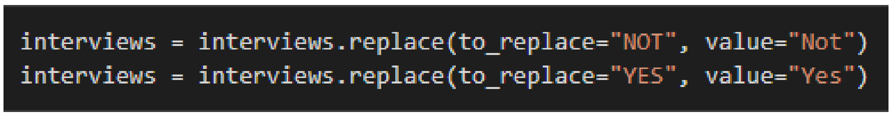
Interview data set is the one which had biggest amount of data and therefore it was a big challenge when cleaning it. The first step was to rename the columns as done with other data sets in the above.



```
interviews.columns = ['mood', 'motivation', 'attention', 'irritability', 'anxiety',
                      'sleep_quality', 'menstrual_cycle', 'nr_cigarettes', 'caffeine',
                      'alcohol', 'other_drugs', 'wake up time', 'going to bed time',
                      'patient', 'date']
```

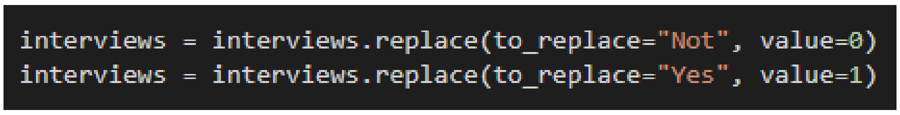
Figure 3.21 Interview data set columns

The next step was to unify the qualitative values (NOT and YES) and map them by giving numerical values (0 and 1) respectively.



```
interviews = interviews.replace(to_replace="NOT", value="Not")
interviews = interviews.replace(to_replace="YES", value="Yes")
```

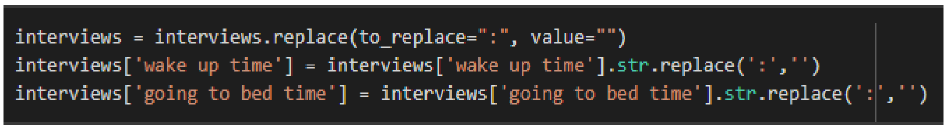
Figure 3.22 Unifying the qualitative values



```
interviews = interviews.replace(to_replace="Not", value=0)
interviews = interviews.replace(to_replace="Yes", value=1)
```

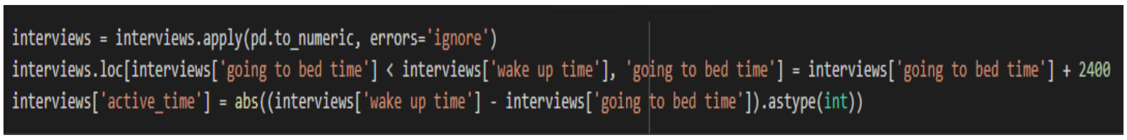
Figure 3.23 Mapping them by giving numerical values 0 and 1 respectively

As the sleeping time overlapped different days (foe an example, the patients might have slept from 10 pm one day to 9 am the day after), it was easier to know how many hours the patient had been active instead of the hours of sleep. If the patient had gone to bed after midnight, it had been to add 24 hours in order to get the correct amount of time that the patient had been active. Before calculate the active time, the time stamps had been to format by removing the columns. This entire process can be seen as follows.



```
interviews = interviews.replace(to_replace=":", value="")
interviews['wake up time'] = interviews['wake up time'].str.replace(':','')
interviews['going to bed time'] = interviews['going to bed time'].str.replace(':','')
```

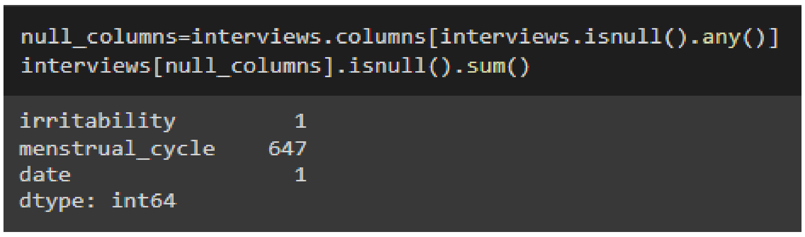
Figure 3.24 Remove the columns from the strings in the data frame



```
interviews = interviews.apply(pd.to_numeric, errors='ignore')
interviews.loc[interviews['going to bed time'] < interviews['wake up time'], 'going to bed time'] = interviews['going to bed time'] + 2400
interviews['active_time'] = abs((interviews['wake up time'] - interviews['going to bed time']).astype(int))
```

Figure 3.25 Calculating the amount of time that the patient has been active

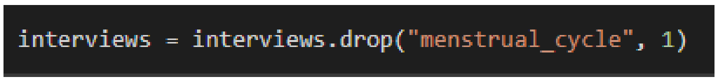
After calculating the amount of active time, both *Wake up time* and *Going to bed time* columns were dropped as they were no longer relevant. As the next step, data set was checked whether it had any empty values or not. As a result of that, menstrual_cycle could be found as a column where all rows were empty. Then, that entire column was dropped because it did not add any value to the data set.



```
null_columns=interviews.columns[interviews.isnull().any()]
interviews[null_columns].isnull().sum()
```

```
irritability         1
menstrual_cycle    647
date                 1
dtype: int64
```

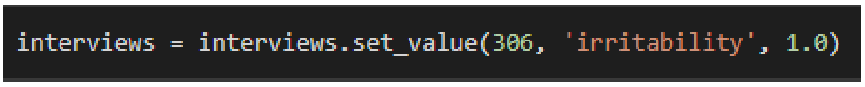
Figure 3.26 Checking which columns have null values and how many there are



```
interviews = interviews.drop("menstrual_cycle", 1)
```

Figure 3.27 Dropping ‘menstrual_cycle’ column since its all rows contain null values

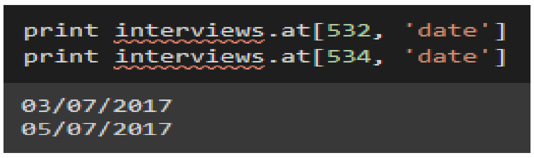
After that, irritability value that was empty with the previous value and the one after it to see if they were similar. Then it could be seen that both were 1.0, so those empty spaces were filled with that value. The other value that was missing was a date, which would most surely be between two other days. Therefore, the value was set to the missing day between them.
In the case of the ‘irritability’ value, as done with HDRS data frame, replaced all null values with 1.0 as shown in the following figure.



```
interviews = interviews.set_value(306, 'irritability', 1.0)
```

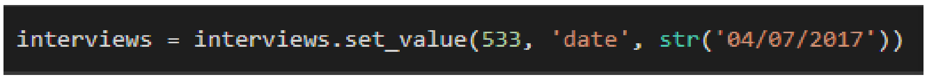
Figure 3.28 Replacing null values with values which are similar as previous and next row values

In case of date, just one date has been missed. Since the interview must have been after the previous one and before the one in the next row, missed date was filled as follows.



```
print interviews.at[532, 'date']
print interviews.at[534, 'date']

03/07/2017
05/07/2017
```

Figure 3.29 Checking the previous and next rows dates



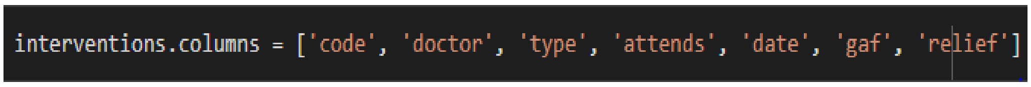
```
interviews = interviews.set_value(533, 'date', str('04/07/2017'))
```

Figure 3.30 Replacing the date

After filling the empty values, all Floating-Point values were changed into Integer and changed the dates to datetime as done in the previous.

3.1.2.5 Intervention data set

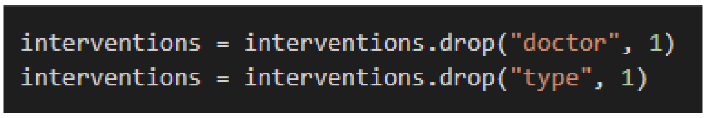
As done in previous, at first data set was renamed.



```
interventions.columns = ['code', 'doctor', 'type', 'attends', 'date', 'gaf', 'relief']
```

Figure 3.31 Intervention data set columns

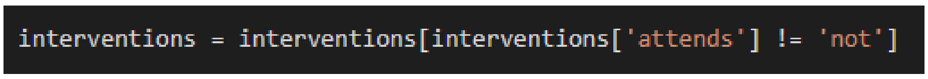
Then after, rows which contained NaN values in the column named gaf were dropped since prevent any risk. The doctor and intervention types were also dropped because they did not add any value to the data set.



```
interventions = interventions.drop("doctor", 1)
interventions = interventions.drop("type", 1)
```

Figure 3.32 Dropping both ‘doctor’ and ‘type’ columns

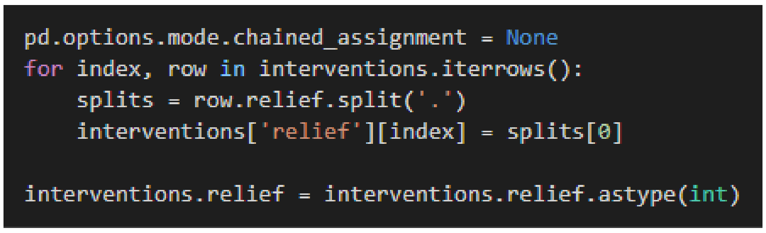
The interventions that the patients did not attend to were not valuable either, so these were dropped from the data set. It means that, columns which indicated if the patient attended or not were got rid.



```
interventions = interventions[interventions['attends'] != 'not']
```

Figure 3.33 Dropping rows with zero attendance of patients

Then after that, the strings were split and taken only the number which range from 1 to 7 as shown in the following figure.



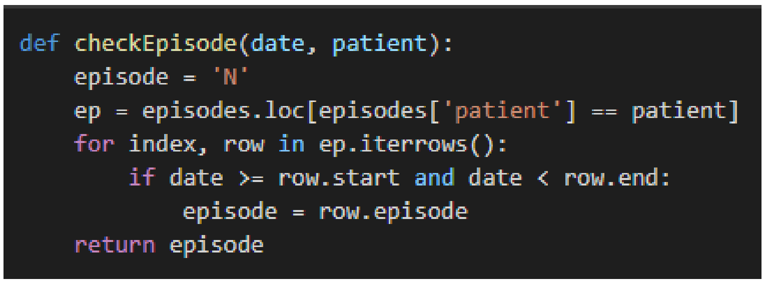
```
pd.options.mode.chained_assignment = None
for index, row in interventions.iterrows():
    splits = row.relief.split('.')
    interventions['relief'][index] = splits[0]

interventions.relief = interventions.relief.astype(int)
```

Figure 3.34 Taking the numbers in range 1 to 7

Ultimately, changed the dates to datetime as done in the previous data sets.

