Final Project: Fine-Tuning MentalBERT for Mental Health Text Classification Using Grid Search and Random Search Optimization

Dominic Boy Almazan
ITE-IV, BSIT
Jose Rizal University
Mandaluyong City, Philippines
dominicboy.almazan@my.jru.edu

Shekinah Rouise Tejada
ITE-IV, BSIT
Jose Rizal University
Mandaluyong City, Philippines
shekinahrouise.tejada@my.jru.edu

Samantha Reign Pajanustan
ITE-IV, BSIT
Jose Rizal University
Mandaluyong City, Philippines
samanthareign.pajanustan@my.jru.edu

Abstract— **Mental health problems shared on social media and online texts have increased the need for automated systems that can detect early signs of distress. In this study, we first preprocessed the text data, then fine-tuned the MentalBERT model for mental health classification, and finally optimized its hyperparameters using both Random Search and Grid Search. Among the base models, Tejada performed best with the lowest evaluation loss (0.075), high accuracy (0.98), and strong F1-score (0.975). After hyperparameter tuning, Almazan achieved the highest accuracy and F1 scores, showing that careful tuning improves model reliability. Pajanustan S. consistently performed worse. This work supports SDG 3, Target 3.4 by promoting mental health awareness and early detection and SDG 4.1 by providing access to knowledge and education about mental well-being.**

Keywords—MentalBERT, Mental Health Classification, Fine-Tuning BERT, Hyperparameter Optimization, Grid Search, Random Search, Transform Models, Natural Language Processing (NLP), Sustainable Development Goal, SDG 3, Good Health and Well Being

I. INTRODUCTION

Social media, forums, and messaging services are online platforms that have gained importance as critical outlets in the past few years to individuals in a distressing state of mental health, including suicidal ideation. It is true that the language that people use to talk about the state of their mind usually carries some minor hints that might signify that there is a high risk of a serious problem, and automated early-warnings systems can be an essential tool to supplement human intervention. NLP provides the capability to analyze large amounts of text in a single step, however, language in the context of mental health has several distinct features which can be missed by generic language models.

Models that are based on transformers have brought a high level of state of the art to NLP, particularly in the tasks associated with semantic understanding and context modeling. Nevertheless, pretraining on domains has been demonstrated to further augment specialization tasks. MentalBERT is one such model in terms of mental health. MentalBERT was originally introduced by Ji et al., and is pre-trained on large mental healthcare corpora and has been shown to perform better and outperform general-purpose models on downstream mental health detection tasks [1].

Advances in Natural Language Processing (NLP) and transformer-based models have enabled more accurate analysis of human language, especially regarding emotional and psychological states. Pre-trained language models such as BERT have demonstrated strong performance across multiple downstream tasks; however, domain-specific pretraining has been shown to improve results for specialized applications. MentalBERT, introduced by Ji et al., is a transformer model pre-trained on mental health–related corpora and designed to capture linguistic patterns associated with mental disorders and emotional distress [2]. This makes it well-suited for tasks such as detecting suicidal versus non-suicidal content.

Two of the most widely used optimization techniques are Grid Search and Random Search. Grid Search evaluates models using all possible combinations of predefined hyperparameters, offering an exhaustive exploration but often requiring high computational cost as the search space grows [3]. Random Search, on the other hand, samples combinations from probability distributions and has been shown to be significantly more efficient, especially in high-dimensional search spaces where many hyperparameters may have little impact on performance [4].

This study contributes to **Sustainable Development Goal 3 (Good Health and Well-Being)**, **Target 3.4 (Reduce mortality from non-communicable diseases and promote mental health)** by addressing the global challenge of mental health disorders. Suicide and other mental health crises are major public health concerns, and early detection is critical to prevent loss of life and improve well-being. By using AI and Natural Language Processing, this research provides tools that can automatically analyze large amounts of user-generated content from social media, forums, and messaging platforms. These automated systems can help identify individuals showing signs of severe emotional distress or suicidal thoughts, allowing healthcare professionals and support networks to intervene faster. In this way, the study supports reducing premature mortality from mental health conditions and promoting mental health awareness.

Using domain-specific models like MentalBERT improves the ability to detect subtle emotional and linguistic cues in online communication. General-purpose language models may miss these signals, but models fine-tuned on mental health data can capture them more accurately. This improves the reliability of automated detection systems and supports preventive strategies, digital mental health monitoring, and scalable mental health support. By applying advanced AI tools to mental health initiatives, the research demonstrates how technology can help achieve SDG 3 (Good Health and Well-Being), Target 3.4 and promote overall well-being in society.

THE GLOBAL GOALS
HOME
THE 17 GOALS
TAKE ACTION
CAMPAIGNS
NEWS
RESOURCES
PARTNERS
3
GOOD HEALTH AND WELL-BEING
Ensure healthy lives and promote well-being for all at all ages.
Find out more

Figure 1. SDG 3 (Good Health and Well-Being) – AI and NLP for Early Mental Health Detection

II. PROBLEM STATEMENT

Suicide has been ranked as one of the most pressing global health issues, as millions of people show signs of psychological distress, hopelessness, and even suicidal thoughts on social media, forums and support groups online. Communication of mental health persons is usually indirect or subtle, and therefore human moderators or clinicians would find it hard to detect it at an early stage. Consequently, the number of posts of possible self-harm or suicidal intent is enormous and goes unnoticed until it is late. This poses a desperate requirement for automated systems that can read and analyze text related to mental health and recognize people at risk.

Transformer-based models such as MentalBERT have demonstrated encouraging performance on comprehending mental health expressions since they are conditioned with psychological and emotional language related to the domain. Nonetheless, the fine-tuning process is largely reliant on the choice of hyperparameters when using such models. The non-optimal hyperparameters can result in a low sensitivity to detect suicidal or non-suicidal expressions and potentially include cases of false negatives when high-risk individuals are wrongly classified as safe. The consequences of these mistakes can be serious in reality.

Although the proper mental health classification is significant, the current studies do not focus on the influence of various hyperparameter optimization methods on the capacity of MentalBERT to identify suicidal ideation. The existing research is usually based on default settings or manually adjusted parameters, which might be inadequate in case of complexity and sensitivity, as is the case with text related to suicide. Also, there is not much information on which optimization methods are more effective to use exhaustive methods such as Grid Search or stochastic methods such as Random Search in optimizing MentalBERT in the context of suicide risk detection.

III. OBJECTIVE

The objective of this study is to examine how different hyperparameter optimization strategies influence the fine-tuning performance of the MentalBERT model in identifying Normal and Suicidal expressions in social media text. Specifically, the study seeks to.

- Investigate the impact of key hyperparameters Including learning rate, weight decay, and number of training epochs on the stability and effectiveness of MentalBERT during fine-tuning.
- Implement and compare Random Search and Grid Search to determine how each method explores the hyperparameter space and how these differences translate into model performance, computational cost, and training efficiency.
- Evaluate the resulting fine-tuned models using established classification metrics such as accuracy, precision, recall, and F1-score, with particular attention to the model's sensitivity in detecting suicidal intent.
- Identify hyperparameter configurations that consistently support balanced and reliable classification, contributing to improved detection of high-risk mental health cues in online environments.
- Provide practical insights for researchers and practitioners: developing automated mental health assessment systems grounded in transformer-based architectures.

IV. SCOPE OF WORK

This study focuses on evaluating the influence of hyperparameter optimization techniques on the fine-tuning performance of the MentalBERT model for mental health text classification. The scope of the research includes the following activities:

A. Dataset Preparation

The study utilizes the "Sentiment Analysis for Mental Health" dataset from Kaggle. Only the *Normal* and *Suicidal* categories are retained to form a binary classification task. Data cleaning, text preprocessing, tokenization, and label encoding are performed to ensure the dataset is suitable for transformer-based training.

B. Model Configuration and Fine-Tuning

The MentalBERT model is fine-tuned on the processed dataset. Core hyperparameters learning rate, weight decay, and number of epochs are varied across multiple controlled experiments to understand their effect on model behavior.

C. Hyperparameter Optimization Methods

Two optimization strategies, Random Search and Grid Search, are applied. Random Search explores a wide and flexible range of hyperparameter values, while Grid Search evaluates systematically defined combinations. Both methods are compared in terms of model performance and computational requirements.

D. Training and Evaluation

The dataset is divided into training and testing subsets following standard machine learning practices. Training logs, intermediate results, and best-model checkpoints are recorded. Model performance is evaluated using accuracy, precision, recall, and F1-score, with emphasis on detecting suicidal intent.

E. Analysis and Interpretation

Results from all experiments are analyzed to identify trends, optimal configurations, and trade-offs between exhaustive and stochastic search approaches. The study also evaluates how hyperparameter choices impact the model's sensitivity to critical mental health cues.

