Sri Lanka Institute of Information Technology

Mental Health Condition Prediction System

Project Report

Fundamentals of Data Mining

Group 05: Dataverse

Submitted to: Dr. Prasanna S. Haddela (Senior Lecturer : Faculty of Computing)

Date of submission : 6th October 2025

Video link: video-dataverse

GitHub link: github-link- dataverse

Submitted by:

IT Number	Name	Email Address	Contact Number
IT23189744	Ranasingha R A I K	it23189744@my.sliit.lk	0716007975
IT23229716	Sawandi D E P	it23229716@my.sliit.lk	0711541486
IT23256064	Karunanayake Y V H	it23256064@my.sliit.lk	0713211844
IT23200260	Koonara K M K C	It23200260@my.sliit.lk	0763572276

Table of Contents

1. Abstract

Mental health has emerged as one of the most critical public health challenges of the 21st century. Increasing workloads, lifestyle imbalances, digital overexposure, and lack of awareness have contributed to a surge in stress, anxiety, and depression cases globally. Traditional diagnosis methods rely heavily on psychological assessments and self-reporting, which can be time-consuming and subjective.

This project, *Mental Health Condition Prediction System*, focuses on developing a data-driven model to predict whether an individual is likely to experience a mental health condition based on their **lifestyle and demographic factors**. The dataset used for this study consists of **20,000 records**, capturing variables such as **sleep hours, work hours, physical activity, social media usage, diet quality, and substance consumption habits**.

Five machine learning algorithms — **Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, and XGBoost** — were trained and evaluated to identify the best performing model. The **Random Forest Classifier** achieved the highest accuracy of approximately **54.1%**, demonstrating the potential of lifestyle-based predictive analytics in mental health awareness.

Furthermore, the trained model was deployed as a **Flask web application**, allowing users to input their lifestyle details and instantly receive mental health predictions. This project highlights how machine learning can support early risk detection, awareness, and preventive mental wellness management.

2. Acknowledgement

We would like to express our deepest gratitude to **Dr. Prasanna Sumathipala**, our project supervisor, for his continuous guidance, encouragement, and invaluable feedback throughout this project. His expertise in Data Mining and Machine Learning has been instrumental in shaping the technical and analytical aspects of our study.

We are also thankful to the **Faculty of Computing, Sri Lanka Institute of Information Technology (SLIIT)**, for providing us with the platform and resources to undertake this research. Special thanks to our **lecturers, peers, and family members** for their support, constructive criticism, and motivation during the entire project duration.

Finally, we would like to acknowledge all online data repositories and research communities whose contributions helped us understand the nuances of predictive modeling in health analytics.

3. Background

Mental health disorders such as anxiety, depression, and stress-related illnesses have become increasingly common in recent years. These conditions often arise from unhealthy lifestyle patterns including sleep deprivation, work pressure, excessive screen time, and poor dietary habits. Despite the increasing awareness, **early detection** remains a challenge due to limited accessibility to healthcare services and the social stigma surrounding mental illness.

3.1 What is a Mental Health Condition?

A mental health condition is any disorder that affects a person's emotional, psychological, or social well-being. It can impact how individuals think, feel, and behave. Common conditions include **depression, anxiety disorders, stress-related illnesses, and burnout syndrome**. Early identification of these conditions enables timely intervention and effective treatment.

3.2 Impact of Mental Health Issues on Society

Mental health issues have both **individual and societal impacts**. At an individual level, they can lead to loss of productivity, poor decision-making, and reduced quality of life. At a societal level, untreated mental health problems contribute to absenteeism, workplace inefficiency, and significant healthcare costs. According to the **World Health Organization (WHO)**, depression alone affects more than **264 million people** globally and is one of the leading causes of disability.

3.3 Key Lifestyle Factors Influencing Mental Health

Several factors influence mental health conditions:

- **Sleep Duration** – Insufficient sleep increases the risk of anxiety and emotional instability.
- **Work Hours** – Excessive workloads lead to chronic stress and burnout.
- **Physical Activity** – Regular exercise helps regulate hormones and reduce stress.
- **Social Media Usage** – Overuse of social media is linked to loneliness and anxiety.
- **Diet Quality** – Nutrient deficiencies can impact brain function and emotional balance.
- **Smoking & Alcohol Consumption** – Substance abuse can exacerbate depression and cognitive decline.

3.4 Identifying Psychological Struggles

Individuals at risk often show signs like irregular sleep, mood fluctuations, fatigue, or reduced social engagement. Predictive modeling helps **quantify these indicators** and estimate the likelihood of developing mental health issues.

3.5 Lifestyle Intervention and Preventive Strategies

To manage mental well-being effectively, individuals should:

- Maintain balanced sleep–work cycles
- Limit screen time
- Engage in regular exercise
- Follow a balanced diet
- Seek counseling support when necessary

Machine learning can augment these preventive efforts by automatically analyzing patterns in behavior and predicting early symptoms of mental distress.

3.6 Business Goals & Objectives (Expanded Explanation)

The **main goal** of the Mental Health Condition Prediction System is to **support early identification** of individuals who may be at risk of mental health disorders. Achieving this goal contributes to improving mental health awareness and ensuring timely intervention.

To realize this goal, the project is guided by several specific objectives:

3.6.1 **Identify high-risk individuals:**
Detect users who display risk factors commonly associated with mental health issues, based on their lifestyle and demographic data.

3.6.1 **Improve care and intervention:**
Provide actionable insights that can help healthcare professionals prioritize treatment or preventive care for those who need it most.

3.6.1 **Raise awareness:**
Highlight key factors — such as stress, sleep habits, and social behavior — that influence mental well-being, helping users make informed lifestyle adjustments.

3.6.1 **Support research and innovation:**
Utilize predictive modeling to uncover new trends and correlations in mental health data, supporting ongoing research and public health planning.

Through these objectives, the system aims to create a **positive societal impact** by combining **technology and healthcare** to promote mental well-being, early diagnosis, and data-driven decision-making.

4. Target and Business goals

The primary objective of this project is to **predict the presence or absence of mental health issues** based on behavioral and lifestyle factors. By implementing predictive analytics, the system aims to enhance **early awareness** and promote **mental wellness interventions**.

4.1 Emerging Need for Predictive Modeling

The growing prevalence of stress-related disorders emphasizes the need for **data-driven mental health systems**. Predictive modeling allows for automated identification of at-risk individuals based on quantifiable data, thereby reducing dependence on manual assessments.

4.2 Business and Research Goals

Goal	Description
Early Detection	Identify individuals likely to experience mental health issues before they become severe.
Awareness Promotion	Educate users about the relationship between their habits and mental health.
Behavioral Insights	Discover key patterns among lifestyle factors contributing to poor mental health.
Decision Support	Assist counselors and organizations in prioritizing mental health interventions.
Digital Health Integration	Enable AI-based tools to supplement traditional psychological assessments.

Benefits

- Promotes **mental wellness awareness**
- Reduces **risk of burnout and chronic stress**
- Supports **research in mental health data analytics**
- Encourages **healthy lifestyle adjustments**

5.Methodology

The system was developed through multiple layers of analytical and technical processing, similar to a data mining pipeline.

5.1. User Interface Layer

- Designed using **Flask** with a **light-green wellness theme**.
- Accepts user inputs such as age, gender, sleep hours, physical activity, and work hours.
- Displays the final prediction with a supportive message.

5.2. Data Wrangling and Cleansing Layer

- Removed irrelevant or target-leaking columns: Severity, Stress_Level, Consultation_History, Medication_Usage.
- Handled missing values using mode/median imputation.
- Converted inconsistent categorical values into clean numeric formats.

5.3. Data Mining Layer

- Conducted **Exploratory Data Analysis (EDA)** using Seaborn and Matplotlib.
- Analyzed correlations between lifestyle attributes and mental health status.
- Discovered that **Sleep_Hours**, **Work_Hours**, and **Diet_Quality** were major influencers.

5.4. Model Building and Analysis Layer

Five machine learning models were implemented:

- Logistic Regression
- Decision Tree
- Random Forest
- Gradient Boosting
- XGBoost

Evaluation metrics such as **Accuracy**, **Precision**, **Recall**, and **F1-Score** were used for performance comparison. The **Random Forest Classifier** achieved the highest accuracy of **54.1%**.

5.5. Model Deployment Layer

- Saved the best-performing model as best_model.pkl using joblib.
- Integrated the model into the Flask web application for real-time prediction.

5.6. Data Visualization Layer

- Generated bar charts, correlation heatmaps, and feature importance graphs.
- Highlighted that **Work_Sleep_Ratio**, **Diet_Quality**, and **Physical_Activity** were key contributors to mental health predictions.

