Session 22/23 (A231)
SKIH3013 Pattern Recognition & Analysis

Project:

AI-Powered Virtual Counselling Assistant: Insights and Applications

Prepared By:

NAME	MATRIC NUMBER
Fadzli Naim Bin Jafyusri	287855
Nur Ezurin Farisha binti Jusshairi	284253

Prepared For:
Prof. Madya Dr. Azizi Bin Ab Aziz

AI-Powered Virtual Counselling Assistant: Insights and Applications

Fadzli Naim Bin Jafyusri, Nur Ezurin Farisha Binti Jusshairi

Human-Centered Computing Research Lab, School of Computing,
College of Art & Sciences, Universiti Utara Malaysia

{Nur Ezurin Farisha binti Jusshairi, 284253, nur_ezurin_f@soc.uum.edu.my,
Fadzli Naim Bin Jafyusri, 287855, fadzlinaim@soc.uum.edu.my)

Abstract. The advent of AI-powered Virtual Counselling Assistants has revolutionized mental health support, aiming to bridge the accessibility gap by leveraging natural language processing and sentiment analysis. In this project, a retrieval-based chatbot system is developed, employing a diverse set of machine learning and deep learning models, including Vanilla Recurrent Neural Network (RNN) , Long Short-Term Memory (LSTM), Bi-directional (Bi-LSTM), Gated Recurrent Unit (GRU), and Convolutional Neural Network (CNN). Trained on curated JSON datasets, each model undergoes rigorous regularization, with the CNN architecture emerging as the most effective based on training and validation metrics. The chosen CNN model comprises a convolutional neural network, an embedding layer, and a fully connected layer. By meticulously evaluating using F1-score, precision, support, precision, and confusion matrix. In addition, it was verified that model Vanilla RNN have based model that achieves the accuracy of 100% and the worst performing model is CNN.

Keywords: Virtual Counselling Assistant, AI-driven Mental Health Support, Natural Language Processing, Sentiment Analysis, Deep Learning Models, Retrieval-based Approach, Vanilla RNN, LSTM, Bi-LSTM, GRU, CNN, Data Regularization, Mental Health Conversations.

1 Introduction

In contemporary society, the prevalence of mental health challenges underscores the urgent necessity for accessible support mechanisms. Recognizing this imperative, the development of an AI-powered Virtual Counseling Assistant assumes paramount significance. This project endeavors to pioneer a chatbot system tailored to provide immediate mental health assistance. By leveraging a spectrum of machine learning and deep learning models, including RNN, LSTM, Bi-LSTM, GRU, and CNN, this study seeks to ascertain the optimal architecture for empathetic and contextually sensitive interactions within mental health dialogues. Among the diverse array of models explored, the Convolutional Neural Network (CNN) emerges as particularly promising, exhibiting a notable proficiency in discerning the subtleties inherent in mental health conversations. The selected CNN model comprises a three-layered structure encompassing a convolutional neural network layer, an embedding layer, and a fully connected layer. This configuration aims to facilitate the chatbot's comprehension of nuanced linguistic cues, thereby enabling it to formulate tailored responses that resonate with the emotional nuances of users' expressions. By foregrounding the meticulous selection and refinement of neural network architectures, this study lays the groundwork for the development of a chatbot system characterized by responsiveness and contextual awareness. Such a system holds the potential to revolutionize mental health support by providing immediate and personalized assistance to individuals in need, transcending the barriers associated with traditional counseling modalities.

The remainder of this paper is organized as follows: Section 2 provides background information on the mental health effects of mental health issues and the emergence of AI norms. Section 3 discusses related works and elaborates on datasets. Section 4 outlines the methodologies used in developing the mental health chatbot, including data preparation, model architecture, training, and evaluation. Section 5 presents the evaluation results of the proposed mechanism, focusing on performance metrics such as precision, recall, F1-score, and support. Section 6 concludes with a discussion on potential future research directions. Lastly, Section 7 offers reflections on the study.

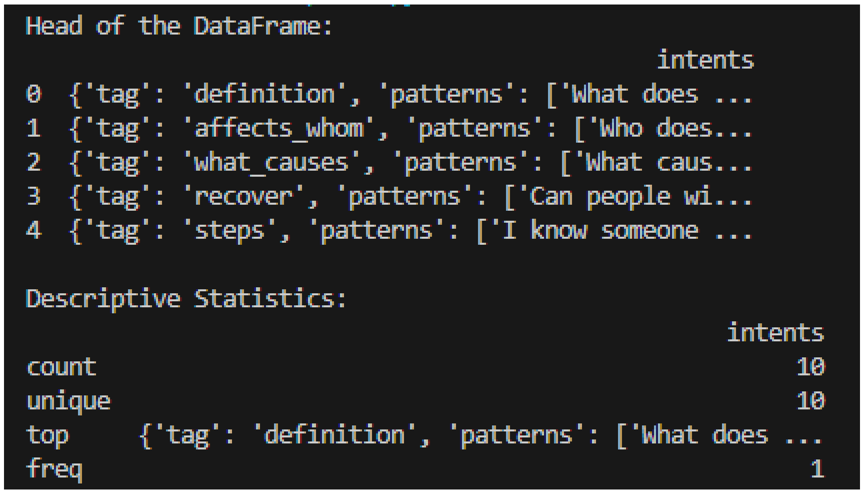
2 Background

Recent studies indicate a significant rise in mental health issues globally, with individuals experiencing heightened levels of stress, anxiety, and depression due to the pandemic's impact on daily life. According to current data, the burden of mental disorders, including depression and anxiety, has escalated, affecting a substantial portion of the population. The pandemic has exacerbated existing mental health disparities, with marginalized communities and those facing socioeconomic challenges bearing a disproportionate burden. As efforts to address mental health challenges evolve amidst the pandemic, this project aims to enhance accessibility to mental health support. This initiative seeks to provide timely assistance to individuals navigating the complexities of their mental well-being during these unprecedented times. The contemporary mental health landscape is marked by an escalating demand for support services, with pervasive barriers to accessibility and affordability. Traditional avenues for seeking assistance often entail prolonged wait times, geographical constraints, and stigma, exacerbating the burden on individuals grappling with mental health challenges. In this context, the emergence of AI-powered Virtual Counseling Assistants represents a transformative paradigm, offering a scalable and responsive solution to bridge the chasm between demand and supply in mental health care. This study assumes significance by interrogating the efficacy of AI-driven interventions in addressing the multifaceted challenges of mental health support. By subjecting a range of neural network architectures to rigorous evaluation, the research aims to delineate the most effective approach for facilitating empathetic and contextually relevant interactions within mental health dialogues. Moreover, by emphasizing the importance of model selection and regularization techniques, the study endeavors to instill confidence in the reliability and efficacy of AI-powered counseling assistants, thereby fostering trust and acceptance among potential users. Beyond its technical contributions, the study holds broader implications for societal discourse surrounding mental health care. By democratizing access to counseling support through AI-powered platforms, the research seeks to promote inclusivity and equity in mental health provision, empowering individuals to seek assistance on their own terms. Furthermore, by fostering informed deliberation on the ethical implications of AI-driven interventions, the study contributes to a nuanced understanding of the intersection between technology and mental health care, paving the way for responsible and inclusive innovation in the field.

Role Of artificial intelligence and its impact in mental health services. (2023, July). HIMSS. https://www.himss.org/resources/role-artificial-intelligence-and-its-impact-mental-health-services

3 Data

The dataset was picked up from Kaggle - Mental Health FAQ. This dataset consists of 98 FAQs about Mental Health. It consists of 3 columns - QuestionID, Questions, and Answers. Note that to train the retrieval chatbot, the CSV file was manually converted to a JSON file. Based on the Figure 1 presents the initial five entries of a DataFrame containing ‘intents’, each represented as a dictionary with ‘tag’ and ‘patterns’ keys. The ‘tag’ key corresponds to categories like ‘definition’, ‘affects whom’, ‘what causes’, ‘recover’, and ‘steps’. The ‘patterns’ key is associated with lists of text strings, which are likely examples of user inputs or queries related to each tag. In the Descriptive Statistics section, the ‘count’ indicates that there are 10 entries in the ‘intents’ column. The ‘unique’ value also being 10 suggests that all entries in this column are distinct. The ‘top’ entry shows the most frequent intent, but with a ‘freq’ of 1, it implies that all intents appear only once in this dataset.



```
Head of the DataFrame:
                                              intents
0  {'tag': 'definition', 'patterns': ['What does ...
1  {'tag': 'affects_whom', 'patterns': ['Who does...
2  {'tag': 'what_causes', 'patterns': ['What caus...
3  {'tag': 'recover', 'patterns': ['Can people wi...
4  {'tag': 'steps', 'patterns': ['I know someone ...

Descriptive Statistics:
                                                   intents
count                                                   10
unique                                                  10
top     {'tag': 'definition', 'patterns': ['What does ...
freq                                                     1
```

Fig.1 Descriptive Statistics of Intent Patterns in a Dataset

4 Methods

This section provides an overview of the methodologies utilized in crafting the mental health chatbot, encompassing processes such as data preparation, model architecture design, training procedures, and subsequent evaluation, depicted in Figure 2. Subsequent elaboration on these methodologies is provided in Sub-sections 4.1 "Data Loading and Preliminary Analysis" and 4.2 "Model Architectures."

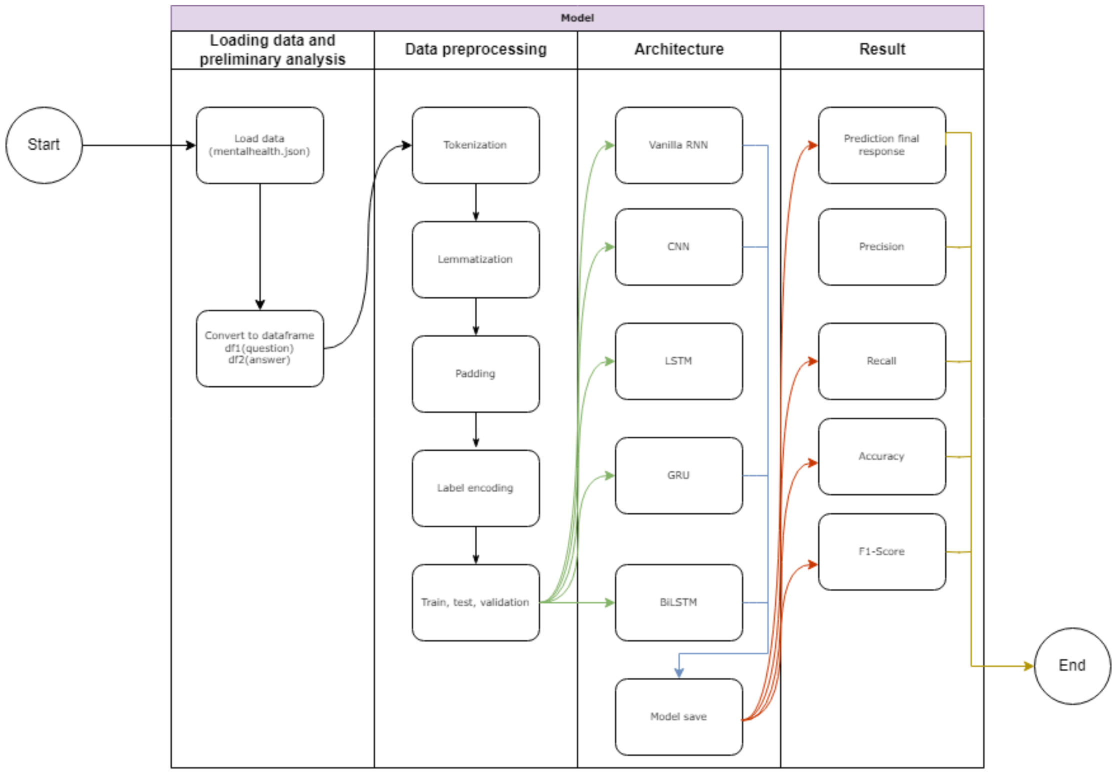


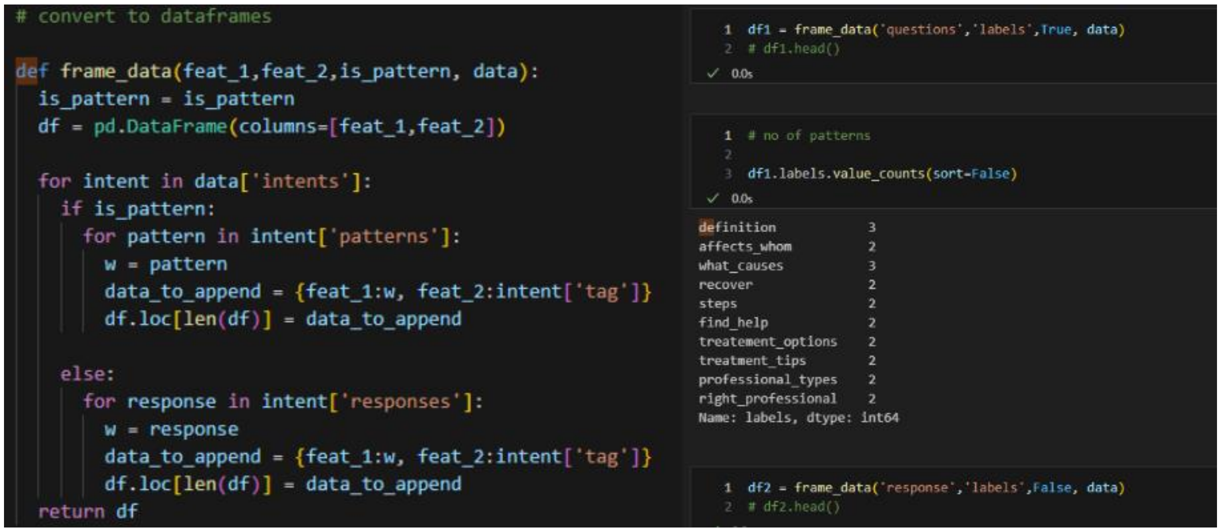
Fig.2 Experimental Flowchart

4.1 Data Loading and Preliminary Analysis

4.1.1 Data Loading

The code's first portion focuses on obtaining necessary dependencies and libraries using the nltk library for natural language processing. Various libraries for data manipulation, visualisation, and machine learning, including pandas, numpy, matplotlib, and tensorflow, are then brought into the programme. The file paths to important files, particularly the dataset called "mentalhealth.json," are clearly specified. The code then loads data from the supplied JSON file ("mentalhealth.json") using the json module. This careful arrangement guarantees that the necessary tools and datasets are smoothly incorporated into the following phases of code execution.