3.1.3 Data Combination

This part of the project represents the process of combining the data in different ways to make new data set with different combinations of features. The goal was to find data set combinations that have enough data for the algorithms to process. A function is used to get the combinations. This function obtained the date of each entry and compared it with the different episodes of depression and mania in the Episode data set, which was the target of the prediction. It was assumed that if the dates from the entries of the data set are not include in the episode intervals from the Episode dataset, the patient was in a normal state. This way there are three possible states a patient can be in (Depression -> ‘D’, Mania -> ‘M’ and Normal -> ‘N’).

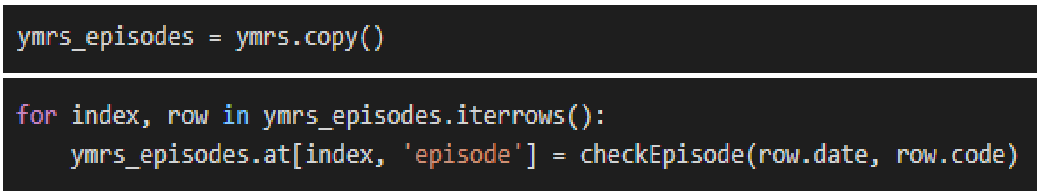


```python
def checkEpisode(date, patient):
    episode = 'N'
    ep = episodes.loc[episodes['patient'] == patient]
    for index, row in ep.iterrows():
        if date >= row.start and date < row.end:
            episode = row.episode
    return episode
```

Figure 3.35 Function defined to obtain the episode

3.1.3.1 Combining YMRS and Episode data sets

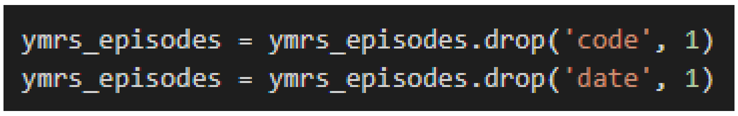
The data sets were combined with the help of the function that returned the episode by using the date of each row and create a new column in the YMRS data set named episode.



```python
ymrs_episodes = ymrs.copy()
```

```python
for index, row in ymrs_episodes.iterrows():
    ymrs_episodes.at[index, 'episode'] = checkEpisode(row.date, row.code)
```

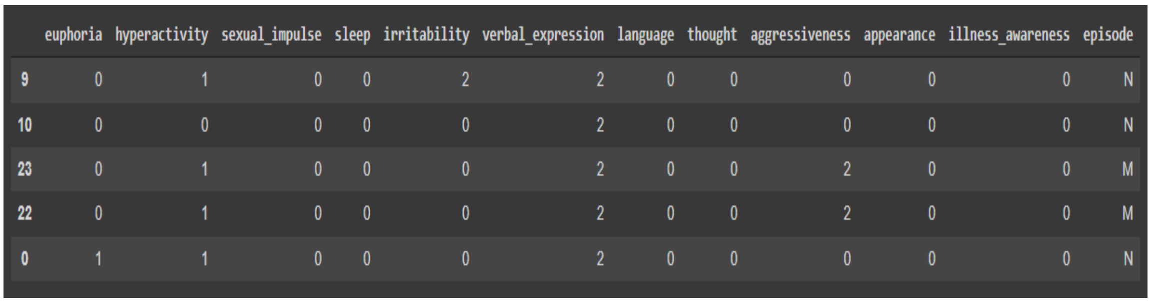
Figure 3.36 Combing YMRS and Episode data sets



```python
ymrs_episodes = ymrs_episodes.drop('code', 1)
ymrs_episodes = ymrs_episodes.drop('date', 1)
```

Figure 3.37 Dropping code and date columns in YMRS and Episode data sets

The ‘code’ and ‘date’ columns were dropped after merging, as they were no longer needed. The dataset that was created after combining YMRS and Episode datasets as follows.

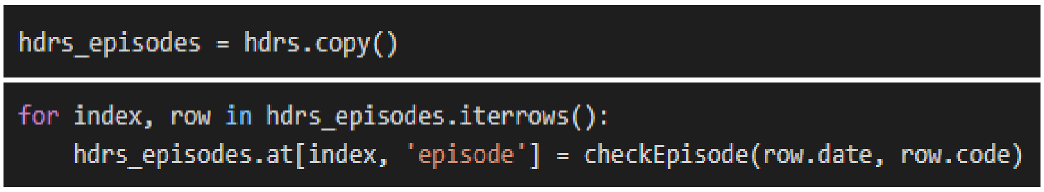


	euphoria	hyperactivity	sexual_impulse	sleep	irritability	verbal_expression	language	thought	aggressiveness	appearance	illness_awareness	episode
9	0	1	0	0	2	2	0	0	0	0	0	N
10	0	0	0	0	0	2	0	0	0	0	0	N
23	0	1	0	0	0	2	0	0	2	0	0	M
22	0	1	0	0	0	2	0	0	2	0	0	M
0	1	1	0	0	0	2	0	0	0	0	0	N

Figure 3.38 YMRS_Episode data set

3.1.3.2 Combining HDRS and Episode data sets

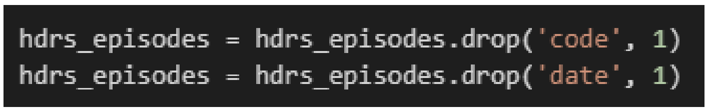
The data sets were combined with the help of the function used in previous and after merging both sets, again 'code' and 'date' columns were dropped since no longer needed.



```
hdrs_episodes = hdrs.copy()
```

```
for index, row in hdrs_episodes.iterrows():
    hdrs_episodes.at[index, 'episode'] = checkEpisode(row.date, row.code)
```

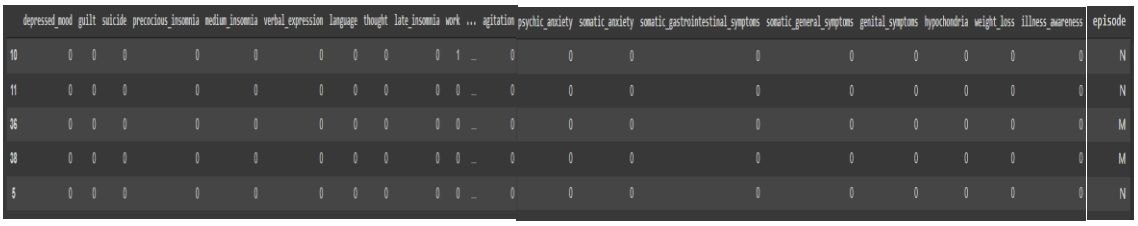
Figure 3.39 Combing HDRS and Episode data sets



```
hdrs_episodes = hdrs_episodes.drop('code', 1)
hdrs_episodes = hdrs_episodes.drop('date', 1)
```

Figure 3.40 Dropping code and date columns in HDRS and Episode data sets

The data set that was created after combining HDRS and Episode datasets as follows.

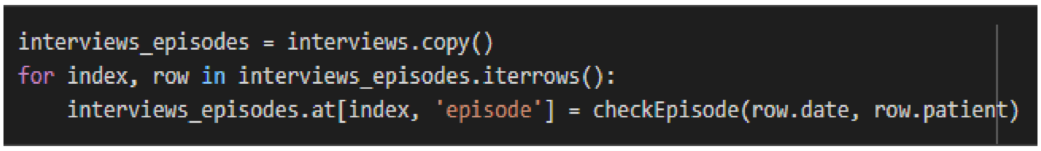


	depressed_mood	guilt	suicide	precocious_insomnia	medium_insomnia	verbal_expression	language	thought	late_insomnia	work	...	agitation	psychic_anxiety	somatic_anxiety	somatic_gastrointestinal_symptoms	somatic_general_symptoms	genital_symptoms	hypochondria	weight_loss	illness_awareness	episode
10	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	N
11	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	N
36	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	M
38	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	M
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	N

Figure 3.41 HDRS_Episode data set

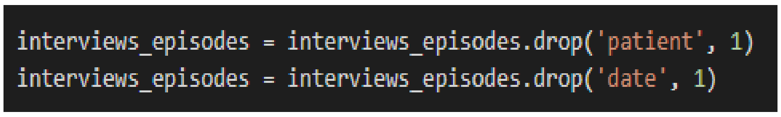
3.1.3.3 Combining Interview and Episode data sets

The data sets were combined with the help of the function used in previous and after merging both sets, 'date' and 'patient' columns were dropped because they did not give any useful information for the prediction.



```
interviews_episodes = interviews.copy()
for index, row in interviews_episodes.iterrows():
    interviews_episodes.at[index, 'episode'] = checkEpisode(row.date, row.patient)
```

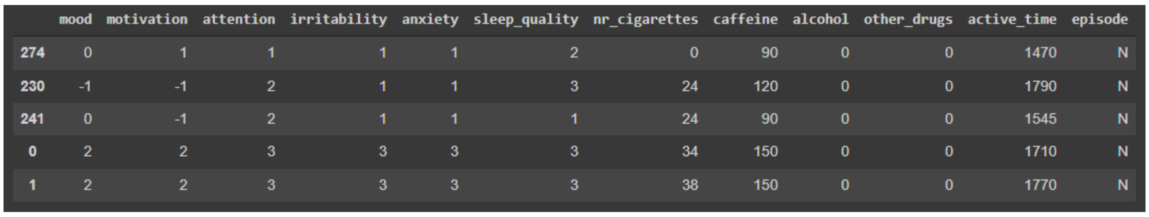
Figure 3.42 Combing Interview and Episode data sets



```
interviews_episodes = interviews_episodes.drop('patient', 1)
interviews_episodes = interviews_episodes.drop('date', 1)
```

Figure 3.43 Dropping patient and date columns in Interview and Episode data sets

The data set that was created after combining Interview and Episode data sets as follows.