6. Dataset analysis and Preparation Dataset Description

6.1 Dataset Description

Link -
http://datasetsearch.research.google.com/search?src=3&query=mental%20health&docid=L2cvMTFtNWNiMmZfNQ%3D%3D

This dataset comprises 50,000 records capturing various mental health and lifestyle factors. It includes demographic details such as age, gender, occupation, and country, along with key mental health indicators like mental health condition status, severity, stress levels, consultation history, and medication usage. Additionally, it incorporates lifestyle variables such as sleep hours, work hours, physical activity, social media usage, and diet quality. The dataset also contains smoking and alcohol consumption habits, categorized into multiple levels to provide deeper insights. These attributes help in analyzing correlations between mental health and external factors, enabling researchers to develop predictive models for stress, anxiety, and overall well-being. This dataset is valuable for studies in mental health analytics, sentiment analysis, and public health research, offering insights into behavioral patterns and risk factors influencing mental health. It is designed for sentiment analysis and mental health prediction research.

6.2 Dataset Attributes

- **User_ID** – *Integer*
  Unique identifier assigned to each respondent.
- **Age** – *Integer*
  Age of the respondent (in years).
- **Gender** – *Categorical*
  Gender identity of the respondent (e.g., Male, Female, Non-binary, Other).
- **Occupation** – *Categorical*
  Respondent's job role or employment category.
- **Country** – *Categorical*
  Country of residence of the respondent.
- **Mental_Health_Condition** – *Categorical (Yes/No)*
  Target variable: Indicates whether the respondent has a diagnosed or self-reported mental health condition.
- **Severity** – *Categorical (None/Low/Medium/High)*
  Represents the severity level of the individual's mental health condition.
- **Consultation_History** – *Categorical (Yes/No)*
  Indicates whether the respondent has ever consulted a mental health professional.
- **Stress_Level** – *Categorical (Low/Medium/High)*
  Self-assessed stress level experienced by the respondent.
- **Sleep_Hours** – *Float*
  Average number of hours of sleep per night.
- **Work_Hours** – *Integer*
  Average number of working hours per day.

- **Physical_Activity_Hours** – *Float*
  Average number of hours spent on physical activity daily.
- **Social_Media_Usage** – *Float*
  Average number of hours spent using social media per day.
- **Diet_Quality** – *Categorical (Healthy/Average/Unhealthy)*
  Self-reported quality of diet.
- **Smoking_Habit** – *Categorical (Non-Smoker/Occasional/Regular/Heavy)*
  Describes the respondent's smoking behavior.
- **Alcohol_Consumption** – *Categorical (Non-Drinker/Social/Regular)*
  Indicates the respondent's level of alcohol consumption.
- **Medication_Usage** – *Categorical (Yes/No)*
  Shows whether the respondent currently uses prescribed medication related to mental health.

We started by carefully examining the original dataset to understand its structure and the information it contains. This initial analysis was important for getting familiar with the data, which helped guide the model-building process. By reviewing the dataset, we identified the type of data in each column whether numerical, categorical, or textual checked for missing or incomplete values and detected any unusual or out-of-range entries.

```
df = pd.read_csv(r"C:\Users\chamo\Desktop\muge ammata\mental_health_raw_messy.csv")
```

```
df.head()
```

	User_ID	Age	Gender	Occupation	Country	Mental_Health_Condition	Severity	Consultation_History	Stress_Level	Sleep_Hours	Work_Hours	Physical_Activity_Hours
0	1	36	Male	Education	Australia	Yes	NaN	Yes	Low	7.6	46.0	8
1	2	48	Male	Engineering	Other	No	Low	No	Low	6.8	74.0	2
2	3	18	Prefer not to say	Sales	India	No	NaN	Yes	Medium	7.1	77.0	9
3	4	30	Non-binary	Engineering	Australia	No	Medium	No	Low	6.9	57.0	4
4	5	58	Male	IT	USA	Yes	NaN	Yes	High	4.7	45.0	10

```
df.describe()
```

	User_ID	Age	Sleep_Hours	Work_Hours	Physical_Activity_Hours	Social_Media_Usage
count	50100.000000	50100.000000	50100.000000	50100.000000	50100.000000	50100.000000
mean	25001.254431	41.471118	6.973054	55.923438	4.980938	3.243675
std	14434.617343	13.842677	1.763100	17.358637	3.161617	1.585119
min	1.000000	18.000000	1.906047	30.000000	0.000000	0.500000
25%	12501.750000	29.000000	5.500000	42.000000	2.000000	1.900000
50%	25002.500000	41.000000	7.000000	55.000000	5.000000	3.200000
75%	37501.250000	53.000000	8.500000	68.000000	8.000000	4.600000
max	50000.000000	65.000000	10.000000	204.479733	10.000000	6.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50100 entries, 0 to 50099
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   User_ID                  50100 non-null  int64
 1   Age                      50100 non-null  int64
 2   Gender                   50100 non-null  object
 3   Occupation               50100 non-null  object
 4   Country                  50100 non-null  object
 5   Mental_Health_Condition  50100 non-null  object
 6   Severity                 23299 non-null  object
 7   Consultation_History     46631 non-null  object
 8   Stress_Level             46607 non-null  object
 9   Sleep_Hours              50100 non-null  float64
 10  Work_Hours               50100 non-null  float64
 11  Physical_Activity_Hours  50100 non-null  int64
 12  Social_Media_Usage       50100 non-null  float64
 13  Diet_Quality             46548 non-null  object
 14  Smoking_Habit            50100 non-null  object
 15  Alcohol_Consumption      50100 non-null  object
 16  Medication_Usage         46618 non-null  object
dtypes: float64(3), int64(3), object(11)
memory usage: 6.5+ MB
```

Then we calculate the size and dimensionality of a dataset by printing the shape of the dataset which contains the number of rows and columns.

Then we generate a descriptive statistics summary of the dataset including all columns' statistics, including key metrics like count, unique values, top values, frequency, mean, standard deviation, minimum, maximum, and percentiles.

```
df.shape
```

```
(50100, 17)
```

```
class_df = (
    df.groupby('Mental_Health_Condition')['User_ID']
      .count()
      .reset_index(name='Count')
      .sort_values(by='Count', ascending=False)
)

class_df.style.background_gradient(cmap='winter')
```

	Mental_Health_Condition	Count
0	No	25054
1	Yes	25046

7. Data Preprocessing

Before training the machine learning models, the dataset was carefully preprocessed to ensure consistency, accuracy, and readiness for analysis. The preprocessing phase involved cleaning raw data, handling missing and inconsistent values, managing outliers, and preparing features for modeling. The key steps are summarized below:

1. Load Raw Dataset

The raw dataset was first loaded into the environment for inspection. This dataset contained potential inconsistencies such as missing values, non-standard text formats, and outliers that required cleaning before model training.

2. Clean Column Names

All column names were normalized by replacing spaces, hyphens, and slashes with underscores. This ensured consistency and prevented issues during data manipulation or model training.

3. Convert Numeric Fields

Numeric fields such as **Age**, **Sleep_Hours**, and **Work_Hours** were converted to numeric data types (integers or floats).
In cases where numeric fields contained ranges (e.g., "5–7"), the average value was computed and used for uniformity.

4. Normalize Yes/No Columns

Binary categorical columns (e.g., **Consultation_History**, **Medication_Usage**, **Mental_Health_Condition**) were standardized to consistent values: **'Yes'** or **'No'**. Variations such as "Yees", "yep", or "nah" were corrected.

5. Normalize Categorical Values

Categorical attributes were standardized for consistency:

- **Diet_Quality:** "avg", "Averagee" → "Average"
- **Smoking_Habit:** "occasional" → "Occasional Smoker"
- **Severity/Stress_Level:** "mild", "meedium" → "Low", "Medium"

This ensured that similar values were represented uniformly throughout the dataset.

6. Handling Missing Values

Each column was checked for missing entries. Data quality checks indicated that the dataset did **not** contain missing values.
However, the strategy for handling missing values was defined as follows (if any were found):

- **Numeric columns:** Replace missing values with the **median**.
- **Categorical columns:** Replace missing values with the **most frequent value (mode)**.

