Mental Health Analyzer

Submitted by

T.MohanRam

VU21CSEN0102093

Under the supervision

of

**Sandeep Varma.K**



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(Deemed to be University)

VISAKHAPATNAM

|  |  |  |
| --- | --- | --- |
|  | Table Of Contents | Page Number |
| 1 | Abstract | 3 |
| 2 | Introduction | 4 |
| 3 | Background And Related Work | 6 |
| 4 | Advantages And Applications | 6 |
| 5 | Challenges And Limitations | 8 |
| 6 | Solution | 10 |
| 7 | Implementation | 11 |
| 8 | Lesson Learned and Self Reflection | 14 |
| 9 | Conclusions | 15 |
| 10 | References | 16 |

**Title: Exploring Predictive Analysis for Mental Health Assessment**

**Abstract:**

This study delves into the dynamics influencing mental health prediction through data analytics methodologies. Leveraging datasets obtained from self-reported mental health assessments, encompassing diverse attributes including personal characteristics, lifestyle factors, and emotional well-being indicators, the study employs advanced analytical techniques to uncover underlying patterns. Python libraries including Pandas, NumPy, Plotly Express, and Scikit-learn are utilized for data processing, exploratory data analysis, and predictive modeling.

Exploratory data analysis is conducted to discern correlations and trends among various factors affecting mental health. Visualization tools such as Plotly Express are employed to reveal significant associations between lifestyle choices, emotional states, and mental health outcomes. Additionally, statistical models are applied to elucidate relationships, with logistic regression highlighting statistically significant predictors of mental health status.

Furthermore, machine learning methodologies are implemented to construct a predictive model for mental health assessment. Utilizing algorithms such as Random Forest or Gradient Boosting, the model integrates personal attributes, lifestyle factors, and emotional indicators to forecast mental health conditions accurately. Model training and evaluation are conducted utilizing cross-validation techniques, ensuring robustness and generalizability.

The developed predictive model exhibits promising performance, enabling individuals to assess their mental health based on pertinent factors. This research contributes valuable insights into proactive mental health management, early intervention strategies, and personalized well-being approaches.

**1. Introduction:**

In today's fast-paced society, mental health has become a significant concern, with increasing awareness of the importance of proactive self-assessment and early intervention. This project embarks on a transformative journey to harness the power of data analytics, exploring a multifaceted approach encompassing exploratory data analysis (EDA), predictive modeling, data visualization, and user-friendly interface creation.

**1. Exploratory Data Analysis (EDA):**

The foundation of this project lies in the thorough exploration of self-reported mental health assessment datasets to uncover hidden patterns, trends, and indicators. Through comprehensive EDA techniques, participants delve deep into the provided datasets, scrutinizing variables and relationships to gain nuanced insights into the underlying factors affecting mental health.

**2. Statistical Modelling:**

Statistical models serve as powerful tools for uncovering correlations and predicting mental health outcomes based on self-reported data. Participants utilize techniques such as logistic regression to elucidate the relationships between various factors, such as personal attributes, lifestyle choices, and emotional well-being indicators, and mental health status.

**3. Predictive Modelling with Machine Learning:**

Machine learning algorithms offer advanced capabilities for predictive modeling in mental health assessment, enabling individuals to anticipate potential mental health conditions based on various factors. Through the implementation of algorithms such as Random Forest or Gradient Boosting, participants develop a predictive model to forecast mental health outcomes accurately. Model training and evaluation are conducted utilizing cross-validation techniques to ensure reliability and generalizability.

**4. Data Visualization and Interpretation:**

Effective communication of insights derived from data analysis is essential for empowering individuals to assess their mental health effectively. Participants employ interactive data visualization tools to create visually compelling representations of complex information, facilitating a deeper understanding of the relationships between personal attributes, lifestyle factors, emotional indicators, and mental health outcomes.

**5. User-Friendly Interface Development:**

The project includes the development of a user-friendly interface that enables individuals to input their personal information, lifestyle choices, and emotional well-being indicators for mental health assessment. The interface provides intuitive visualizations and interpretable results, empowering users to make informed decisions about their mental well-being.