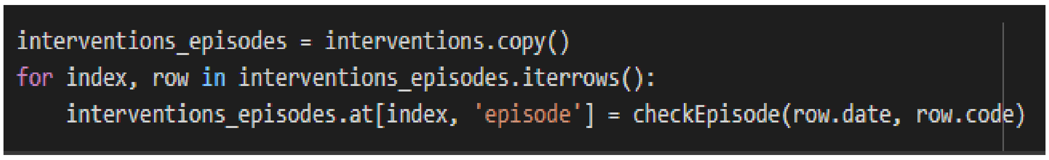


	mood	motivation	attention	irritability	anxiety	sleep_quality	nr_cigarettes	caffeine	alcohol	other_drugs	active_time	episode
274	0	1	1	1	1	2	0	90	0	0	1470	N
230	-1	-1	2	1	1	3	24	120	0	0	1790	N
241	0	-1	2	1	1	1	24	90	0	0	1545	N
0	2	2	3	3	3	3	34	150	0	0	1710	N
1	2	2	3	3	3	3	38	150	0	0	1770	N

Figure 3.44 Interview_Episode data set

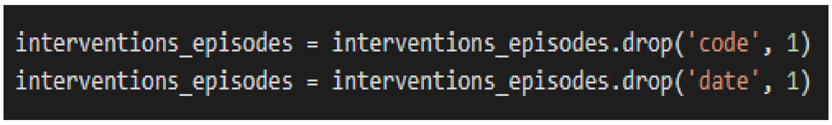
3.1.3.4 Combining Intervention and Episode data sets

The data sets were combined with the help of the function used in the above and after merging the both data sets, again 'code' and 'date' columns were dropped since they did not give any useful information for the prediction.



```
interventions_episodes = interventions.copy()
for index, row in interventions_episodes.iterrows():
    interventions_episodes.at[index, 'episode'] = checkEpisode(row.date, row.code)
```

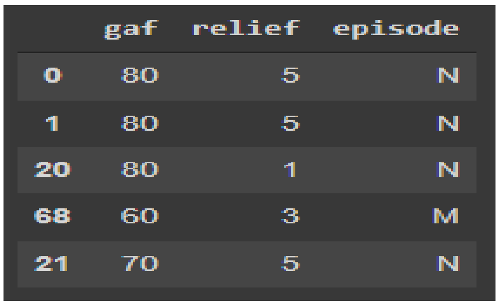
Figure 3.45 Combing Intervention and Episode data sets



```
interventions_episodes = interventions_episodes.drop('code', 1)
interventions_episodes = interventions_episodes.drop('date', 1)
```

Figure 3.46 Dropping code and date columns in Intervention and Episode data sets

The data set that was created after combining Intervention and Episode data sets as follows.



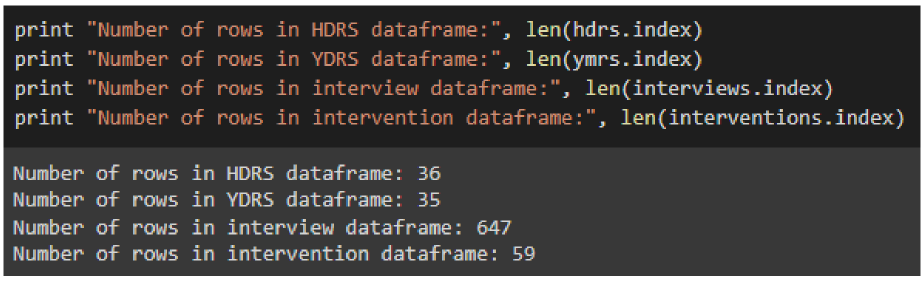
	gaf	relief	episode
0	80	5	N
1	80	5	N
20	80	1	N
68	60	3	M
21	70	5	N

Figure 3.47 Intervention_Episode data set

3.1.3.5 Combining YMRS and HDRS data sets

Since, the ultimate goal is to combine all the data sets by date to get a data set with many features, in order to later check if the prediction accuracy is higher than the rest of data sets. Then, after making different combination of data sets, it can be seen that there are enough entries to test the algorithms.

Before combining the data sets, let’s have a look on the number of rows they have.



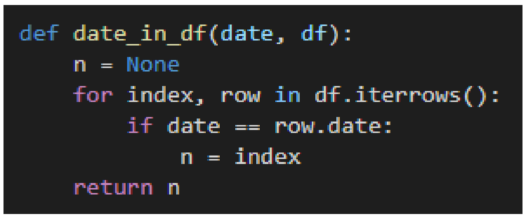
```
print "Number of rows in HDRS dataframe:", len(hdrs.index)
print "Number of rows in YDRS dataframe:", len(ymrs.index)
print "Number of rows in interview dataframe:", len(interviews.index)
print "Number of rows in intervention dataframe:", len(interventions.index)
```

```
Number of rows in HDRS dataframe: 36
Number of rows in YDRS dataframe: 35
Number of rows in interview dataframe: 647
Number of rows in intervention dataframe: 59
```

Figure 3.48 Num of rows in each data set

As shown in the above figure, YDRS data set have relatively less rows. Therefore, let’s iterate over it and combine it with the other three data sets.

At first, in order to find the entries with the same date, a special function was defined that checked if there was an entry in a data set with given date and, if it existed, returned the index entry, as shown in the following figure.

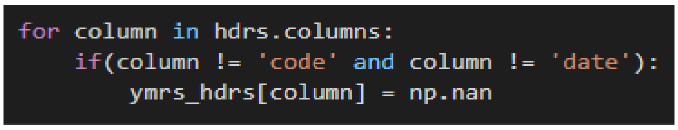


```
def date_in_df(date, df):
    n = None
    for index, row in df.iterrows():
        if date == row.date:
            n = index
    return n
```

Figure 3.49 Function defined to match a given date

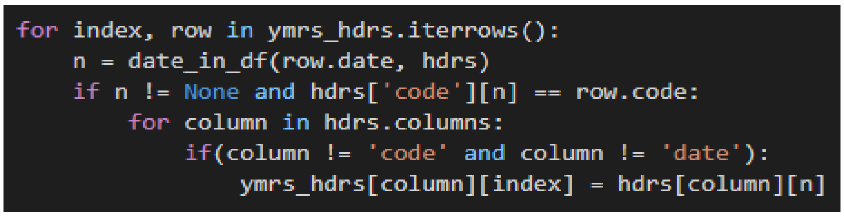
For each date in the YMRS data set, this function can be checked if it existed in HDRS data set too by iterative over it, so that it could be able to combine the features of the rows in both data sets into a same row of the combined data set.

After initializing the empty columns in the YMRS data set, it was combined with the HDRS data set by iterating over every entry and checking the date in the HDRS data set as follows.



```
for column in hdrs.columns:
    if(column != 'code' and column != 'date'):
        ymrs_hdrs[column] = np.nan
```

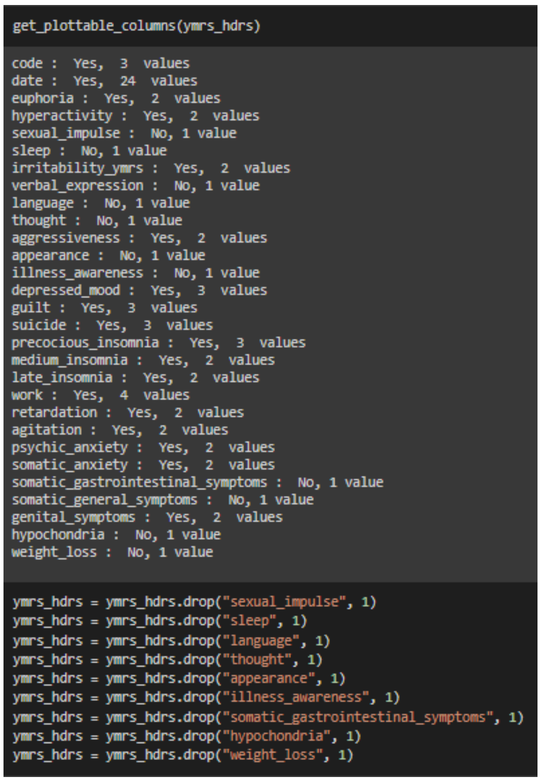
Figure 3.50 Initializing empty columns in the YMRS data set



```
for index, row in ymrs_hdrs.iterrows():
    n = date_in_df(row.date, hdrs)
    if n != None and hdrs['code'][n] == row.code:
        for column in hdrs.columns:
            if(column != 'code' and column != 'date'):
                ymrs_hdrs[column][index] = hdrs[column][n]
```

Figure 3.51 Combining YMRS and HDRS data sets

Then after that, YMRS_HDRS data set was proceeded to drop the columns that only have one value as follows.



```
get_plottable_columns(ymrs_hdrs)
```

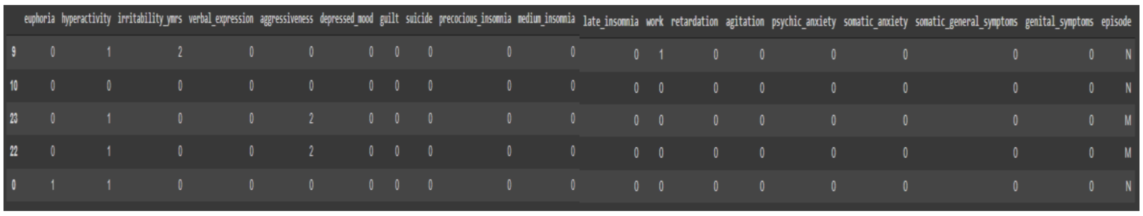
```
code :  Yes,  3  values
date :  Yes,  24  values
euphoria :  Yes,  2  values
hyperactivity :  Yes,  2  values
sexual_impulse :  No, 1 value
sleep :  No, 1 value
irritability_ymrs :  Yes,  2  values
verbal_expression :  No, 1 value
language :  No, 1 value
thought :  No, 1 value
aggressiveness :  Yes,  2  values
appearance :  No, 1 value
illness_awareness :  No, 1 value
depressed_mood :  Yes,  3  values
guilt :  Yes,  3  values
suicide :  Yes,  3  values
precocious_insomnia :  Yes,  3  values
medium_insomnia :  Yes,  2  values
late_insomnia :  Yes,  2  values
work :  Yes,  4  values
retardation :  Yes,  2  values
agitation :  Yes,  2  values
psychic_anxiety :  Yes,  2  values
somatic_anxiety :  Yes,  2  values
somatic_gastrointestinal_symptoms :  No, 1 value
somatic_general_symptoms :  No, 1 value
genital_symptoms :  Yes,  2  values
hypochondria :  No, 1 value
weight_loss :  No, 1 value
```

```
ymrs_hdrs = ymrs_hdrs.drop("sexual_impulse", 1)
ymrs_hdrs = ymrs_hdrs.drop("sleep", 1)
ymrs_hdrs = ymrs_hdrs.drop("language", 1)
ymrs_hdrs = ymrs_hdrs.drop("thought", 1)
ymrs_hdrs = ymrs_hdrs.drop("appearance", 1)
ymrs_hdrs = ymrs_hdrs.drop("illness_awareness", 1)
ymrs_hdrs = ymrs_hdrs.drop("somatic_gastrointestinal_symptoms", 1)
ymrs_hdrs = ymrs_hdrs.drop("hypochondria", 1)
ymrs_hdrs = ymrs_hdrs.drop("weight_loss", 1)
```

Figure 3.52 Dropping the columns that only have one value

After dropping columns with one value, as done in YMRS_Episode, HDRS_Episode, interview_Episode and Intervention_Episode data sets, after checking the data types of rest of the all columns, each Floating-Point data type is converted onto Integer type. Then after, it is combined with the Episode data set and both ‘date’ and ‘code’ columns are dropped as no longer needed for the future predictions.

The YMRS_HDRS data set with 25 entries after completing all of these steps as follows.