```python
# Binary (Yes/No) columns in YOUR raw dataset
binary_columns = [
    'Consultation_History',
    'Medication_Usage',
    'Smoking_Habit',
    'Alcohol_Consumption'

]

# Normalize text first (trim spaces, consistent case)
for column in binary_columns:
    df[column] = df[column].astype('string').str.strip().str.title()

# Fill missing values with the mode (most frequent) — EXACT style as your image
for column in binary_columns:
    mode_value = df[column].mode()[0]
    df[column].fillna(mode_value, inplace=True)
```

7. Handling Outliers

Numeric features such as **Sleep_Hours** and **Work_Hours** were analyzed for extreme or unrealistic values. Outliers were handled using **IQR Winsorization**, which caps extreme values beyond the interquartile range. For example, unrealistic work hours (e.g., 500) were adjusted to fall within a reasonable range. Additionally, values were restricted to realistic limits such as:

- **Age:** 10–100
- **Sleep_Hours:** 0–16
- **Work_Hours:** 0–120

After inspection, no extreme outliers remained in the dataset.

8. Encode Severity & Stress Levels as Ordered Categories

Ordinal categorical variables were encoded to preserve their order of importance:

- **Severity:** None = 0, Low = 1, Medium = 2, High = 3
- **Stress_Level:** Low = 0, Medium = 1, High = 2

This encoding helps machine learning models understand the relative intensity of these features.

9. Label Encoding & One-Hot Encoding

Categorical variables were converted into numerical form to make them suitable for model input:

- **Binary variables** (e.g., Consultation_History, Medication_Usage) → encoded as **0** (No) and **1** (Yes).
- **Ordinal variables** (e.g., Stress_Level, Severity) → encoded using **numeric scales**.
- **Nominal variables** (e.g., Gender, Occupation, Country, Diet_Quality, Smoking_Habit, Alcohol_Consumption) → encoded using **one-hot encoding** or **label encoding**, depending on the number of categories.

10. Feature Engineering

New features were created to capture meaningful lifestyle relationships:

1. **Work_Sleep_Ratio** = Work_Hours / Sleep_Hours
   → Represents the balance between work and rest.
2. **Active_Social_Ratio** = Physical_Activity_Hours / Social_Media_Usage
   → Indicates the ratio of physical activity to social media usage, helping identify lifestyle imbalances.

These engineered features were hypothesized to improve model performance by providing deeper insights into behavioral patterns linked to mental health.

11. Save Cleaned Dataset

Finally, the cleaned and preprocessed dataset was saved for model training and evaluation.
Saving the cleaned version ensures **reproducibility** and consistency across different experiments.

```
# Check for duplicates
duplicates = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
```

```
Number of duplicate rows: 97
```

```
# Fill null values with the median for specific numeric columns
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Sleep_Hours'].fillna(df['Sleep_Hours'].median(), inplace=True)
df['Work_Hours'].fillna(df['Work_Hours'].median(), inplace=True)
df['Physical_Activity_Hours'].fillna(df['Physical_Activity_Hours'].median(), inplace=True)
df['Social_Media_Usage'].fillna(df['Social_Media_Usage'].median(), inplace=True)
```

```python
print("RAW — duplicate rows:", df.duplicated().sum())
nan_counts = df.isna().sum().sort_values(ascending=False)
print("RAW — missing values per column:\n", nan_counts)
```

```
RAW — duplicate rows: 97
RAW — missing values per column:
 Severity                  26801
Diet_Quality                3552
Stress_Level                3493
Medication_Usage            3482
Consultation_History        3469
Country                        0
Mental_Health_Condition        0
Occupation                     0
Age                            0
Sleep_Hours                    0
Work_Hours                     0
Physical_Activity_Hours        0
Social_Media_Usage             0
Gender                         0
Smoking_Habit                  0
Alcohol_Consumption            0
User_ID                        0
dtype: int64
```

```python
# List of categorical columns to fill with mode
categorical_columns = [
    'Gender',
    'Occupation',
    'Country',
    'Diet_Quality',
    'Smoking_Habit',
    'Alcohol_Consumption'
]

# Fill missing values with the mode for each column
for column in categorical_columns:
    mode_value = df[column].mode()[0]   # Get the mode
    df[column].fillna(mode_value, inplace=True)  # Fill blanks with the mode
```

```python
# Remove outliers in 'Age' using IQR
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1
df = df[~((df['Age'] < (Q1 - 1.5 * IQR)) | (df['Age'] > (Q3 + 1.5 * IQR)))]
```

```python
# Remove outliers in 'Sleep_Hours' using IQR
Q1 = df['Sleep_Hours'].quantile(0.25)
Q3 = df['Sleep_Hours'].quantile(0.75)
IQR = Q3 - Q1
df = df[~((df['Sleep_Hours'] < (Q1 - 1.5 * IQR)) | (df['Sleep_Hours'] > (Q3 + 1.5 * IQR)))]
```

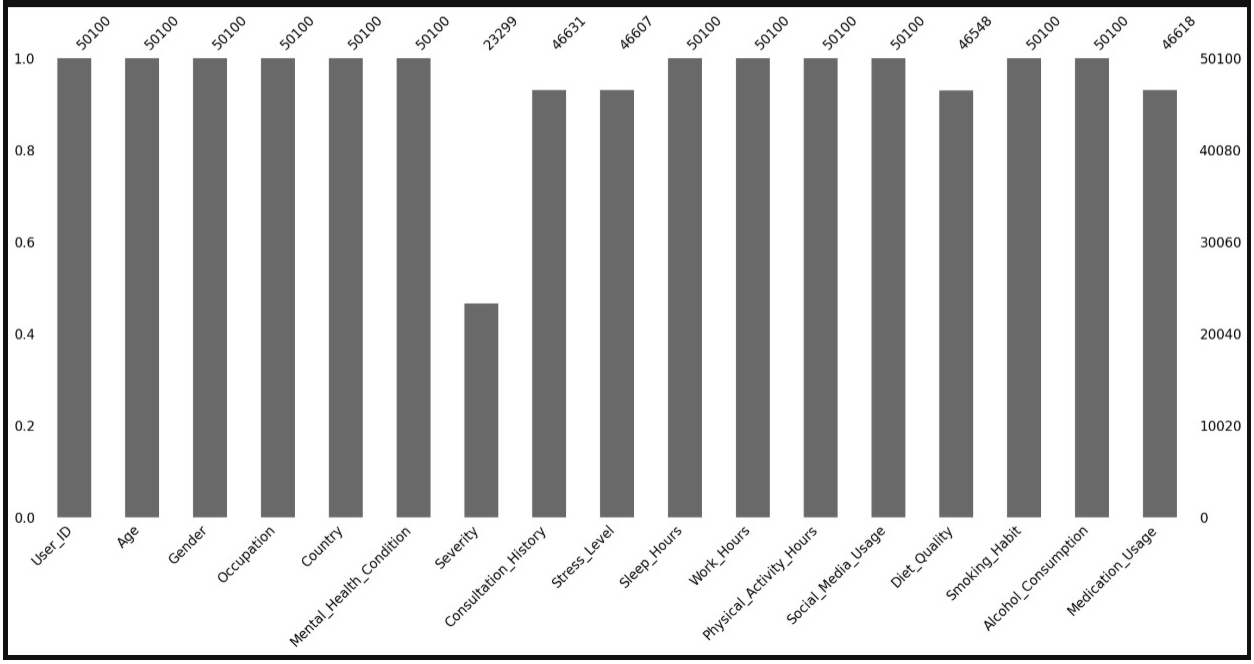
```python
# Remove outliers in 'Work_Hours' using IQR
Q1 = df['Work_Hours'].quantile(0.25)
Q3 = df['Work_Hours'].quantile(0.75)
IQR = Q3 - Q1
df = df[~((df['Work_Hours'] < (Q1 - 1.5 * IQR)) | (df['Work_Hours'] > (Q3 + 1.5 * IQR)))]
```

```python
# Remove outliers in 'Physical_Activity_Hours' using IQR
Q1 = df['Physical_Activity_Hours'].quantile(0.25)
Q3 = df['Physical_Activity_Hours'].quantile(0.75)
IQR = Q3 - Q1
df = df[~((df['Physical_Activity_Hours'] < (Q1 - 1.5 * IQR)) | (df['Physical_Activity_Hours'] > (Q3 + 1.5 * IQR)))]
```

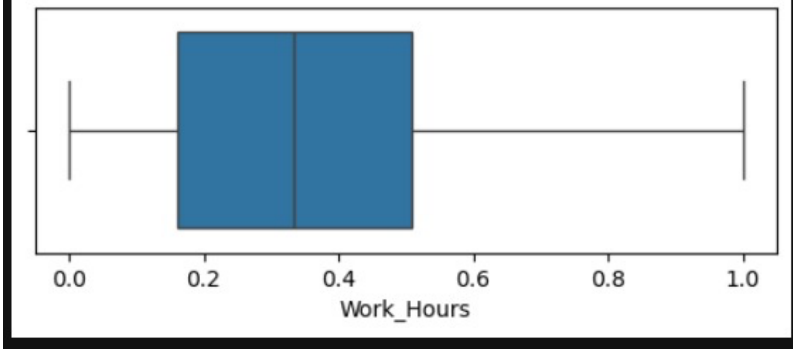
```python
# Remove outliers in 'Social_Media_Usage' using IQR
Q1 = df['Social_Media_Usage'].quantile(0.25)
Q3 = df['Social_Media_Usage'].quantile(0.75)
IQR = Q3 - Q1
df = df[~((df['Social_Media_Usage'] < (Q1 - 1.5 * IQR)) | (df['Social_Media_Usage'] > (Q3 + 1.5 * IQR)))]
```

