Mental Health Prediction

SmartInternz

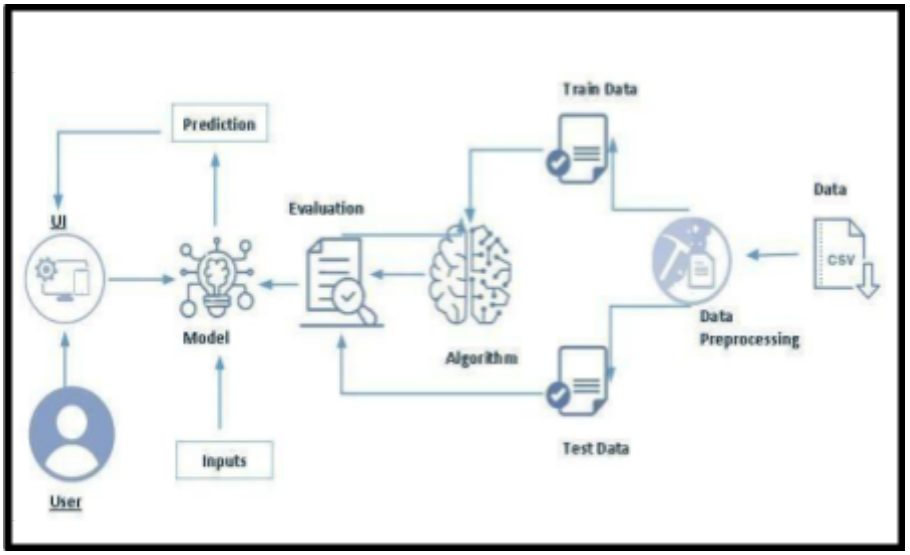
www.smartinternz.com

Mental Health Prediction Using Machine Learning

Mental Health disorders affect millions of people worldwide, impacting their emotional, psychological, and social well-being. Early detection and intervention are crucial to providing timely support and improving quality of life. However, many individuals remain undiagnosed or untreated due to lack of awareness, stigma, or limited access to mental health services. In recent years, the increasing prevalence of stress, anxiety, depression, and other mental health conditions has highlighted the need for proactive measures. Traditional diagnostic methods rely heavily on self-reported symptoms and clinical evaluations, which can be subjective or delayed. This is where predictive analytics can play a transformative role.

A Mental Health Prediction System leverages machine learning and data-driven approaches to assess risk factors, behavioral patterns, and psychological indicators that may signal potential mental health issues. By analyzing historical and real-time data—such as patient health records, lifestyle habits, social interactions, and biometric signals—the system can identify early warning signs and predict the likelihood of mental health disorders.

Technical Architecture:



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on

the UI To accomplish this, we have to complete all the activities listed below,

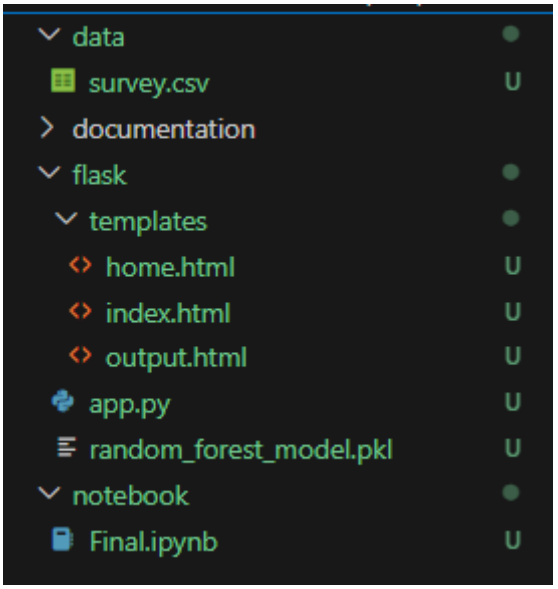
- Define Problem / Problem Understanding
    - Specify the business problem
    - Business requirements
    - Literature Survey
    - Social or Business Impact.
- Data Collection & Preparation
    - Collect the dataset
    - Data Preparation
- Exploratory Data Analysis
    - Descriptive statistical
    - Visual Analysis
- Model Building
    - Training the model in multiple algorithms
    - Testing the model
- Performance Testing & Hyperparameter Tuning
    - Testing model with multiple evaluation metrics
    - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
    - Save the best model
    - Integrate with Web Framework
- Project Demonstration & Documentation
    - Record explanation Video for project end to end solution
    - Project Documentation-Step by step project development procedure

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- ML Concepts
    - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
    - Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
    - Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
    - Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
    - KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
    - Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
    - Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- random_forest_model.pkl is our saved model. Further we will use this model for flask integration.
- Data Folder contains the Dataset used
- The Notebook file contains procedures for building the model.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the problem

The project aims to develop a predictive system for early detection of mental health conditions (e.g., depression, anxiety, or stress) using machine learning. With rising mental health concerns globally, timely intervention is critical. This system will analyze behavioral, demographic, and physiological data to identify at-risk individuals, enabling healthcare providers to offer proactive support.