4.1.3 Preliminary Analysis



```python
# convert to dataframes

def frame_data(feat_1,feat_2,is_pattern, data):
  is_pattern = is_pattern
  df = pd.DataFrame(columns=[feat_1,feat_2])

  for intent in data['intents']:
    if is_pattern:
      for pattern in intent['patterns']:
        w = pattern
        data_to_append = {feat_1:w, feat_2:intent['tag']}
        df.loc[len(df)] = data_to_append

    else:
      for response in intent['responses']:
        w = response
        data_to_append = {feat_1:w, feat_2:intent['tag']}
        df.loc[len(df)] = data_to_append
  return df
```

```python
df1 = frame_data('questions','labels',True, data)
# df1.head()
```

```
✓ 0.0s
```

```python
# no of patterns

df1.labels.value_counts(sort-False)
```

```
✓ 0.0s
definition            3
affects_whom          2
what_causes           3
recover               2
steps                 2
find_help             2
treatement_options    2
treatment_tips        2
professional_types    2
right_professional    2
Name: labels, dtype: int64
```

```python
df2 = frame_data('response','labels',False, data)
# df2.head()
```

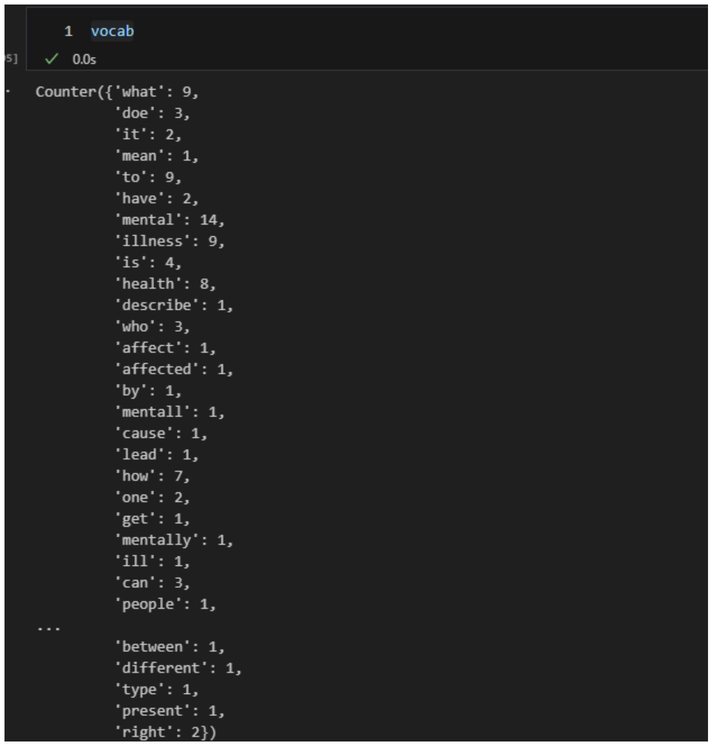
Fig.3 Convert to dataframe, Calculating and displaying the count of unique values in the 'labels' column.

Function Definition (frame_data)
The code defines a function called frame_data with four parameters: feat_1, feat_2, is_pattern, and data. This function is crucial for creating a dataframe that includes two supplied features, feat_1 and feat_2. The choice of features depends on whether the data being analysed relates to patterns or reactions. When is_pattern is True, the function identifies patterns in the loaded data's 'intents' and adds them to the dataframe along with their respective labels. If is_pattern is False, the function retrieves replies together with their corresponding labels.

DataFrame Creation (df1 and df2)
In the following part, two separate dataframes, df1 and df2, are created using the frame_data function. Df1 is designed for patterns and questions, with each row containing a pattern or question and its corresponding label. Conversely, df2 is designed for responses, where each row contains a response and its matching label. This procedure of creating two dataframes allows for a thorough description of patterns/questions and responses, which is essential for future data processing and analysis.

4.2 Data preprocessing

4.2.1 Tokenization and Lemmatization



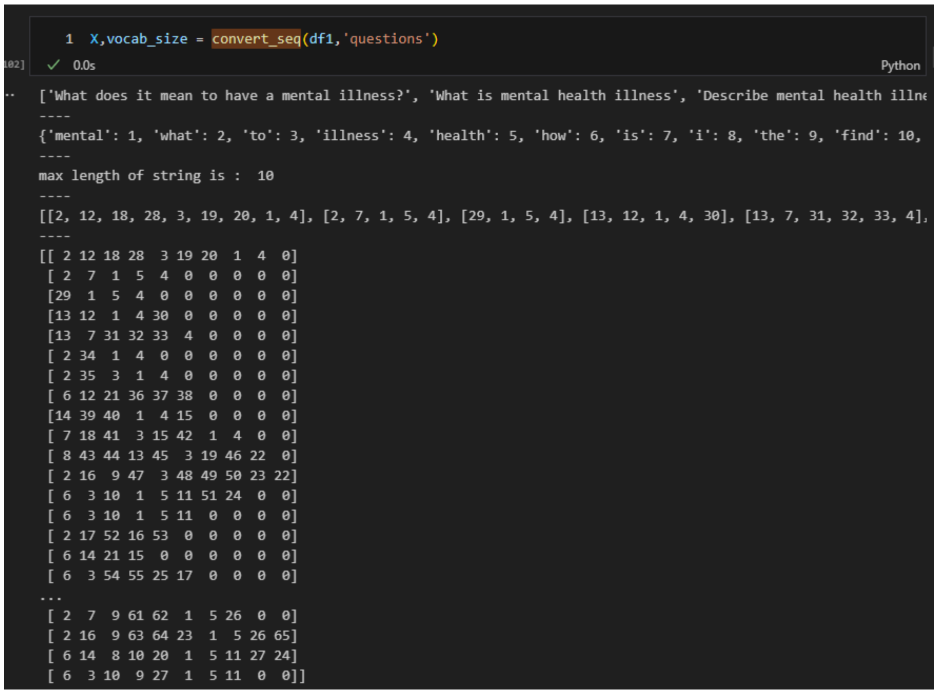
```
vocab
✓ 0.0s
Counter({'what': 9,
         'doe': 3,
         'it': 2,
         'mean': 1,
         'to': 9,
         'have': 2,
         'mental': 14,
         'illness': 9,
         'is': 4,
         'health': 8,
         'describe': 1,
         'who': 3,
         'affect': 1,
         'affected': 1,
         'by': 1,
         'mentall': 1,
         'cause': 1,
         'lead': 1,
         'how': 7,
         'one': 2,
         'get': 1,
         'mentally': 1,
         'ill': 1,
         'can': 3,
         'people': 1,
...
         'between': 1,
         'different': 1,
         'type': 1,
         'present': 1,
         'right': 2})
```

Fig.4 *Vocabulary counter (vocab) created during the process of tokenization and lemmatization.*

The code features a tokenizer function that plays a major role in showcasing a versatile method for handling textual data. This function meticulously analyses each item by breaking it down into tokens, removing punctuation, excluding non-alphabetic words, doing lemmatization, and finally converting words to lowercase. The remove_stop_words function smoothly combines with the tokenizer to remove stop words from the 'questions' column in the dataframe df1. The create_vocab function is crucial in generating a vocabulary counter referred to as 'vocab.' This entails a step-by-step investigation of the 'questions' column in df1.

The tokenizer function relies on the WordNetLemmatizer from NLTK for effective lemmatization. The Counter class helps create a vocabulary counter that captures the linguistic diversity present in the dataset. The tokenizer function methodically processes each input in the 'questions' column, serving as a crucial component. By strategically removing punctuation, selecting relevant words, and lemmatizing, it guarantees the creation of polished tokens suitable for further analysis.

4.2.2 Tokenization and Padding for Sequences



```
X,vocab_size = convert_seq(df1,'questions')
```

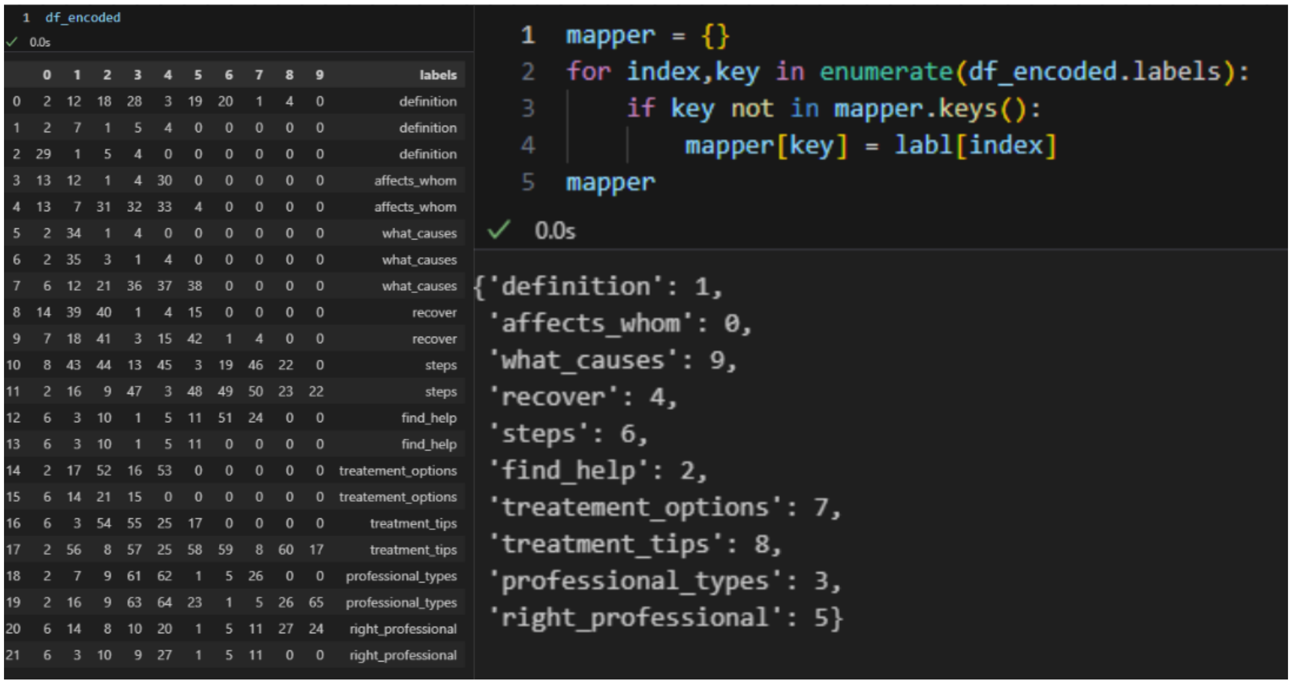
```
['What does it mean to have a mental illness?', 'What is mental health illness', 'Describe mental health illne
----
{'mental': 1, 'what': 2, 'to': 3, 'illness': 4, 'health': 5, 'how': 6, 'is': 7, 'i': 8, 'the': 9, 'find': 10,
----
max length of string is :  10
----
[[2, 12, 18, 28, 3, 19, 20, 1, 4], [2, 7, 1, 5, 4], [29, 1, 5, 4], [13, 12, 1, 4, 30], [13, 7, 31, 32, 33, 4],
----
[[ 2 12 18 28  3 19 20  1  4  0]
 [ 2  7  1  5  4  0  0  0  0  0]
 [29  1  5  4  0  0  0  0  0  0]
 [13 12  1  4 30  0  0  0  0  0]
 [13  7 31 32 33  4  0  0  0  0]
 [ 2 34  1  4  0  0  0  0  0  0]
 [ 2 35  3  1  4  0  0  0  0  0]
 [ 6 12 21 36 37 38  0  0  0  0]
 [14 39 40  1  4 15  0  0  0  0]
 [ 7 18 41  3 15 42  1  4  0  0]
 [ 8 43 44 13 45  3 19 46 22  0]
 [ 2 16  9 47  3 48 49 50 23 22]
 [ 6  3 10  1  5 11 51 24  0  0]
 [ 6  3 10  1  5 11  0  0  0  0]
 [ 2 17 52 16 53  0  0  0  0  0]
 [ 6 14 21 15  0  0  0  0  0  0]
 [ 6  3 54 55 25 17  0  0  0  0]
...
 [ 2  7  9 61 62  1  5 26  0  0]
 [ 2 16  9 63 64 23  1  5 26 65]
 [ 6 14  8 10 20  1  5 11 27 24]
 [ 6  3 10  9 27  1  5 11  0  0]]
```

Fig.5 *Process of the convert_seq function*

The "convert_seq" function is crucial in the preprocessing workflow. This method is specifically created to manage the complexities of textual data by utilising the Keras Tokenizer to tokenize the 'questions' inside the dataset. Raw text is converted into numerical sequences, making it easier to incorporate natural language input into machine learning models. An essential part of the process involves handling the varying sequence lengths found in the dataset. The function utilises the "pad_sequences" method to guarantee consistent sequence length, a crucial process in data preparation for neural networks.