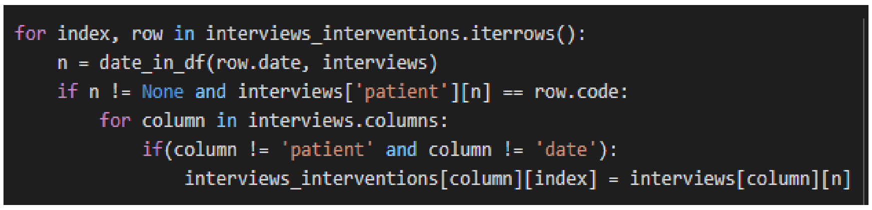


	euphoria	hyperactivity	irritability_ymrs	verbal_expression	aggressiveness	depressed_mood	guilt	suicide	precocious_insomnia	medium_insomnia	late_insomnia	work	retardation	agitation	psychic_anxiety	somatic_anxiety	somatic_general_symptoms	genital_symptoms	episode
9	0	1	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	N
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	N
23	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	M
22	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	M
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	N

Figure 3.53 YMRS_HDRS data set

3.1.3.6 Combining Interview and Intervention data sets

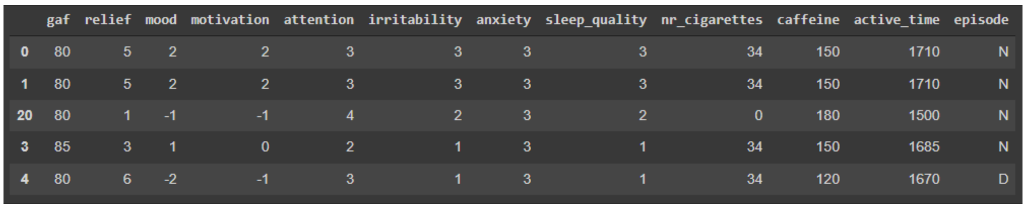
These two data sets are combined in the exact same way as done with YMRS and HDRS data sets, by iterating over the Intervention data set which has fewer values than the Interview data set.



```python
for index, row in interviews_interventions.iterrows():
    n = date_in_df(row.date, interviews)
    if n != None and interviews['patient'][n] == row.code:
        for column in interviews.columns:
            if(column != 'patient' and column != 'date'):
                interviews_interventions[column][index] = interviews[column][n]
```

Figure 3.54 Combining Interview and Intervention data sets

After dropping NaN values and columns with just one value, as done in the previous, after checking the data types of the rest of the all columns, each Floating Point data type is converted into Integer data type. Then after, it is merged with the Episode data set and both ‘date’ and ‘code’ columns are dropped as no longer needed for the future prediction. The Interview_Intervention data set after completing all of those steps as follows.



	gaf	relief	mood	motivation	attention	irritability	anxiety	sleep_quality	nr_cigarettes	caffeine	active_time	episode
0	80	5	2	2	3	3	3	3	34	150	1710	N
1	80	5	2	2	3	3	3	3	34	150	1710	N
20	80	1	-1	-1	4	2	3	2	0	180	1500	N
3	85	3	1	0	2	1	3	1	34	150	1685	N
4	80	6	-2	-1	3	1	3	1	34	120	1670	D

Figure 3.55 Interview_Intervention data set

3.1.4 Application of algorithms

Same as Machine Learning, the two most common types of Deep Learning which are used at present are supervised leaning and unsupervised learning.

Typically, the supervised learning techniques based on a model that is created with the help of an algorithm which will predict an output for a certain input. It often counts with a set of data for which the input and output are known. This set of data is called as training set.

Basically, there are two types of Supervised Learning problems. One is regression and the other one is classification. Regression is a problem in which the output is quantitative. Classification problem is a problem in which the output is qualitative.

The target or the value that tried to predict in this project is the state of a patient who is in Episode, Depression or Euthymic state. Therefore, according to the ultimate target of this project, the problem that is presented at here is a classification problem.

Therefore, in this part, three classification deep learning algorithms are utilized on the data sets that was previously obtained from the data combination part.

Since the target is prediction a patient's state as mentioned in the above, all the combined data sets had to have a column with the episode associated to the rest of the features, which were specified in the combining data sets section. Such as,

- **YMRS**: Young Mania Rating Scale data (ymrs_episodes)
- **HDRS**: Hamilton Depression Rating Scale data (hdrs_episode)
- **Interviews**: Interview data (interview_episode)
- **Interventions**: Intervention data (intervention_episode)
- **YMRS_HDRS**: The combination of YMRS and HDRS data (ymrs_hdrs)
- **Interview_Intervention**: The combination of interview and intervention data (interview_intervention)

The classification algorithms that are used for this project are: **MLP** (Multi-Layer Perceptron), **ANN** (Artificial Neural Networks) using Keras-Sequential Model and **CNN** (Convolutional Neural Network).
The fact that these algorithms have been applied on this project does not mean that they are the most appropriate options for the classification of bipolar disorder states. Because, with such a small amount of data as was available for this project, it was really hard to obtain a higher accuracy for the ultimate target. But, those three classification Deep Learning algorithms were the most suitable ones which are most satisfied with the given amount of data and number of features used in this project.

As mentioned in the above, before applying each algorithm, it is essential to split each combined data sets into a training set and testing set. At first, the training data would be used to train the prediction models and after that, the testing data would be used to compare the output of the model with the real targets.

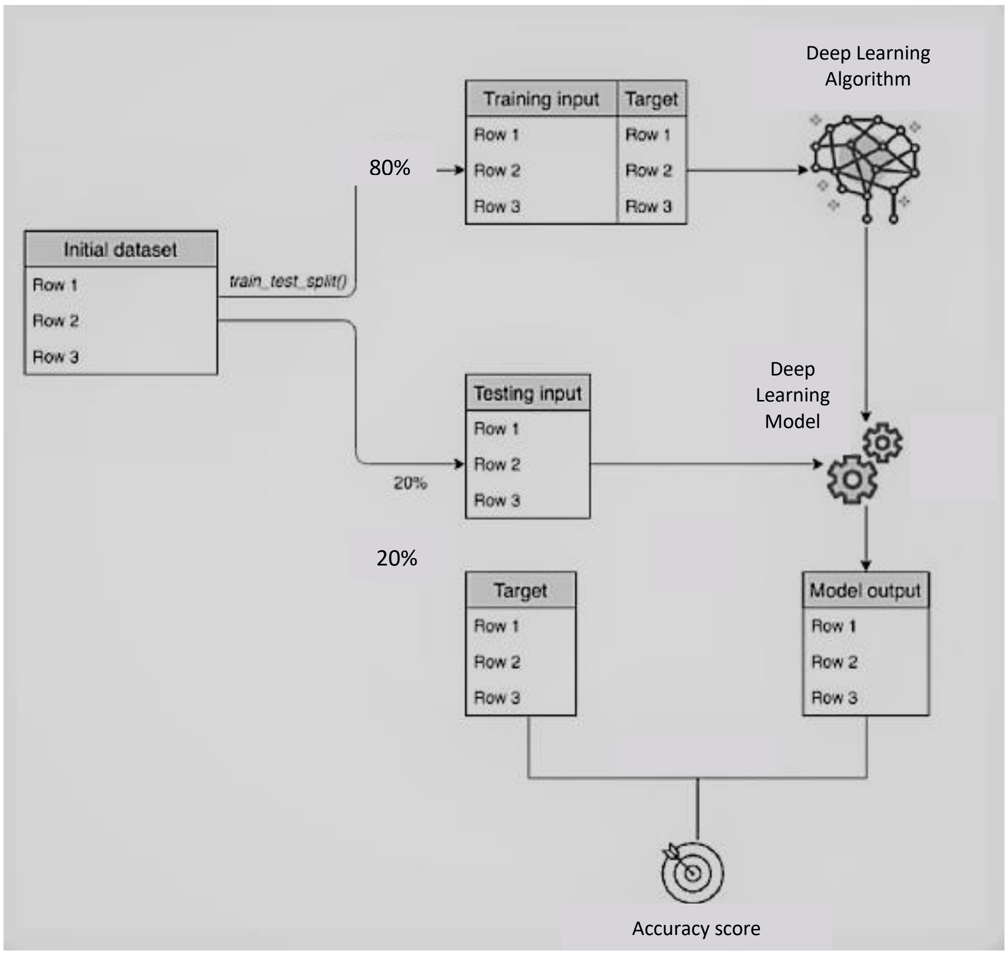


Figure 3.56 Deep Learning algorithm application process

The testing set is used for obtaining the accuracy of the model, as mentioned in the above. It is done by comparing the output which was obtained from the testing input and the real output of the testing set.

In order to divide the original combined data sets, *train_test_split()* function from the scikit-learn library is used. Then after that, all three algorithms are implemented. The detailed implementation can be seen under the Chapter 3.3. After the algorithms are applied, algorithm performance evaluation is done by using relevant functions. The overview of that part can be seen under the Chapter 3.2.

3.1.5 Algorithm performance evaluation

In this part, the performance of each algorithm when predicting the state of the patient is evaluated.

3.2 Selection of the section to be implemented

The classification algorithms that are used for this project are: **MLP** (Multi-Layer Perceptron), **ANN** (Artificial Neural Networks) using Keras-Sequential Model [13] and **CNN model** using keras Conv1D layer.

As mentioned in the chapter 3.1.4, theses three classification Deep Learning algorithms are the most suitable ones which are most satisfied with the given amount of data and number of features used in this project.

- **ANN** (Artificial Neural Networks) using Keras-Sequential Model

  Keras is a user-friendly neural network library which is written in Python. This model is appropriate for plane stack of layers where each layer has just one input tensor and one output tensor. So, this model can be used to create ANN (Artificial Neural Network).

  Since that categorical data sets can be used with this model, it has been decided as one appropriate algorithm.

- **MLP** (Multi-Layer Perceptron)

  This is also one of the type of artificial neural network (ANN). Simplest MLP can be consisted of at least three layers as one input layer, one hidden layer and one output layer. As well as that output layer can be available without any activation function. It can be utilized with single as well as multiple target value regression.

  As mentioned in the above, it is relatively easy to work with this algorithm using our data sets.

- **CNN** (Convolutional Neural Network) using keras Conv1D layer

  CNNs are feed-forward Artificial Neural Networks (ANNs) with alternating convolutional and subsampling layers. Deep 2D CNNs with many hidden layers and millions of parameters have the ability to learn complex objects and patterns. Sine this ability, 2D CNNs can be implemented for various engineering applications for 2D signals such as images and video frames.
  Yet, this may not be a viable option in numerous applications over ID signals. 1D CNNs have recently been proposed to address this issue and it can be implemented for personalized biomedical data classification and early diagnosis. That's the reason for selecting 1D CNN for this project.