Activity 2: Business requirements

- Data Accuracy: Use reliable, anonymized datasets (e.g., clinical records, wearable device data) to ensure prediction credibility.
- Scalability: Design the system to handle diverse user inputs (e.g., survey responses, activity logs) for broad applicability.
- Regulatory Compliance: Adhere to healthcare data privacy laws (e.g., HIPAA, GDPR) and ethical guidelines.
- User-Centric Design: Develop an intuitive interface for both clinicians and end-users (e.g., patients or caregivers).
- Explainability: Provide interpretable model outputs to support clinical decision-making.

Activity 3: Literature Survey

A comprehensive review of existing mental health prediction systems reveals:

- Techniques: Prior studies used ML models (e.g., Random Forests,Logistic Regression etc) on datasets like PHQ-9 surveys or social media activity.
- Gaps: Limited real-time prediction capabilities and bias in underrepresented populations.
- Opportunities: Integration of multimodal data (e.g., sleep patterns, speech analysis) for higher accuracy.

Activity 4: Social or Business Impact.

Social Impact :-

- Early Intervention: Reduce undiagnosed cases by flagging high-risk individuals.
- Stigma Reduction: Normalize mental health screening through accessible tools.

Business Model/Impact :-

- Healthcare Cost Savings: Lower treatment costs via preventive care.
- Product Potential: Commercial applications in telemedicine platforms or employer wellness programs

.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: https://www.kaggle.com/datasets/osmi/mental-health-in-tech-survey

As the dataset is downloaded. We read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image.

```
from google.colab import files
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import pickle
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

Activity 1.2: Read the Dataset

Our dataset format is .csv. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
df = pd.read_csv("survey.csv")
df.head()
```

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_health_consequence	phys_health_consequence
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	Often	6-25	...	Somewhat easy	No	No
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	Rarely	More than 1000	...	Don't know	Maybe	No
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Rarely	6-25	...	Somewhat difficult	No	No
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100	...	Somewhat difficult	Yes	Yes
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	Never	100-500	...	Don't know	No	No

N

Activity 2: Data Preparation

Now that we have an understanding of the dataset, the next step is to pre-process the collected data. The raw dataset may contain noise or inconsistencies, making it unsuitable for training a machine learning model directly. To improve model performance, it's important to clean the data effectively. This pre-processing stage typically involves:

- Handling missing values
- Handling outliers

Activity 2.1: Handling missing values

- For checking the null values, df.isna().any( ) function is used. To sum those null values we use .sum() function.

```
print(df.isnull().sum())
sns.heatmap(df.isnull(), cbar = False, cmap = "viridis")
plt.title("Missing Values")
plt.show()
```

```
Timestamp                      0
Age                            0
Gender                         0
Country                        0
state                        515
self_employed                 18
family_history                 0
treatment                      0
work_interfere               264
no_employees                   0
remote_work                    0
tech_company                   0
benefits                       0
care_options                   0
wellness_program               0
seek_help                      0
anonymity                      0
leave                          0
mental_health_consequence      0
phys_health_consequence        0
coworkers                      0
supervisor                     0
mental_health_interview        0
phys_health_interview          0
mental_vs_physical             0
obs_consequence                0
comments                    1095
dtype: int64
```

- Dropping the irrelevant columns like timestamp, comments, etc, and filling the missing values with the help of .fillna() function