**Conclusion:**

This project represents a holistic exploration of data analytics methodologies for mental health assessment, ranging from exploratory analysis to predictive modeling and user-friendly interface development. By leveraging these techniques, individuals can gain invaluable insights into their mental health status, empowering them to take proactive steps towards well-being and seek appropriate support when needed.

**6.References:**

Previous research in the field of mental health assessment and predictive analytics provides a foundation for this project, emphasizing the importance of data-driven approaches in promoting well-being and early intervention strategies. Techniques such as logistic regression and machine learning algorithms have been successfully applied in various contexts, paving the way for proactive mental health management and personalized interventions.

窗体顶端

窗体底端

**7. ⁠Implementation:**

*7.1 Data import and Explorations:*

import pandas as pd

from flask import Flask, render\_template, request, redirect, url\_for

import pickle

app = Flask(\_\_name\_\_, static\_url\_path='/static')

# Load the pickled machine learning model

with open('model.pkl', 'rb') as model\_file:

    model = pickle.load(model\_file)

@app.route('/', methods=['GET', 'POST'])

def index():

    if request.method == 'POST':

        # Get input values from the form

        age = float(request.form['Age'])

        g = str(request.form['Gender'])

        if len(g)==4:

            gender=1

        else:

            gender=0

        screen\_time = float(request.form['ScreenTime'])

        active\_lifestyle = float(request.form['ActiveLifestyle'])

        sleep\_time = float(request.form['SleepTime'])

        stress\_levels = float(request.form['StressLevels'])

        mood = float(request.form['Mood'])

        social\_relationship = float(request.form['SocialRelationship'])

        # Make predictions using the loaded model

        inputs = [[age, gender, screen\_time, active\_lifestyle, sleep\_time, stress\_levels, mood, social\_relationship]]

        p = model.predict(inputs)[0]

        if p==0:

            prediction='Mental Health Status: Severe '

        if p==1:

            prediction='Mental Health Status: Moderate '

        if p==2:

            prediction='Mental Health Status: Mild '

        if p==3:

            prediction='Mental health status: Good '

        if p==4:

            prediction='Mental health status: Optimal '

        # Redirect to the results page with the prediction

        return redirect(url\_for('results', prediction=prediction))

    return render\_template('index.html')

@app.route('/results/<prediction>')

def results(prediction):

    return render\_template('results.html', prediction=prediction)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

* *import pandas as pd:* Imports the Pandas library, which is used for data manipulation and analysis, and aliases it as `pd` for easier referencing in the code.
* *import numpy as np:* Imports the NumPy library, which is used for numerical computing and working with arrays, and aliases it as `np` for easier referencing in the code.
* *import plotly.express as px:* Imports the Plotly Express module from the Plotly library, which is used for creating interactive plots and charts, and aliases it as `px` for easier referencing in the code.
* *import statsmodels.api as sm:* Imports the statsmodels library, which is used for statistical modelling and hypothesis testing, and aliases it as `sm` for easier referencing in the code.

* *Data Import:* Loads the dataset "deliverytime - train.csv" into a Pandas DataFrame named data.
* *Data Exploration:* Displays the first few rows of the dataset and provides information about its structure using the head() and info() functions.

*7.2 Filling the Missing values with preprocessing techniqes*

Age,Gender,Screen Time (hours),Active Lifestyle (Scale 1-5),Sleep Time (hours),Stress levels (Scale 1-5),Mood (Scale 1-5),Social Relationship scale (Scale 1-5),Label