3.3 Detailed design of the section to be implemented

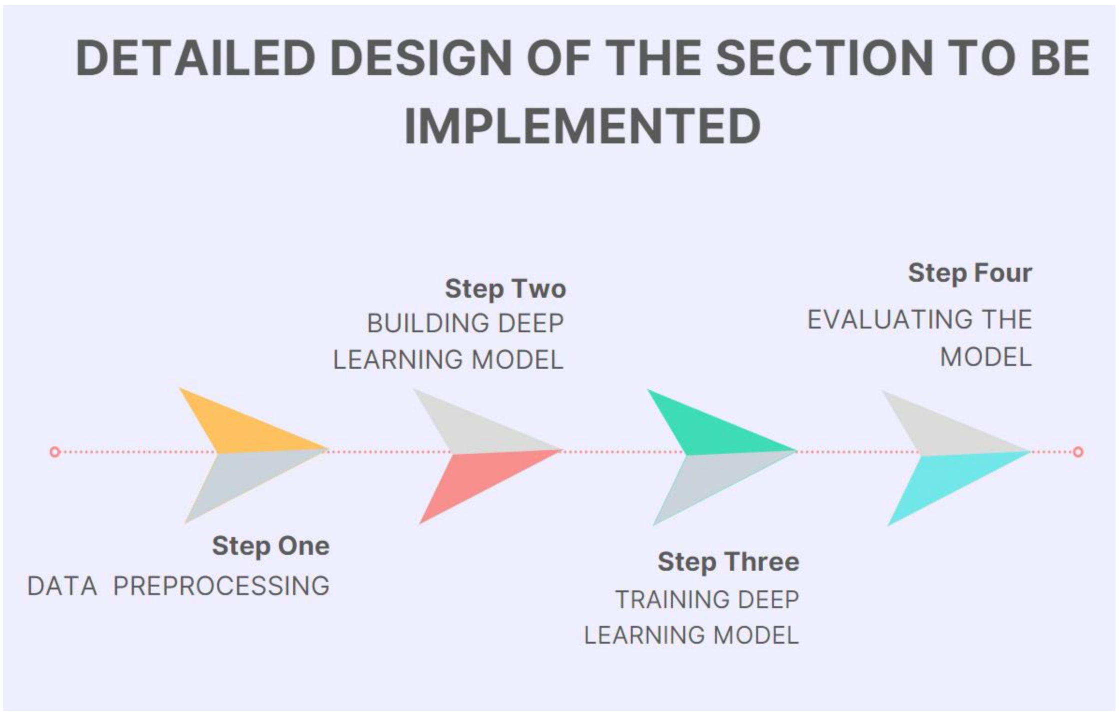

DETAILED DESIGN OF THE SECTION TO BE IMPLEMENTED
Step One
DATA PREPROCESSING
Step Two
BUILDING DEEP LEARNING MODEL
Step Three
TRAINING DEEP LEARNING MODEL
Step Four
EVALUATING THE MODEL

Figure 3. 57 Detailed designing of the section

As shown in the above figure, detailed designing of this project is going to be explained under this chapter. As mentioned in the chapter 3.1, the entire process that should be completed before the detailed designing part has been explained as the overall designing. So, in this chapter, data preprocessing, building deep learning model, training deep learning model and evaluating the model will be covered.

4 Implementation

In this section implementation of the three deep learning algorithms is described in step by step.

4.1 Technologies used

4.1.1 Python

The programming language used in this project is Python []. Python is a programming language that is widely used in Machine Learning. It is especially useful for data cleaning and data preprocessing as the language is designed for legibility and ease of use. There are quite a few libraries designed for this task, like Pandas [], which includes a lot of methods for data frame handling. Scikit-learn []is the Machine Learning library of choice for this project because it includes preprocessing and as well as all the known baseline Machine Learning algorithms. Keras [] is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow [] library.

4.1.2 Google Collaboratory

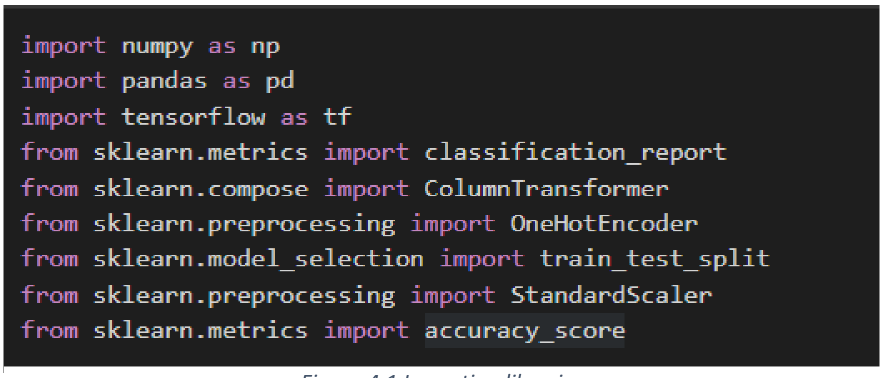
Google Colab [] is a powerful platform for learning and quickly developing machine learning models in Python. It is based on Jupyter notebook [] and supports collaborative development and It is Prebuilt with lots of python libraries. The main reason this environment was chosen is team members can share and concurrently edit the notebooks, even remotely.

4.2 Implementation of ANN using Keras-Sequential model API

Keras is a neural network Application Programming Interface (API) for Python that is tightly integrated with TensorFlow, which is used to build machine learning models. Its models offer a simple, user-friendly way to define a neural network. The Sequential model API [13] is used to create deep learning model where an instance of the Sequential class is created, then model layers are created and added to it. In this model, the data flow from one layer to another layer. The flow of data is continued until the data reaches the final layer.

4.2.1 Data preprocessing

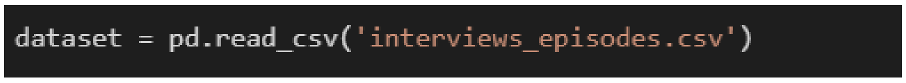
Importing libraries



```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.metrics import classification_report
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
```

Figure 4.1 Importing libraries

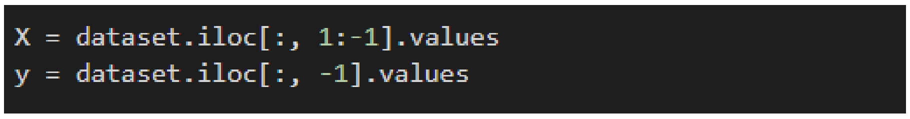
Importing dataset



```
dataset = pd.read_csv('interviews_episodes.csv')
```

Figure 4.2 Importing dataset

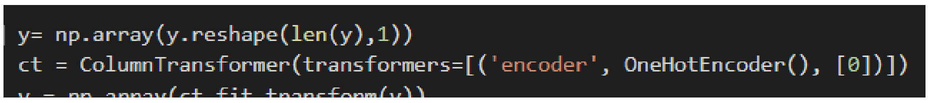
Creating the matrix of features(X) and the dependent variable vector(y)



```
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

Figure 4.3 Creating the matrix of features(X) and the dependent variable vector(y)

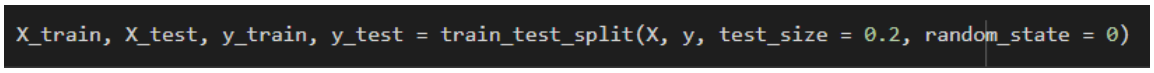
Encoding dependent variable vector(y)



```
y= np.array(y.reshape(len(y),1))
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])])
y = np.array(ct.fit_transform(y))
```

Figure 4.4 Encoding dependent variable vector(y)

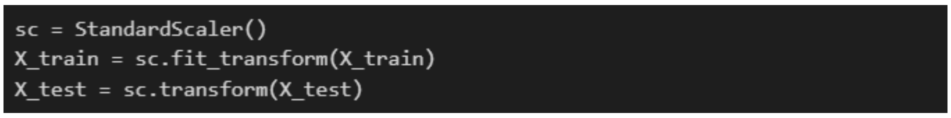
Splitting the dataset into the Training set and Test set



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Figure 4.5 Splitting the dataset into the Training set and Test set

Feature Scaling

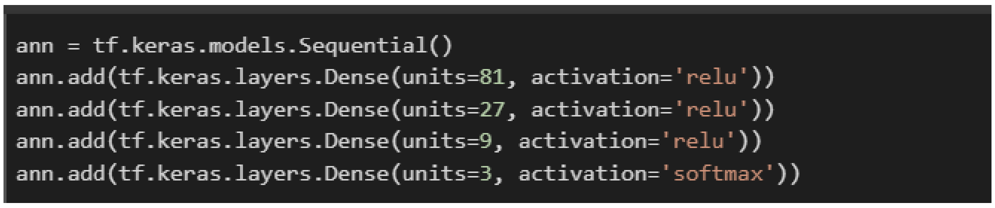


```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Figure 4.6 Feature Scaling

4.2.2 Building deep learning model

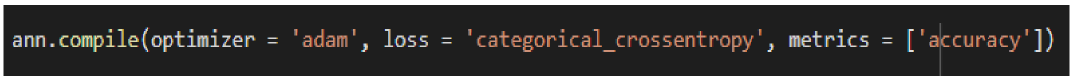
Creating model and adding input, hidden and output layers



```
ann = tf.keras.models.Sequential()
ann.add(tf.keras.layers.Dense(units=81, activation='relu'))
ann.add(tf.keras.layers.Dense(units=27, activation='relu'))
ann.add(tf.keras.layers.Dense(units=9, activation='relu'))
ann.add(tf.keras.layers.Dense(units=3, activation='softmax'))
```

Figure 4.7 Creating model and adding input, hidden and output layers

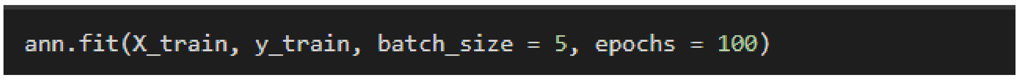
Compiling model



```
ann.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

Figure 4.8 Compiling model

4.2.3 Training deep learning model

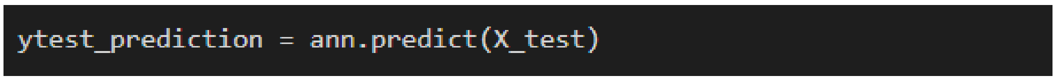


```
ann.fit(X_train, y_train, batch_size = 5, epochs = 100)
```

Figure 4.9 Training deep learning model

4.2.4 Evaluating the model

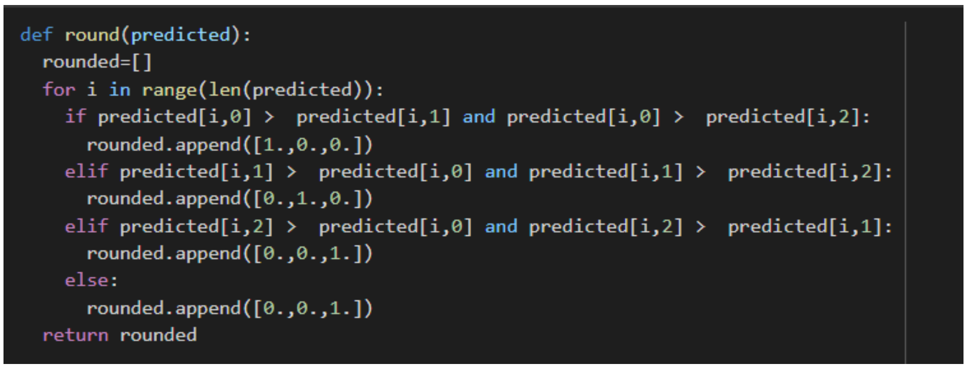
Predicting test set.