F. Documentation and Reporting

All procedures, experimental settings, and evaluation results are documented to ensure reproducibility. Insights gained from the analysis are presented to guide future research and practical implementation of transformer-based models for mental health assessment.

V. METHODOLOGY

A. Dataset Description

The dataset used in this study was obtained from Kaggle, titled "Sentiment Analysis for Mental Health". It consists of user-generated tweets that express a variety of emotions and mental states, making it a suitable resource for research on mental health–related sentiment analysis. The dataset was selected because it reflects real-world language used in social media, where individuals often express their thoughts and emotional well-being openly and informally. This kind of data provides a valuable foundation for training models to recognize the linguistic patterns associated with normal and potentially suicidal expressions.

The dataset is structured with three main columns: ID, Statement, and Tweet. Each serves a specific role in organizing and preparing the data for the sentiment classification task.

- ID

  This column serves as the unique identifier for each record in the dataset. It ensures proper indexing and tracking of data instances during preprocessing, model training, and evaluation. Each ID corresponds to one unique tweet entry.

- Status

  The Status column contains the sentiment label assigned to each tweet. The original dataset includes five categories that represent different emotional or psychological conditions expressed in the text. These categories are designed to capture a wide range of mental health–related sentiments. However, for the purpose of this study, only two categories were utilized Normal and Suicidal. This binary classification setup simplifies the model's objective to distinguish between typical emotional expressions and those indicating potential suicidal intent. Tweets labeled under other categories were excluded to maintain a clear focus on detecting critical mental health cues.

- Statement

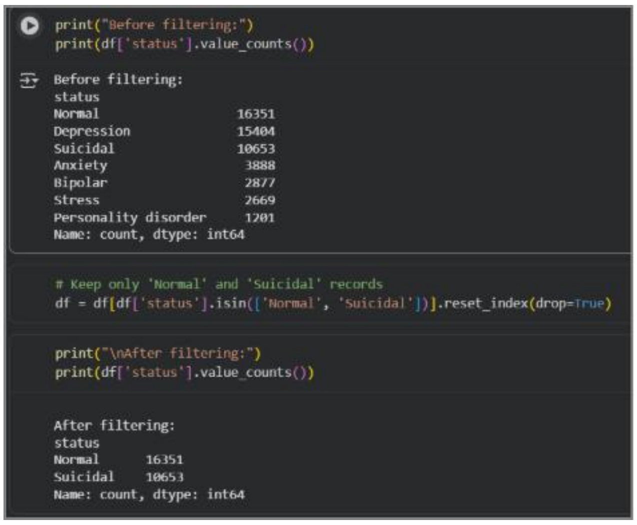
  The Statement column holds the actual text data collected from Twitter. Each entry reflects real social media content where individuals express their daily thoughts, emotions, and mental states. The text often includes informal language, emojis, abbreviations, and grammatical variations, making it representative of authentic online communication. This column serves as the main input for the MentalBERT model after preprocessing and tokenization.

By focusing on the Normal and Suicidal classes, the dataset provides a meaningful foundation for studying how fine-tuned transformer models can differentiate between neutral and high-risk mental health expressions within real-world social media data.

B. Data Preprocessing

Before training the model, several preprocessing steps were performed to prepare the dataset for fine-tuning. These steps ensured that the input data was properly formatted and consistent with the requirements of transformer-based models such as MentalBERT.

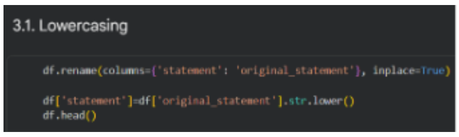
The first step involved filtering the dataset to retain only the relevant classes Normal and Suicidal from the original five sentiment labels. This process simplified the problem into a binary classification task. Once the filtered data was obtained, the Status column was encoded into numerical form using label encoding, where Normal was represented as 0 and Suicidal as 1. This numerical conversion was necessary because machine learning models operate on numeric values rather than textual labels.



```
print("Before filtering:")
print(df['status'].value_counts())

Before filtering:
status
Normal                 16351
Depression             15404
Suicidal               10653
Anxiety                 3888
Bipolar                 2877
Stress                  2669
Personality disorder    1201
Name: count, dtype: int64

# Keep only 'Normal' and 'Suicidal' records
df = df[df['status'].isin(['Normal', 'Suicidal'])].reset_index(drop=True)

print("\nAfter filtering:")
print(df['status'].value_counts())

After filtering:
status
Normal      16351
Suicidal    10653
Name: count, dtype: int64
```

Figure 2. Snippet Code of keeping the 'Normal' and 'Suicidal' in the column of 'status'

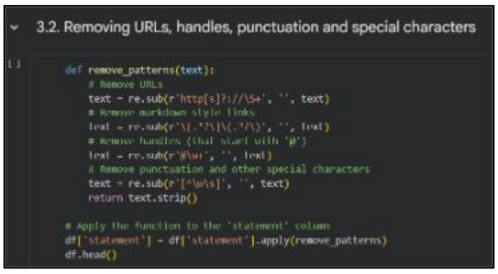
After labeling, text cleaning was applied to the Tweet column to remove unnecessary elements such as special characters, punctuation marks, repeated whitespace, and URLs. This step helped standardize the language and minimize noise in the input text. The cleaned tweets were then converted to lowercase to ensure that the model treated words like "Sad" and "sad" as the same token.



3.1. Lowercasing

```
df.rename(columns={'statement': 'original_statement'}, inplace=True)

df['statement']=df['original_statement'].str.lower()
df.head()
```

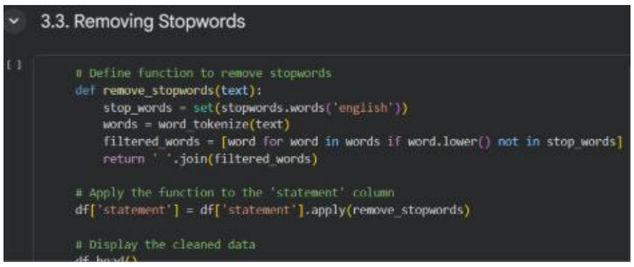
Figure 3. Snippet Code of Lowercasing



3.2. Removing URLs, handles, punctuation and special characters

```
def remove_patterns(text):
    # Remove URLs
    text = re.sub(r'http[s]?://\S+', '', text)
    # Remove markdown-style links
    text = re.sub(r'\[.*?\]\(.*?\)', '', text)
    # Remove handles (that start with '@')
    text = re.sub(r'@\w+', '', text)
    # Remove punctuation and other special characters
    text = re.sub(r'[^\w\s]', '', text)
    return text.strip()

# Apply the function to the 'statement' column
df['statement'] = df['statement'].apply(remove_patterns)
df.head()
```

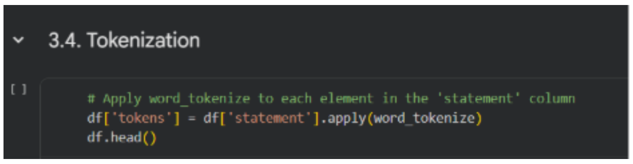
Figure 4. Snippet Code of Removing URL,punctuations and special characters.



3.3. Removing Stopwords

```
# Define function to remove stopwords
def remove_stopwords(text):
    stop_words = set(stopwords.words('english'))
    words = word_tokenize(text)
    filtered_words = [word for word in words if word.lower() not in stop_words]
    return ' '.join(filtered_words)

# Apply the function to the 'statement' column
df['statement'] = df['statement'].apply(remove_stopwords)

# Display the cleaned data
```

Figure 5. Snippet Code of Removing Stopwords

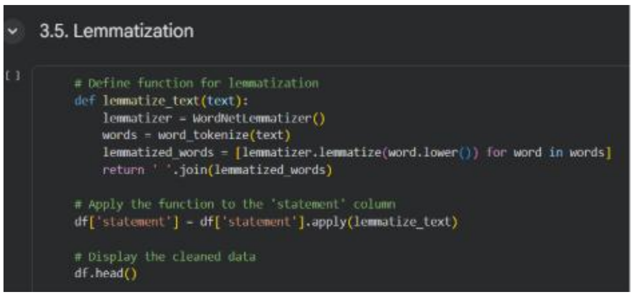


3.4. Tokenization

```
# Apply word_tokenize to each element in the 'statement' column
df['tokens'] = df['statement'].apply(word_tokenize)
df.head()
```

Figure 6. Snippet Code of Removing Tokenization

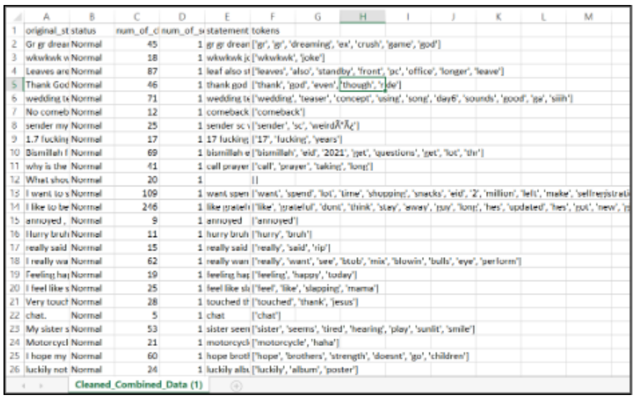
Next, the text data was tokenized using the MentalBERT tokenizer, which splits each tweet into individual subword tokens and converts them into corresponding integer IDs. The tokenizer also added special tokens ([CLS] and [SEP]) required by the BERT architecture and truncated or padded each sequence to a fixed maximum length of 128 tokens. This step ensures uniform input size across all samples, improving training stability.