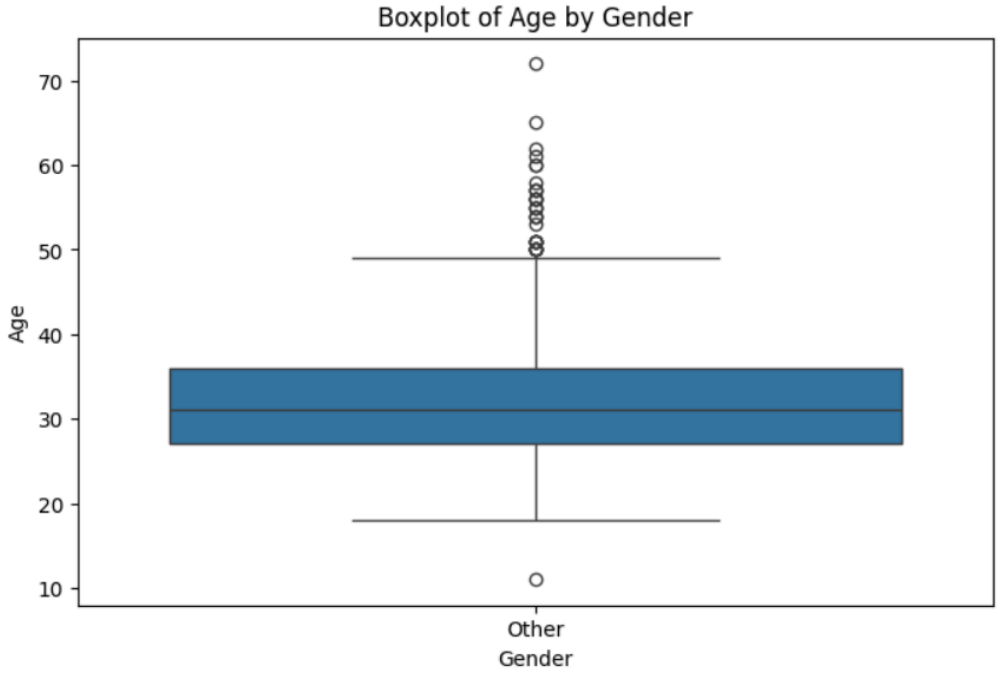
```
# Dropping columns that are either irrelevant for prediction or may induce bias (e.g., country)
# Also handling missing values in categorical columns

df.drop(["Timestamp", "Country","state","comments"],axis = 1, inplace = True)
df["work_interfere"] = df["work_interfere"].fillna("Unknown")
df["self_employed"] = df["self_employed"].fillna("No")
```

```
def clean_gender(gender):
    gender = str(gender).strip().lower()

    if any(x in gender for x in ['female', 'woman', 'f',"F"]):
        return 'Female'
    elif any(x in gender for x in ['male', 'man',"cis-man","m","M"]):
        return 'Male'
    else:
        return 'Other'

df['Gender'] = df['Gender'].apply(clean_gender)
```

Activity 2.2:

Handling Outliers

With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of age feature with some mathematical formula.



- From the below diagram, we could visualize that policy_annual_premium feature has outliers. Boxplot from seaborn library is used here.

- While outliers were identified and quantified, no treatment (removal or transformation) was applied at this stage. The decision to retain outliers was made to:
  - Preserve the original data distribution for initial exploratory analysis
  - Avoid premature data modification that might affect subsequent analyses
  - Allow for comprehensive evaluation of the raw dataset

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
df.describe(include = "all")
```

	Age	Gender	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	tech_company	benefits	...	leave	mental_health_consequence
count	1252.000000	1252	1252.000000	1252.000000	1252.000000	1252.000000	1252.000000	1252.000000	1252.000000	1252.000000	...	1252.000000	1252.000000
unique	NaN	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
top	NaN	Other	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
freq	NaN	1252	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
mean	32.059904	NaN	0.114217	0.390575	0.504792	2.337061	2.789137	0.297125	0.819489	1.052716	...	1.407348	0.849840
std	7.309669	NaN	0.318202	0.488074	0.500177	1.374008	1.737427	0.457174	0.384766	0.837052	...	1.507310	0.766906
min	11.000000	NaN	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000
25%	27.000000	NaN	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.000000	...	0.000000	0.000000
50%	31.000000	NaN	0.000000	0.000000	1.000000	3.000000	3.000000	0.000000	1.000000	1.000000	...	1.000000	1.000000
75%	36.000000	NaN	0.000000	1.000000	1.000000			1.000000	1.000000	2.000000	...	2.000000	1.000000
max	72.000000	NaN	1.000000	1.000000	1.000000		0	1.000000	1.000000	2.000000	...	4.000000	2.000000

11 rows × 24 columns

Here, the columns details of the dataset