```
ytest_prediction = ann.predict(X_test)
```

Figure 4.10 Predicting test set.

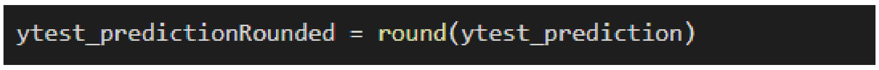
For each sample in the test set we are getting a probability that maps to the patient is in depression state, in mania state and in neutral state. The following function is used to get only the most probable prediction for each sample in the test set.



```python
def round(predicted):
  rounded=[]
  for i in range(len(predicted)):
    if predicted[i,0] >  predicted[i,1] and predicted[i,0] >  predicted[i,2]:
      rounded.append([1.,0.,0.])
    elif predicted[i,1] >  predicted[i,0] and predicted[i,1] >  predicted[i,2]:
      rounded.append([0.,1.,0.])
    elif predicted[i,2] >  predicted[i,0] and predicted[i,2] >  predicted[i,1]:
      rounded.append([0.,0.,1.])
    else:
      rounded.append([0.,0.,1.])
  return rounded
```

Figure 4.11 The function used to get only the most probable prediction

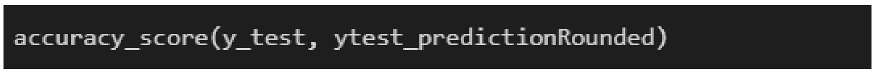
Getting the most probable prediction.



```python
ytest_predictionRounded = round(ytest_prediction)
```

Figure 4.12 Getting the most probable prediction.

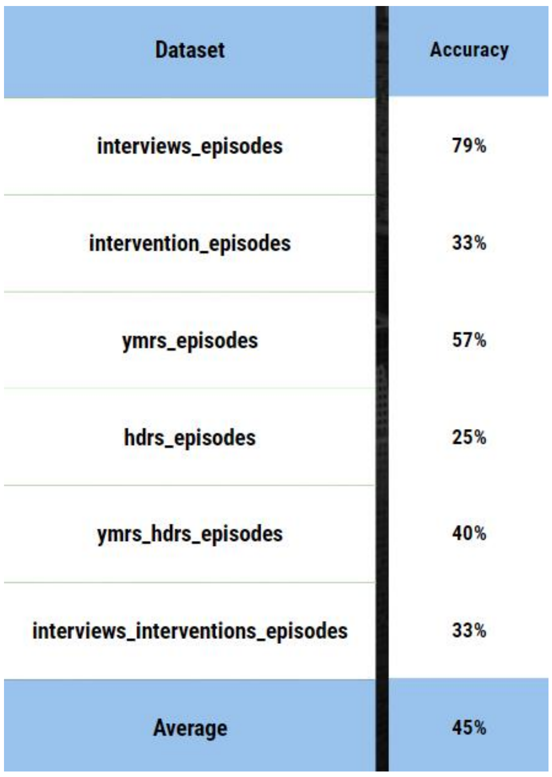
After having that accuracy score is calculated.



```python
accuracy_score(y_test, ytest_predictionRounded)
```

Figure 4.13 Calculating accuracy score

The same process was followed for each dataset. The accuracy score obtained for each data set is shown below.



Dataset	Accuracy
interviews_episodes	79%
intervention_episodes	33%
ymrs_episodes	57%
hdrs_episodes	25%
ymrs_hdrs_episodes	40%
interviews_interventions_episodes	33%
Average	45%

Figure 4.14 Accuracy Of datasets with ANN model

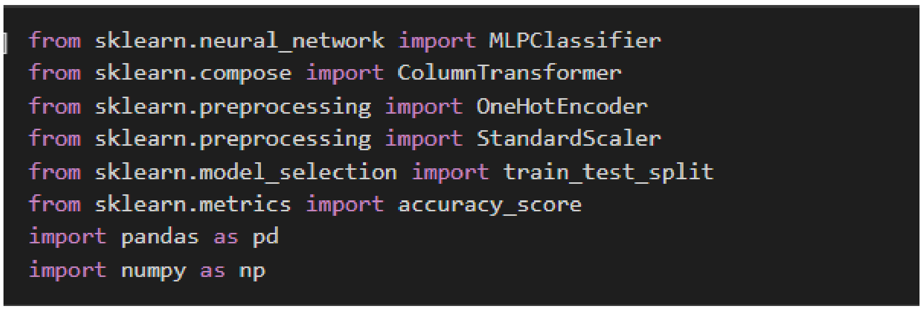
As shown in the above figure sequential model had the best accuracy with the interviews_episodes dataset which is 79%. Model performed very poorly on intervention_episodes, hdrs_episodes, ymrs_hdrs_episodes and interviews_interventions_episodes datasets with 33%, 25%, 40%, and 33% respectively. This is because the data were very scarce in these datasets. The only data set that resulted in a high prediction accuracy with this algorithm was the interview data set.

4.3 Implementation of multilayer perceptron (MLP) using sklearn

The multilayer perceptron (MLP) is a feedforward artificial neural network model that maps input data sets to a set of appropriate outputs. It consists of multiple layers and each layer is fully connected to the following one. The nodes of the layers are neurons with nonlinear activation functions, except for the nodes of the input layer. Between the input and the output layer there may be one or more nonlinear hidden layers. The classification model is implemented using multilayer perceptron classifier (MLPClassifier) [14] which is contained in sklearn.neural_network. Class MLPClassifier implements a multi-layer perceptron (MLP) algorithm that trains using backpropagation.

4.3.1 Data preprocessing

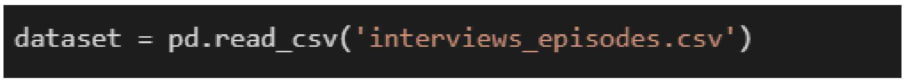
Importing libraries.



```
from sklearn.neural_network import MLPClassifier
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np
```

Figure 4.15 Importing libraries.

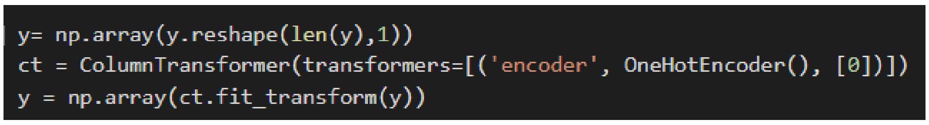
Importing dataset.



```
dataset = pd.read_csv('interviews_episodes.csv')
```

Figure 4.16 Importing dataset.

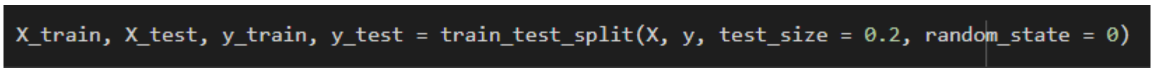
Encoding dependent variable vector(y)



```
y= np.array(y.reshape(len(y),1))
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])])
y = np.array(ct.fit_transform(y))
```

Figure 4.17 Encoding dependent variable vector(y)

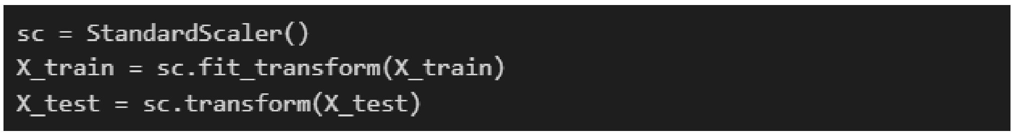
Splitting the dataset into the Training set and Test set



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Figure 4.18 Splitting the dataset into the Training set and Test set

Feature Scaling

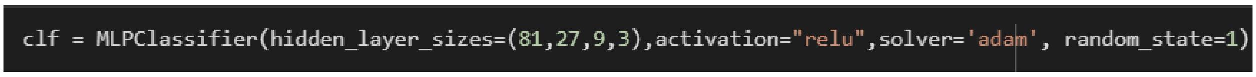


```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Figure 4.19 Feature Scaling

4.3.2 Building deep learning model

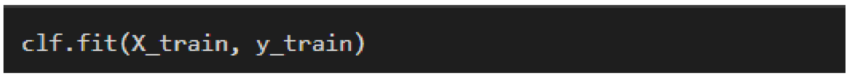
Creating model and hidden layers



```
clf = MLPClassifier(hidden_layer_sizes=(81,27,9,3),activation="relu",solver='adam', random_state=1)
```

Figure 4.20 Creating model and hidden layers

4.3.3 Training deep learning model

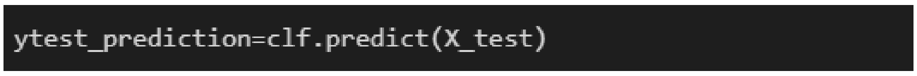


```
clf.fit(X_train, y_train)
```

Figure 4.21 Training Model

4.2.4 Evaluating the model

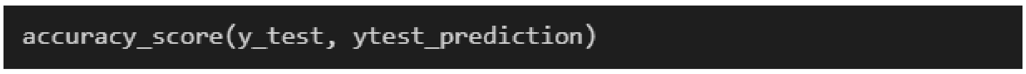
Predicting test set.



```
ytest_prediction=clf.predict(X_test)
```

Figure 4.22 Predicting test set.

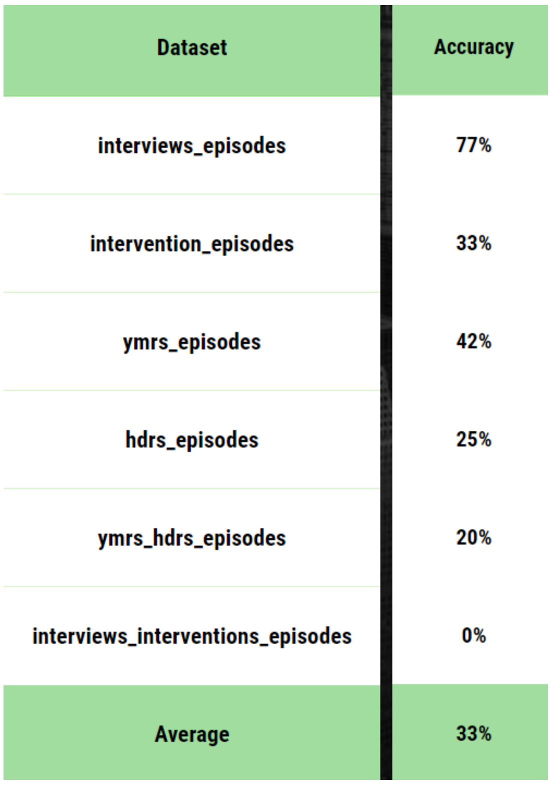
Obtaining accuracy score.



```
accuracy_score(y_test, ytest_prediction)
```

Figure 4.23 Calculate accuracy score

The same process was followed for each dataset. The accuracy score obtained for each data set is shown below.