3.5. Lemmatization

```
# Define function for lemmatization
def lemmatize_text(text):
    lemmatizer = WordNetLemmatizer()
    words = word_tokenize(text)
    lemmatized_words = [lemmatizer.lemmatize(word.lower()) for word in words]
    return ' '.join(lemmatized_words)

# Apply the function to the 'statement' column
df['statement'] = df['statement'].apply(lemmatize_text)

# Display the cleaned data
df.head()
```

Figure 7. Snippet Code of Removing Tokenization



	A	B	C	D	E	F
1	original_st	status	num_of_c	num_of_s	statement	tokens
2	Gr gr drear	Normal	45	1	gr gr drear	['gr', 'gr', 'dreaming', 'ex', 'crush', 'game', 'god']
3	wkwkwk w	Normal	18	1	wkwkwk jc	['wkwkwk', 'joke']
4	Leaves are	Normal	87	1	leaf also st	['leaves', 'also', 'standby', 'front', 'pc', 'office', 'longer', 'leave']
5	Thank God	Normal	46	1	thank god	['thank', 'god', 'even', 'though', 'ride']
6	wedding te	Normal	71	1	wedding te	['wedding', 'teaser', 'concept', 'using', 'song', 'day6', 'sounds', 'good', 'ga', 'siiih']
7	No comeb	Normal	12	1	comeback	['comeback']
8	sender my	Normal	25	1	sender sc v	['sender', 'sc', 'weirdÃ'Ã¿']
9	1.7 fucking	Normal	17	1	17 fucking	['17', 'fucking', 'years']
10	Bismillah I	Normal	69	1	bismillah e	['bismillah', 'eid', '2021', 'get', 'questions', 'get', 'lot', 'thr']
11	why is the	Normal	41	1	call prayer	['call', 'prayer', 'taking', 'long']
12	What shou	Normal	20	1		[]
13	I want to s	Normal	109	1	want spen	['want', 'spend', 'lot', 'time', 'shopping', 'snacks', 'eid', '2', 'million', 'left', 'make', 'sellregistrati
14	I like to be	Normal	246	1	like gratefu	['like', 'grateful', 'dont', 'think', 'stay', 'away', 'guy', 'long', 'hes', 'updated', 'hes', 'put', 'new', 'g
15	annoyed ,	Normal	9	1	annoyed	['annoyed']
16	Hurry bruh	Normal	11	1	hurry bruh	['hurry', 'bruh']
17	really said	Normal	15	1	really said	['really', 'said', 'rip']
18	I really wa	Normal	62	1	really wan	['really', 'want', 'see', 'btob', 'mix', 'blowin', 'bulls', 'eye', 'perform']
19	Feeling haj	Normal	19	1	feeling hap	['feeling', 'happy', 'today']
20	I feel like s	Normal	25	1	feel like sl	['feel', 'like', 'slapping', 'mama']
21	Very touch	Normal	28	1	touched th	['touched', 'thank', 'jesus']
22	chat.	Normal	5	1	chat	['chat']
23	My sister s	Normal	53	1	sister seen	['sister', 'seems', 'tired', 'hearing', 'play', 'sunlit', 'smile']
24	Motorcycl	Normal	21	1	motorcycl	['motorcycle', 'haha']
25	I hope my	Normal	60	1	hope brotl	['hope', 'brothers', 'strength', 'doesnt', 'go', 'children']
26	luckily not	Normal	24	1	luckily albu	['luckily', 'album', 'poster']

Cleaned_Combined_Data (1)

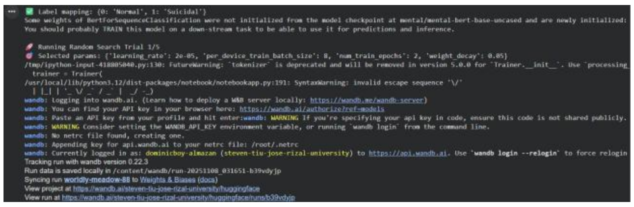
Figure 6. Overview of the Dataset after applying the Pre Processing

After the data cleaning and preprocessing stages, the dataset was refined to include six key columns: original_statement, status, number_of_characters, number_of_sentences, statement, and tokens. The original_statement retains the unprocessed tweet text, serving as a reference for the initial raw data. The status column contains the assigned sentiment label (Normal or Suicidal), which guides the model's classification task. The number_of_characters and number_of_sentences columns provide basic linguistic features that help describe the length and structure of each text sample. The statement column holds the cleaned and standardized version of the tweet used as model input, while the tokens column contains the tokenized representation of the text after applying the MentalBERT tokenizer. Collectively, these columns ensure that both the raw and processed forms of data are preserved, supporting effective training, validation, and interpretability of the fine-tuning process.

C. *Model Setup*

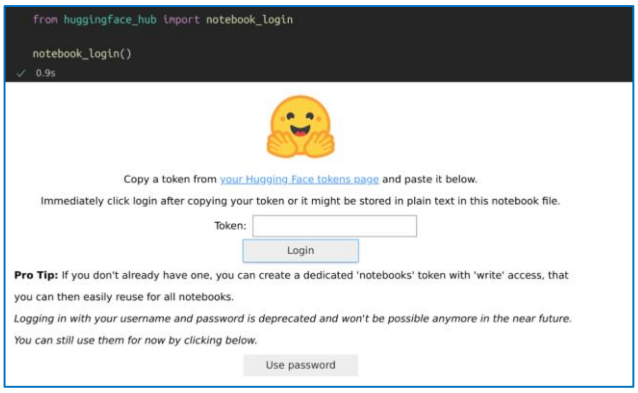
After the dataset was cleaned, encoded, and tokenized, it was divided into training and testing subsets to support effective model evaluation. Approximately 80% of the data was used for training, while the remaining 20% was reserved for testing. Each subset was then converted into a format compatible with the Hugging Face Trainer API, which simplified the overall workflow for training, validation, and performance monitoring. To further enhance efficiency, a DataLoader was used to batch and shuffle the data automatically, optimizing GPU memory utilization and ensuring that each training epoch presented the model with varied input sequences. These steps collectively prepared the dataset for fine-tuning while preserving the contextual and linguistic integrity of each tweet.

The study utilized the MentalBERT model, a domain-specific variant of BERT that has been pretrained on mental health–related text. This model was selected due to its strong capability to capture psychological nuances, emotional cues, and linguistic patterns commonly found in mental health discourse. Accessing MentalBERT requires authentication through the Hugging Face API, which is available at no cost upon creating an account on the Hugging Face platform. After obtaining API access, the model can be securely downloaded and integrated into the training pipeline. The fine-tuning process then adapts MentalBERT's pretrained parameters for the binary classification task of determining whether a tweet expresses a Normal or Suicidal sentiment.



```
✅ Label mapping: {0: 'Normal', 1: 'Suicidal'}
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at mental/mental-bert-base-uncased and are newly initialized:
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

🧪 Running Random Search Trial 1/5
🎯 Selected params: {'learning_rate': 2e-05, 'per_device_train_batch_size': 8, 'num_train_epochs': 2, 'weight_decay': 0.05}
/tmp/ipython-input-418805040.py:130: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_
  trainer = Trainer(
/usr/local/lib/python3.12/dist-packages/notebook/notebookapp.py:191: SyntaxWarning: invalid escape sequence '\/'
  | |_| | '_ \/ _` / _` |  _/ -_)
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize?ref=models
wandb: Paste an API key from your profile and hit enter:wandb: WARNING If you're specifying your api key in code, ensure this code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running `wandb login` from the command line.
wandb: No netrc file found, creating one.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
wandb: Currently logged in as: dominicboy-almazan (steven-tiu-jose-rizal-university) to https://api.wandb.ai. Use `wandb login --relogin` to force relogin
Tracking run with wandb version 0.22.3
Run data is saved locally in /content/wandb/run-20251108_031651-b39vdyjp
Syncing run worldly-meadow-88 to Weights & Biases (docs)
View project at https://wandb.ai/steven-tiu-jose-rizal-university/huggingface
View run at https://wandb.ai/steven-tiu-jose-rizal-university/huggingface/runs/b39vdyjp
```

Figure 8 . Snippet showing the authentication and initialization process of Weights & Biases (W&B) used for experiment logging during MentalBERT fine-tuning.



```
from huggingface_hub import notebook_login

notebook_login()
✓ 0.9s
```

Copy a token from your Hugging Face tokens page and paste it below.