```
df.columns
```

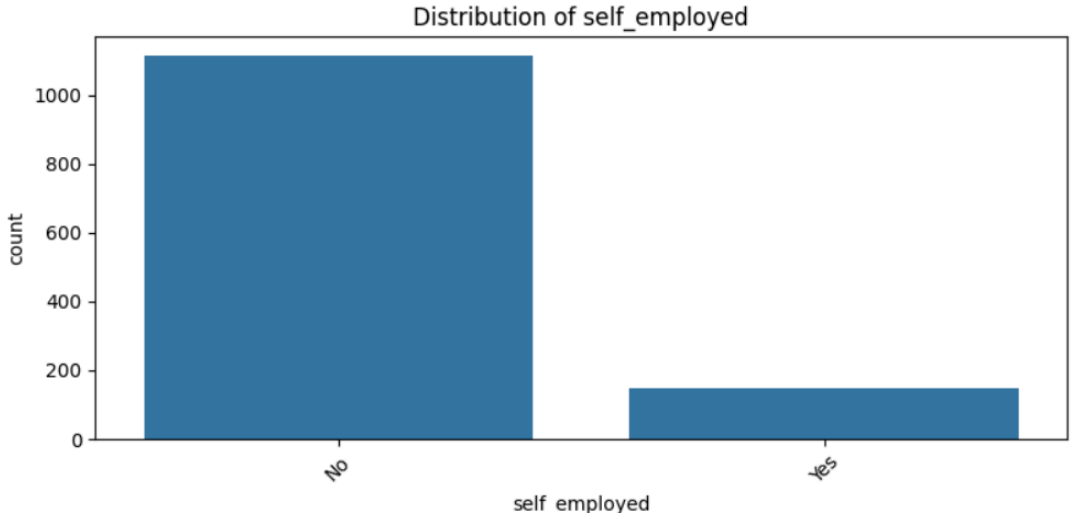
```
Index(['Timestamp', 'Age', 'Gender', 'Country', 'state', 'self_employed',
       'family_history', 'treatment', 'work_interfere', 'no_employees',
       'remote_work', 'tech_company', 'benefits', 'care_options',
       'wellness_program', 'seek_help', 'anonymity', 'leave',
       'mental_health_consequence', 'phys_health_consequence', 'coworkers',
       'supervisor', 'mental_health_interview', 'phys_health_interview',
       'mental_vs_physical', 'obs_consequence', 'comments'],
      dtype='object')
```

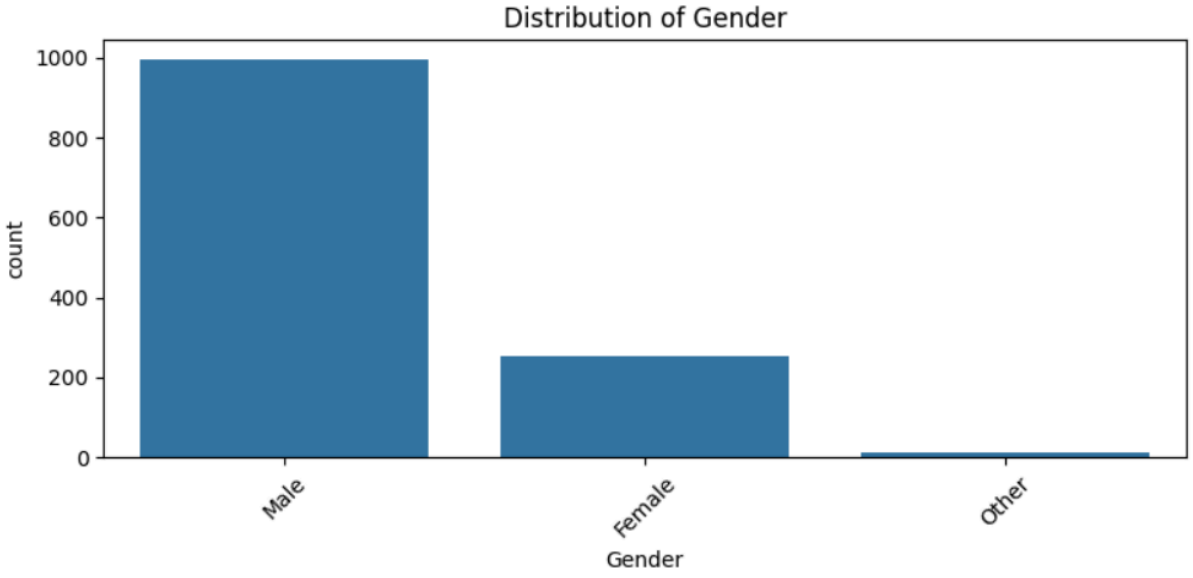
Activity 2: Visual analysis

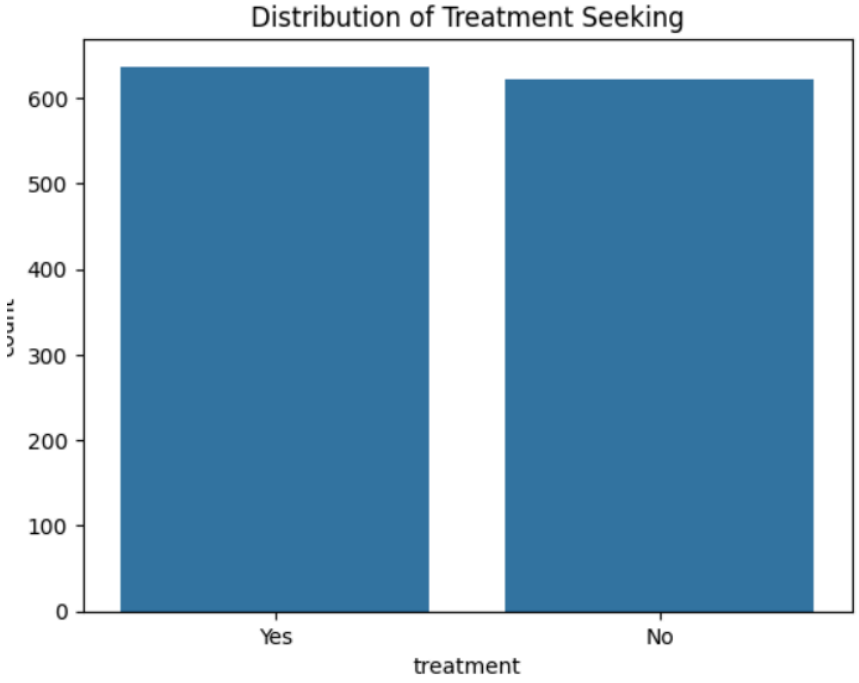
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Activity 2.1: Univariate analysis

Univariate analysis is understanding the data with a single feature. Here we have bar charts for different functions.

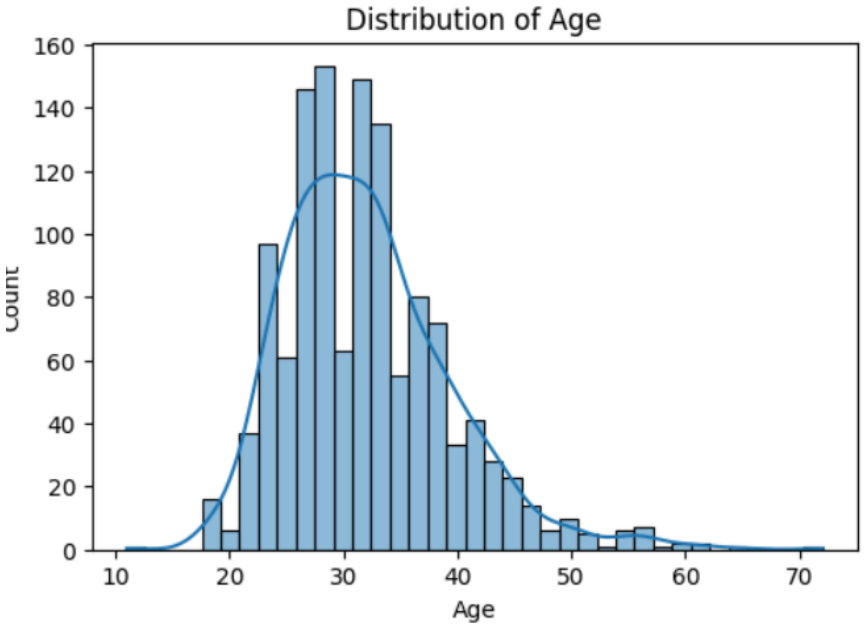

Distribution of self_employed
count
1000
800
600
400
200
0
No
Yes
self_employed


Distribution of Gender
count
1000
800
600
400
200
0
Male
Female
Other
Gender


Distribution of Treatment Seeking
600
500
400
300
200
100
0
Yes
No
treatment

- The preceding visualizations were generated to analyze the distribution and relationships within the categorical variables of the dataset. Bar plots were employed to examine the frequency and proportions of different categories, providing insights into dominant trends, imbalances, or unexpected patterns