46,Male,8.3,2,1.6,4,5,1,1

33,Male,2.1,3,8.4,2,3,5,3

32,Female,2.6,5,5.9,1,1,3,3

31,Female,1.5,4,7.9,2,3,4,3

46,Male,5.7,1,1.2,1,1,4,0

38,Male,7.8,1,5.8,1,5,4,1

18,Female,6.8,1,4.5,4,4,2,0

19,Male,3.6,3,3.0,4,2,3,2

36,Female,5.1,1,5.2,5,4,2,1

44,Male,8.1,2,8.3,4,5,3,2

31,Female,7.0,2,7.9,1,2,2,1

27,Female,5.8,1,7.0,5,5,1,1

17,Male,2.8,4,3.6,5,4,1,2

50,Female,2.2,5,1.9,4,1,1,1

27,Female,1.5,5,4.6,3,3,5,3

20,Female,7.3,1,6.8,3,5,4,1

32,Male,2.3,5,6.5,1,1,4,3

26,Female,2.4,5,2.9,4,2,2,2

33,Female,5.6,2,8.4,1,1,1,1

21,Female,5.1,2,8.5,1,5,5,2

40,Male,4.4,5,6.8,4,4,3,3

43,Male,7.0,2,7.8,5,2,3,1

27,Female,1.6,5,8.5,5,1,3,3

48,Female,1.1,3,7.7,5,2,3,3

23,Male,6.0,2,4.2,3,4,4,1

16,Female,2.6,4,8.7,5,1,3,3

17,Female,6.3,1,6.4,3,5,4,1

44,Female,5.1,2,4.4,5,4,1,1

39,Female,9.0,2,6.9,1,1,3,1

e.

Age,Gender,Screen Time (hours),Active Lifestyle (Scale 1-5),Sleep Time (hours),Stress levels (Scale 1-5),Mood (Scale 1-5),Social Relationship scale (Scale 1-5),Label

46,Male,8.3,2,1.6,4,5,1,1

33,Male,2.1,3,8.4,2,3,5,3

32,Female,2.6,5,5.9,1,1,3,3

31,Female,1.5,4,7.9,2,3,4,3

46,Male,5.7,1,1.2,1,1,4,0

38,Male,7.8,1,5.8,1,5,4,1

18,Female,6.8,1,4.5,4,4,2,0

19,Male,3.6,3,3.0,4,2,3,2

36,Female,5.1,1,5.2,5,4,2,1

44,Male,8.1,2,8.3,4,5,3,2

31,Female,7.0,2,7.9,1,2,2,1

27,Female,5.8,1,7.0,5,5,1,1

17,Male,2.8,4,3.6,5,4,1,2

50,Female,2.2,5,1.9,4,1,1,1

27,Female,1.5,5,4.6,3,3,5,3

20,Female,7.3,1,6.8,3,5,4,1

32,Male,2.3,5,6.5,1,1,4,3

26,Female,2.4,5,2.9,4,2,2,2

33,Female,5.6,2,8.4,1,1,1,1

21,Female,5.1,2,8.5,1,5,5,2

40,Male,4.4,5,6.8,4,4,3,3

43,Male,7.0,2,7.8,5,2,3,1

27,Female,1.6,5,8.5,5,1,3,3

48,Female,1.1,3,7.7,5,2,3,3

23,Male,6.0,2,4.2,3,4,4,1

16,Female,2.6,4,8.7,5,1,3,3

17,Female,6.3,1,6.4,3,5,4,1

44,Female,5.1,2,4.4,5,4,1,1

39,Female,9.0,2,6.9,1,1,3,1

*Traing the model:*

import numpy as np

import pandas as pd

import pickle

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import  LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

import tensorflow as tf

data = pd.read\_csv("data.csv")

X = data.drop("Label", axis=1)

y = data["Label"]

label\_encoder = LabelEncoder()

X['Gender'] = label\_encoder.fit\_transform(X['Gender'])

X\_train, x\_test, Y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

print("X\_train shape:", X\_train.shape)

print("X\_test shape:", x\_test.shape)

print("y\_train shape:", Y\_train.shape)

print("y\_test shape:", y\_test.shape)

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_classifier.fit(X\_train, Y\_train)

y\_pred = rf\_classifier.predict(x\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

accuracy=accuracy\*100

print(f'Accuracy: {accuracy:.2f}')

f1 = f1\_score(y\_test, y\_pred,average='micro')

f1=f1\*100

print(f'F1 Score: {f1:.2f}')

with open('model.pkl', 'wb') as model\_file:

    pickle.dump(rf\_classifier, model\_file)

with open('model.pkl', 'rb') as file:

    model = pickle.load(file)

**Predecting the Results :**

As an AI language model, I can provide an overview of potential results and outcomes for the mental health prediction analyzer project based on the described methodology:

Exploratory Data Analysis (EDA) Results:

Identification of significant correlations between various personal attributes, lifestyle factors, emotional well-being indicators, and mental health outcomes.

Discovery of key trends and patterns in the data, such as certain lifestyle choices being associated with higher or lower mental health scores.