Dataset	Accuracy
interviews_episodes	77%
intervention_episodes	33%
ymrs_episodes	42%
hdrs_episodes	25%
ymrs_hdrs_episodes	20%
interviews_interventions_episodes	0%
Average	33%

Figure 4.24 Dataset accuracy with MLP model

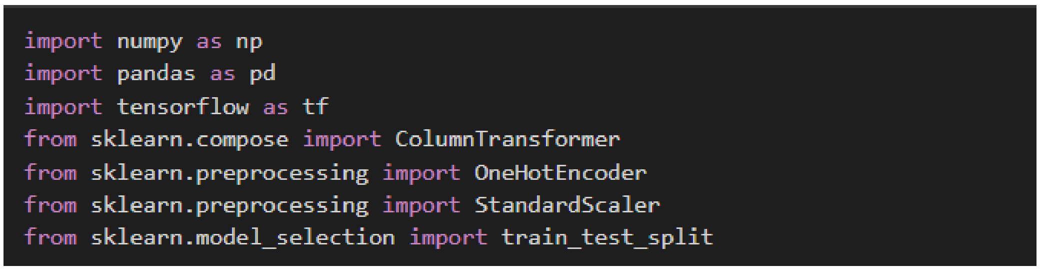
The Interview data set returned a score of 77% which is an acceptable performance because it has much more data than the rest of the data sets. It performed very poorly on the rest of the data sets. The model was clearly underfitted because data were very scarce in these datasets.

4.4 Implementation of CNN

A convolutional neural network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification. CNNs are playing a major role in diverse tasks/functions like image processing problems, computer vision tasks like localization and segmentation, video analysis, to recognize obstacles in self driving cars, as well as speech recognition in natural language processing. Here the CNN model is implemented using sequential model with keras Conv1D layer [15] and keras MaxPooling1D [16] layer. According to keras documentation Conv1D layer creates a convolution kernel that is convolved with the layer input over a single spatial dimension to produce a tensor of outputs and MaxPooling1D layer performs Max pooling operation for 1D temporal data. To train and test CNN model, train and test data were reshaped so that Conv1D layer can work on them.

4.4.1 Data preprocessing

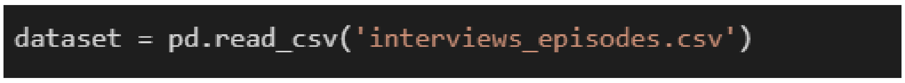
Importing libraries



```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

Figure 4.25 Importing libraries

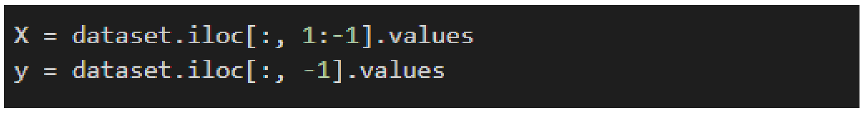
Importing dataset



```
dataset = pd.read_csv('interviews_episodes.csv')
```

Figure 4.26 Importing dataset

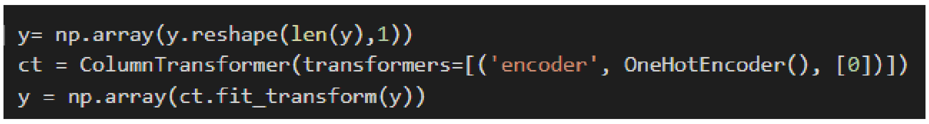
Creating the matrix of features(X) and the dependent variable vector(y)



```
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

Figure 4.27 Creating the matrix of features(X) and the dependent variable vector(y)

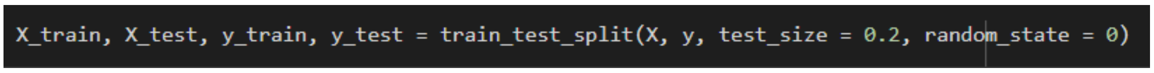
Encoding dependent variable vector(y)



```
y= np.array(y.reshape(len(y),1))
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])])
y = np.array(ct.fit_transform(y))
```

Figure 4.28 Encoding dependent variable vector(y)

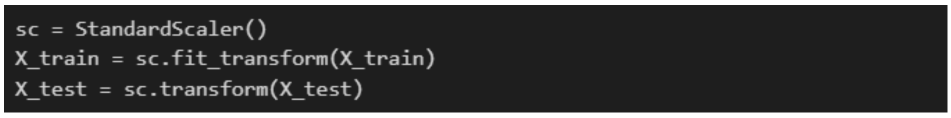
Splitting the dataset into the Training set and Test set



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Figure 4.29 Splitting the dataset into the Training set and Test set

Feature Scaling

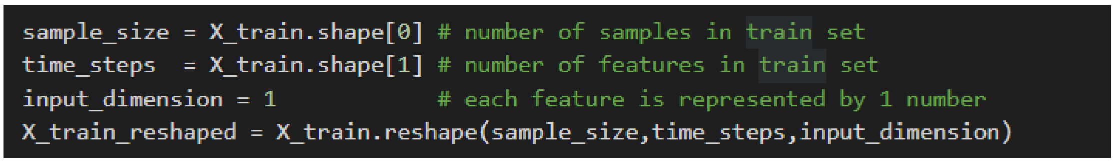


```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Figure 4.30 Feature Scaling

Reshape data set

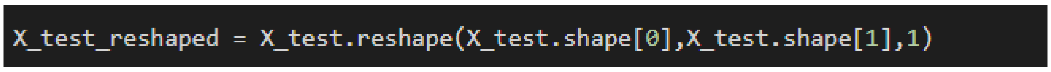
Conv1D layer expects input shape in 3D as [batch_size, time_steps, input_dimension] But current data is in the shape of 2D as [batch_size, features]. In current data sets each sample have multiple features and no time steps. In order to convert 2D input to 3D input features of the dataset is converted to time steps using numpy.reshape [17] function .



```
sample_size = X_train.shape[0] # number of samples in train set
time_steps  = X_train.shape[1] # number of features in train set
input_dimension = 1            # each feature is represented by 1 number
X_train_reshaped = X_train.reshape(sample_size,time_steps,input_dimension)
```

Figure 4.31 Reshape the train data

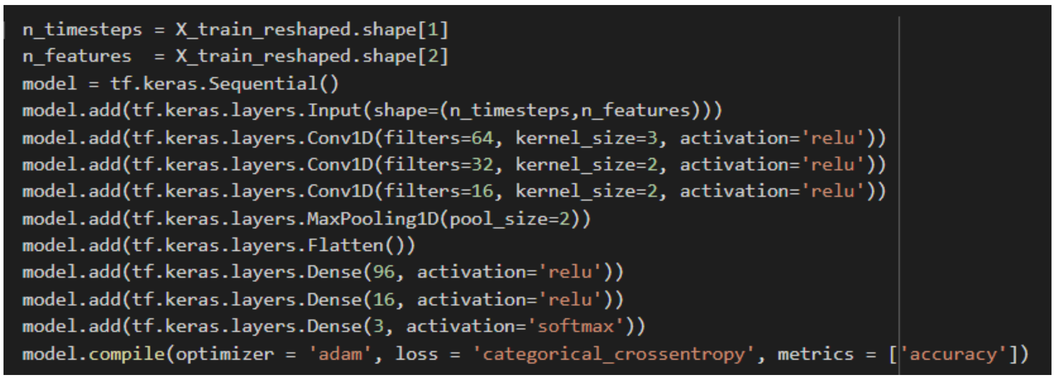
After conversion, train data shape is [batch_size, time_steps, input_dimension]. Test data is also reshaped.



```
X_test_reshaped = X_test.reshape(X_test.shape[0],X_test.shape[1],1)
```

Figure 4.32 Reshape the test data

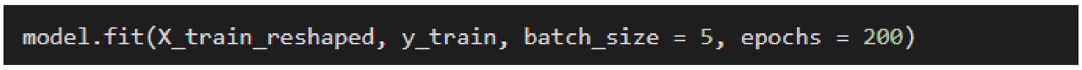
4.4.2 Building deep learning model



```python
n_timesteps = X_train_reshaped.shape[1]
n_features  = X_train_reshaped.shape[2]
model = tf.keras.Sequential()
model.add(tf.keras.layers.Input(shape=(n_timesteps,n_features)))
model.add(tf.keras.layers.Conv1D(filters=64, kernel_size=3, activation='relu'))
model.add(tf.keras.layers.Conv1D(filters=32, kernel_size=2, activation='relu'))
model.add(tf.keras.layers.Conv1D(filters=16, kernel_size=2, activation='relu'))
model.add(tf.keras.layers.MaxPooling1D(pool_size=2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(96, activation='relu'))
model.add(tf.keras.layers.Dense(16, activation='relu'))
model.add(tf.keras.layers.Dense(3, activation='softmax'))
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

Figure 4.33 Building deep learning model

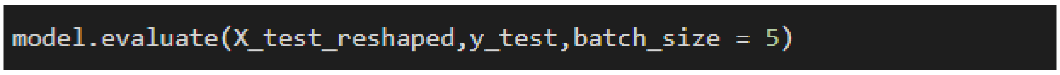
4.4.3 Training deep learning model



```python
model.fit(X_train_reshaped, y_train, batch_size = 5, epochs = 200)
```

Figure 4.34 Training deep learning model

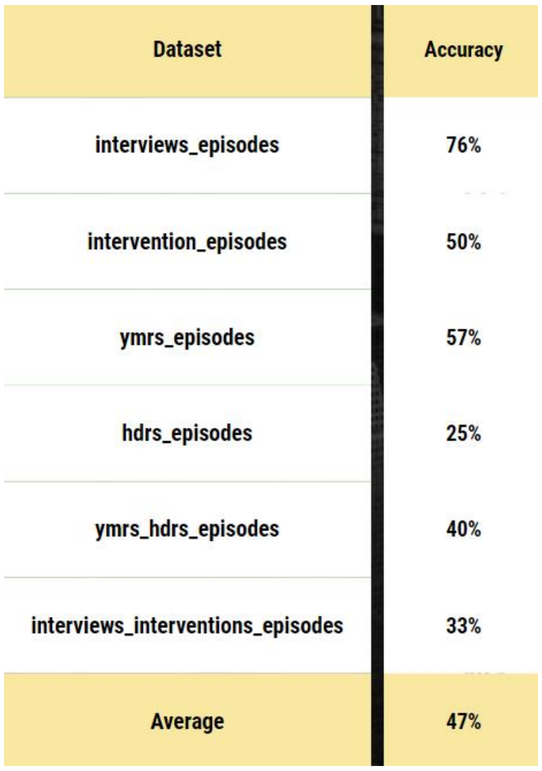
4.4.4 Evaluating the model



```python
model.evaluate(X_test_reshaped,y_test,batch_size = 5)
```

Figure 4.35 Evaluating the model

The same process was followed for each dataset. The accuracy score obtained for each data set is shown below



Dataset	Accuracy
interviews_episodes	76%
intervention_episodes	50%
ymrs_episodes	57%
hdrs_episodes	25%
ymrs_hdrs_episodes	40%
interviews_interventions_episodes	33%
Average	47%

Figure 4.36 Accuracy of datasets with CNN model

This model obtained same accuracy scores as sequential model on ymrs_episodes, hdrs_episodes, ymrs_hdrs_episodes, interviews_interventions_episodes data sets and again was very low.