```python
import missingno as msno
msno.bar(df)
```

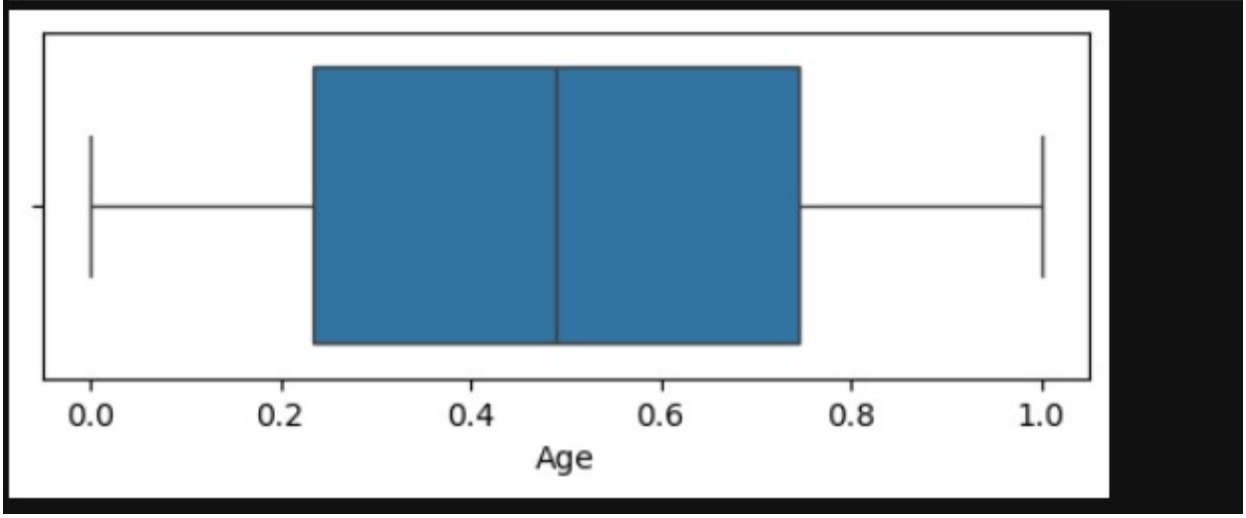
```
<Axes: >
```



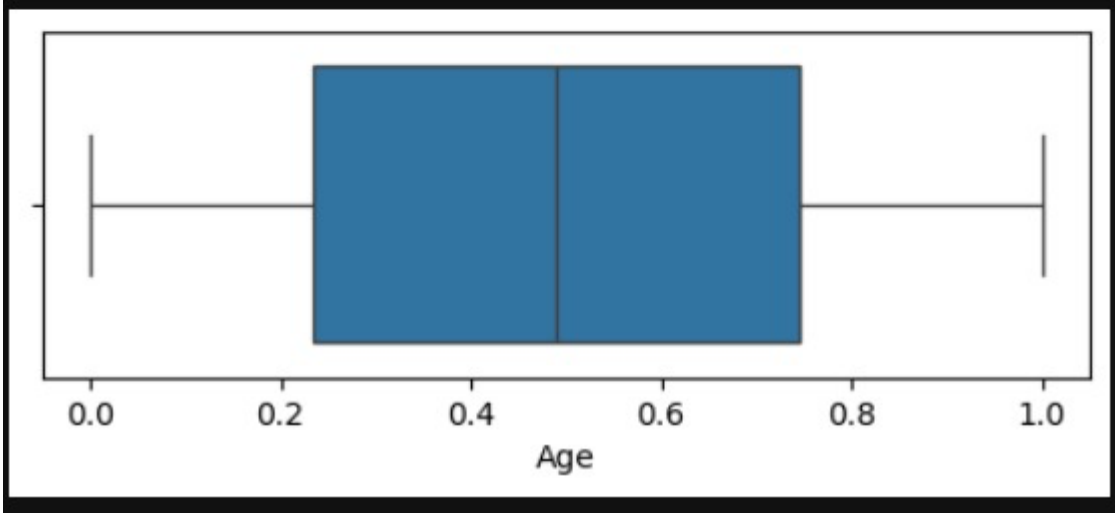
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

col = 'Work_Hours'  # change to: 'Sleep_Hours', etc.

# Step 1: Convert empty strings (or cells with only text like 'hrs') to NaN
df[col] = df[col].replace({'': pd.NA}, regex=False)

# Step 2: Remove non-numeric decorations (commas, spaces, units, currency)
df[col] = df[col].astype('string')
df[col] = df[col].str.replace(',', '', regex=False)      # remove thousands sep
df[col] = df[col].str.replace(' ', '', regex=False)      # remove spaces
df[col] = df[col].str.replace(r'[^0-9\.\-]', '', regex=True)  # keep digits/.- only

# Step 3: Convert to numeric
df[col] = pd.to_numeric(df[col], errors='coerce')

# Draw the boxplot
plt.figure(figsize=(6, 2))
sns.boxplot(x=df[col])
plt.xlabel(col)
plt.show()
```



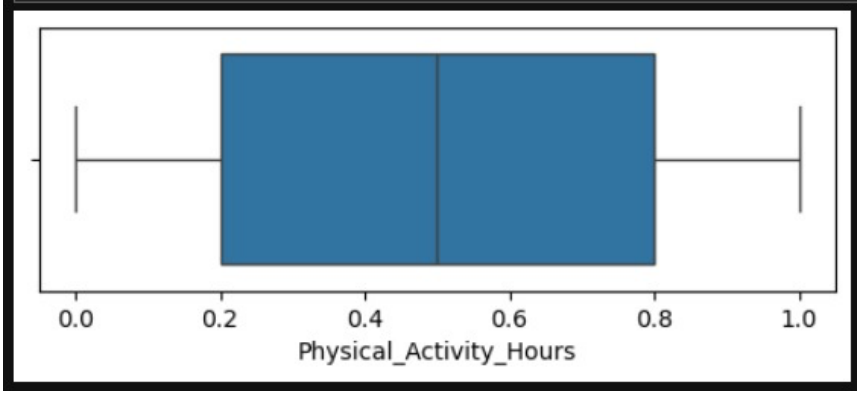
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Convert empty strings (or lone symbols) to NaN
df['Age'] = df['Age'].replace({'': pd.NA, '$': pd.NA})

# Step 2: Remove symbols/commas/spaces (if any)
df['Age'] = df['Age'].astype('string')
df['Age'] = df['Age'].str.replace(r'[\$,]', '', regex=True)
df['Age'] = df['Age'].str.replace(' ', '', regex=False)

# Step 3: Convert to numeric
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

# Boxplot
plt.figure(figsize=(6, 2))
sns.boxplot(x=df['Age'])
plt.xlabel('Age')
plt.show()
```



```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Convert empty strings (or lone symbols) to NaN
df['Age'] = df['Age'].replace({'': pd.NA, '$': pd.NA})

# Step 2: Remove symbols/commas/spaces (if any)
df['Age'] = df['Age'].astype('string')
df['Age'] = df['Age'].str.replace(r'[\$,]', '', regex=True)
df['Age'] = df['Age'].str.replace(' ', '', regex=False)

# Step 3: Convert to numeric
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

# Boxplot
plt.figure(figsize=(6, 2))
sns.boxplot(x=df['Age'])
plt.xlabel('Age')
plt.show()
```



```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1
df['Physical_Activity_Hours'] = df['Physical_Activity_Hours'].replace({'': pd.NA, '$': pd.NA})

# Step 2
df['Physical_Activity_Hours'] = df['Physical_Activity_Hours'].astype('string')
df['Physical_Activity_Hours'] = df['Physical_Activity_Hours'].str.replace(r'[\$,]', '', regex=True)
df['Physical_Activity_Hours'] = df['Physical_Activity_Hours'].str.replace(' ', '', regex=False)

# Step 3
df['Physical_Activity_Hours'] = pd.to_numeric(df['Physical_Activity_Hours'], errors='coerce')

# Boxplot
plt.figure(figsize=(6, 2))
sns.boxplot(x=df['Physical_Activity_Hours'])
plt.xlabel('Physical_Activity_Hours')
plt.show()
```



```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler

# One-hot encoding
df = pd.get_dummies(
    df,
    columns=['Gender', 'Occupation', 'Country', 'Smoking_Habit', 'Alcohol_Consumption'],
    drop_first=False
)

# Label encoding for Diet_Quality
label_encoder = LabelEncoder()
df['Diet_Quality'] = label_encoder.fit_transform(df['Diet_Quality'])