Immediately click login after copying your token or it might be stored in plain text in this notebook file.

Token:

Pro Tip: If you don't already have one, you can create a dedicated 'notebooks' token with 'write' access, that you can then easily reuse for all notebooks.

Logging in with your username and password is deprecated and won't be possible anymore in the near future.

You can still use them for now by clicking below.

Figure 9 . Snippet: Hugging Face API Authentication for Accessing and Loading MentalBERT

.

Several hyperparameters were adjusted across different experiments to analyze their influence on model performance. These included:

- **Learning Rate** - Varied between 2e-5 and 5e-5 to evaluate the trade-off between convergence speed and model stability.
- **Weight Decay** - Adjusted between 0.01 and 0.5 to prevent overfitting by controlling the magnitude of weight updates.
- **Number of Epochs** - Tested between 2 and 5 epochs to observe how extended training affected accuracy and generalization
- **Save Strategy** – Switched between saving every epoch and every set of steps to see which method produces more stable training results.
- **Logging Steps** – Tested different logging intervals (50 to 400 steps) to find a balance between frequent progress tracking and efficient training.
- **Learning Rate** – Tried values from 5e-06 to 5e-05 to observe how fast or slow the model should learn for the best performance.
- **per_device_train_batch_size -** Tested different training batch sizes to see how many samples per step affect training speed, stability, and GPU memory.
- **per_device_eval_batch_size -** Tried different evaluation batch sizes to balance evaluation speed and memory use.
- **fp16 (Mixed-Precision Training) -** Turned on/off half-precision training to check if it speeds up training and saves memory without affecting accuracy.

During training, evaluation was conducted at the end of each epoch using the F1-score as the main metric for selecting the best-performing model. Other metrics such as accuracy, precision, and recall were also computed to assess classification of balance and consistency.

All experiment runs were logged and tracked through Excel for performance visualization and comparison. This monitoring helped identify trends such as overfitting, optimal learning rate regions, and the most effective regularization parameters.

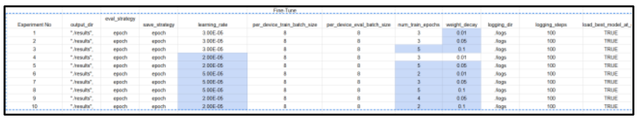
The trained models were then saved locally for further evaluation and testing on unseen text samples. The combination of systematic hyperparameter tuning and domain-specific fine-tuning enabled the MentalBERT model to effectively recognize subtle emotional differences between Normal and Suicidal statements, reflecting strong potential for practical applications in automated mental health assessment systems.

D. *Fine-Tuning Process*

i. *Almazan Fine Tuning Experimentation*

The finetuning process involved running a series of controlled experiments to understand how different hyperparameters influence the behavior and training performance of the model. Each experiment used the same dataset, evaluation setup, and output structure, ensuring that only the hyperparameters being tested changed between runs. This allowed a clearer view of how training conditions shape the model's learning process. All experiments were executed using a consistent logging setup, evaluation strategy, and model-saving mechanism, ensuring that every run was monitored in the same way.

Across the ten experiments, variations were mainly applied to three key hyperparameters: learning rate, weight decay, and number of epochs. The learning rate was adjusted across low to moderately high values to see how fast or slow the model should update its weights during training. Weight decay was also changed to observe how different levels of regularization influence the model's ability to avoid overfitting. Finally, the number of epochs was expanded or reduced depending on the experiment, allowing exploration of how longer or shorter training durations affect the model during the finetuning phase.



Fine Tune											
Experiment No	output_dir	eval_strategy	save_strategy	learning_rate	per_device_train_batch_size	per_device_eval_batch_size	num_train_epochs	weight_decay	logging_dir	logging_steps	load_best_model_at_
1	"./results",	epoch	epoch	3.00E-05	8	8	3	0.01	./logs	100	TRUE
2	"./results",	epoch	epoch	3.00E-05	8	8	3	0.05	./logs	100	TRUE
3	"./results",	epoch	epoch	3.00E-05	8	8	5	0.1	./logs	100	TRUE
4	"./results",	epoch	epoch	2.00E-05	8	8	3	0.01	./logs	100	TRUE
5	"./results",	epoch	epoch	2.00E-05	8	8	5	0.05	./logs	100	TRUE
6	"./results",	epoch	epoch	5.00E-05	8	8	2	0.01	./logs	100	TRUE
7	"./results",	epoch	epoch	5.00E-05	8	8	3	0.05	./logs	100	TRUE
8	"./results",	epoch	epoch	5.00E-05	8	8	5	0.1	./logs	100	TRUE
9	"./results",	epoch	epoch	2.00E-05	8	8	4	0.05	./logs	100	TRUE
10	"./results",	epoch	epoch	2.00E-05	8	8	2	0.1	./logs	100	TRUE

Figure 10. Snippet of The Experimentation Logs

Each experiment followed a structured approach where the batch sizes, evaluation strategy, and training environment remained constant. This consistency ensured that the only factors influencing the model's training behavior were the hyperparameters being tested. For example, some experiments used shorter training durations but higher learning rates, while others used longer training with stronger regularization. Every setup was designed intentionally to test a specific combination and observe how the model adapts under different levels of training pressure.

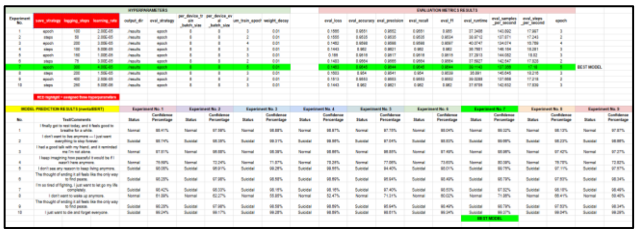
Throughout all runs, standardized logging steps and automatic tracking of the best-performing model were enabled. This allowed the researcher to capture training patterns such as loss of behavior, training stability, and runtime efficiency. By exploring multiple combinations of learning rate, epoch count, and weight decay, the experimentation phase created a clear picture of how tuning these parameters impacts the model's learning environment. This systematic approach provides a solid foundation for understanding which hyperparameter setups are more suitable for stable, efficient, and well-balanced finetuning.

ii. *Pajanustan S. Fine Tuning Experimentation*

Across the ten fine-tuning experiments, I changed several training settings to find which setup would give the best performance for MentalBERT. The assigned hyperparameters were the save strategy, logging steps, and learning rate. The save strategy was switched between saving every epoch and saving after a fixed number of steps to see which approach helped keep the training more stable. The logging steps varied from 50 to 400 to check how often training progress should be recorded.

The learning rate was tested using different values, ranging from very small to moderately high, to observe how fast or slow the model should learn during fine-tuning. The number of epochs was also adjusted between 2 and 5 to understand whether shorter or longer training would improve the model's performance.

After running all the experiments, each configuration was evaluated using accuracy, precision, recall, and F1-score. Based on the results, Experiment 7 achieved the best performance overall.



HYPERPARAMETERS									EVALUATION METRICS RESULTS									
Experiment No.	save_strategy	logging_steps	learning_rate	output_dir	eval_strategy	per_device_train_batch_size	per_device_eval_batch_size	um_train_epoch	weight_decay	eval_loss	eval_accuracy	eval_precision	eval_recall	eval_f1	eval_runtime	eval_samples_per_second	eval_steps_per_second	epoch
1	epoch	100	2.00E-05	./results	epoch	8	8	3	0.01	0.1586	0.9651	0.9652	0.9651	0.965	37.3486	143.692	17.967	3
2	steps	50	2.00E-05	./results	epoch	8	8	2	0.01	0.1555	0.9635	0.9635	0.9635	0.9634	38.9712	137.671	17.243	2
3	epoch	200	3.00E-05	./results	epoch	8	8	4	0.01	0.1482	0.9598	0.9598	0.9598	0.9597	40.0747	134.074	16.769	4
4	steps	100	5.00E-06	./results	epoch	8	8	3	0.01	0.1443	0.962	0.9621	0.962	0.962	36.7801	146.104	18.281	3
5	epoch	150	1.00E-05	./results	epoch	8	8	3	0.01	0.166	0.9616	0.9617	0.9616	0.9615	37.2913	144.082	18.02	3
6	steps	75	3.00E-05	./results	epoch	8	8	2	0.01	0.1483	0.9664	0.9665	0.9664	0.9664	37.5927	142.947	17.828	2
7	epoch	300	4.00E-05	./results	epoch	8	8	5	0.01	0.1463	0.9645	0.9644	0.9645	0.9644	39.1143	137.368	17.18	5
8	steps	200	1.50E-05	./results	epoch	8	8	3	0.01	0.1503	0.954	0.9541	0.954	0.9539	35.891	145.645	18.216	3
9	epoch	400	2.50E-05	./results	epoch	8	8	2	0.01	0.1513	0.9653	0.9653	0.9653	0.9652	39.3288	137.668	17.218	2
10	steps	250	5.00E-06	./results	epoch	8	8	3	0.01	0.1443	0.962	0.9621	0.962	0.962	37.6709	142.632	17.830	3