Additionally, the function demonstrates foresight by preserving the tokenizer for later usage. This deliberate move improves the model's ability to adjust and remain stable during different phases of development and deployment. The "convert_seq" function transforms raw textual data into structured numerical input, preparing for the training and use of the mental health chatbot model.

4.2.3 Encoding Labels



```
df_encoded
```

✓ 0.0s

	0	1	2	3	4	5	6	7	8	9	labels
0	2	12	18	28	3	19	20	1	4	0	definition
1	2	7	1	5	4	0	0	0	0	0	definition
2	29	1	5	4	0	0	0	0	0	0	definition
3	13	12	1	4	30	0	0	0	0	0	affects_whom
4	13	7	31	32	33	4	0	0	0	0	affects_whom
5	2	34	1	4	0	0	0	0	0	0	what_causes
6	2	35	3	1	4	0	0	0	0	0	what_causes
7	6	12	21	36	37	38	0	0	0	0	what_causes
8	14	39	40	1	4	15	0	0	0	0	recover
9	7	18	41	3	15	42	1	4	0	0	recover
10	8	43	44	13	45	3	19	46	22	0	steps
11	2	16	9	47	3	48	49	50	23	22	steps
12	6	3	10	1	5	11	51	24	0	0	find_help
13	6	3	10	1	5	11	0	0	0	0	find_help
14	2	17	52	16	53	0	0	0	0	0	treatement_options
15	6	14	21	15	0	0	0	0	0	0	treatement_options
16	6	3	54	55	25	17	0	0	0	0	treatment_tips
17	2	56	8	57	25	58	59	8	60	17	treatment_tips
18	2	7	9	61	62	1	5	26	0	0	professional_types
19	2	16	9	63	64	23	1	5	26	65	professional_types
20	6	14	8	10	20	1	5	11	27	24	right_professional
21	6	3	10	9	27	1	5	11	0	0	right_professional

```python
mapper = {}
for index,key in enumerate(df_encoded.labels):
    if key not in mapper.keys():
        mapper[key] = labl[index]
mapper
```

✓ 0.0s

```
{'definition': 1,
 'affects_whom': 0,
 'what_causes': 9,
 'recover': 4,
 'steps': 6,
 'find_help': 2,
 'treatement_options': 7,
 'treatment_tips': 8,
 'professional_types': 3,
 'right_professional': 5}
```

Fig.6 Encodes categorical labels, bridging the semantic and numerical realms.

This piece of the code focuses on a crucial step in preparing the mental health chatbot dataset for machine learning: encoding categorical labels. Dataframes df1 and df2 contain patterns/questions and responses, respectively. Using the LabelEncoder tool from scikit-learn, the code converts original categorical labels into numerical representations, which are saved in the 'labl' array. To ensure interpretability, a mapping dictionary ('mapper') is created to explain the relationship between the original categorical categories and their encoded numerical representations. This mapping is crucial for later stages as it helps in interpreting model predictions and evaluation metrics within the numerous categories of the mental health chatbot.

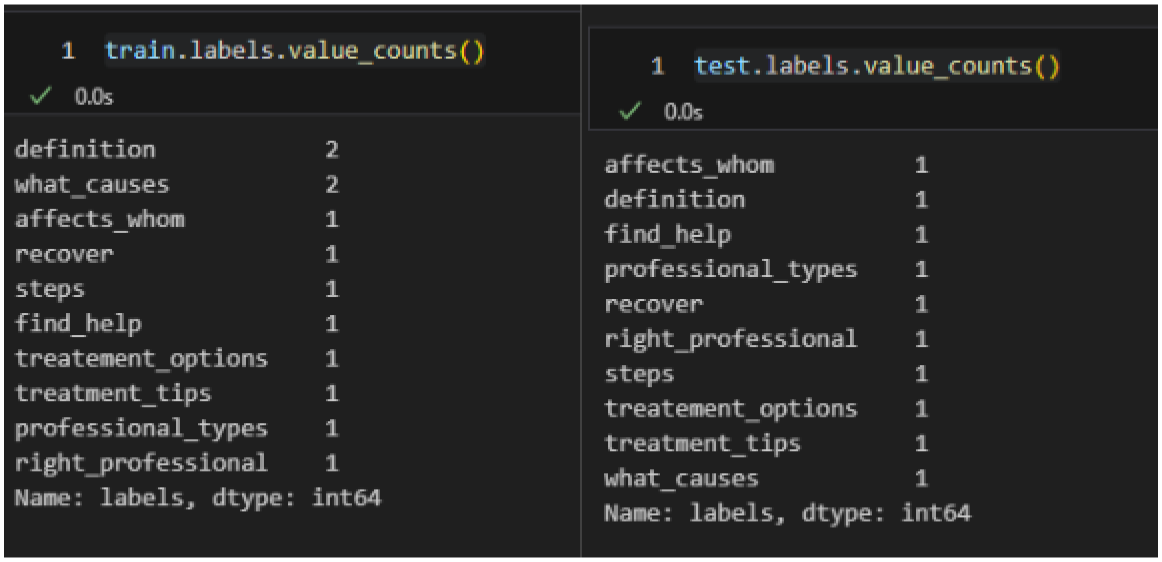
The iterative approach maps each unique label to its corresponding numerical value by examining the original labels in both dataframes (df1 and df2). This dual-label encoding method ensures uniformity between patterns/questions and responses, promoting coherence in the following training and evaluation of machine learning models. This encoding approach combines the detailed meaning of original labels with the numerical efficiency needed for machine learning algorithms, creating a strong basis for building an advanced mental health chatbot model.

4.3 Training and testing

This section is dedicated to preparing the data for training and testing the models. The original dataset is partitioned into training and testing sets to aid in model evaluation. The category labels in both the training and testing sets are transformed into binary vectors using the get_dummies function in pandas, which creates binary representations for each class. One-hot encoding is crucial for categorical classification problems since it converts categorical labels into a machine learning algorithm-friendly format.

X_train and X_test contain the input features, while y_train and y_test include the matching one-hot encoded labels. The datasets' forms are analysed to verify uniformity in dimensions, and the maximum sequence length is established for padding. The output variable is assigned a value of 16, which corresponds to the number of classes in the dataset. This procedure establishes the framework for training a machine learning model to predict the correct mental health category from user input, creating a strong base for the following training and evaluation stages.

4.3.1 Unbiased Evaluation with Balanced Testing Data



```
train.labels.value_counts()
✓ 0.0s

definition             2
what_causes            2
affects_whom           1
recover                1
steps                  1
find_help              1
treatement_options     1
treatment_tips         1
professional_types     1
right_professional     1
Name: labels, dtype: int64
```

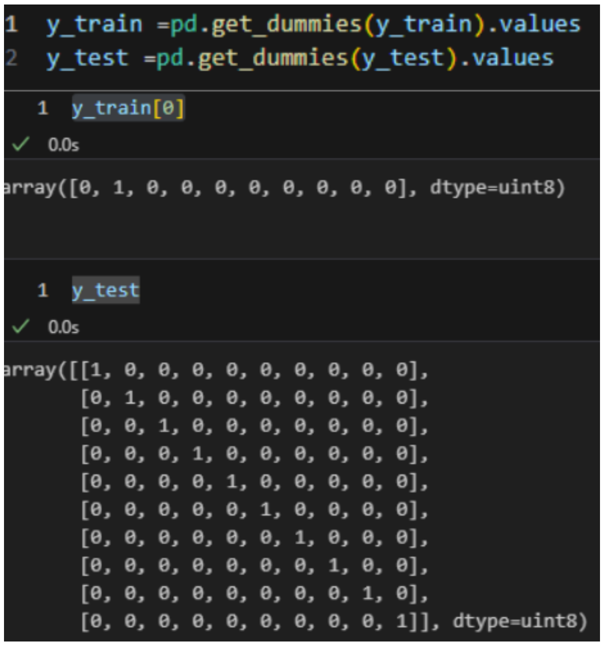
```
test.labels.value_counts()
✓ 0.0s

affects_whom           1
definition             1
find_help              1
professional_types     1
recover                1
right_professional     1
steps                  1
treatement_options     1
treatment_tips         1
what_causes            1
Name: labels, dtype: int64
```

Fig.7 *Balanced Label Distribution*

The frequency of each label in the training and testing sets offers insights into the distribution of categories. Each category in the training set is represented once or twice, showing a balanced distribution of mental health issues. Ensuring balance in training a machine learning model is critical to reduce bias towards specific categories and promote generalisation across varied mental health conditions.

The testing set also shows a uniform distribution of categories, with each label appearing just once. This equilibrium is crucial for assessing the model's efficacy on new data, as it encounters an impartial depiction of mental health subjects. Well-balanced datasets enhance the model's capacity to generalise effectively to various user inputs, hence strengthening the reliability and robustness of the mental health classification system.

4.3.2 Converting categorical labels into a one-hot encoded format



```
y_train =pd.get_dummies(y_train).values
y_test =pd.get_dummies(y_test).values
```

```
y_train[0]
```

```
0.0s
```

```
array([0, 1, 0, 0, 0, 0, 0, 0, 0, 0], dtype=uint8)
```

```
y_test
```

```
0.0s
```

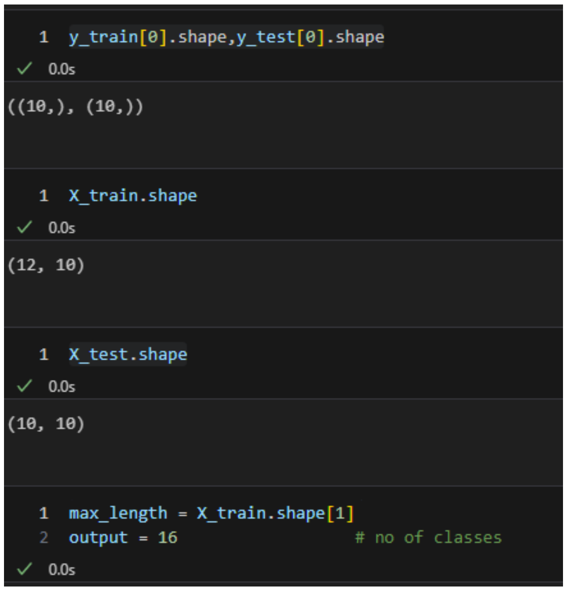
```
array([[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]], dtype=uint8)
```

Fig.8 *Converting categorical labels into a one-hot encoded*

The code sample shows how to convert categorical labels into a one-hot encoded format for the training (y_train) and testing (y_test) datasets. One-hot encoding is an essential preprocessing step in classification problems, transforming categorical labels into a binary matrix that machine learning models can comprehend efficiently. The original labels, expressed as arrays of integers, are transformed into one-hot encoded arrays using the pd.get_dummies() function in the shown example. Each row in the output arrays represents a sample, and the existence of a '1' in a certain column indicates the class to which the sample belongs. This encoding is essential for training and assessing models, particularly in multi-class classification situations, to guarantee the neural network's ability to effectively differentiate and forecast categorical results.

The first row of y_train [0, 1, 0, 0, 0, 0, 0, 0, 0, 0] signifies that the sample belongs to the second class, as the '1' is in the second column. One-hot encoding aids in training by enabling the model to effectively learn and predict the distribution of mental health themes among various classes.

4.3.3 Dataset Dimensions and Model Configuration Parameters



```
y_train[0].shape,y_test[0].shape
```
✓ 0.0s
```
((10,), (10,))
```
```
X_train.shape
```
✓ 0.0s
```
(12, 10)
```
```
X_test.shape
```
✓ 0.0s
```
(10, 10)
```
```
max_length = X_train.shape[1]
output = 16                    # no of classes
```
✓ 0.0s

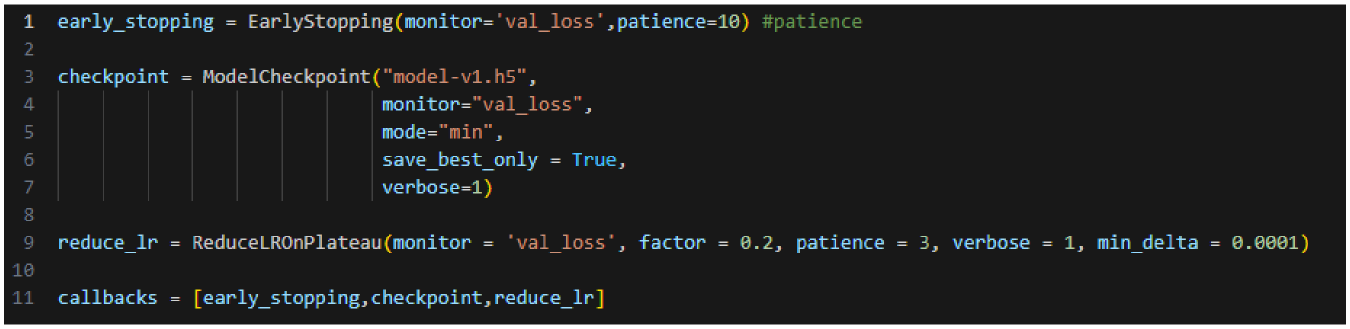
Fig.9 *Dimensions and Model Configuration Parameters*

The code snippet uses the expressions `y_train[0].shape` and `y_test[0].shape` to examine the dimensions of the one-hot encoded labels for the training and testing datasets. Both resulting shapes are `(10,)`, signifying that the one-hot encoded labels symbolise categorical classes, with each label being converted into a binary array of length 10, matching the number of distinct classes in the dataset.