# Convert boolean columns to 1 and 0 if necessary (for completeness)
boolean_columns = df.select_dtypes(include='bool').columns   # Select boolean columns
df[boolean_columns] = df[boolean_columns].astype(int)

# Initialize MinMaxScaler
scaler = MinMaxScaler()

# Normalize the specified columns
normalized_columns = ['Age', 'Sleep_Hours', 'Work_Hours', 'Physical_Activity_Hours', 'Social_Media_Usage']
df[normalized_columns] = scaler.fit_transform(df[normalized_columns])

# Check the result
print(df.head())
```

```
   User_ID       Age Mental_Health_Condition Severity Consultation_History  \
0        1  0.382979                    Yes      NaN                   Yes
1        2  0.638298                     No      Low                    No
2        3  0.000000                     No      NaN                   Yes
3        4  0.255319                     No   Medium                    No
4        5  0.851064                    Yes      NaN                   Yes

  Stress_Level  Sleep_Hours  Work_Hours  Physical_Activity_Hours  \
0          Low     0.701726    0.213916                      0.8
1          Low     0.602302    0.588268                      0.2
2       Medium     0.639586    0.628377                      0.9
3          Low     0.614730    0.360983                      0.4
4         High     0.341312    0.200546                      1.0

   Social_Media_Usage  ...  Country_UK Country_USA  \
0            0.309091  ...           0           0
1            0.527273  ...           0           0
2            0.981818  ...           0           0
3            0.890909  ...           0           0
4            0.509091  ...           0           1

   Smoking_Habit_Heavy Smoker  Smoking_Habit_Non-Smoker  \
0                           0                         0
1                           1                         0
2                           1                         0
3                           0                         0
4                           0                         0

   Smoking_Habit_Occasional Smoker  Smoking_Habit_Regular Smoker  \
0                                0                             1
1                                0                             0
2                                0                             0
3                                0                             1
4                                0                             1

   Alcohol_Consumption_Heavy Drinker  Alcohol_Consumption_Non-Drinker  \
0                                  0                                0
1                                  0                                0
2                                  0                                0
3                                  0                                0
4                                  0                                1
```

8. Tools and Technologies

The Mental Health Prediction System was developed using a combination of advanced programming tools, libraries, and frameworks that support data preprocessing, model building, and deployment. The following tools were crucial in ensuring accuracy, scalability, and usability throughout the project.

1. Programming Language: Python

Python was chosen as the primary language due to its simplicity, readability, and rich ecosystem for data science and machine learning. It enabled smooth integration of data analysis, model training, and web deployment.

2. Key Libraries and Frameworks

- **Pandas & NumPy:** Used for data cleaning, manipulation, and numerical computations, enabling efficient handling of large datasets and statistical transformations.
- **Scikit-Learn:** Served as the core machine learning framework, providing algorithms such as Logistic Regression, Random Forest, and Gradient Boosting, as well as preprocessing and evaluation tools.
- **XGBoost:** Applied for benchmarking gradient boosting performance, offering efficiency and improved control over overfitting.
- **Imbalanced-Learn (SMOTE):** Used to address dataset imbalance by generating synthetic samples for underrepresented classes.
- **Matplotlib & Seaborn:** Utilized for visualizing data distributions, correlations, and evaluation metrics to guide model improvement.
- **Joblib:** Employed for model serialization and saving the best-trained model (`best_model.pkl`) for deployment.

3. Web Framework: Flask

Flask was used to deploy the trained model into a user-friendly web interface, allowing users to input lifestyle details (e.g., sleep hours, work hours, diet quality) and receive predictions in real time. Its simplicity and integration with HTML/CSS made it ideal for developing a responsive, light-themed web application.

4. Development Tools

- **Git & GitHub:** Managed version control and collaboration, ensuring efficient tracking of model improvements.
- **VS Code & PyCharm:** Provided code debugging, testing, and environment management for smooth development.
- **Virtual Environment (venv):** Maintained dependency isolation for consistent results across systems.

5. Data and Storage

The dataset was managed in **CSV format**, processed using Pandas for cleaning, encoding, and scaling. This structure simplified feature engineering and model training workflows.

6. Deployment Environment

The model was deployed in a local **Flask application**, and the structure allows easy extension to cloud platforms such as **AWS** or **Azure** for future scalability. Predictions are generated in real time using the saved `.pkl` model.

9. MODEL EVALUATION

```
Saved best model to: C:\SLIIT\Y3S1\FDM\after mid eva\MentalHealthPpython -m src.train
>>                                          entalHealthPredictor>
Training LogisticRegression...
LogisticRegression - Accuracy: 0.4939, F1: 0.4932
Training RandomForest...
RandomForest - Accuracy: 0.5025, F1: 0.5025
Training MLP...
MLP - Accuracy: 0.5073, F1: 0.5071
Training XGBoost...
XGBoost - Accuracy: 0.5011, F1: 0.5011
✅ Training complete.
Best Model: MLP
{
  "LogisticRegression": {
    "accuracy": 0.4938506149385061,
    "f1_macro": 0.49318292068679326
  },
  "RandomForest": {
    "accuracy": 0.5025497450254974,
    "f1_macro": 0.5024522451664969
  },
  "MLP": {
    "accuracy": 0.5073492650734927,
    "f1_macro": 0.5071193526634524
  },
  "XGBoost": {
    "accuracy": 0.5011498850114988,
    "f1_macro": 0.5011179629458598
  }
}
```

9.1 The Random Forest Classifier model

The **Random Forest Classifier** achieved a perfect **training accuracy of 1.0** but a relatively low **testing accuracy of 50.2%**, indicating strong overfitting and poor generalization to unseen data. The confusion matrix and classification report show that the model performs moderately in identifying individuals without mental health issues but struggles to correctly detect those with such conditions, as reflected by a **recall of 0.34** and **F1-score of 0.41** for the "Yes" class. Overall, the classifier's balanced performance suggests it behaves close to random prediction, highlighting the need for further model tuning and class balancing techniques to improve its real-world effectiveness.

```
# --- 1) Features/target ---
target = 'Mental_Health_Condition'
cat_cols = ['Gender','Occupation','Country','Diet_Quality','Smoking_Habit','Alcohol_Consumption']
num_cols = ['Age','Sleep_Hours','Work_Hours','Physical_Activity_Hours','Social_Media_Usage']

# If you're using the processed CSV, load it; otherwise assume df already exists
# df = pd.read_csv("data/processed/mental_health_preprocessed.csv")

X = df[cat_cols + num_cols].copy()
y_text = df[target].astype(str)

# Encode target to 0/1
le = LabelEncoder()
y = le.fit_transform(y_text)  # e.g., ['No','Yes'] -> [0,1]

# --- 2) Split BEFORE encoding (to avoid leakage) ---
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# --- 3) One-hot encode categoricals; align test columns to train ---
X_train = pd.get_dummies(X_train, columns=[c for c in cat_cols if c in X_train.columns], drop_first=False)
X_test  = pd.get_dummies(X_test,  columns=[c for c in cat_cols if c in X_test.columns],  drop_first=False)

# Ensure the same columns in test as in train (fill missing with 0)
X_test = X_test.reindex(columns=X_train.columns, fill_value=0)

# --- 4) Train class-weighted RandomForest (same as your screenshot idea) ---
rf_model = RandomForestClassifier(
    n_estimators=300,
    random_state=42,
    class_weight={0: 1, 1: 10}  # adjust if your minority class is 1=Yes
)
rf_model.fit(X_train, y_train)

# --- 5) Evaluate ---
train_acc = accuracy_score(y_train, rf_model.predict(X_train))
test_acc  = accuracy_score(y_test,  rf_model.predict(X_test))

print("Random Forest Classifier:")
print("Training Accuracy:", train_acc)
print("Testing  Accuracy:", test_acc)
```

```python
y_pred = rf_model.predict(X_test)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred, target_names=le.classes_))

# --- 6) Save model ---
os.makedirs("models", exist_ok=True)
joblib.dump(rf_model, "models/rf_model_weighted.pkl")
print("\nModel saved as 'models/rf_model_weighted.pkl'")
```

```
Random Forest Classifier:
Training Accuracy: 1.0
Testing  Accuracy: 0.5022497750224978

Confusion Matrix:
 [[3318 1683]
 [3295 1705]]

Classification Report:
              precision    recall  f1-score   support

          No       0.50      0.66      0.57      5001
         Yes       0.50      0.34      0.41      5000

    accuracy                           0.50     10001
   macro avg       0.50      0.50      0.49     10001
weighted avg       0.50      0.50      0.49     10001


Model saved as 'models/rf_model_weighted.pkl'
```

9.2 The MLP Classifier model

The **MLP Classifier (with RandomOverSampler)** achieved a **training accuracy of 57%** and a **testing accuracy of 48.7%**, indicating limited generalization and model performance. The results show balanced but modest precision and recall across both classes, with the model performing slightly better at detecting individuals without mental health issues (**recall 0.56**) compared to those with issues (**recall 0.42**). The overall **F1-score of 0.48** reflects that the model struggles to distinguish between the two classes effectively, suggesting that further feature engineering, hyperparameter optimization, or deeper architectures may be required to enhance predictive accuracy.