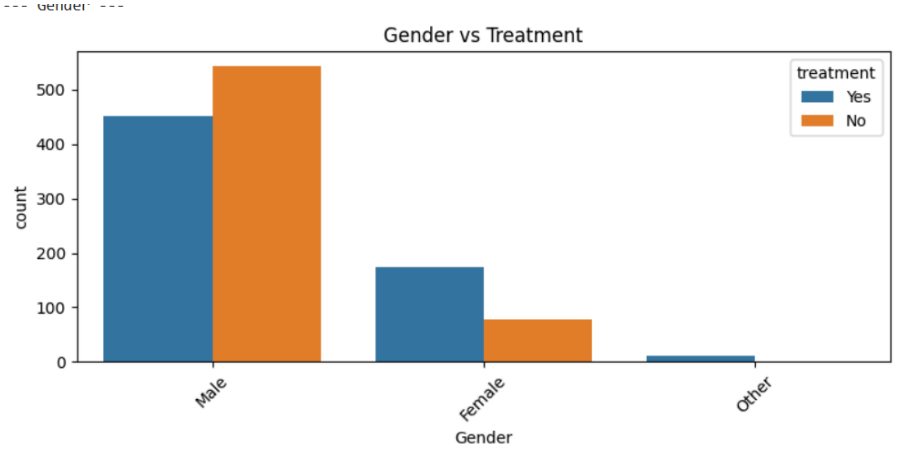
- From the below histogram we can say that ‘age’ Feature is almost Normally distributed. Majority of Insurance claims are of age 20 to 35.



Activity 2.2: Bivariate analysis

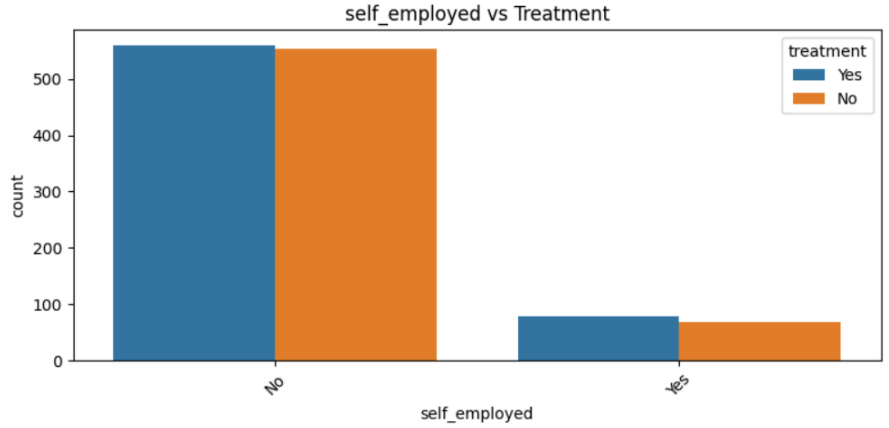
To find the relation between two features we use bivariate analysis. Here we can used barplot. Barplot is used here. As a 1st parameter we are passing Gender and as a 2nd parameter we are passing Treatement which we are treating as our target variable.

- From the below plot you can understand that distribution of gender on Treatement.
- We can concur from the following data that while more number of men were diagnosed with a mental illness they were less likely to get treatment when compared to women.
- From the self employment vs treatment graph given below we can gather that the self employment is not affecting if the person is getting treatment or not



treatment	No	Yes
Gender		
Female	30.952381	69.047619
Male	54.627767	45.372233
Other	7.692308	92.307692

--- self_employed ---



treatment	No	Yes
self_employed		
No	49.775382	50.224618
Yes	46.575342	53.424658

Encoding the Categorical Features:

- The categorical Features are can't be passed directly to the Machine Learning Model. So we convert them into Numerical data based on their order. This Technique is called Encoding.
- Here we are importing Label Encoder from the Sklearn Library.
- Here we are applying fit_transform to transform the categorical features to numerical features.