Statistical Modeling Results:

Determination of statistically significant predictors of mental health status through logistic regression analysis.

Insights into the relative importance of different factors in predicting mental health outcomes, potentially highlighting areas for intervention or support.

Predictive Modeling Results:

Development of a predictive model (e.g., Random Forest, Gradient Boosting) to forecast mental health conditions based on input variables.

Evaluation of the model's performance using metrics such as accuracy, precision, recall, and F1-score.

Generation of predictions for mental health status based on individual input data, providing users with actionable insights into their well-being.

Data Visualization and Interpretation Results:

Creation of interactive visualizations depicting relationships between personal attributes, lifestyle factors, emotional indicators, and mental health outcomes.

Clear presentation of model predictions and their associated confidence levels, facilitating interpretation by users.

User-Friendly Interface Results:

Development of an intuitive interface enabling users to input their information and receive personalized mental health assessments.

Implementation of features for visualizing prediction results, providing explanations for predictions, and offering suggestions for improving mental well-being.

import pickle

import numpy as np

with open('model.pkl', 'rb') as model\_file:

    loaded\_model = pickle.load(model\_file)

age=int(input("Enter the age: "))

g=str(input("Enter the Gender: "))

if g=='male' or 'Male':

    gender=1

elif g=='female' or 'Female':

    gender=0

screen\_time= float(input("Enter the first input: "))

Active\_lifestyle = float(input("Enter the second input: "))

sleep\_time = float(input("Enter the third input: "))

stress\_level = float(input("Enter the fourth input: "))

mood = float(input("Enter the fivth input: "))

social\_relation = float(input("Enter the sixth input: "))

user\_input\_data = np.array([[age,gender,screen\_time, Active\_lifestyle,sleep\_time,stress\_level, mood,social\_relation]])

model\_prediction = loaded\_model.predict(user\_input\_data)

if model\_prediction[0]==0:

    print("Mental Health Status: Severe")

elif model\_prediction[0]==1:

    print("Mental Health Status: Moderate")

elif model\_prediction[0]==2:

    print("Mental Health Status: Mild")

elif model\_prediction[0]==3:

    print("Mental Health Status: Good")

else:

    print("Mental Health Status: Optimal")

*Key Parameters:*

1. Sleeping Hours
2. Screen Time
3. Stress Levels
4. Social Relationship
5. Active life style

*Importing Libraries:*The code imports the train\_test\_split function from the model\_selection module in the scikit-learn library.

*Data Preparation:*

x is created as a NumPy array, containing the predictor variables: delivery person age, delivery person ratings, and distance.

y is created as a NumPy array, containing the target variable: time taken for delivery.

*Train-Test Split:*

train\_test\_split(x, y, test\_size=0.2, random\_state=42) function is called to split the dataset into training and testing sets.

xtrain and ytrain represent the features and target variable for the training set, respectively.

xtest and ytest represent the features and target variable for the testing set, respectively.

test\_size=0.10 specifies that 10% of the data should be allocated for testing, and the remaining 90% is used for training.

random\_state=42 ensures reproducibility by fixing the random seed, meaning the same split will be obtained each time the code is run.

*Outcome:*

The dataset is now divided into two subsets: one for training the machine learning model (xtrain and ytrain) and the other for evaluating its performance (xtest and ytest).

This separation allows for training the model on a subset of the data and assessing its performance on unseen data, helping to gauge the model's generalisation capabilities.

* *Data Splitting:* Splits the dataset into training and testing sets using train\_test\_split() from scikit-learn.

X\_train, x\_test, Y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

print("X\_train shape:", X\_train.shape)

print("X\_test shape:", x\_test.shape)

print("y\_train shape:", Y\_train.shape)

print("y\_test shape:", y\_test.shape)

Our Model: “Randomforest”

The Random Forest model is a powerful ensemble learning technique used for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputting the mode (classification) or mean (regression) prediction of the individual trees.

Here are some key aspects of the Random Forest model:

Ensemble Learning: Random Forest belongs to the ensemble learning family, which combines multiple models to improve performance and robustness over a single model. It aggregates predictions from multiple decision trees to make the final prediction.