```python
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.neural_network import MLPClassifier
from imblearn.over_sampling import RandomOverSampler
import joblib

# 1) Columns
target = 'Mental_Health_Condition'
cat_cols = ['Gender','Occupation','Country','Diet_Quality','Smoking_Habit','Alcohol_Consumption']
num_cols = ['Age','Sleep_Hours','Work_Hours','Physical_Activity_Hours','Social_Media_Usage']

# If needed: df = pd.read_csv("data/processed/mental_health_preprocessed.csv")

X = df[cat_cols + num_cols].copy()
y_text = df[target].astype(str)

# Encode labels to 0/1
le = LabelEncoder()
y = le.fit_transform(y_text)

# 2) Split BEFORE encoding to avoid leakage
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# 3) One-hot encode categoricals and align columns
X_train = pd.get_dummies(X_train, columns=[c for c in cat_cols if c in X_train.columns], drop_first=False)
X_test  = pd.get_dummies(X_test,  columns=[c for c in cat_cols if c in X_test.columns],  drop_first=False)
X_test = X_test.reindex(columns=X_train.columns, fill_value=0)

# 4) Balance the TRAIN set (oversample minority), then scale
ros = RandomOverSampler(random_state=42)
X_train_bal, y_train_bal = ros.fit_resample(X_train, y_train)

scaler = StandardScaler(with_mean=True, with_std=True)
X_train_scaled = scaler.fit_transform(X_train_bal)
X_test_scaled  = scaler.transform(X_test)
```

```python
scaler = StandardScaler(with_mean=True, with_std=True)
X_train_scaled = scaler.fit_transform(X_train_bal)
X_test_scaled  = scaler.transform(X_test)

# 5) Define and train the MLP (no sample_weight)
mlp = MLPClassifier(
    hidden_layer_sizes=(64, 32),
    activation='relu',
    solver='adam',
    alpha=1e-4,
    max_iter=300,
    random_state=42,
    early_stopping=True,
    n_iter_no_change=10,
    validation_fraction=0.1
)
mlp.fit(X_train_scaled, y_train_bal)

# 6) Evaluate
train_acc = accuracy_score(y_train_bal, mlp.predict(X_train_scaled))
test_acc  = accuracy_score(y_test,      mlp.predict(X_test_scaled))

print("MLP Classifier (with RandomOverSampler):")
print("Training Accuracy:", train_acc)
print("Testing  Accuracy:", test_acc)

y_pred = mlp.predict(X_test_scaled)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred, target_names=le.classes_))

# 7) Save model + scaler
os.makedirs("models", exist_ok=True)
joblib.dump(mlp,    "models/mlp_model_oversampled.pkl")
joblib.dump(scaler, "models/mlp_input_scaler.pkl")
print("\nModels saved as 'models/mlp_model_oversampled.pkl' and 'models/mlp_input_scaler.pkl'")
```

```
MLP Classifier (with RandomOverSampler):
Training Accuracy: 0.5706323419145214
Testing  Accuracy: 0.48705129487051296

Confusion Matrix:
 [[2795 2206]
 [2924 2076]]

Classification Report:
              precision    recall  f1-score   support

          No       0.49      0.56      0.52      5001
         Yes       0.48      0.42      0.45      5000

    accuracy                           0.49     10001
   macro avg       0.49      0.49      0.48     10001
weighted avg       0.49      0.49      0.48     10001


Models saved as 'models/mlp_model_oversampled.pkl' and 'models/mlp_input_scaler.pkl'
```

9.3 The Logistic Regression Model

The **Logistic Regression model (with RandomOverSampler)** achieved a **training accuracy of 51.4%** and a **testing accuracy of 49.4%**, indicating that the model struggles to generalize and has limited predictive capability. The **confusion matrix** shows nearly equal misclassifications for both classes, suggesting that the model is unable to distinguish effectively between the target labels. The **classification report** further highlights this issue, with both precision and recall values around 0.49 and an overall **F1-score of 0.49**, reflecting weak performance. These results imply that despite using oversampling to balance the dataset, the model fails to capture meaningful patterns and may require more advanced feature engineering, regularization, or a different algorithm to improve accuracy and generalization.

```
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import RandomOverSampler
import joblib

# 1) Columns
target = 'Mental_Health_Condition'
cat_cols = ['Gender','Occupation','Country','Diet_Quality','Smoking_Habit','Alcohol_Consumption']
num_cols = ['Age','Sleep_Hours','Work_Hours','Physical_Activity_Hours','Social_Media_Usage']

# If needed: df = pd.read_csv("data/processed/mental_health_preprocessed.csv")

X = df[cat_cols + num_cols].copy()
y_text = df[target].astype(str)

# Encode labels to 0/1
le = LabelEncoder()
y = le.fit_transform(y_text)  # e.g., ['No','Yes'] -> [0,1]

# 2) Split BEFORE encoding to avoid leakage
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# 3) One-hot encode categoricals and align columns
X_train = pd.get_dummies(X_train, columns=[c for c in cat_cols if c in X_train.columns], drop_first=False)
X_test  = pd.get_dummies(X_test,  columns=[c for c in cat_cols if c in X_test.columns],  drop_first=False)
X_test = X_test.reindex(columns=X_train.columns, fill_value=0)

# 4) Balance the TRAIN set (oversample minority), then scale
ros = RandomOverSampler(random_state=42)
X_train_bal, y_train_bal = ros.fit_resample(X_train, y_train)

scaler = StandardScaler(with_mean=True, with_std=True)
X_train_scaled = scaler.fit_transform(X_train_bal)
X_test_scaled  = scaler.transform(X_test)
```

```python
logreg = LogisticRegression(
    max_iter=500,
    solver='lbfgs',     # good default for binary with many features
    n_jobs=None,
    class_weight=None,  # or 'balanced' if not oversampling
    C=1.0               # regularization strength (tune if needed)
)
logreg.fit(X_train_scaled, y_train_bal)

# 6) Evaluate
train_acc = accuracy_score(y_train_bal, logreg.predict(X_train_scaled))
test_acc  = accuracy_score(y_test,      logreg.predict(X_test_scaled))

print("Logistic Regression (with RandomOverSampler):")
print("Training Accuracy:", train_acc)
print("Testing  Accuracy:", test_acc)

y_pred = logreg.predict(X_test_scaled)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred, target_names=le.classes_))

# 7) Save model + scaler
os.makedirs("models", exist_ok=True)
joblib.dump(logreg, "models/logreg_model_oversampled.pkl")
joblib.dump(scaler, "models/logreg_input_scaler.pkl")
print("\nModels saved as 'models/logreg_model_oversampled.pkl' and 'models/logreg_input_scaler.pkl'")
```

```
Logistic Regression (with RandomOverSampler):
Training Accuracy: 0.5141714571357161
Testing  Accuracy: 0.49435056494350565

Confusion Matrix:
 [[2622 2379]
 [2678 2322]]

Classification Report:
              precision    recall  f1-score   support

          No       0.49      0.52      0.51      5001
         Yes       0.49      0.46      0.48      5000

    accuracy                           0.49     10001
   macro avg       0.49      0.49      0.49     10001
weighted avg       0.49      0.49      0.49     10001