BEST MODEL

RED highlight = assigned three hyperparameters

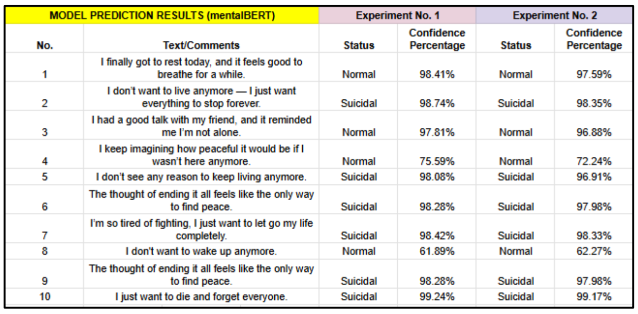
MODEL PREDICTION RESULTS (mentalBERT)		Experiment No. 1		Experiment No. 2		Experiment No. 3		Experiment No. 4		Experiment No. 5		Experiment No. 6		Experiment No. 7		Experiment No. 8		Experiment No. 9	
No.	Text/Comments	Status	Confidence Percentage	Status	Confidence Percentage	Status	Confidence Percentage	Status	Confidence Percentage	Status	Confidence Percentage	Status	Confidence Percentage	Status	Confidence Percentage	Status	Confidence Percentage	Status	Confidence Percentage
1	I finally got to rest today, and it feels good to breathe for a while.	Normal	98.41%	Normal	97.59%	Normal	98.88%	Normal	98.87%	Normal	97.75%	Normal	98.04%	Normal	99.32%	Normal	98.13%	Normal	97.87%
2	I don't want to live anymore — I just want everything to stop forever.	Suicidal	98.74%	Suicidal	98.35%	Suicidal	99.31%	Suicidal	98.98%	Suicidal	97.04%	Suicidal	98.83%	Suicidal	99.98%	Suicidal	98.23%	Suicidal	98.88%
3	I had a good talk with my friend, and it reminded me I'm not alone.	Normal	97.81%	Normal	96.88%	Normal	98.38%	Normal	98.86%	Normal	98.85%	Normal	97.49%	Normal	98.98%	Normal	97.42%	Normal	97.27%
4	I keep imagining how peaceful it would be if I wasn't here anymore.	Normal	75.59%	Normal	72.24%	Normal	71.07%	Normal	70.26%	Normal	77.06%	Normal	73.63%	Normal	80.38%	Normal	78.78%	Normal	72.82%
5	I don't see any reason to keep living anymore.	Suicidal	98.08%	Suicidal	96.91%	Suicidal	99.28%	Suicidal	98.95%	Suicidal	94.40%	Suicidal	98.01%	Suicidal	99.75%	Suicidal	97.11%	Suicidal	97.67%
6	The thought of ending it all feels like the only way to find peace.	Suicidal	98.28%	Suicidal	97.98%	Suicidal	98.58%	Suicidal	98.89%	Suicidal	95.64%	Suicidal	98.49%	Suicidal	98.79%	Suicidal	97.93%	Suicidal	98.34%
7	I'm so tired of fighting, I just want to let go my life completely.	Suicidal	98.42%	Suicidal	98.33%	Suicidal	98.18%	Suicidal	98.16%	Suicidal	97.40%	Suicidal	98.53%	Suicidal	97.52%	Suicidal	98.18%	Suicidal	98.49%
8	I don't want to wake up anymore.	Normal	61.89%	Normal	62.27%	Normal	55.85%	Normal	52.47%	Normal	71.31%	Normal	60.02%	Normal	71.08%	Normal	68.41%	Normal	68.40%
9	The thought of ending it all feels like the only way to find peace.	Suicidal	98.28%	Suicidal	97.98%	Suicidal	98.58%	Suicidal	98.89%	Suicidal	95.64%	Suicidal	98.49%	Suicidal	98.79%	Suicidal	97.93%	Suicidal	98.34%
10	I just want to die and forget everyone.	Suicidal	99.24%	Suicidal	99.17%	Suicidal	99.28%	Suicidal	99.89%	Suicidal	99.61%	Suicidal	99.34%	Suicidal	99.37%	Suicidal	99.04%	Suicidal	99.39%

BEST MODEL

Figure 11. Snippet of The Experimentation Logs

The prediction results show how each of the ten fine-tuned models labeled different mental-health related texts. For clear positive or calm messages, all models correctly predicted "Normal" with very high confidence. For the more direct and obvious suicidal statements, every model also predicted "Suicidal" with strong confidence, usually above 98 percent. This means the models were good at identifying texts that were clearly safe or clearly harmful.

The biggest difference appeared in messages that were unclear or indirect. For example, the sentence "I don't want to wake up anymore" was marked as Normal by all models, and the confidence scores were much lower compared to the other samples. This shows that the models had a harder time understanding messages that were not explicit. Overall, Experiment 7 gave the most consistent and confident results, which matches its strong performance in the evaluation metrics.

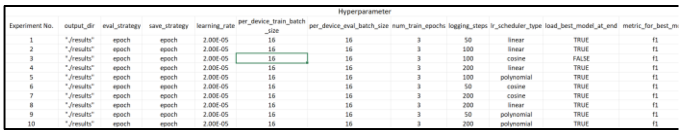


MODEL PREDICTION RESULTS (mentalBERT)		Experiment No. 1		Experiment No. 2	
No.	Text/Comments	Status	Confidence Percentage	Status	Confidence Percentage
1	I finally got to rest today, and it feels good to breathe for a while.	Normal	98.41%	Normal	97.59%
2	I don't want to live anymore — I just want everything to stop forever.	Suicidal	98.74%	Suicidal	98.35%
3	I had a good talk with my friend, and it reminded me I'm not alone.	Normal	97.81%	Normal	96.88%
4	I keep imagining how peaceful it would be if I wasn't here anymore.	Normal	75.59%	Normal	72.24%
5	I don't see any reason to keep living anymore.	Suicidal	98.08%	Suicidal	96.91%
6	The thought of ending it all feels like the only way to find peace.	Suicidal	98.28%	Suicidal	97.98%
7	I'm so tired of fighting, I just want to let go my life completely.	Suicidal	98.42%	Suicidal	98.33%
8	I don't want to wake up anymore.	Normal	61.89%	Normal	62.27%
9	The thought of ending it all feels like the only way to find peace.	Suicidal	98.28%	Suicidal	97.98%
10	I just want to die and forget everyone.	Suicidal	99.24%	Suicidal	99.17%

Figure 12. Snippet of Model Prediction Result

iii. *Tejada Fine Tuning Experimentation*

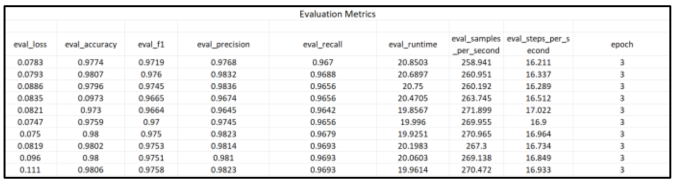
In every experiment, I would optimize important parameters that included the type of learning rate scheduler and the logging steps because these parameters were the ones that most likely affected the training stability and performance. The learning rate scheduler is used to control the rate at which the learning rate varies with time, whereas logging steps influence the frequency with which training feedback is written. I tried to observe changes in the parameters with a view to establishing trends which would aid in better model generalization.



						Hyperparameter					
Experiment No.	output_dir	eval_strategy	save_strategy	learning_rate	per_device_train_batch_size	per_device_eval_batch_size	num_train_epochs	logging_steps	lr_scheduler_type	load_best_model_at_end	metric_for_best_m
1	"./results"	epoch	epoch	2.00E-05	16	16	3	50	linear	TRUE	f1
2	"./results"	epoch	epoch	2.00E-05	16	16	3	100	linear	TRUE	f1
3	"./results"	epoch	epoch	2.00E-05	16	16	3	100	cosine	FALSE	f1
4	"./results"	epoch	epoch	2.00E-05	16	16	3	200	linear	TRUE	f1
5	"./results"	epoch	epoch	2.00E-05	16	16	3	100	polynomial	TRUE	f1
6	"./results"	epoch	epoch	2.00E-05	16	16	3	50	cosine	TRUE	f1
7	"./results"	epoch	epoch	2.00E-05	16	16	3	200	cosine	TRUE	f1
8	"./results"	epoch	epoch	2.00E-05	16	16	3	200	linear	TRUE	f1
9	"./results"	epoch	epoch	2.00E-05	16	16	3	50	polynomial	TRUE	f1
10	"./results"	epoch	epoch	2.00E-05	16	16	3	200	polynomial	TRUE	f1

Figure 13. Snippet of The Experimentation Logs

Once all experiments were done, I compared the evaluation metrics that were obtained such as eval_loss, accuracy, F1 score, precision, recall, runtime, and samples per second. Experiment 2, with the highest overall performance where F1 score is **0.976**, and accuracy is high **0.9807** and constant values of precision-recall, was the best of the ten trials. The particular trial was linear learning rate scheduler and logging steps of 100 that seemed to provide a more stable learning curve than other runs that used the cosine and the polynomial learning rate schedulers.