```
#Label Encoding
le = LabelEncoder()
for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = le.fit_transform(df[col])
```

Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set

The following code snippet prepares the dataset for a supervised machine learning task by separating features (X) from the target variable (y) and splitting them into training and testing sets: For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

Scaling

- Scaling is a technique used to transform the values of a dataset to a similar scale to improve the performance of machine learning algorithms. Scaling is important because many machine learning algorithms are sensitive to the scale of the input features.
- We usually use scaling while using Logistic Regression.

```
log_model = make_pipeline(StandardScaler(), LogisticRegression(max_iter=1000, random_state=42))
log_model.fit(X_train, y_train)
```

Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

Activity 1.1: Decision tree model

The DecisionTreeClassifier from sklearn.tree is initialized with random_state=42 and fitted to x_train and y_train. Predictions for the test set are generated and stored in y_pred_dt. The model's performance is then assessed by printing its accuracy, a classification report, and the confusion matrix.

```
#Decision Tree
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

y_pred_dt = dt_model.predict(X_test)

print("=== Decision Tree Classifier ===")
print("Accuracy: {:.2f}%".format(accuracy_score(y_test, y_pred_dt) * 100))
print("\nClassification Report:\n", classification_report(y_test, y_pred_dt))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
```

Activity 1.2: Random forest model

The RandomForestClassifier from sklearn.ensemble is initialized with random_state=42 and trained on X_train and y_train. Predictions are made on X_test and stored in y_pred. The model's performance is then evaluated by displaying the accuracy score, a classification report, and the confusion matrix against y_test. Finally, the trained Random Forest model is saved as "random_forest_model.pkl" using pickle.

```
#Random Forest
final_model = RandomForestClassifier(random_state=42)
final_model.fit(X_train, y_train)

y_pred = final_model.predict(X_test)

print("Model Accuracy: {:.2f}%".format(accuracy_score(y_test, y_pred) * 100))
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

with open("random_forest_model.pkl", "wb") as f:
    pickle.dump(final_model, f)
```

Activity 1.3: Logistic Regression model

A machine learning pipeline is created using make_pipeline, which first applies StandardScaler for feature scaling, followed by LogisticRegression with max_iter=1000 and random_state=42. This log_model pipeline is then trained on X_train and y_train. Predictions are made on X_test and saved as y_pred_log. The model's performance, including accuracy, classification report, and confusion matrix, is then printed.