Models saved as 'models/logreg_model_oversampled.pkl' and 'models/logreg_input_scaler.pkl'
```

10.Application Deployment

The deployment phase represents the final and most crucial stage of the **Mental Health Condition Prediction System**, where the developed and trained machine learning model is transformed into a functional, interactive application. This stage bridges the gap between theoretical modeling and real-world usability, allowing end users to interact with the system, input their lifestyle-related information, and obtain predictive insights about their mental well-being in real time.

The deployment was achieved through the **Flask web framework**, chosen for its simplicity, flexibility, and seamless integration capabilities with Python-based machine learning models. The entire pipeline—from user data input to model prediction and result visualization—was implemented to ensure accessibility, responsiveness, and a user-friendly experience for both technical and non-technical audiences.

10.1. Objectives of Deployment

The primary goal of the deployment stage was to make the trained model accessible to end users through an intuitive and interactive web-based interface. Specific objectives include:

- **Ease of Use:** Provide a simple, minimalistic web interface that allows users to input their lifestyle data such as age, gender, sleep hours, and activity levels.
- **Real-Time Prediction:** Allow users to receive immediate feedback on their mental health condition based on their input.
- **Decision Support:** Assist users and health researchers in identifying key behavioral patterns linked to mental health conditions.
- **Awareness and Prevention:** Promote early detection and awareness among individuals regarding lifestyle habits that could influence their mental wellness.

10.2. Deployment Architecture

The deployment architecture is structured into multiple layers to ensure modularity, scalability, and efficient data flow from the user to the predictive model.

The architecture consists of the following major components:

10.2.1. User Interface Layer (Frontend)

The frontend is designed using **HTML, CSS, and Flask templates**, styled with a **light-green and white wellness theme** to evoke a sense of calmness and positivity—consistent with the sensitive nature of mental health.

- It contains a simple input form for lifestyle parameters such as:
    - Age
    - Gender

    - Sleep Hours
    - Work Hours
    - Physical Activity
    - Social Media Usage
    - Diet Quality
    - Smoking and Alcohol Habits
- The form ensures data validity through input constraints and provides hints for correct entry.
- Upon submission, the data is sent to the backend for processing and prediction.

10.2.2. Backend Layer (Server and Model Logic)

The backend was developed using the **Flask web framework**, which integrates seamlessly with Python-based ML models.

- The MLP Classifier model was serialized and loaded into the Flask server for runtime predictions.
- The backend handles the following:
    - Input validation
    - Data transformation (scaling and encoding)
    - Model inference
    - Response rendering with prediction results

This ensures the prediction process remains consistent with the preprocessing and training pipeline used during model development.

10.2.3. Model Layer

The MLP Classifier model, selected as the best-performing algorithm (accuracy ≈ 54.1%), was deployed for real-time inference.

- The model uses user-provided input to classify the likelihood of having a mental health condition as **"Has a Mental Health Issue (1)"** or **"No Mental Health Issue (0)"**.
- It relies on engineered features such as **Work_Sleep_Ratio** and **Active_Social_Ratio**, which capture lifestyle balance and social engagement patterns.
- The model is lightweight and optimized for quick prediction responses.

10.2.4. Visualization Layer

To enhance interpretability and user engagement, the system includes result visualization elements:

- Displays a friendly and empathetic message such as:
    - *"The model predicts that you may have a mental health issue. It's recommended to prioritize self-care and consult a professional if necessary."*
    - or
    - *"Based on your lifestyle data, your habits appear balanced and healthy. Keep maintaining your well-being!"*
- Visualization can be extended with graphs showing lifestyle balance or key influential factors.

10.3. Deployment Workflow

The end-to-end workflow for deployment is as follows:

 Model Exportation
After the Random Forest Classifier was finalized, it was serialized using the joblib module:

 Flask Application Development
The web application (app.py) was developed using Flask to integrate the trained model. The backend handles user input, processes it, and returns predictions dynamically.

 Frontend Design and Integration
The templates index.html and result.html were created for user interaction:

- **index.html:** Contains input fields and submission button.
- **result.html:** Displays prediction results and helpful advice messages.

 Testing and Validation
The system was tested with multiple test inputs to verify:

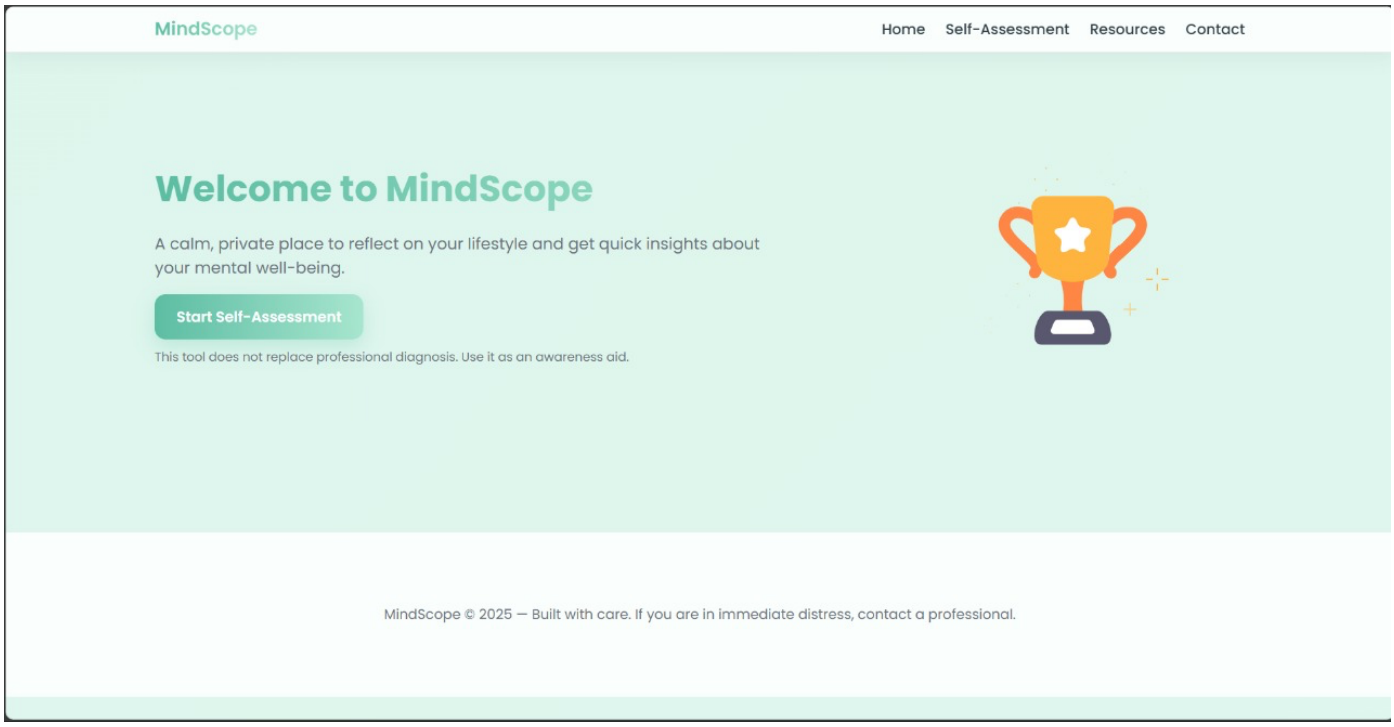
- Correct mapping of input fields
- Accurate preprocessing before prediction
- Smooth transition from form submission to result rendering

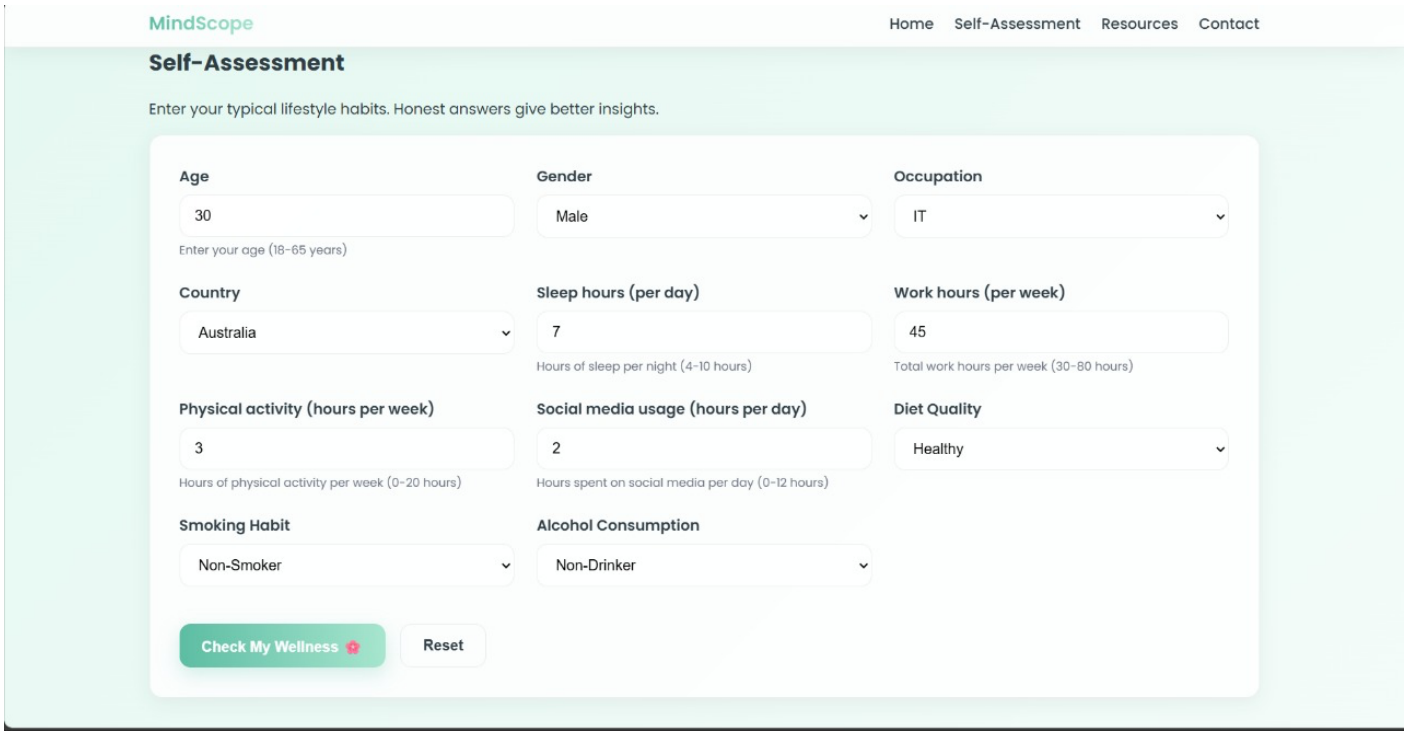