				Evaluation Metrics				
eval_loss	eval_accuracy	eval_f1	eval_precision	eval_recall	eval_runtime	eval_samples_per_second	eval_steps_per_second	epoch
0.0783	0.9774	0.9719	0.9768	0.967	20.8503	258.941	16.211	3
0.0793	0.9807	0.976	0.9832	0.9688	20.6897	260.951	16.337	3
0.0886	0.9796	0.9745	0.9836	0.9656	20.75	260.192	16.289	3
0.0835	0.0973	0.9665	0.9674	0.9656	20.4705	263.745	16.512	3
0.0821	0.973	0.9664	0.9645	0.9642	19.8567	271.899	17.022	3
0.0747	0.9759	0.97	0.9745	0.9656	19.996	269.955	16.9	3
0.075	0.98	0.975	0.9823	0.9679	19.9251	270.965	16.964	3
0.0819	0.9802	0.9753	0.9814	0.9693	20.1983	267.3	16.734	3
0.096	0.98	0.9751	0.981	0.9693	20.0603	269.138	16.849	3
0.111	0.9806	0.9758	0.9823	0.9693	19.9614	270.472	16.933	3

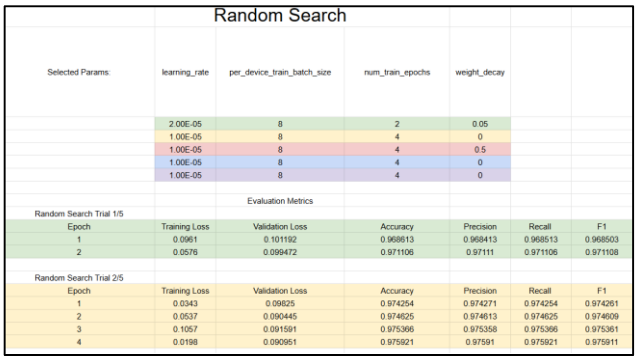
Figure 14. Snippet of Evaluation Metrics

According to these findings, I chose the hyperparameters of Experiment 2 as the best setting for my fine-tuned model. The experimentation process enabled me to critically assess which training dynamics were most favorable to the model to classify text related to mental health with the highest accuracy and efficiency.

E. *Random Search and Grid Search*

i. *Almazan Random Search and Grid Search Experimentation*

The hyperparameter search phase involved two approaches Random Search and Grid Search to understand how different training settings influence model behavior. In Random Search, the goal was broad exploration, so the hyperparameters were sampled from wider and more flexible ranges. The learning rate values included small and moderate settings such as **1e-5** and **2e-5**, which allowed the model to learn at different speeds. The number of epochs varied between **2 and 4**, giving the experiment a mix of short and extended training durations. Weight decay had the broadest range, from **0.0 up to 0.5**, introducing conditions with no regularization up to very strong regularization. Batch size remained constant at **8** to maintain training stability while the other parameters changed freely. This randomness helped capture combinations that would not normally be tested in a structured setup, allowing the search to uncover unexpected but meaningful configurations.

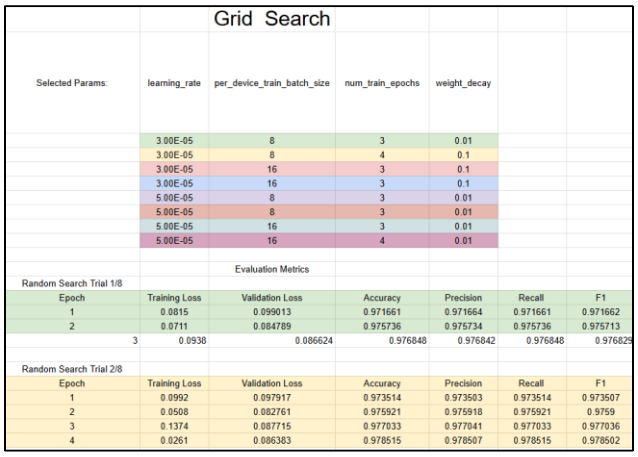


Random Search

Selected Params:	learning_rate	per_device_train_batch_size	num_train_epochs	weight_decay		
	2.00E-05	8	2	0.05		
	1.00E-05	8	4	0		
	1.00E-05	8	4	0.5		
	1.00E-05	8	4	0		
	1.00E-05	8	4	0		
		Evaluation Metrics				
Random Search Trial 1/5						
Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.0961	0.101192	0.968613	0.968413	0.968513	0.968503
2	0.0576	0.099472	0.971106	0.97111	0.971106	0.971108
Random Search Trial 2/5						
Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.0343	0.09825	0.974254	0.974271	0.974254	0.974261
2	0.0537	0.090445	0.974625	0.974613	0.974625	0.974609
3	0.1057	0.091591	0.975366	0.975358	0.975366	0.975361
4	0.0198	0.090951	0.975921	0.97591	0.975921	0.975911

Figure 15. The Experimentation of logs of Random Search

Each Random Search trial produced a unique mix of these hyperparameters, resulting in diverse training conditions. Some trials paired extremely low learning rates with longer training durations, while others combined short epochs with high weight decay values. This approach allowed the experiment to observe a wide range of training behaviors without being limited to a predictable pattern. The purpose of using broad and flexible ranges was to maximize coverage of the hyperparameter space, especially in areas that a researcher might not naturally consider. By doing so, Random Search offered the chance to identify strong configurations that could emerge from unusual combinations of learning rate, epochs, and weight decay.



Grid Search

Selected Params:	learning_rate	per_device_train_batch_size	num_train_epochs	weight_decay		
	3.00E-05	8	3	0.01		
	3.00E-05	8	4	0.1		
	3.00E-05	16	3	0.1		
	3.00E-05	16	3	0.1		
	5.00E-05	8	3	0.01		
	5.00E-05	8	3	0.01		
	5.00E-05	16	3	0.01		
	5.00E-05	16	4	0.01		
		Evaluation Metrics				
Random Search Trial 1/8						
Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.0815	0.099013	0.971661	0.971664	0.971661	0.971662
2	0.0711	0.084789	0.975736	0.975734	0.975736	0.975713
3	0.0938	0.086624	0.976848	0.976842	0.976848	0.976829
Random Search Trial 2/8						
Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.0992	0.097917	0.973514	0.973503	0.973514	0.973507
2	0.0508	0.082761	0.975921	0.975918	0.975921	0.9759
3	0.1374	0.087715	0.977033	0.977041	0.977033	0.977036
4	0.0261	0.086383	0.978515	0.978507	0.978515	0.978502

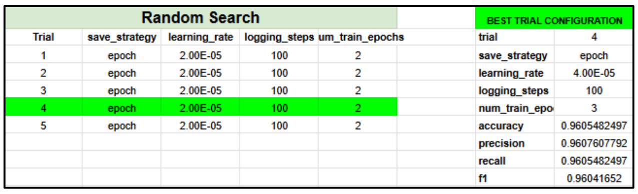
Figure 16. The Experimentation of logs of Grid Search

In contrast, the Grid Search method followed a much more organized structure. The hyperparameters were restricted to a fixed list of values: learning rates such as **3e-5**, **5e-5**, and **2e-5**; batch sizes of **8 and 16**; epoch counts of **3 and 4**; and weight decay values commonly set at **0.01 or 0.1**. Every possible combination of these values was tested systematically, ensuring that no pairing was skipped. This structured approach allowed direct comparison between controlled variations, such as how the model behaved when only the batch size changed while the learning rate stayed constant. Together, Random Search and Grid Search provided two complementary perspectives, one wide and exploratory, the other structured and exhaustive, giving the researcher a complete view of how different hyperparameters influence the finetuning process.

ii. *Pajanustan S. Random Search and Grid Search Experimentation*

In the Random Search experiments, four main hyperparameters were tested to improve the performance of MentalBERT. The save strategy determined whether the model checkpoints were saved after each epoch or after a fixed number of steps. The learning rate controlled how quickly the model updated its weights during training. The logging steps specified how often the training progress and metrics were recorded. Finally, the number of training epochs set how many times the model would go through the entire training dataset. By varying these parameters, the experiment aimed to find the combination that gave the best balance of accuracy and efficiency.