```
#Training Models
#Logistic Regression
log_model = make_pipeline(StandardScaler(), LogisticRegression(max_iter=1000, random_state=42))
log_model.fit(X_train, y_train)

y_pred_log = log_model.predict(X_test)

print("=== Logistic Regression with Scaling ===")
print("Accuracy: {:.2f}%".format(accuracy_score(y_test, y_pred_log) * 100))
print("\nClassification Report:\n", classification_report(y_test, y_pred_log))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_log))
```

Activity 2: Testing the model

For the purpose of evaluating our mental health analysis models, we did not conduct separate, individual comparisons between the Logistic Regression, Decision Tree, and Random Forest models. Instead, all three models were run concurrently on the same test dataset (X_test), and their respective performance metrics (accuracy, classification report, and confusion matrix) were given as part of a collective evaluation. This approach allowed us to observe the results of each model side-by-side after the training, providing an overview of their individual prediction accuracy within the same testing setup.

Milestone 5: Performance Testing & Hyperparameter Tuning

Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Activity 1.1: Compare the model

For comparing the above three models, the individual performance of each model is as follows:

```
=== Logistic Regression with Scaling ===
Accuracy: 71.83%

Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.74      0.73       129
           1       0.72      0.69      0.71       123

    accuracy                           0.72       252
   macro avg       0.72      0.72      0.72       252
weighted avg       0.72      0.72      0.72       252

Confusion Matrix:
 [[96 33]
 [38 85]]
```

```
=== Decision Tree Classifier ===
Accuracy: 72.62%

Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.72      0.73       129
           1       0.71      0.73      0.72       123

    accuracy                           0.73       252
   macro avg       0.73      0.73      0.73       252
weighted avg       0.73      0.73      0.73       252

Confusion Matrix:
 [[93 36]
 [33 90]]
```

```
Model Accuracy: 83.33%

Classification Report:

              precision    recall  f1-score   support

           0       0.87      0.79      0.83       129
           1       0.80      0.88      0.84       123

    accuracy                           0.83       252
   macro avg       0.84      0.83      0.83       252
weighted avg       0.84      0.83      0.83       252

Confusion Matrix:
[[102  27]
 [ 15 108]]
```

After calling the function, the results of models are displayed as output. From the above models the performance can be seen.

Milestone 6: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance.This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
with open("random_forest_model.pkl", "wb") as f:
    pickle.dump(final_model, f)
```

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

Activity 2.1: Building Html Page:

For this project create HTML file namely

- index.html

and save them in the templates folder. Refer this link for templates.

Activity 2.2: Build Python code:

Import the libraries

```
from flask import Flask, render_template, request, redirect, url_for
import pickle
import numpy as np
import pandas as pd
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (name) as argument.

```
with open(r'flask\random_forest_model.pkl', 'rb') as f:
    model = pickle.load(f)
```

Render HTML page:

```
@app.route('/')
def home():
    return render_template('home.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
@app.route('/submit', methods=['POST'])
def submit():
    user_input = {}

    for feature in FEATURES:
        value = request.form.get(feature)
        if feature == "Age":
            user_input[feature] = int(value)
        elif feature in multi_map:
            user_input[feature] = multi_map[feature].get(value, 0)
        else:
            user_input[feature] = simple_map.get(value, 0)

    # Convert to DataFrame for model input
    input_df = pd.DataFrame([user_input])[FEATURES]

    # Predict
    prediction = model.predict(input_df)[0]  # assume 1 = needs treatment

    result = 'yes' if prediction == 1 else 'no'
    return redirect(url_for('output', result=result))
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