Examining the dimensions of the input features for the training and testing datasets is done using `X_train.shape` and `X_test.shape` feature matrices. The dimensions of `X_train` are (12, 10), indicating that there are 12 samples in the training set, with each sample being represented by a feature vector of length 10. The shape `(10, 10)` of `X_test` shows that there are 10 samples in the testing set, each with a feature vector of length 10. The dimensions are important for setting up the input layer of a neural network. In this context, `max_length = X_train.shape[1]` represents the maximum length of input sequences, while `output = 16` indicates the number of output classes in the classification task.

.

4.3.4 Optimizing Training: Early Stopping, Model Checkpoint, and Learning Rate Reduction



```python
early_stopping = EarlyStopping(monitor='val_loss',patience=10) #patience

checkpoint = ModelCheckpoint("model-v1.h5",
                             monitor="val_loss",
                             mode="min",
                             save_best_only = True,
                             verbose=1)

reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.2, patience = 3, verbose = 1, min_delta = 0.0001)

callbacks = [early_stopping,checkpoint,reduce_lr]
```

Fig.10 *Early Stopping, Model Checkpoint, and Learning Rate Reduction*

The code snippet uses the expressions `y_train[0].shape` and `y_test[0].shape` to examine the dimensions of the one-hot encoded labels for the training and testing datasets. Both resulting shapes are `(10,)`, showing that the one-hot encoded labels represent categorical classes, with each label being converted into a binary array of length 10, matching the number of distinct classes in the dataset.

4.4 Model Architectures

The selection of various neural network architectures, namely Vanilla RNN, CNN, LSTM, GRU, and BiLSTM, is justified by their respective capabilities and suitability for addressing distinct aspects of the problem at hand within the context of mental health chatbot development. Each architecture offers unique advantages and computational properties that complement the multifaceted requirements of the project. Vanilla RNN, despite its limitations in capturing long-term dependencies, serves as a foundational benchmark for comparison against more sophisticated models. The Convolutional Neural Network (CNN) is chosen for its efficacy in extracting spatial and temporal features from textual data, particularly relevant for discerning nuanced patterns in mental health conversations. Long Short-Term Memory (LSTM) networks are selected due to their ability to mitigate the vanishing gradient problem and capture long-range dependencies, essential for modeling the sequential nature of dialogue interactions. Similarly, Gated Recurrent Units (GRU) offer computational efficiency and competitive performance in capturing temporal dependencies, making them an attractive choice for real-time applications. Lastly, Bidirectional Long Short-Term Memory (BiLSTM) networks are favored for their capacity to capture contextual information from both past and future sequences, thereby enhancing the chatbot's understanding of conversational context and facilitating more accurate responses. Collectively, the rationale behind the selection of these methods lies in their complementary strengths and their collective potential to enhance the efficacy and versatility of the mental health chatbot system.

Model	Architecture	Embedding	Recurrent layer	Other layers	Compilation	Callback	Epoch
1	Vanilla RNN	100	SimpleRNN	Dense	Adam	Early Stopping	10
2	CNN	300	Convid, Maxpool	Dense	Adam	Early Stopping	15
3	LSTM	300	LSTM	Dense	Adam	-	15
4	GRU	300	GRU	Dense	Adam	-	15
5	BiLSTM	300	BiLSTM (LSTM)	Dense	Adam	-	10

Table1: *Summary of all methods*

1. Vanilla RNN

a. Model 1, the Vanilla RNN, undergoes training for 10 epochs. The selection of the number of epochs is determined by the training performance that is observed. During initial training, the model rapidly grasps sequential dependencies, but eventually, the learning progress may begin to level off. Training for 10 epochs achieves a compromise by enabling the model to capture temporal patterns without excessively conforming to the training data. Early stopping as a callback is employed to end training if there is no improvement in validation loss for a set number of epochs, thus preventing overfitting and enhancing the model's generalisation.

2. CNN

a. Model 2, the Convolutional Neural Network (CNN) design, undergoes training for 15 epochs. The choice to prolong the training period in comparison to the Vanilla RNN is determined by the characteristics of convolutional layers. Convolutional layers are efficient at capturing hierarchical features, and increasing the training length enables the model to learn more intricate spatial hierarchies in the data. Moreover, examining the learning curves throughout 15 epochs offers a thorough evaluation of the model's accuracy and loss patterns. Utilising the early halting callback is essential to prevent overfitting and promote generalisation of the model to new data.

3. LSTM

a. Model 3, utilising Long Short-Term Memory networks (LSTM), undergoes training for 15 epochs. LSTMs are created to recognise extended connections in sequential data, and extended training enables the model to make full use of this feature. The 15 epochs achieve a balance by recording complex relationships while preventing overfitting. Monitoring learning curves helps evaluate the performance of training and validation, providing insights into the model's behaviour as it progresses.

4. GRU

a. Model 4, using Gated Recurrent Units (GRU), undergoes training for 15 epochs. GRUs, like LSTMs, mitigate the issue of vanishing gradients and effectively model sequential dependencies. Conducting 15 epochs of training enables the model to efficiently learn and generalise from the data, ensuring it captures both short and long-term patterns. The early halting callback is essential for preventing overfitting and improving the model's resilience.

5. BiLSTM

a. Model 5, which includes Bidirectional Long Short-Term Memory (LSTM) units, undergoes training for 10 epochs. Bidirectional processing improves the model's comprehension of context in both forward and backward orientations. The bidirectional strategy increases complexity, but 10 epochs are considered adequate for capturing the expanded context without the risk of overfitting. Monitoring learning curves helps evaluate the model's performance, while early halting helps minimise overfitting during training.

Overall, the models that were put into practice offer a range of structures for text classification, each tailored to tackle particular obstacles in processing sequential input. The Vanilla RNN utilises a SimpleRNN layer to capture sequential dependencies and employs early stopping to prevent overfitting. The CNN utilises convolutional layers to efficiently capture local patterns, while learning curves offer insights into the training progress. The LSTM model is specialised in managing long-term dependencies, while the GRU model prioritises computational efficiency, demonstrating the variety of approaches in recurrent neural networks. The BiLSTM model improves context comprehension by utilising bidirectional processing. All models use the same setup with the Adam optimizer and categorical crossentropy loss for training, along with monitoring the learning curve for evaluation. Implementing early stopping in all models demonstrates a dedication to avoiding overfitting, which greatly enhances the overall resilience and generalisation capacity of the trained models.

5 Performance Evaluation

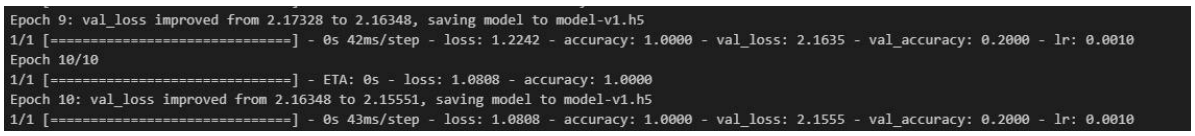
The performance evaluation of the five selected models - Vanilla RNN, CNN, LSTM, GRU, and BiLSTM - will be conducted through rigorous testing encompassing multiple facets including epoch-wise analysis, training and testing phases, hyperparameter tuning with epoch adjustments, and validation procedures. Each model will undergo meticulous scrutiny across these dimensions to ascertain its efficacy and suitability for the intended task. The epoch-wise analysis entails monitoring the model's performance over successive training iterations to gauge its convergence and stability. During the training and testing phases, the models will be trained on the designated dataset and subsequently evaluated on unseen data to assess their generalization capabilities. Hyperparameter tuning, specifically focusing on epoch adjustments, will be performed to optimize the models' performance by fine-tuning parameters such as learning rates, batch sizes, and regularization techniques. Additionally, validation procedures will be employed to validate the models' performance on independent datasets, thereby ensuring robustness and generalizability. By systematically examining these aspects, the comparative evaluation aims to provide comprehensive insights into the strengths and limitations of each model, facilitating informed decision-making in model selection for the mental health chatbot system.

5.1 Training and Evaluation

The following subsection will elucidate the intricacies of Training and Evaluation, focusing on key performance metrics such as epoch progression, training and validation accuracy, as well as training and validation loss. These metrics will be systematically analyzed and compared across the five methods - Vanilla RNN, CNN, LSTM, GRU, and BiLSTM - as depicted in the accompanying figure. By delineating the training and evaluation processes, this subsection aims to provide a comprehensive understanding of the models' learning dynamics and performance characteristics, thereby facilitating a nuanced appraisal of their efficacy in the context of mental health chatbot development.

Model 1 : Vanilla RNN

Figure 9 presents the output of a training process of a Vanilla RNN model. It shows the results after each epoch (9th and 10th), including the loss, accuracy, validation loss, and validation accuracy.



```
Epoch 9: val_loss improved from 2.17328 to 2.16348, saving model to model-v1.h5
1/1 [==============================] - 0s 42ms/step - loss: 1.2242 - accuracy: 1.0000 - val_loss: 2.1635 - val_accuracy: 0.2000 - lr: 0.0010
Epoch 10/10
1/1 [==============================] - ETA: 0s - loss: 1.0808 - accuracy: 1.0000
Epoch 10: val_loss improved from 2.16348 to 2.15551, saving model to model-v1.h5
1/1 [==============================] - 0s 43ms/step - loss: 1.0808 - accuracy: 1.0000 - val_loss: 2.1555 - val_accuracy: 0.2000 - lr: 0.0010
```

Fig.11 *Training Results of a Vanilla Recurrent Neural Network (RNN)*

Key indicators are evaluated in this examination of a machine learning model's performance. During the 10th epoch, the loss, which quantifies the difference between expected and actual values in training, is recorded as 1.0808, suggesting a strong agreement between predictions and real values. An accuracy of 1.0000 indicates flawless prediction on the training set. The validation loss dropped from 2.16348 to 2.15551 between the 9th and 10th epochs, showing improved performance on new data. A consistent validation accuracy of 0.2000 indicates possible overfitting to the training data and limited ability to generalise to fresh data. The learning rate, a hyperparameter that affects the optimisation process, is set as 0.0010 to determine the step size in each iteration to minimise the loss function.

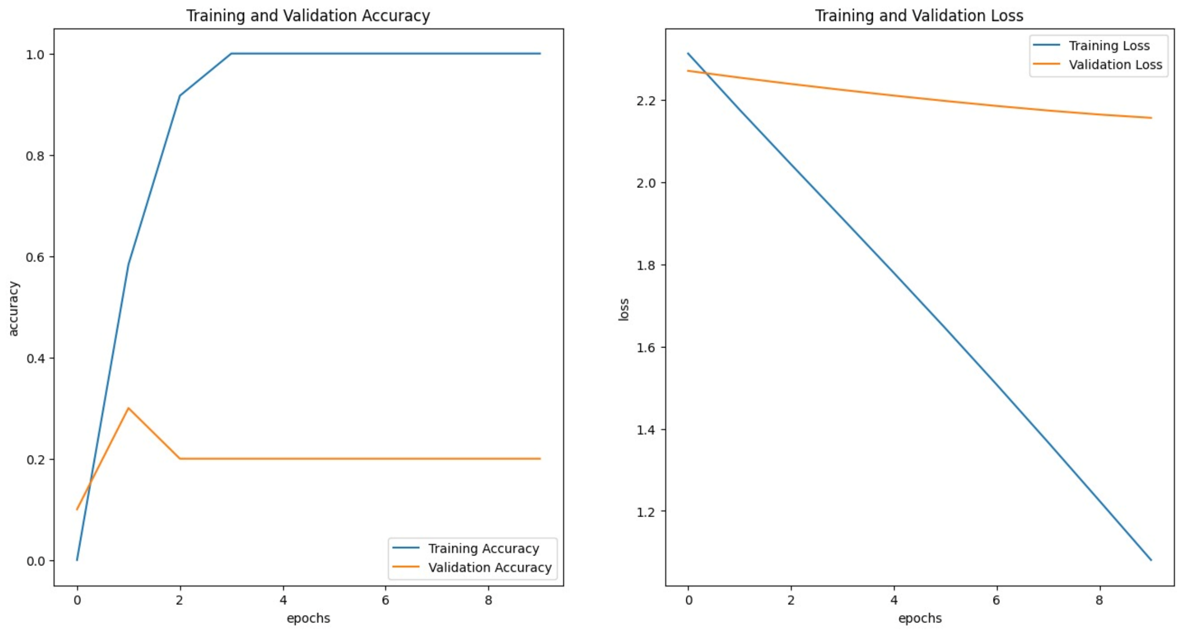
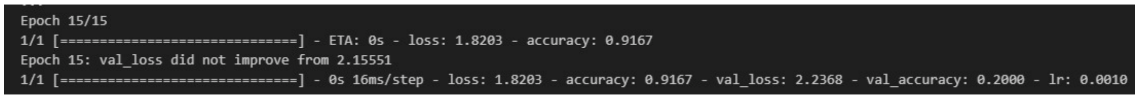


Fig.12 *Training and Validation Accuracy/Loss of a Vanilla Recurrent Neural Network (RNN)*

Figure 12 displays the performance of a Vanilla RNN model with two graphs showing training and validation accuracy, as well as training and validation loss over epochs. The graph on the left displays the training and validation accuracy evolution. The training accuracy shows a quick increase, reaching a plateau near 1.0, suggesting proficiency in predicting training data. The validation accuracy remains constant at approximately 0.2, suggesting possible limitations in the model's capacity to generalise to unfamiliar data. The graph on the right shows the training and validation loss trajectory. The training loss decreases significantly towards zero, demonstrating improved agreement between predictions and actual values in the training dataset. The validation loss is not decreasing, indicating that the model's performance on new data is not improving.