After running five trials, **Trial 4** was identified as the best configuration. In this trial, the model used epoch-based saving, a learning rate of 4e-05, logging every 100 steps, and 3 training epochs. This setup achieved the highest evaluation metrics, with an accuracy of 0.9605, precision of 0.9608, recall of 0.9605, and F1-score of 0.9604. These results show that Random Search was effective in finding a strong-performing combination without testing every possible setting, saving time while maintaining good model performance.

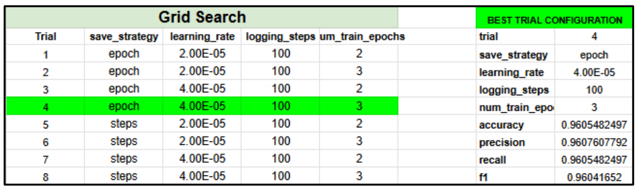


Random Search						BEST TRIAL CONFIGURATION	
Trial	save_strategy	learning_rate	logging_steps	um_train_epochs		trial	4
1	epoch	2.00E-05	100	2		save_strategy	epoch
2	epoch	2.00E-05	100	2		learning_rate	4.00E-05
3	epoch	2.00E-05	100	2		logging_steps	100
4	epoch	2.00E-05	100	2		num_train_epo	3
5	epoch	2.00E-05	100	2		accuracy	0.9605482497
						precision	0.9607607792
						recall	0.9605482497
						f1	0.96041652

Figure 17. The Experimentation of logs of Random Search

In the Grid Search experiments, the same set of core hyperparameters was tested, but this time every possible combination within the predefined values was explored. The save strategy was switched between epoch-based and step-based saving. The learning rate included two values, 2e-05 and 4e-05, to observe how different learning speeds affect performance. The logging steps were kept constant at 100 to ensure consistent monitoring across all trials. The number of training epochs was tested at both 2 and 3 to compare shorter and slightly longer training durations. With Grid Search, each combination of these hyperparameters was tested to systematically identify the best setup.

From the eight trials conducted, **Trial 4** produced the strongest overall performance. This configuration used epoch-based saving, a learning rate of 4e-05, logging every 100 steps, and 3 training epochs. It achieved the highest evaluation metrics, with an accuracy of 0.9605, precision of 0.9608, recall of 0.9605, and an F1-score of 0.9604. These results show that Grid Search successfully identified an optimal combination by exhaustively testing all preset options, confirming the value of structured exploration in fine-tuning MentalBERT.



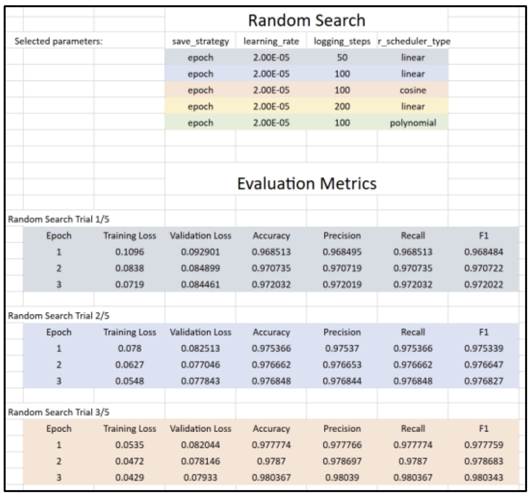
Grid Search						BEST TRIAL CONFIGURATION	
Trial	save_strategy	learning_rate	logging_steps	um_train_epochs		trial	4
1	epoch	2.00E-05	100	2		save_strategy	epoch
2	epoch	2.00E-05	100	3		learning_rate	4.00E-05
3	epoch	4.00E-05	100	2		logging_steps	100
4	epoch	4.00E-05	100	3		num_train_epo	3
5	steps	2.00E-05	100	2		accuracy	0.9605482497
6	steps	2.00E-05	100	3		precision	0.9607607792
7	steps	4.00E-05	100	2		recall	0.9605482497
8	steps	4.00E-05	100	3		f1	0.96041652

Figure 18. The Experimentation of logs of Grid Search

iii. *Tejada Random Search and Grid Search Experimentation*

To further refine the performance of my model, I did a Random Search hyperparameter experiment which lets the training process more heavily sample the space of potential parameter combinations without necessarily considering all combinations. My aim was to find a high-performing set of hyperparameters effectively, in particular, those that affect training stability and convergence, e.g., the logging steps and the type of learning rate scheduler.

I also predefined the set of possible hyperparameter values and left the Random Search procedure to choose combinations randomly in five independent experiments to do this experiment. The model was trained in three epochs in every trial, and I observed significant performance measures, including training loss, validation loss, accuracy, precision, recall, and F1 score. Contrary to the Grid Search that considers all the possible combinations in the Random Search, Random Search samples configurations more flexibly, which is another effective algorithm when hyperparameters have a few that greatly affect the performance.



Random Search

Selected parameters:	save_strategy	learning_rate	logging_steps	r_scheduler_type
	epoch	2.00E-05	50	linear
	epoch	2.00E-05	100	linear
	epoch	2.00E-05	100	cosine
	epoch	2.00E-05	200	linear
	epoch	2.00E-05	100	polynomial

Evaluation Metrics

Random Search Trial 1/5

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.1096	0.092901	0.968513	0.968495	0.968513	0.968484
2	0.0838	0.084899	0.970735	0.970719	0.970735	0.970722
3	0.0719	0.084461	0.972032	0.972019	0.972032	0.972022

Random Search Trial 2/5

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.078	0.082513	0.975366	0.97537	0.975366	0.975339
2	0.0627	0.077046	0.976662	0.976653	0.976662	0.976647
3	0.0548	0.077843	0.976848	0.976844	0.976848	0.976827

Random Search Trial 3/5

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.0535	0.082044	0.977774	0.977766	0.977774	0.977759
2	0.0472	0.078146	0.9787	0.978697	0.9787	0.978683
3	0.0429	0.07933	0.980367	0.98039	0.980367	0.980343

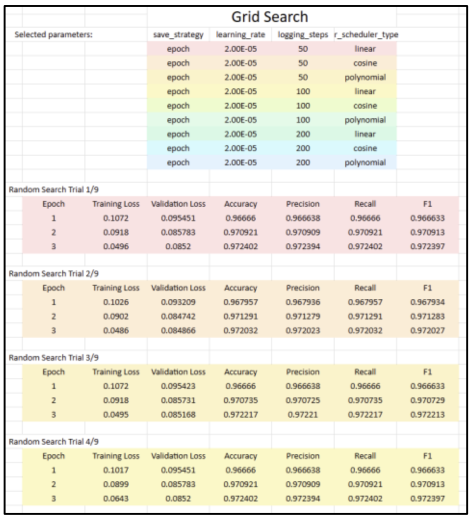
Figure 19. The Experimentation of logs of Random Search

In all five trials I witnessed a definite disparity in learning behavior and measures of evaluation. Despite the fact that all runs yielded comparatively high accuracy and F1 scores, there was always one configuration that performed well. Random Search Trial 3 had the highest level of performance with an F1 score of **0.980343,** which is the highest of all trials. The linear learning rate scheduler and logging steps of 200 were used in this experiment, and it seemed to offer a more gradual and less noisy pattern of gradient updates. The accuracy **(0.980367)**, the precision **(0.980309)**, and the recall **(0.980367)** were also fair and strong and supported the reliability of this set-up.

On the basis of these results, I chose Trial 3 to be the best hyperparameter configuration in the Random Search experiment. This was a necessary procedure since it enabled the model to experiment with the possibility of configurations that otherwise would not have been intuitively selected, yet high performance would have been achieved. Finally, the use of Random Search further allowed me to locate an effective combination in the optimal way, and it led to the enhanced generalization of the model.

For the Grid Search, the hyperparameter tuning phase of my model, I used a succession of trial and error with the help of both Grid Search and Random Search to establish the combination of parameters that would produce the optimal result. I began by configuring a Grid Search, in which I provided some possible values of key hyperparameters (type of learning rate scheduler (linear, cosine, and polynomial) and number of logging steps (50, 100, and 200) manually). To ensure consistency in the trials, I used the same learning rate of 2e-5 and the save strategy of epoch. I had three epochs of training the model and noted its training loss, validation loss, its accuracy, precision, recall, and F1-score per combination. This enabled me to note the effect of every parameter setting on the learning behavior of the model.

Once all the Grid Searchs were finished, I followed with a Random Search where I get to search more area of parameter combinations in a more flexible manner. During this stage, I performed nine randomized trials, which were also taking place over three epochs. The Random Search assisted me with determining the ability of randomly chosen settings to be more effective than the structured combinations of the Grid Search. In the process, I compared the performance metrics of each trial in order to identify the most stable and highest results configuration.