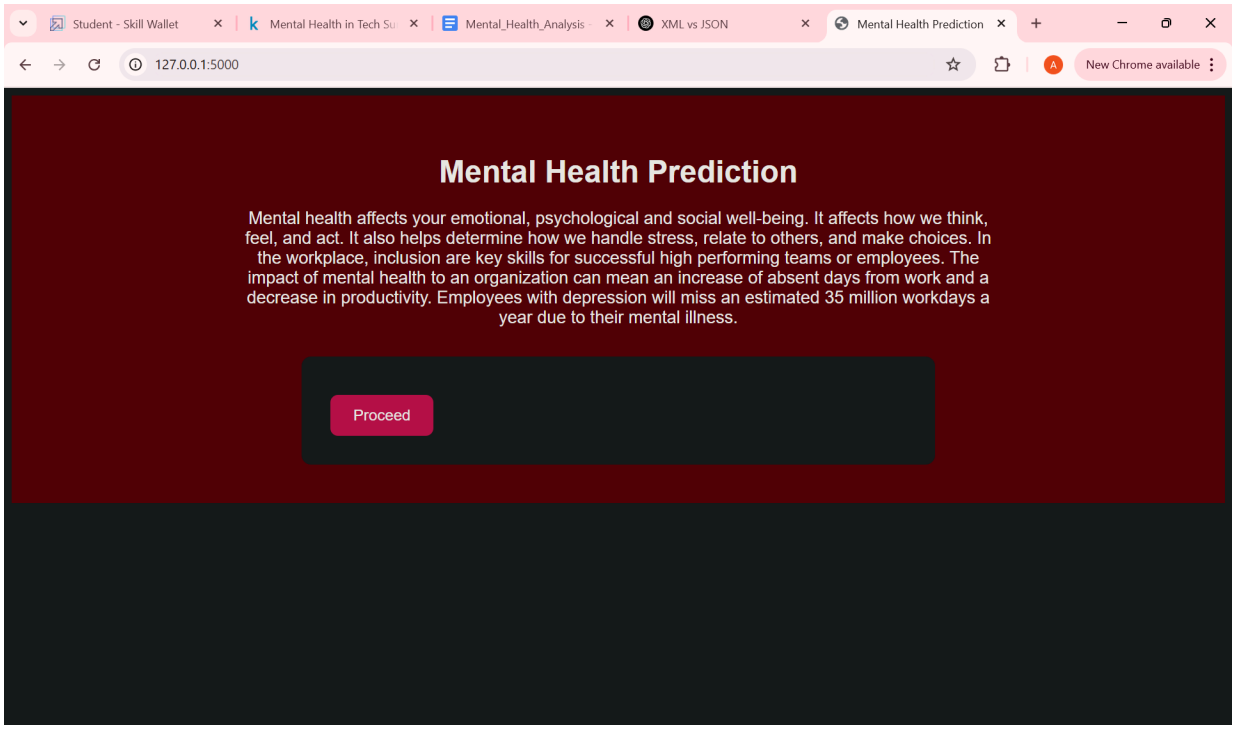
```
if __name__ == '__main__':
    app.run(debug=True)
```

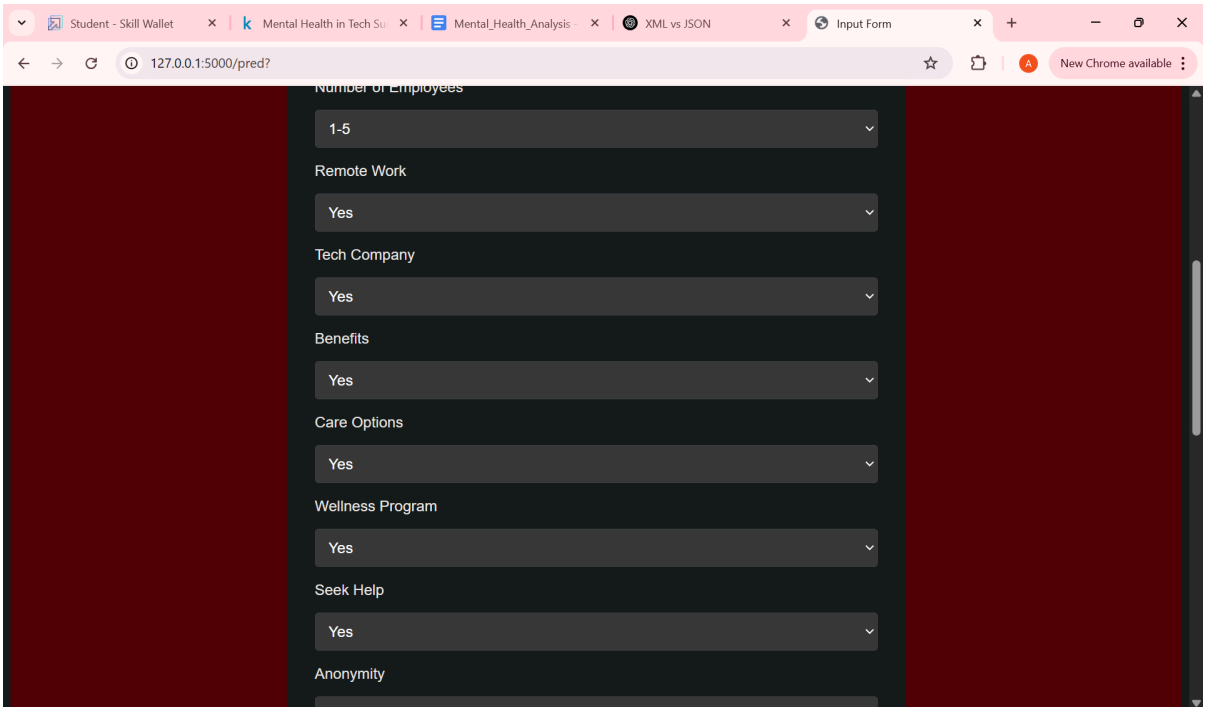
Activity 2.3: Run the web application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result





Milestone 7: Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

Activity 1:- Record explanation Video for project end to end solution

Activity 2:- Project Documentation-Step by step project development procedure

Create document as per the template provided