Decision Trees: Each tree in the Random Forest is built using a random subset of the features and a random subset of the training data. This randomness helps to decorrelate the individual trees, reducing the risk of overfitting.

Bootstrap Aggregating (Bagging): Random Forest employs a technique called bagging, where each tree is trained on a bootstrap sample of the training data (sampling with replacement). This further enhances the diversity among the trees.

Feature Importance: Random Forest can provide insights into feature importance, indicating which features are most influential in making predictions. This can be valuable for understanding the underlying factors driving the target variable.

Robustness to Overfitting: Random Forest tends to be less prone to overfitting compared to individual decision trees, especially when hyperparameters like the number of trees and maximum depth are properly tuned.

Scalability: Random Forest can handle large datasets efficiently and is parallelizable, making it suitable for high-dimensional data with many features.

Interpretability: While Random Forest provides feature importance scores, the individual trees are typically not interpretable in the same way as simpler models like linear regression. However, ensemble techniques like Random Forest often provide better predictive performance at the expense of interpretability.

Random Forest is a popular choice for various machine learning tasks due to its robustness, ease of use, and ability to handle complex datasets. It's often considered a go-to model for classification and regression problems in both research and industry applications

Model Configuration:

model.compile(optimizer='adam', loss='mean\_squared\_error'): Configures the model for training by specifying the optimizer and loss function.

optimizer='adam': Random Forest doesn't require an optimizer like neural networks do. Instead, it typically uses techniques like bagging and feature randomization.

loss='mean\_squared\_error': Random Forest doesn't directly minimize a loss function during training like neural networks. Instead, it optimizes based on reducing impurity (e.g., Gini impurity for classification or mean squared error for regression) during tree construction.

Training the Model:

model.fit(xtrain, ytrain, batch\_size=1, epochs=9): Trains the compiled model using the specified training data.

xtrain: The input features for training (features matrix).

ytrain: The target variable for training (labels vector).

batch\_size=1: Random Forest doesn't use batch training like neural networks. Each tree is trained on the entire training set.

epochs=9: Random Forest doesn't have epochs like neural networks. The number of trees (or iterations) is specified by the n\_estimators parameter.

Training Process:

The Random Forest model is trained using the provided input features (xtrain) and target variable (ytrain).

Each decision tree in the forest is trained independently using bootstrapped samples of the training data and feature randomization.

The number of trees (or epochs) is specified by the n\_estimators hyperparameter. Increasing the number of trees can improve model performance but may also increase computational cost.

During training, each tree seeks to minimize impurity in the nodes, such as Gini impurity for classification tasks or mean squared error for regression tasks.

The final predictions are typically based on aggregating the predictions of all the trees in the forest (e.g., averaging for regression or voting for classification).

Outcome:

After training, the Random Forest model is ready to make predictions on new, unseen data.

The trained model's performance can be evaluated using appropriate metrics such as accuracy, precision, recall, F1-score, or mean squared error, depending on the task (classification or regression).

Random Forests are powerful ensemble learning models known for their robustness and effectiveness in a wide range of tasks, including classification and regression.

sleep\_hours = float(input("Hours of Sleep: ")) # User inputs the number of hours of sleep as a floating-point number.

stress\_level = float(input("Stress Level (0-10): ")) # User inputs their stress level on a scale of 0 to 10.

screen\_time = float(input("Screen Time (hours): ")) # User inputs their screen time in hours.

social\_relationship = int(input("Social Relationship Score (1-5): ")) # User inputs their social relationship score on a scale of 1 to 5.

# Feature Array

import numpy as np

features = np.array([[sleep\_hours, stress\_level, screen\_time, social\_relationship]]) # Creates a NumPy array containing the user-inputted values.

# Prediction

predicted\_mental\_health\_status = model.predict(features) # Uses the trained model to predict the mental health status based on the provided features.

# Display Prediction

print("Predicted Mental Health Status: ", predicted\_mental\_health\_status) # Prints the predicted mental health status to the console.

In this adaptation:

The user inputs information related to factors affecting mental health, including hours of sleep, stress level, screen time, and social relationship score.

The input values are used to create a NumPy array features structured to match the expected input shape for the model.

The trained model is then used to predict the mental health status based on the provided features.