Grid Search

Selected parameters:	save_strategy	learning_rate	logging_steps	r_scheduler_type
	epoch	2.00E-05	50	linear
	epoch	2.00E-05	50	cosine
	epoch	2.00E-05	50	polynomial
	epoch	2.00E-05	100	linear
	epoch	2.00E-05	100	cosine
	epoch	2.00E-05	100	polynomial
	epoch	2.00E-05	200	linear
	epoch	2.00E-05	200	cosine
	epoch	2.00E-05	200	polynomial

Random Search Trial 1/9

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.1072	0.095451	0.96666	0.966638	0.96666	0.966633
2	0.0918	0.085783	0.970921	0.970909	0.970921	0.970913
3	0.0496	0.0852	0.972402	0.972394	0.972402	0.972397

Random Search Trial 2/9

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.1026	0.093209	0.967957	0.967936	0.967957	0.967934
2	0.0902	0.084742	0.971291	0.971279	0.971291	0.971283
3	0.0486	0.084866	0.972032	0.972023	0.972032	0.972027

Random Search Trial 3/9

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.1072	0.095423	0.96666	0.966638	0.96666	0.966633
2	0.0918	0.085731	0.970735	0.970725	0.970735	0.970729
3	0.0495	0.085168	0.972217	0.97221	0.972217	0.972213

Random Search Trial 4/9

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.1017	0.095451	0.96666	0.966638	0.96666	0.966633
2	0.0899	0.085783	0.970921	0.970909	0.970921	0.970913
3	0.0643	0.0852	0.972402	0.972394	0.972402	0.972397

Figure 20. The Experimentation of logs of Grid Search

These experiments allowed me to discover that the most successful setup was Random Search Trial 1 that employed 50 logging steps and a linear learning rate scheduler. This configuration recorded the best accuracy, precision, recall, and F1-score in all the trials. These experimentations enabled me to test, compare, and validate various hyperparameter settings in a systematic manner, thus enabling me to choose the most efficient hyperparameter configuration to use in my final model.

VI. RESULTS

The fine-tuning of the MentalBERT model produced noticeable improvements in its ability to classify social media text as either Normal or Suicidal. Across the training runs, the model demonstrated progressive enhancements in learning stability, loss reduction, and overall classification performance as it adapted to the characteristics of the dataset. Evaluation metrics such as accuracy, precision, recall, and F1-score reflect how well the model internalized the linguistic cues present in mental health–related expressions. The following section presents the quantitative outcomes of the fine-tuning process, highlighting how the model's performance evolved over the training epochs and how effectively it distinguished between the two target classes.

The hyperparameter optimization experiments, using both Grid Search and Random Search, were conducted independently by each research member. These experiments aimed to identify the best configurations for fine-tuning the MentalBERT model to classify social media text as Normal or Suicidal. Across the trials, members explored different learning rates, weight decay values, batch sizes, and epoch counts. Evaluation metrics such as accuracy, precision, recall, and F1-score were used to compare the outcomes, highlighting how systematic and stochastic approaches affected the model's performance for each member. The following table summarizes the best results obtained by each researcher using both optimization methods.

Table I. Best Fine-Tuning Results from Each Research Member's Hyperparameter Experiments:

Researcher	**Experiment No.**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Eval Loss**
Almazan	8	0.98	0.98	0.9827	0.983	0.0796
Pajanustan S.	7	0.9645	0.9644	0.9645	0.9644	0.1463
Tejada	7	0.98	0.9823	0.9679	0.975	0.075

Among the three models, Tejada performed the best overall. It had the lowest evaluation loss at 0.075, which indicates that it made the fewest errors when predicting the evaluation data. This means the model is more confident and precise in its predictions compared to the others. Tejada also achieved a high accuracy of 0.98, showing that it correctly classified the majority of the texts, and a strong F1-score of 0.975, which means it balanced both precision and recall well.

In comparison, the Almazan model also had high accuracy and F1-score, but its evaluation loss was slightly higher (0.0796), which suggests it made more mistakes than Tejada. The Pajanustan S. model had lower accuracy and F1-score and the highest evaluation loss (0.1463), making it the least reliable of the three.

Overall, Tejada offered the best combination of low error rate, high accuracy, and balanced predictions, making it the most stable and dependable model for classifying mental health-related texts. This makes it the recommended choice for deployment or further analysis.

Table II. Best Fine-Tuning Results by Member for Random Search

Researcher	**Random Search Trial**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Almazan	5/5	0.980737	0.980782	0.980737	0.980709
Pajanustan S.	4/5	0.960548249675865	0.960760779154779	0.960548249675865	0.960416519954017
Tejada	3/5	0.9803667345804778	0.9803904362307762	0.9803667345804778	0.9803427053172904

Among the three models tested with Random Search, Almazan performed the best overall. It achieved the highest accuracy of 0.9807, indicating that it correctly classified the majority of the texts. The model also had the highest F1-score of 0.9807, showing that it maintained a good balance between precision and recall. This means the model was confident and consistent in its predictions compared to the others.

In comparison, the Tejada model also had high accuracy and F1-score, but its values were slightly lower (accuracy 0.9804, F1 0.9803), suggesting it was slightly less precise. The Pajanustan S. model had lower accuracy and F1-score (accuracy 0.9605, F1 0.9604), making it the least reliable among the three.

Overall, Almazan offered the best combination of high accuracy, strong F1-score, and reliable predictions, making it the most stable and dependable model from the Random Search experiments. This makes it the recommended choice for deployment or further analysis.

Table III. Best Fine-Tuning Results by Member for Grid Search

Researcher	Random Search Trial	Accuracy	Precision	Recall	F1 Score
Almazan	6/8	0.981293	0.981291	0.981293	0.981279
PajanustanS.	4/5	0.960548249675865	0.960760779154779	0.960548249675865	0.960416519954017
Tejada	1/9	0.9724022967216152	0.9723937580352047	0.9724022967216152	0.9723971625881617

Among the three models tested with Grid Search, Almazan performed the best overall. It achieved the highest accuracy of 0.9813, showing that it correctly classified the majority of the texts. The model also had the highest F1-score of 0.9813, indicating a strong balance between precision and recall. This means Almazan was the most consistent and confident in its predictions compared to the others.

In comparison, the Tejada model had slightly lower accuracy and F1-score (accuracy 0.9724, F1 0.9724), suggesting it was a bit less precise. The Pajanustan S. model again had the lowest accuracy and F1-score (accuracy 0.9605, F1 0.9604), making it the least reliable among the three.

Overall, Almazan offered the best combination of high accuracy, strong F1-score, and stable predictions, making it the most dependable model from the Grid Search experiments and the recommended choice for deployment or further analysis.

VII. CONCLUSION

The experiments show that different models can perform differently when classifying mental health texts. Among the base models, Tejada performed the best overall. It had the lowest evaluation loss at 0.075, high accuracy of 0.98, and a strong F1-score of 0.975, indicating it made fewer errors and balanced precision and recall well. In comparison, Almazan was slightly less precise, and Pajanustan S. had the lowest accuracy and F1-score, making it the least reliable.

When looking at hyperparameter optimization, Almazan achieved the best performance in both Random Search and Grid Search. With tuned settings for learning rate, save strategy, logging steps, and number of epochs, it achieved the highest accuracy and F1-scores. This shows that optimizing hyperparameters can improve the model's stability and prediction quality compared to using default settings.

Overall, the results suggest that Tejada is the best choice among the base models, while Almazan is the most reliable when hyperparameters are optimized. Both models demonstrated strong performance, low error rates, and consistent predictions. Pajanustan S., on the other hand, was consistently weaker and less stable. These findings highlight the importance of model selection and hyperparameter tuning for developing accurate and dependable mental health classification systems.

VIII. REFERENCE

[1] Ji, S., Zhang, T., Ansari, L., Fu, J., Tiwari, P., & Cambria, E. (2022). MentalBERT: Publicly available Pretrained Language Models for Mental Healthcare. https://aaltodoc.aalto.fi/items/a5d3341d-7ad5-479b-87ce-b70faf81015c

[2] Purnima. (2025, February 16). Hyperparameter Tuning in Machine Learning: Grid Search vs. Random Search. TechieFreak. https://techiefreak.org/index.php/blog/machine-learning/hyperparameter-tuning-in-machine-learning--grid-search-vs--random-search

[3] Wikipedia contributors. (2025, September 5). Hyperparameter optimization. Wikipedia. https://en.wikipedia.org/wiki/Hyperparameter_optimization?utm_source=chatgpt.com

[4] Grid Search vs. Random Search in Machine Learning: A Comprehensive Comparison for Hyperparameter Optimization. (n.d.). techiny.com. https://techiny.com/machine-learning/grid-search-vs-random-search

[5] S. Sarkar, "Sentiment Analysis for Mental Health," Kaggle, 2022. Available: https://www.kaggle.com/datasets/suchintikasarkar/sentiment-analysis-for-mental-health