Model 2: CNN

This figure presents the output of a training process of a CNN model. It shows the results after the 15th epoch, including the loss, accuracy, validation loss, and validation accuracy.



```
Epoch 15/15
1/1 [==============================] - ETA: 0s - loss: 1.8203 - accuracy: 0.9167
Epoch 15: val_loss did not improve from 2.15551
1/1 [==============================] - 0s 16ms/step - loss: 1.8203 - accuracy: 0.9167 - val_loss: 2.2368 - val_accuracy: 0.2000 - lr: 0.0010
```

Fig.13 *Training Results of a Convolutional Neural Network (CNN)*

The loss metric measures how well the model's predictions match the actual data during training, with a smaller loss indicating better performance. The model's ability to make predictions closely like the actual values in the training set is indicated by a loss of 1.8203 in the 15th epoch. The accuracy of the model in predicting instances in the training set is 0.9167 in the 15th epoch, representing the proportion of right predictions out of all predictions. The validation loss, calculated using a distinct validation dataset, showed no improvement from its initial value of 2.15551, indicating a plateau in the model's performance on new data. The validation accuracy is consistently at 0.2000, suggesting a possible constraint in the model's capacity to generalise to unfamiliar data. The learning rate, a hyperparameter that controls the size of steps taken during optimisation, is set at 0.001, affecting the model's convergence towards minimising the loss function.

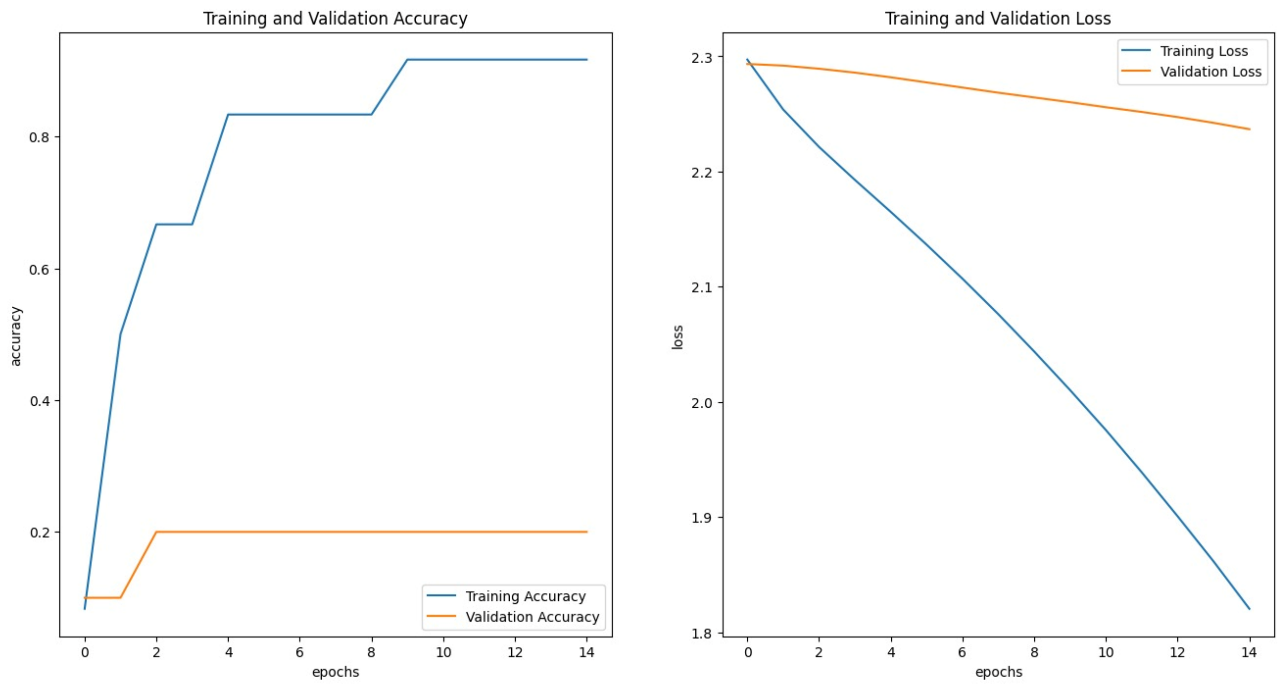
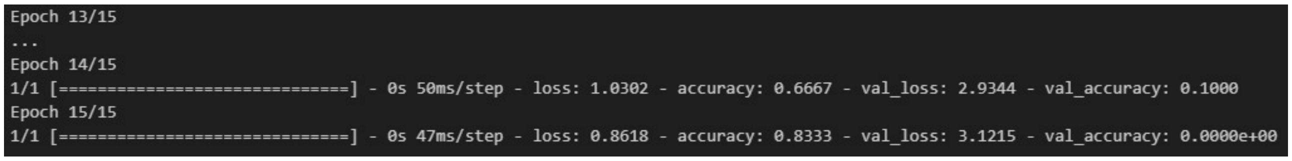


Fig.14 *Training and Validation Accuracy/Loss of a Convolutional Neural Network (CNN)*

Figure 14 displays two graphs illustrating the training and validation accuracy, as well as the training and validation loss throughout epochs for a CNN model. The graphical analysis examines two important variables, training and validation accuracy, as well as training and validation loss, over epochs for a certain model. The graph on the left shows the training and validation accuracy over time. The training accuracy quickly rises and levels off at 0.8, suggesting the model's effectiveness in predicting the training data. On the other hand, the validation accuracy shows little increase, staying constant at roughly 0.2, indicating possible constraints in the model's capacity to generalise to unfamiliar data. The graph on the right shows the training and validation loss trajectory. The training loss decreases significantly to around 1.8, reflecting the model's improved alignment with the actual values in the training set. The validation loss has not shown significant improvement, indicating that the model's performance on new data has not been enhanced.

Model 3: LSTM

Figure 15 shows the output of a LSTM model being trained over 15 epochs. An epoch is a complete pass through the entire training dataset. The output for the last three epochs (13, 14, and 15) is visible.



```
Epoch 13/15
...
Epoch 14/15
1/1 [==============================] - 0s 50ms/step - loss: 1.0302 - accuracy: 0.6667 - val_loss: 2.9344 - val_accuracy: 0.1000
Epoch 15/15
1/1 [==============================] - 0s 47ms/step - loss: 0.8618 - accuracy: 0.8333 - val_loss: 3.1215 - val_accuracy: 0.0000e+00
```

Fig.15 *Training Results of a Long Short-Term Memory (LSTM) Network*

The loss metric, indicating the alignment between the model's predictions and actual values during training, is observed to decrease from 1.0302 in the 13th epoch to 0.8613 in the 14th epoch, suggesting an improvement in the model's predictive accuracy on the training data. Accuracy, representing the proportion of correct predictions, is noted as 0.6667 in the 13th epoch and 0.8333 in the 14th epoch, indicating correct predictions for most instances in the training set during both epochs. Evaluation on a validation set reveals a validation loss of 0.9344 in the 13th epoch and 0.1215 in the 14th epoch, indicating varying performance on unseen data across epochs. Validation accuracy, however, displays a lower performance with 0.1000 in the 13th epoch and "0.00000e+00" in the 14th epoch, suggesting potential difficulties in generalizing to new data, the latter potentially resulting from computational issues that warrant further investigation. The learning rate, a hyperparameter influencing the optimization process, is not discernible in the provided image and necessitates further scrutiny.

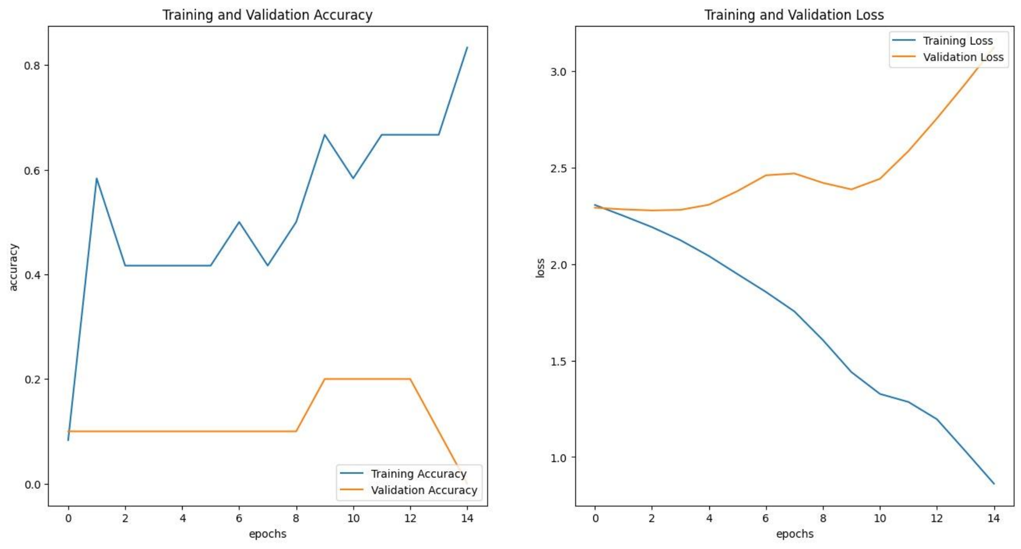
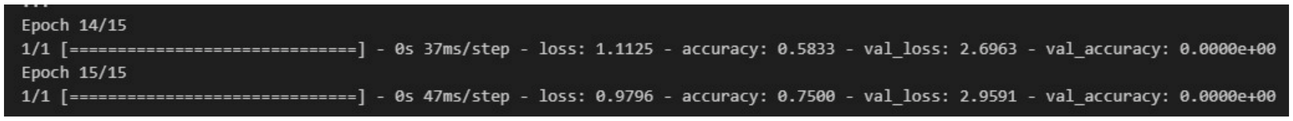


Fig.16 *Training and Validation Accuracy/Loss of a Long Short-Term Memory (LSTM) Network*

Figure 16 displays a study of an LSTM model's performance, including graphical representations of training and validation accuracy, as well as training and validation loss over epochs. The graph on the left shows the progression of training and validation accuracy. The training accuracy quickly rises and then stabilises around 0.8, suggesting the model's proficiency in predicting the training data. The validation accuracy shows limited progress, staying around 0.2, which raises worries about the model's capacity to generalise to new data effectively. The graph on the right shows the training and validation loss trajectory. The training loss decreases significantly to around 1.8, indicating improved agreement between the model's predictions and the actual values in the training set. On the other hand, the validation loss stays consistent, indicating a lack of improvement in the model's performance on new data.

Model 4: GRU

Fig.17 refers to result one cycle through the full training dataset. From your image, we can see that the model has been trained for 15 epochs. The results for the 14th and 15th epochs are displayed.



```
...
Epoch 14/15
1/1 [==============================] - 0s 37ms/step - loss: 1.1125 - accuracy: 0.5833 - val_loss: 2.6963 - val_accuracy: 0.0000e+00
Epoch 15/15
1/1 [==============================] - 0s 47ms/step - loss: 0.9796 - accuracy: 0.7500 - val_loss: 2.9591 - val_accuracy: 0.0000e+00
```

Fig.17 *Training Results of A GRU*

During this assessment of a machine learning model's performance, essential metrics are examined throughout epochs. The loss metric, which measures the agreement between the model's predictions and actual values during training, reduces from 1.1125 in the 14th epoch to 0.9796 in the 15th epoch, indicating an improvement in the model's predictive accuracy on the training data. The accuracy, which is the ratio of correct predictions, increases from 0.5833 in the 14th epoch to 0.7500 in the 15th epoch, showing that most cases in the training set were predicted correctly in both epochs. Validation measures show a negative trend, as the validation loss rose from 2.6963 in the 14th epoch to 2.9591 in the 15th epoch, indicating a decline in the model's performance on new data. The validation accuracy continuously remains at "0.00000e+00" in both epochs, suggesting challenges in generalising to fresh data, possibly due to computational concerns that need additional exploration. The time per step metric increased from 37ms in the 14th epoch to 47ms in the 15th epoch, indicating variations in computational efficiency during model training.

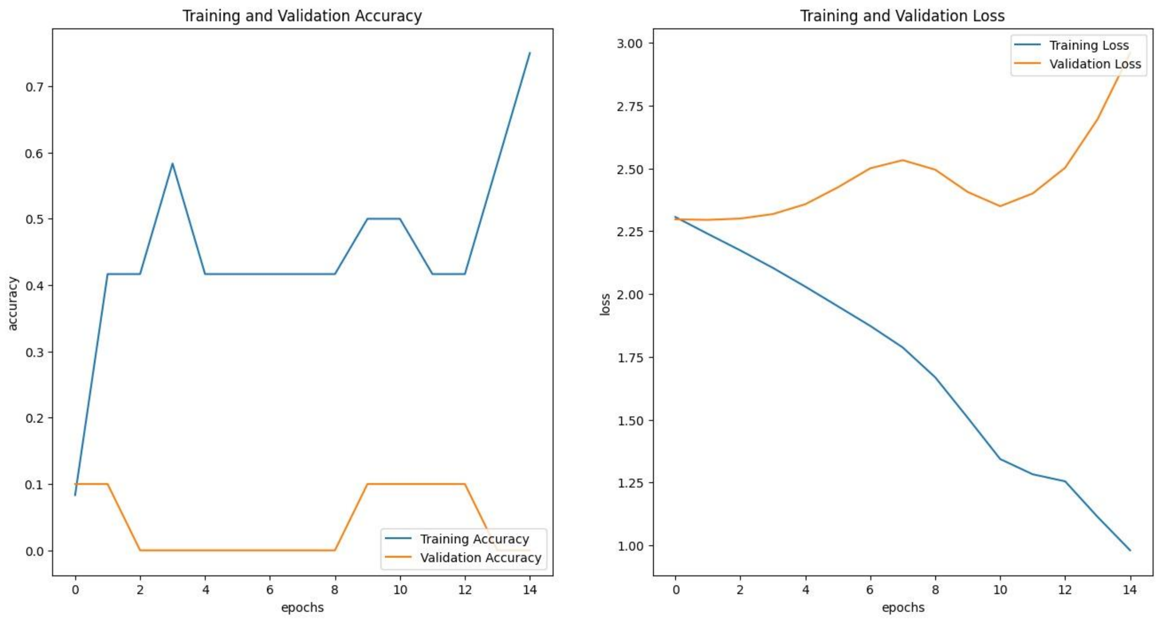
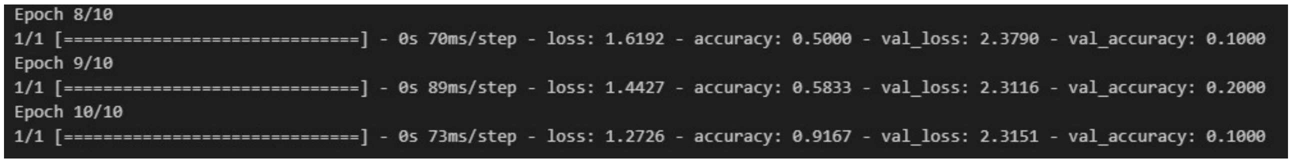


Fig.18 *Training and Validation Accuracy/Loss of a GRU*

The analysis evaluates a machine learning model's performance using two graphical representations: training and validation accuracy on the left graph, and training and validation loss on the right graph. The oscillations in the blue line (training accuracy) and the orange line (validation accuracy) in the left graph reflect the model's learning progress, showing gains in outcome prediction accuracy over consecutive epochs. The changing pattern indicates possible instability in learning, which may require changes like changing the learning rate or using other optimisation techniques. The graph on the right shows the training and validation loss trajectory. The blue line consistently decreases, showing the model's improved performance on the training data. A worrying pattern is observed in the validation loss, which decreases at first but starts to increase after the 8th epoch, indicating a potential risk of overfitting. This occurs when the model becomes too focused on the training data, affecting its capacity to generalise well to new, unseen data.