The predicted mental health status is displayed to the user as the final output.

sleep\_hours = float(input("Hours of Sleep: ")) # User inputs the number of hours of sleep as a floating-point number.

stress\_level = float(input("Stress Level (0-10): ")) # User inputs their stress level on a scale of 0 to 10.

screen\_time = float(input("Screen Time (hours): ")) # User inputs their screen time in hours.

social\_relationship = int(input("Social Relationship Score (1-5): ")) # User inputs their social relationship score on a scale of 1 to 5.

# Feature Array

import numpy as np

features = np.array([[sleep\_hours, stress\_level, screen\_time, social\_relationship]]) # Creates a NumPy array containing the user-inputted values.

# Prediction

predicted\_mental\_health\_status = model.predict(features) # Uses the trained model to predict the mental health status based on the provided features.

# Display Prediction

print("Predicted Mental Health Status: ", predicted\_mental\_health\_status) # Prints the predicted mental health status to the console.

In this adaptation:

The user inputs information related to factors affecting mental health, including hours of sleep, stress level, screen time, and social relationship score.

The input values are used to create a NumPy array features structured to match the expected input shape for the model.

The trained model is then used to predict the mental health status based on the provided features.

The predicted mental health status is displayed to the user as the final output.

**8. Lessons Learned and Self-Reflections:**

*8.1 Insights from Data Analysis:*

Undertaking the data analysis journey provided invaluable insights into the relationships between delivery variables. Exploring correlations between delivery person age, ratings, distance, and delivery time shed light on nuanced patterns within the data. Understanding these relationships highlighted the importance of considering multiple factors in predictive modelling and decision-making processes.

*8.2 Model Development and Evaluation:*

Developing and training machine learning models for predicting delivery times revealed the complexities involved in model selection and evaluation. Experimenting with different algorithms and techniques, such as linear regression and LSTM neural networks, underscored the need for a thorough understanding of model assumptions and limitations. Evaluating model performance through metrics like mean squared error provided valuable feedback for refining and optimising predictive models.

*8.3 Practical Applications and Real-World Considerations:*

Applying data analytics techniques to real-world scenarios uncovered practical challenges and considerations in the delivery industry. Analysing factors like delivery person age, ratings, and distance showcased the potential impact on delivery times and operational efficiency. Recognizing the significance of these insights in optimising delivery processes highlighted the practical applications of data analytics in enhancing business operations and customer satisfaction.

*8.4 Continuous Learning and Improvement:*

Engaging in machine learning projects reinforced the importance of continuous learning and skill development. Exploring new methodologies and technologies, such as data preprocessing techniqes for data filling and predicting algorithms for data predecting , expanded the toolkit for tackling analytical challenges. Embracing a growth mindset and seeking opportunities for learning and improvement remain essential for staying ahead in the rapidly evolving field of machine learnig algorithms.

**9. Conclusions:**

In conclusion, this project represents a comprehensive exploration of machine learning algorithms applied to the domain of mental health analysis and prediction. By analyzing factors such as screen time, sleeping hours, stress levels, and social relationships, valuable insights have been gained into the intricate relationships influencing mental well-being. Machine learning models, including Random Forest has enabled the development of predictive tools to forecast mental health status with reasonable accuracy.

The project has underscored the importance of data-driven decision-making in optimizing mental health assessment processes and enhancing overall well-being. By identifying key variables impacting mental health status and evaluating their effects through rigorous analysis and modeling, opportunities for intervention and support have been illuminated. These insights can inform personalized interventions and proactive strategies for mental health management.

Furthermore, the project highlights the significance of continuous learning and skill development in the field of machine learnig , particularly in the context of mental health. Engaging in hands-on exploration of diverse methodologies and technologies has expanded the toolkit for addressing mental health challenges and adapting to evolving trends. Embracing a growth mindset and a commitment to ongoing education will be essential for advancing the field of mental health analytics and improving outcomes for individuals.

Overall, this project demonstrates the transformative potential of machine learning in optimizing mental health assessment processes and promoting well-being. By harnessing the power of data to uncover actionable insights and inform personalized interventions, organizations and individuals alike can contribute to better mental health outcomes and foster a culture of well-being in society.

**10. References:**

* Google Sheets
* Visual Studio Code
* GitHub