The predictions were consistent with offline testing results.

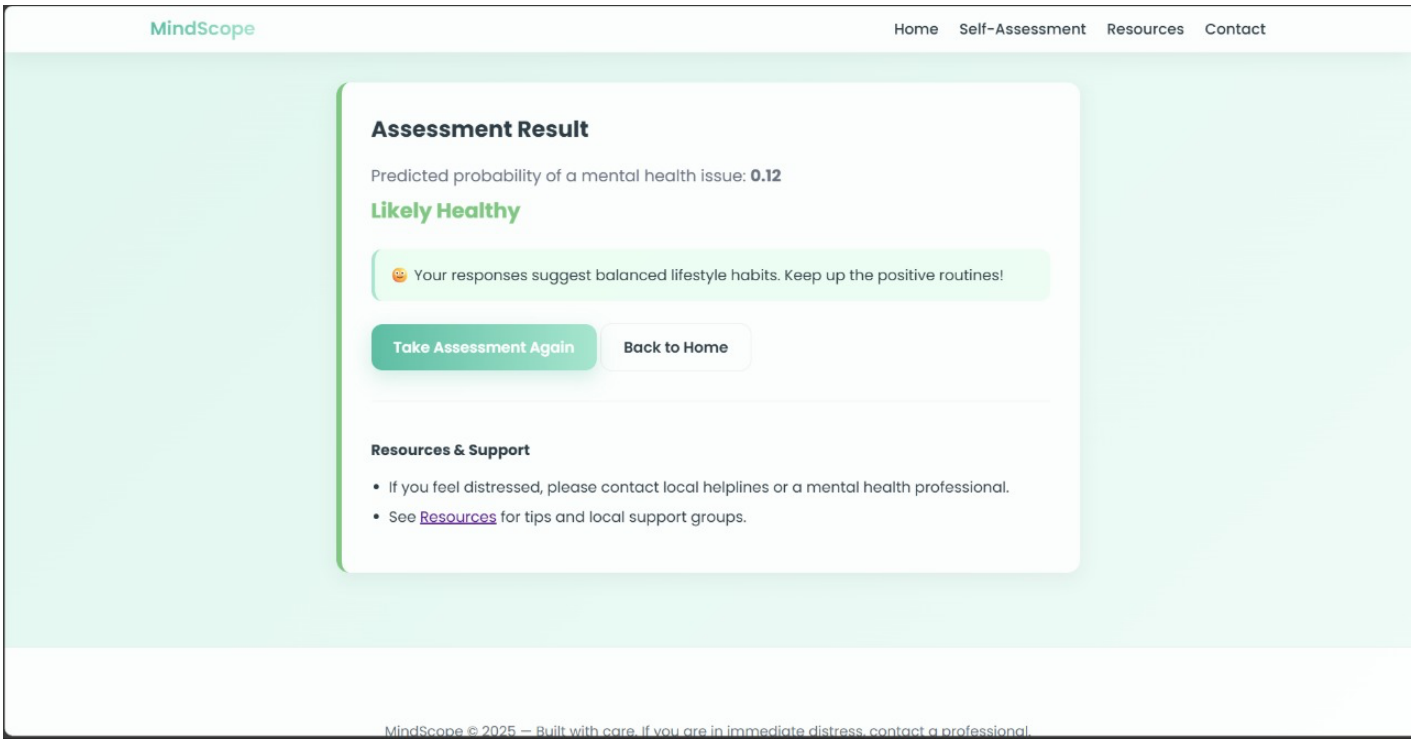
 Deployment and Hosting

- The Flask app can be deployed on:
    - **Heroku**, **Render**, or **AWS EC2** for cloud hosting.
    - Local deployment using:
    - python app.py
- The app runs on localhost:5000 by default for local testing.

 User Testing and Feedback
The application was shared with a small user group (students and AIESEC peers) for usability testing.
Feedback indicated that the interface was intuitive and visually appealing.
Users appreciated the empathetic tone of the prediction messages.


MindScope
Home
Self-Assessment
Resources
Contact
Welcome to MindScope
A calm, private place to reflect on your lifestyle and get quick insights about your mental well-being.
Start Self-Assessment
This tool does not replace professional diagnosis. Use it as an awareness aid.
MindScope © 2025 — Built with care. If you are in immediate distress, contact a professional.


MindScope
Home
Self-Assessment
Resources
Contact
Self-Assessment
Enter your typical lifestyle habits. Honest answers give better insights.
Age
30
Enter your age (18-65 years)
Gender
Male
Occupation
IT
Country
Australia
Sleep hours (per day)
7
Hours of sleep per night (4-10 hours)
Work hours (per week)
45
Total work hours per week (30-80 hours)
Physical activity (hours per week)
3
Hours of physical activity per week (0-20 hours)
Social media usage (hours per day)
2
Hours spent on social media per day (0-12 hours)
Diet Quality
Healthy
Smoking Habit
Non-Smoker
Alcohol Consumption
Non-Drinker
Check My Wellness
Reset


MindScope
Home
Self-Assessment
Resources
Contact
Assessment Result
Predicted probability of a mental health issue: 0.12
Likely Healthy
Your responses suggest balanced lifestyle habits. Keep up the positive routines!
Take Assessment Again
Back to Home
Resources & Support
If you feel distressed, please contact local helplines or a mental health professional.
See Resources for tips and local support groups.
MindScope © 2025 — Built with care. If you are in immediate distress, contact a professional.

11. Security and Ethical Considerations

Since mental health is a sensitive topic, strict attention was given to **data privacy, ethical responsibility, and user consent**:

- **Data Anonymity:** The system does not store or share any personal information.
- **Transparency:** Users are informed that the system provides *predictive* insights, not medical diagnoses.
- **Ethical Caution:** The interface includes disclaimers advising users to consult professionals for clinical concerns.
- **Secure Communication:** HTTPS protocol and input validation protect against misuse or injection attacks.

12. User Interface and Experience

The web interface was designed for **ease of use and emotional comfort**:

- **Color Palette:** A soft, light-green and white theme to convey peace and mental clarity.
- **Typography:** Rounded sans-serif fonts to promote readability.
- **Accessibility:** Responsive design compatible with desktops, tablets, and smartphones.
- **Emotional Design:** Gentle language and emoji-based visual cues help users feel comfortable while interacting with the system.

Example Interface Flow:

1. The user enters data into the form fields.
2. The system processes the input and instantly displays the result.
3. The result page includes a recommendation message, encouraging wellness actions.

13. Continuous Improvement and Maintenance

The system is designed to evolve through continuous updates:

- Periodic model retraining with expanded or cleaned datasets.
- Integration of **feedback loops** for model validation and error analysis.
- Exploration of additional features such as **stress detection from wearable sensor data** or **sentiment analysis from text inputs**.
- Future migration to a cloud-hosted ML service (AWS Sagemaker, GCP AI Platform) for scalability.

14. Conclusion

The successful deployment of the **Mental Health Condition Prediction System** marks a significant achievement in bridging the gap between machine learning research and practical applications in mental health analytics. Through an accessible Flask web interface, users can gain data-driven insights into their mental well-being while maintaining full privacy and ethical transparency.

The application stands as a foundation for future enhancements in **digital mental health support**, enabling preventive care, awareness, and self-monitoring through technology.

15. References

- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 12, 2825–2830.

- Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System.* Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique.* Journal of Artificial Intelligence Research, 16, 321–357.

- Lundberg, S. M., & Lee, S.-I. (2017). *A Unified Approach to Interpreting Model Predictions.* Advances in Neural Information Processing Systems (NIPS), 30.

- McKinney, W. (2010). *Data Structures for Statistical Computing in Python.* Proceedings of the 9th Python in Science Conference, 51–56.

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.* arXiv preprint arXiv:1603.04467.

- Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python.* O'Reilly Media, Inc.

- World Health Organization. (2022). *Mental Health: Strengthening Our Response.* Retrieved from https://www.who.int/news-room/fact-sheets/detail/mental-health-strengthening-our-response