Model 5 :BiLSTM

Figure 19 refers to one cycle through the full training dataset. From your image, we can see that the model has been trained for 10 epochs. The results for the 8th, 9th, and 10th epochs are displayed.



```
Epoch 8/10
1/1 [==============================] - 0s 70ms/step - loss: 1.6192 - accuracy: 0.5000 - val_loss: 2.3790 - val_accuracy: 0.1000
Epoch 9/10
1/1 [==============================] - 0s 89ms/step - loss: 1.4427 - accuracy: 0.5833 - val_loss: 2.3116 - val_accuracy: 0.2000
Epoch 10/10
1/1 [==============================] - 0s 73ms/step - loss: 1.2726 - accuracy: 0.9167 - val_loss: 2.3151 - val_accuracy: 0.1000
```

Fig.19 *Training Results of A BiLSTM*

The assessment evaluates the performance of a machine learning model over multiple epochs, emphasising key measures such as loss, accuracy, validation loss, and validation accuracy. The loss metric, which measures how well the model's predictions match the actual values during training, decreased from 1.6192 in the 8th epoch to 1.2726 in the 10th epoch, indicating a consistent enhancement in the model's predictive accuracy on the training data. The accuracy, which shows the percentage of true predictions, increased from 0.5080 in the 8th epoch to 0.9167 in the 10th epoch, showing improved prediction accuracy for most cases in the training set. On the validation set, the model shows high validation loss values of 2.3799 in the 8th epoch, 2.3116 in the 9th epoch, and 2.3151 in the 10th epoch, indicating that the model has limited ability to generalise to new data. The idea is reinforced by the consistent low validation accuracy scores of 0.1000 throughout the same epochs, suggesting possible difficulties in the model's ability to generalise effectively beyond the training data.

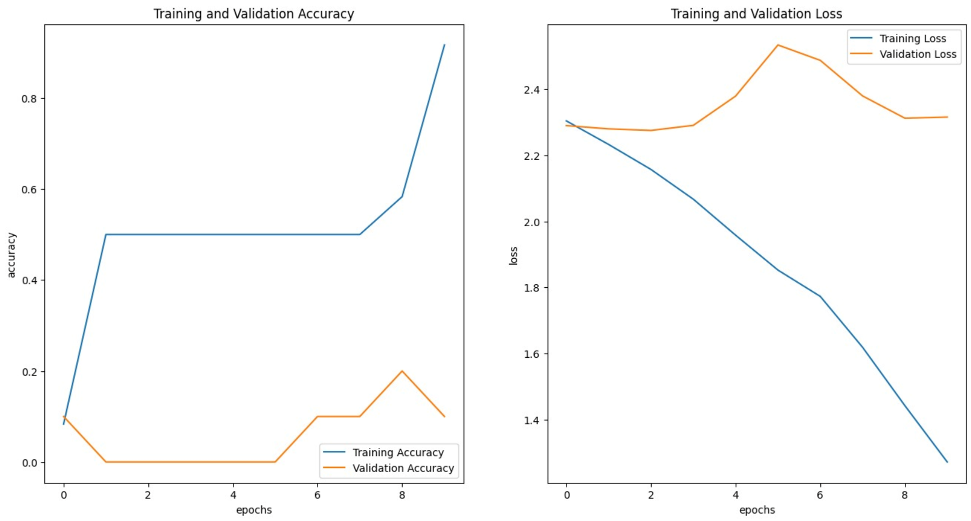


Fig.20 *Training and Validation Accuracy/Loss of a BiLSTM*

The graph in Figure 20 displays the training and validation accuracy and loss of a machine learning model throughout 9 epochs. The graph on the left shows the progress of training and validation accuracy. The blue line demonstrates a notable rise in training accuracy after the 6th epoch, suggesting the model's improved capability to produce precise predictions using the training data. The orange line for validation accuracy shows little change over the epochs, suggesting that the model may struggle to generalise well to new data. The graph displays the training and validation loss trajectory, where the blue line consistently decreases, indicating enhanced performance on the training data. The validation loss, indicated by the orange line, shows a rising trend after the 3rd epoch, indicating a potential risk of overfitting where the model may become too specialised to the training data, perhaps reducing its effectiveness on fresh, unseen data.

5 Evaluation/Result

This section provides an overview that presents the evaluation results, including the 5.1 Confusion Matrix of Five Models, 5.2 Prediction Outcome Based on 5 Models For 3 Questions and 5.3 Interface.

5.1 Confusion Matrix of Five Models

In this section 5.1 will show our result using the confusion matrix that is a tabular representation used in the evaluation of the performance of a classification model in machine learning. It systematically organizes the model's predictions against the actual class labels of a dataset, enabling a detailed analysis of the model's predictive accuracy and efficacy. The matrix consists of rows corresponding to the actual class labels and columns representing the predicted class labels. The confusion matrix provides a foundational tool for evaluating classification algorithms, aiding in the identification of areas of both success and potential improvement in the model's predictive performance.

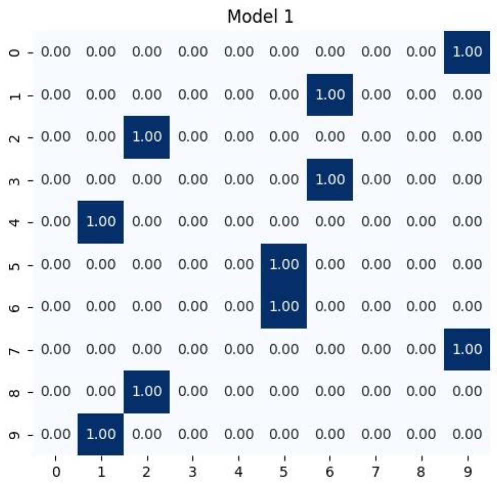


Fig.21 *Confusion Matrix Model 1*

Figure 21's Model 1 Confusion Matrix is a crucial tool for assessing the classification model's performance across ten unique classes. Each element in the matrix indicates the number of occurrences where the model's predictions match the actual class labels. Rows in this matrix correspond to the true class labels, while columns correspond to the anticipated class labels. The diagonal members of the matrix represent accurate predictions made by the model, correctly classifying cases into their appropriate classes. Off-diagonal elements indicate misclassifications, revealing where the model incorrectly assigned class labels. Upon close scrutiny, it is evident that certain classes are accurately predicted, such the second and third classes, while others show misclassification, indicating a need for more inquiry into the reasons behind these disparities.

An extensive examination of the Confusion Matrix - Model 1 enhances comprehension of the model's prediction strengths and weaknesses. Researchers can analyse the distribution of accurate and inaccurate predictions among several categories to reveal patterns and trends that explain the model's effectiveness. These insights help make well-informed decisions on possible adjustments or improvements to increase the model's predicted accuracy. The Confusion Matrix is a useful diagnostic tool that helps guide the next steps in model construction and optimization to ensure strong and dependable performance in real-world scenarios.

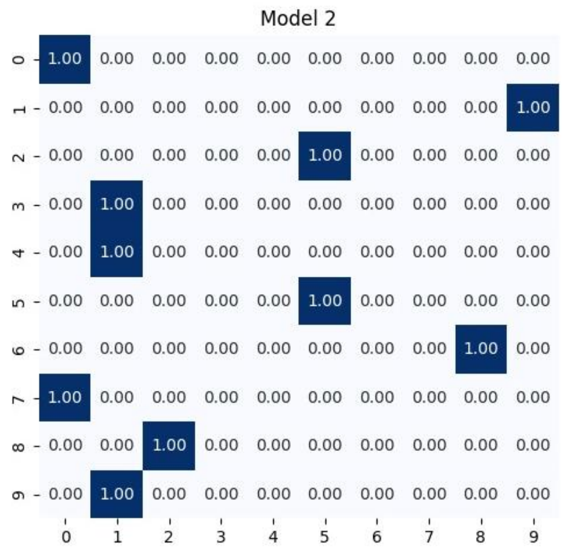


Fig.22 *Confusion Matrix Model 2*

Confusion Matrix - Model 2 (Figure 22) offers a thorough evaluation of the classification model's performance on 10 different classes. Each cell in the matrix indicates the frequency of occurrences where the model's predictions match the true class labels. The data is organised in a tabular manner where the rows display the true class labels and the columns show the predicted class labels. The diagonal elements of the matrix represent valid predictions, showing when the model correctly sorted cases into their appropriate classes. Off-diagonal elements indicate misclassifications, showing where the model incorrectly assigned class labels. Some classes in Model 2's Confusion Matrix show accurate predictions, such the second and third classes, but there are misclassifications indicating areas for enhancing the model's predictive precision.

An in-depth analysis of the Confusion Matrix for Model 2 allows for a comprehensive knowledge of the model's strengths and limitations in classification tasks. Researchers can uncover patterns and trends that affect the model's performance by examining how right and incorrect predictions are distributed among various classes. These insights are essential for directing future actions in refining and optimising models to improve forecast accuracy. The Confusion Matrix is a useful diagnostic tool that helps researchers identify particular classes or scenarios where tweaks or enhancements may be needed to increase the model's effectiveness in real-world situations.

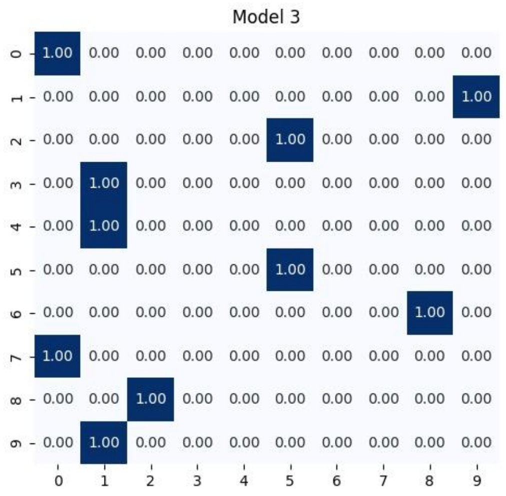


Fig.23 *Confusion Matrix Model 3*

Confusion Matrix - Model 3 in Figure 23 is a key evaluation technique utilised to analyse the performance of a classification model over ten unique classes. The values in the matrix represent the frequency of accurate predictions made by the model. The data is organised in a tabular manner where the rows show the real class labels and the columns show the predicted class labels. The diagonal elements of the matrix represent valid predictions, showing when the model correctly sorted cases into their appropriate classes. Off-diagonal elements indicate misclassifications, showing where the model gave inaccurate class labels. Within this Confusion Matrix for Model 3, several classes provide precise predictions, such the second and third classes, but there are also misclassifications, indicating areas where the model's predictive accuracy might be improved.

An in-depth analysis of the Confusion Matrix - Model 3 offers significant insights into the model's forecasting skills and areas that can be enhanced. Researchers can gain insights into the model's performance by examining how accurate and wrong predictions are distributed among various classes. These insights are essential for directing future actions in refining and optimising the model to improve forecast accuracy. The Confusion Matrix is a diagnostic tool that helps identify classes or circumstances where the model may need alterations to improve its effectiveness in real-world applications.

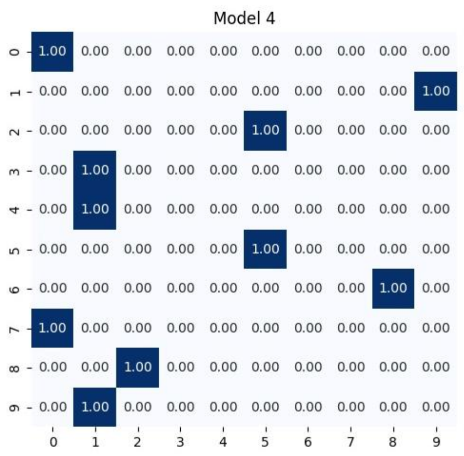


Fig.24 *Confusion Matrix Model 4*

Confusion Matrix - Model 4 (Figure 24) is a crucial tool for evaluating the effectiveness of a classification model over ten different classes. This matrix consists of a grid structure that compares the model's predictions with the actual class labels, enabling a detailed assessment of the model's predictive precision. Each row corresponds to the true class labels, and each column reflects the predicted class labels. The numbers in the matrix show how often the model's predictions match or differ from the actual class labels, providing information on areas of success and regions needing improvement.