5. Discussions and Conclusions

5.1 Overall discussion

The outcomes of this project are divided into different sections. Each section corresponds main parts of the project that discussed in the design section. Each section contains a summary of the results that were obtained in the corresponding part of the project.

The data sets used for this project was obtained from Application of Machine Learning Algorithms for Bipolar Disorder Crisis Prediction [12] project. Data set was not cleaned and was in an unorganized way. So, the data cleaning and the data combination processes conducted in the Application of Machine Learning for Bipolar Disorder Crisis Prediction project were also carried out in this project in order to get data sets that can be used to train deep learning algorithms.

5.1.1 data cleaning

Data cleaning part resulted with datasets without incomplete and empty values in them.

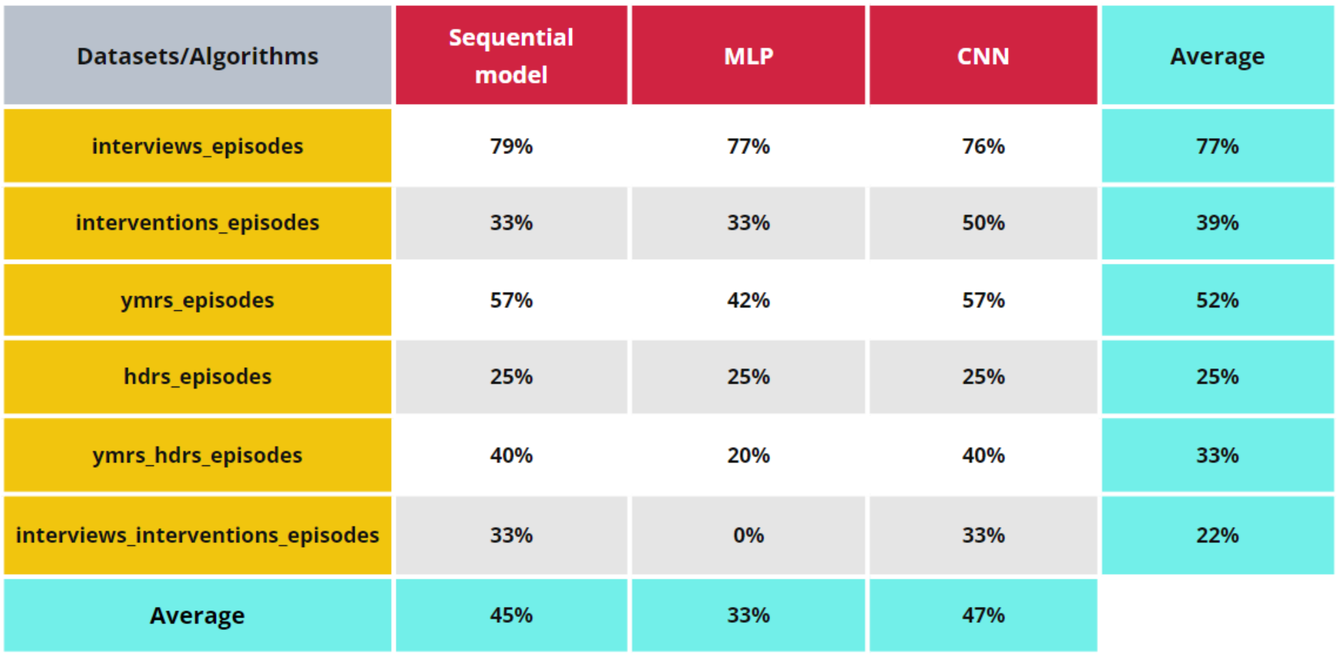
5.1.2 data combination

Data set generated from data combination step was very small like the combination of Interviews and Interventions, which only had 14 rows. ymrs_hdrs_episode contained only 25 entries. But they can be interesting for future implementations since they had a lot of features. The data combination resulted in ymrs_episodes, hdrs_episodes, interview_episode, intervention_episode, ymrs_hdrs_episodes and interview_intervention_episodes data sets.

5.1.3 Application of algorithms

Training and testing deep learning models with each dataset were conducted in this part. We were able to compare the data sets in order to see which ones performed better and which algorithms performed better on those datasets.

The best way to compare the accuracies obtained by the different algorithms with all the data sets was to make an algorithm performance matrix.



Datasets/Algorithms	Sequential model	MLP	CNN	Average
interviews_episodes	79%	77%	76%	77%
interventions_episodes	33%	33%	50%	39%
ymrs_episodes	57%	42%	57%	52%
hdrs_episodes	25%	25%	25%	25%
ymrs_hdrs_episodes	40%	20%	40%	33%
interviews_interventions_episodes	33%	0%	33%	22%
Average	45%	33%	47%	

Figure 5.1 Comparing dataset with model

The dataset returned the best accuracy (77%) in average was interview_episode dataset which was which was a quite reasonable score having in mind that the data set had almost 650 entries. Rest of the datasets had very few entries. The models were clearly underfitted and returned very poor accuracies. The algorithm that performed the best, also in average (47%), was the CNN, as seen on the row farthest down. Both Sequential model and CNN model returned nearly similar accuracy scores on the datasets.

5.2 Conclusions

The CNN returned the best average accuracy score which is 47% So we can say that CNN is the best performing model. But the sequential model also got an average accuracy score of 45% which is close to CNN model's core.

As a conclusion of this project, we can state that Both CNN and sequential models performed fairly similarly and can be used for further implementations. We can also affirm that the amount of data used in a project is a very important factor because with a larger amount of data we will be able to get prediction accuracies with a much higher level of confidence. Also having a deep perception of the theory behind each algorithm used, as well as their different implementations, is crucial in order to get the models to perform in the best way possible.

5.3 Proposed future work

Firstly, we can train and test existing models used in this project on a larger amount of data in order to see if they perform in a similar way.

We also can gather data using mobile phones, wristbands and watches and see how performance of different algorithms on the data gathered from these devices. If we could predict the patient's state with high accuracy, we can implement a drug recommendation system.

References

[1] Mayo Clinic.(2021) *Bipolar disorder - Symptoms and causes*. [online] Available from: https://www.mayoclinic.org/diseases-conditions/bipolar-disorder/symptoms-causes/syc-20355955[Accessed 24 February 2022].

[2] Howland,M.M & Sehamy,A.E.M.(2021) *What Are Bipolar Disorders?*. [online] Available from:https://psychiatry.org/patients-families/bipolar-disorders/what-are-bipolar-disorders[Accessed 24 February 2022].

[3]Brownlee,J.(2020) *What is Deep Learning?*. [online] Available from: https://machinelearningmastery.com/what-is-deep-learning/ [Accessed 24 February 2022].

[4]Wikipedia.(2022) *Deep learning-Wikipedia*.[online]Available from: https://en.wikipedia.org/wiki/Deep_learning [Accessed 24 February 2022].

[5] Reyes.K.(2022) *What is Deep Learning and How Does It Works [Explained]*. [online] Available from: https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-deep-learning [Accessed 24 February 2022].

[6] Besga, A., Termenon, M., Grana, M., Echeveste, J., Perez, J. and Gonzalez-pinto, A., (2012). *Discovering Alzheimer's disease and bipolar disorder white matter effects building computer aided diagnostic systems on brain diffusion tensor imaging features*. [online] Available from: https://www.sciencedirect.com/science/article/abs/pii/S0304394012006799 [Accessed 23 May 2022].

[7] Li,Z. LI,W. Wei,Y. Gui,G. Zhang,R. Liu,H. Chen,Y. & Jiang,Y.(2021) *Deep learning based automatic diagnosis of first-episode psychosis, bipolar disorder and healthy controls*.[online] Available from : https://www.sciencedirect.com/science/article/abs/pii/S0895611121000306 [Accessed 23 May 2022].

[8] Eramenco, K. & pontives, H.(2020) *Deep Learning A-Z™: Hands-On Artificial Neural Networks*. [online] Udemy. Available from: https://www.udemy.com/course/deeplearning/ [Accessed 28 May 2022].

[9] National Institute of Mental Health (NIMH).(2022). *Bipolar Disorder*. [online] Available from: https://www.nimh.nih.gov/health/topics/bipolar-disorder[Accessed 4 April 2022].

[10] HappiestMinds.(2020) *Convolutional Neural Networks (CNN) with Deep Learning*. [online] Available from: https://www.happiestminds.com/insights/convolutional-neural-networks-cnns/#:~:text=Within%20Deep%20Learning%2C%20a%20Convolutional,image%20by%20using%20a%20CNN. [Accessed 21 April 2022].

[11] Simplilearn.(2022) *An Overview on Multilayer Perceptron (MLP)*. [online] Available from: https://www.simplilearn.com/tutorials/deep-learning-tutorial/multilayer-perceptron> [Accessed 2 May 2022].

[12] Leal,A.J.(2018) *Application of Machine Learning Algorithms for Bipolar Disorder Crisis Prediction* [online] Available from: https://eprints.ucm.es/id/eprint/48866/1/076.pdf [Accessed 12 March 2022].

[13] Team, K.(2022)*Keras documentation: The Sequential model*. [online] Keras.io. Available from: https://keras.io/guides/sequential_model/ [Accessed 3 August 2022]. (Team, 2022)

[14] scikit-learn. (2022)*sklearn.neural_network.MLPClassifier*. [online] Available from: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html [Accessed 3 August 2022].

[15] Team, K.(2022)*Keras documentation: Conv1D layer*. [online] Keras.io. Available from: https://keras.io/api/layers/convolution_layers/convolution1d/ [Accessed 3 August 2022].

[16] Team, K.(2022)*Keras documentation: MaxPooling1D layer*. [online] Keras.io. Available from:https://keras.io/api/layers/pooling_layers/max_pooling1d/ [Accessed 3 August 2022].

[17] Numpy.org.(2022) *numpy.reshape — NumPy v1.23 Manual*. [online] Available from: https://numpy.org/doc/stable/reference/generated/numpy.reshape.html[Accessed 3 August 2022].