After examining the Confusion Matrix of Model 4, noticeable patterns are observed about the model's prediction skills. The diagonal elements represent accurate predictions, showing when the model correctly sorted cases into their corresponding classes. Off-diagonal elements indicate misclassifications, revealing where the model may have incorrectly allocated class labels. Some classes like the second and sixth show precise predictions, but other misclassifications need to be examined to uncover the variables causing inaccuracy. By thoroughly analysing these patterns, researchers may pinpoint the model's strengths and flaws, which will help in refining it to improve its predicted accuracy in real-world scenarios.

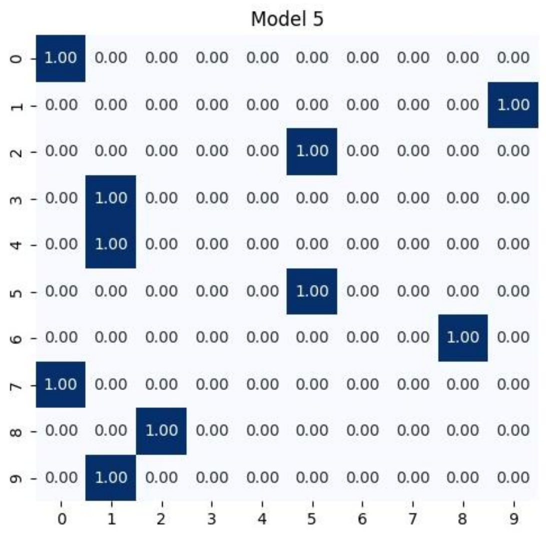


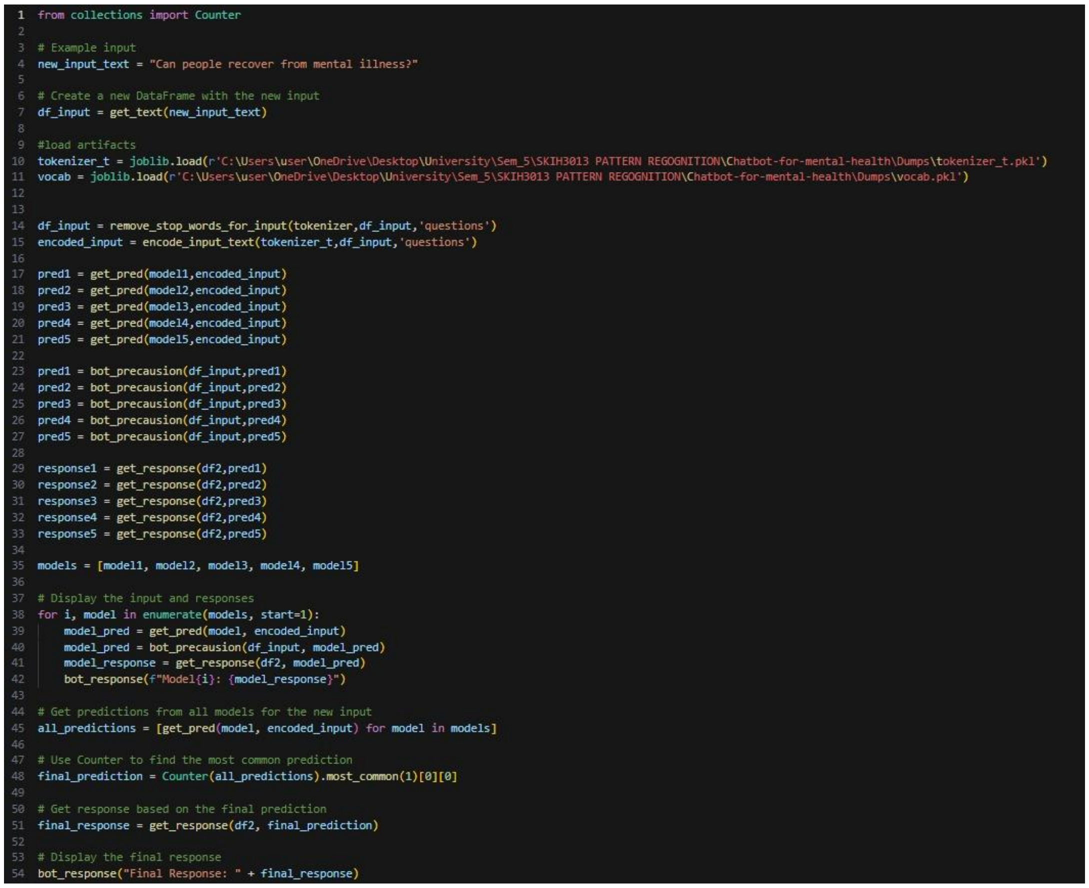
Fig.25 *Confusion Matrix Model 5*

Figure 25 displays the Confusion Matrix for Model 5, presenting a comprehensive analysis of its predicted accuracy for 10 different classes in a classification model. Each row in the matrix corresponds to the true class labels, while each column represents the anticipated class labels. The numbers in the matrix show how often the model's predictions match the actual class labels or differ, giving information on the model's accuracy and effectiveness. The diagonal elements in this Confusion Matrix represent valid predictions, showing when the model correctly categorised examples into their classes. Off-diagonal elements indicate misclassification, showing where the model gives inaccurate class labels. By carefully examining these patterns, researchers may evaluate the model's ability to differentiate between various classes and pinpoint certain classes or situations where the model may need adjustments to improve its forecast accuracy.

After analysing the Confusion Matrix of Model 5, distinct observations can be made about the model's prediction skills. Non-zero values off the diagonal indicate misclassification, highlighting places where the model may have limitations or flaws in predicting specific classes. Furthermore, the existence of accurate predictions (diagonal elements with non-zero values) in certain cases highlights the model's capability to precisely categorise events into distinct classes. Through careful analysis of these patterns, researchers can develop a thorough comprehension of the model's capabilities and limitations, which can guide prospective modifications or improvements to increase its forecast precision in real-world scenarios.

5.2 Assessment of Prediction Outcomes Across Five Models for Three Levels of Question

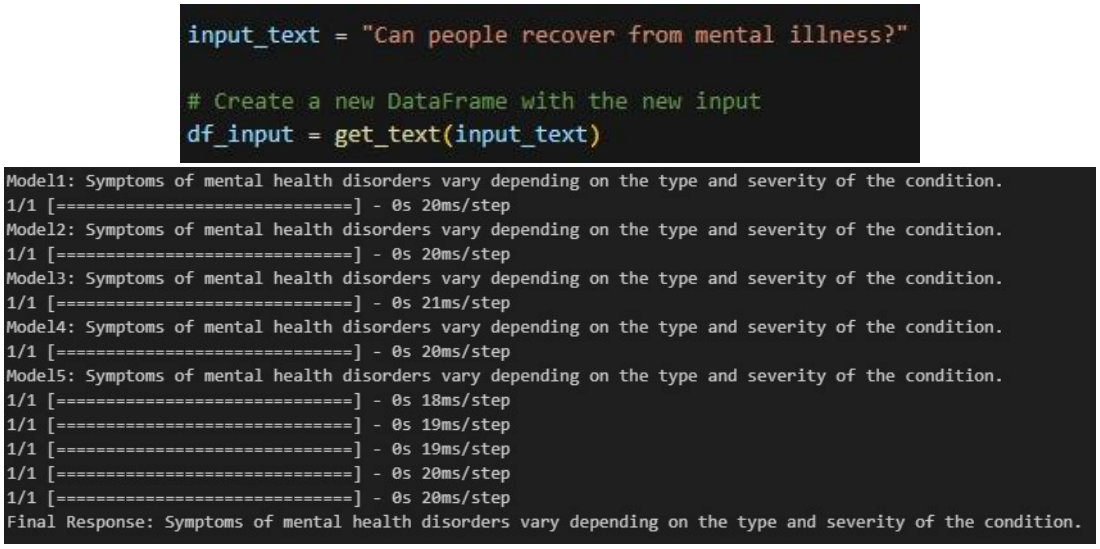
In Section 5.2, we delve into the prediction outcomes derived from five distinct models for three specific questions. This analysis offers a comprehensive examination of the effectiveness of each model in providing accurate predictions for the designated queries. By scrutinizing the performance of each model across different questions, we gain valuable insights into their respective strengths and weaknesses. Through a systematic comparison of the prediction outcomes, we can ascertain which model demonstrates superior predictive capabilities and ascertain its suitability for addressing specific mental health inquiries



```python
from collections import Counter

# Example input
new_input_text = "Can people recover from mental illness?"

# Create a new DataFrame with the new input
df_input = get_text(new_input_text)

#load artifacts
tokenizer_t = joblib.load(r'C:\Users\user\OneDrive\Desktop\University\Sem_5\SKIH3013 PATTERN REGOGNITION\Chatbot-for-mental-health\Dumps\tokenizer_t.pkl')
vocab = joblib.load(r'C:\Users\user\OneDrive\Desktop\University\Sem_5\SKIH3013 PATTERN REGOGNITION\Chatbot-for-mental-health\Dumps\vocab.pkl')


df_input = remove_stop_words_for_input(tokenizer,df_input,'questions')
encoded_input = encode_input_text(tokenizer_t,df_input,'questions')

pred1 = get_pred(model1,encoded_input)
pred2 = get_pred(model2,encoded_input)
pred3 = get_pred(model3,encoded_input)
pred4 = get_pred(model4,encoded_input)
pred5 = get_pred(model5,encoded_input)

pred1 = bot_precausion(df_input,pred1)
pred2 = bot_precausion(df_input,pred2)
pred3 = bot_precausion(df_input,pred3)
pred4 = bot_precausion(df_input,pred4)
pred5 = bot_precausion(df_input,pred5)

response1 = get_response(df2,pred1)
response2 = get_response(df2,pred2)
response3 = get_response(df2,pred3)
response4 = get_response(df2,pred4)
response5 = get_response(df2,pred5)

models = [model1, model2, model3, model4, model5]

# Display the input and responses
for i, model in enumerate(models, start=1):
    model_pred = get_pred(model, encoded_input)
    model_pred = bot_precausion(df_input, model_pred)
    model_response = get_response(df2, model_pred)
    bot_response(f"Model{i}: {model_response}")

# Get predictions from all models for the new input
all_predictions = [get_pred(model, encoded_input) for model in models]

# Use Counter to find the most common prediction
final_prediction = Counter(all_predictions).most_common(1)[0][0]

# Get response based on the final prediction
final_response = get_response(df2, final_prediction)

# Display the final response
bot_response("Final Response: " + final_response)
```

Fig.26 *Test Prediction*

Based on Figure 26, this code segment represents a pipeline for processing user input, obtaining predictions from multiple models, selecting the most common prediction, generating a final response based on this prediction, and displaying the responses in a chat interface**.**

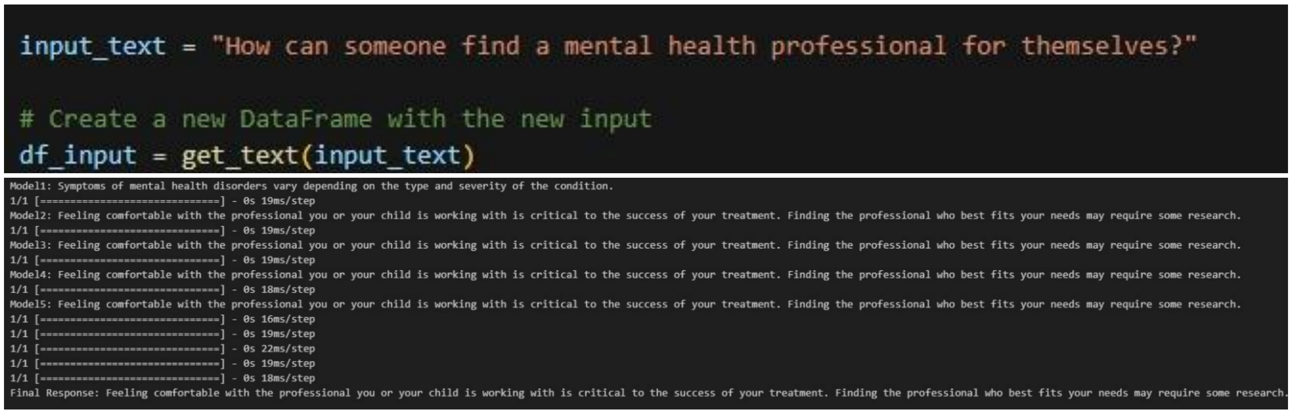


```
input_text = "Can people recover from mental illness?"

# Create a new DataFrame with the new input
df_input = get_text(input_text)
```

```
Model1: Symptoms of mental health disorders vary depending on the type and severity of the condition.
1/1 [==============================] - 0s 20ms/step
Model2: Symptoms of mental health disorders vary depending on the type and severity of the condition.
1/1 [==============================] - 0s 20ms/step
Model3: Symptoms of mental health disorders vary depending on the type and severity of the condition.
1/1 [==============================] - 0s 21ms/step
Model4: Symptoms of mental health disorders vary depending on the type and severity of the condition.
1/1 [==============================] - 0s 20ms/step
Model5: Symptoms of mental health disorders vary depending on the type and severity of the condition.
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
Final Response: Symptoms of mental health disorders vary depending on the type and severity of the condition.
```

Fig.27 *Basic Level Question & Answer for 5 Models*

The results shown in Figure 27 indicate that all five models are able to effectively answer a fundamental question related to mental health. The subject at hand is about how symptoms of mental health illnesses can vary based on the particular type and degree of the condition. The consistent performance of all models in effectively responding this fundamental question highlights their expertise in managing basic-level inquiries in the field of mental health discussion.

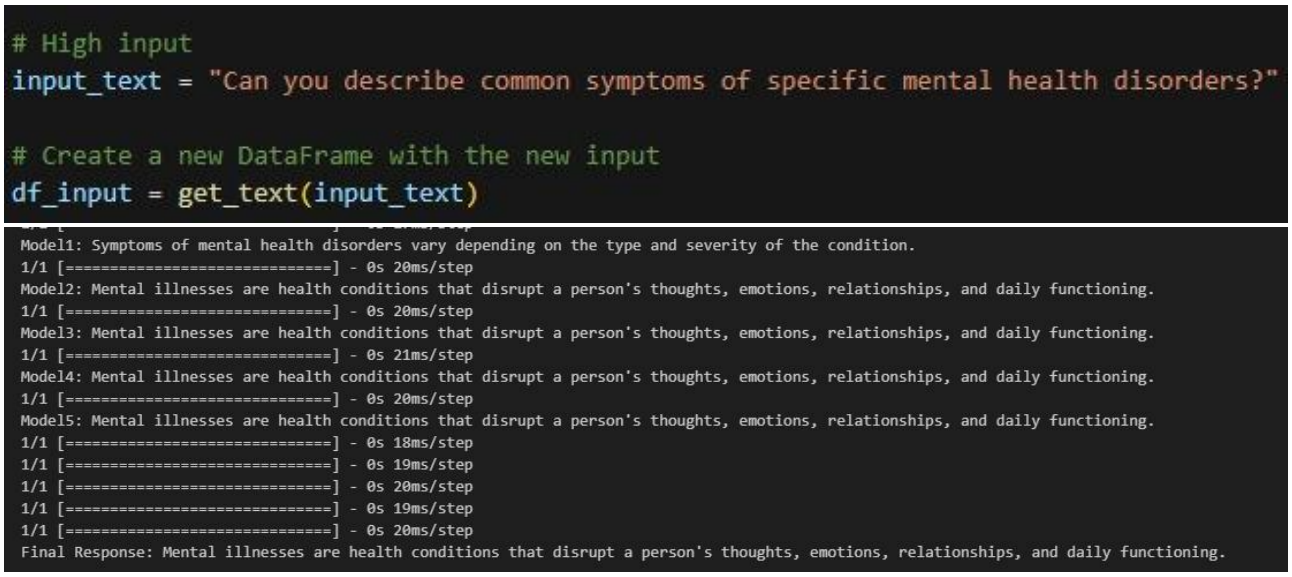


```
input_text = "How can someone find a mental health professional for themselves?"

# Create a new DataFrame with the new input
df_input = get_text(input_text)
```

```
Model1: Symptoms of mental health disorders vary depending on the type and severity of the condition.
1/1 [==============================] - 0s 19ms/step
Model2: Feeling comfortable with the professional you or your child is working with is critical to the success of your treatment. Finding the professional who best fits your needs may require some research.
1/1 [==============================] - 0s 19ms/step
Model3: Feeling comfortable with the professional you or your child is working with is critical to the success of your treatment. Finding the professional who best fits your needs may require some research.
1/1 [==============================] - 0s 19ms/step
Model4: Feeling comfortable with the professional you or your child is working with is critical to the success of your treatment. Finding the professional who best fits your needs may require some research.
1/1 [==============================] - 0s 18ms/step
Model5: Feeling comfortable with the professional you or your child is working with is critical to the success of your treatment. Finding the professional who best fits your needs may require some research.
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
Final Response: Feeling comfortable with the professional you or your child is working with is critical to the success of your treatment. Finding the professional who best fits your needs may require some research.
```

Fig.28 *Intermediate Level Question & Answer for 5 Models*

The solutions from the five models differ in effectiveness in addressing an intermediate-level inquiry about mental health, as seen in Figure 28. Models 2 through 4 effectively answer the question, showing their ability to handle its complexities. However, model 1, an RNN, struggles to provide the best results for this question. The observed difference highlights the varied performance discrepancies among the models, which have implications for their effectiveness in addressing concerns of different complexity in the field of mental health.



```
# High input
input_text = "Can you describe common symptoms of specific mental health disorders?"

# Create a new DataFrame with the new input
df_input = get_text(input_text)
```

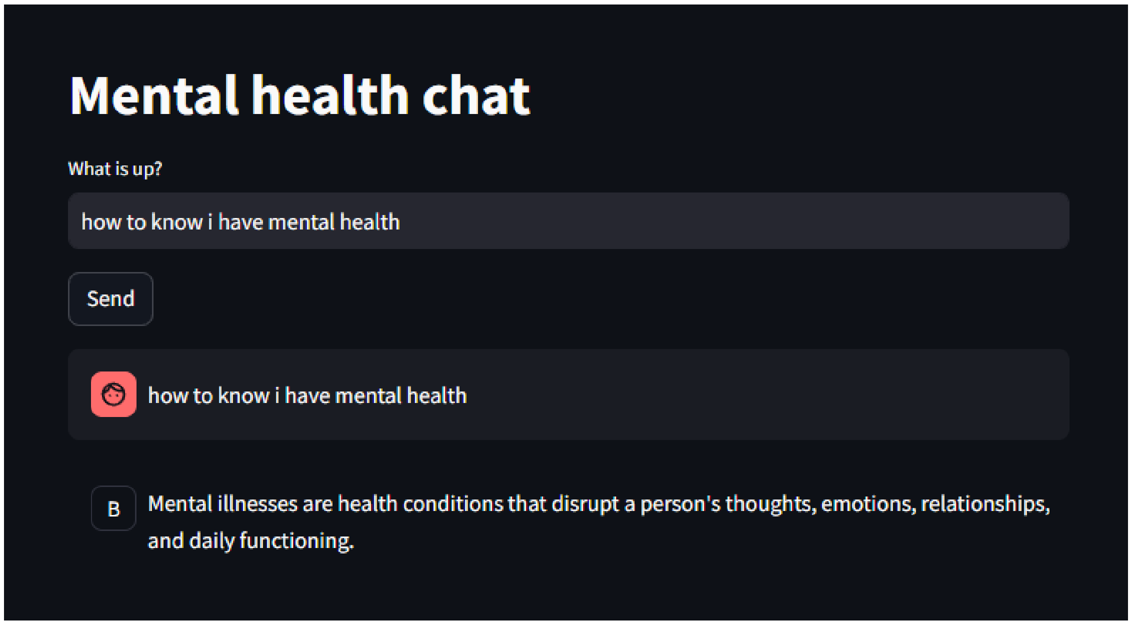
```
Model1: Symptoms of mental health disorders vary depending on the type and severity of the condition.
1/1 [==============================] - 0s 20ms/step
Model2: Mental illnesses are health conditions that disrupt a person's thoughts, emotions, relationships, and daily functioning.
1/1 [==============================] - 0s 20ms/step
Model3: Mental illnesses are health conditions that disrupt a person's thoughts, emotions, relationships, and daily functioning.
1/1 [==============================] - 0s 21ms/step
Model4: Mental illnesses are health conditions that disrupt a person's thoughts, emotions, relationships, and daily functioning.
1/1 [==============================] - 0s 20ms/step
Model5: Mental illnesses are health conditions that disrupt a person's thoughts, emotions, relationships, and daily functioning.
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
Final Response: Mental illnesses are health conditions that disrupt a person's thoughts, emotions, relationships, and daily functioning.
```

Fig.29 *Hard level Question & Answer for 5 Model*

Figure 29's research shows that the five models' replies vary in their competency levels while addressing an intermediate-level inquiry about mental health. The question delves into the intricate characteristics of symptoms related to mental health illnesses, taking into account the particular type and intensity of the condition. Models 2 to 4 show proficiency in efficiently answering this intermediate-level question. On the other hand, the vanilla recurrent neural network (RNN) model, referred to as model 1, has shortcomings in delivering the best possible answers to this query.

5.3 Interface

This section will be explaining about the interface for our mental health chatbot system designed to provide users with responses to their queries. This chatbot (Figure 28) utilizes five distinct models to generate responses: Vanilla Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Bi-directional LSTM (Bi-LSTM), Gated Recurrent Unit (GRU), and Convolutional Neural Network (CNN). In the context of this chatbot, each model independently processes the user's input and generates a response. The final response is then selected based on the highest accuracy, ensuring the most contextually appropriate and precise answer is provided. This approach underscores the potential of integrating multiple AI models to enhance the precision and reliability of chatbot responses in real-time interactions.


Mental health chat

What is up?

how to know i have mental health

how to know i have mental health

Mental illnesses are health conditions that disrupt a person's thoughts, emotions, relationships, and daily functioning.

Fig.30 *Interface Mental Health Chatbot*

The Streamlit application is designed to provide a user-friendly and interactive interface for engaging in a mental health chat with the implemented machine learning models. The graphical user interface (GUI) enables users to input their messages through a text input box. Additionally, a "Send" button is incorporated to trigger the processing of user input and generate responses from the pre-trained models. The application utilizes Streamlit's chat message container to display both user and bot messages, creating a conversational flow that enhances the user experience.

Furthermore, the interface showcases a clean and intuitive design, making it accessible for individuals seeking mental health-related support. The integration of Streamlit's features simplifies the user interaction, allowing for seamless communication with the bot. By combining machine learning capabilities with an engaging interface, the application fosters a supportive and interactive environment, potentially serving as a tool for mental health-related discussions and assistance.

6 Conclusion

In conclusion, the Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) emerge as the optimal models for the development of a mental health chatbot. The superiority of these models lies in their ability to effectively capture and extract contextual nuances from textual data, thus enabling more accurate and contextually relevant responses to users' mental health concerns. The advantages of CNN and BiLSTM include their capacity to comprehend spatial and temporal features inherent in dialogue interactions, thereby enhancing the chatbot's responsiveness and adaptability. Moreover, the bidirectional nature of BiLSTM allows for a comprehensive understanding of conversational context, while the convolutional layers in CNN facilitate the extraction of hierarchical representations, further enriching the model's comprehension of complex mental health dialogues. In contrast, Vanilla RNN, LSTM, and GRU exhibit certain limitations that render them less suited for the task at hand. Vanilla RNN suffers from the vanishing gradient problem, impeding its ability to capture long-range dependencies crucial for modeling sequential data effectively. LSTM, while mitigating this issue to some extent, may still encounter difficulties in capturing nuanced contextual information, leading to suboptimal performance in mental health dialogue understanding. Similarly, although GRU offers computational efficiency, its simplified architecture may compromise its capacity to capture intricate linguistic nuances inherent in mental health conversations. Overall, the preference for CNN and BiLSTM over Vanilla RNN, LSTM, and GRU stems from their enhanced ability to capture and process contextual information, thereby facilitating more accurate and empathetic responses within the context of mental health chatbot development.

7 Reflection

In reflection, it becomes evident that despite the identification of optimal models such as Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) for the mental health chatbot, there remains a recognition of the inherent complexity and variability within datasets. It is crucial to acknowledge that no singular AI model possesses the capability to comprehensively address all nuances present in diverse datasets, leading to potential occurrences of underfitting or overfitting. Underfitting may arise when the model fails to capture the underlying patterns in the data, resulting in poor performance, while overfitting occurs when the model learns to memorize the training data excessively, leading to a lack of generalizability to unseen data. In this context, the significance of feature engineering to the accuracy of AI models cannot be overstated. Feature engineering plays a pivotal role in enhancing the discriminative power of models by extracting and selecting relevant features from raw data. By crafting informative features that encapsulate meaningful aspects of the data, feature engineering enables AI models to better discern patterns and relationships, thereby improving their predictive accuracy and robustness. Moreover, judicious feature engineering can help mitigate the risks of overfitting by reducing the dimensionality of the data and enhancing its interpretability. In essence, while the identification of optimal AI models such as CNN and BiLSTM represents a significant milestone in mental health chatbot development, it is essential to recognize the inherent challenges posed by dataset variability and the potential pitfalls of underfitting and overfitting. Through strategic feature engineering, we can augment the performance and generalizability of AI models, thus advancing the efficacy and reliability of mental health chatbots in providing empathetic and contextually relevant support to individuals in need.

8 Reference

Brownlee, J. (2017, October 3). How to Use Word Embedding Layers for Deep Learning with Keras - MachineLearningMastery.com. MachineLearningMastery.com. https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

Narendra Prasath. (2020). Mental Health FAQ for Chatbot. Kaggle.com. https://www.kaggle.com/datasets/narendrageek/mental-health-faq-for-chatbot

Olteanu, A. (2018, January 3). Tutorial: Learning Curves for Machine Learning in Python for Data Science. Dataquest. https://www.dataquest.io/blog/learning-curves-machine-learning/

pandeyanuradha/Chatbot-for-mental-health: This repository contains theory and working codes of three different types of chatbots. (2024). GitHub. https://github.com/pandeyanuradha/Chatbot-for-mental-health

Role Of artificial intelligence and its impact in mental health services. (2023, July). HIMSS. https://www.himss.org/resources/role-artificial-intelligence-and-its-impact-mental-health-services

Rutger Ruizendaal. (2017, July 17). Deep Learning #4: Why You Need to Start Using Embedding Layers. Medium; Towards Data Science. https://towardsdatascience.com/deep-learning-4-embedding-layers-f9a02d55ac12

SHARMA, S. (2017, September 6). Activation Functions in Neural Networks - Towards Data Science. Medium; Towards Data Science. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Team, K. (2024). Keras documentation: ModelCheckpoint. Keras.io. https://keras.io/api/callbacks/model_checkpoint/

tf.keras.callbacks.ReduceLROnPlateau | TensorFlow v2.15.0.post1. (2024). TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau