

Mental health prediction using Machine Learning

Problem Description: Early identification and intervention are crucial for managing mental health conditions. However, traditional methods often rely on self-reporting or clinical assessment, which can be limited by stigma, accessibility, and individual biases. Machine learning presents a potential solution by analyzing data to predict the risk of developing mental health issues.

Project Objectives:

- Develop a machine learning model to predict the likelihood of individuals experiencing specific mental health conditions (e.g., depression, anxiety).
- Identify crucial data features associated with increased mental health risk.
- Explore the ethical implications and limitations of using machine learning for mental health prediction.

Data:

- The specific data used will depend on the chosen mental health condition(s) and target population. Potential sources include:
 - Self-reported surveys on mood, sleep, and stress levels.
 - Social media activity analysis (with user consent).
 - Wearable device data on heart rate, sleep patterns, and activity levels.
 - Electronic health records (anonymized and ethically sourced).
- Data preprocessing and feature engineering will be crucial to prepare the data for machine learning algorithms.

Machine Learning Methods:

- Explore various supervised learning algorithms suitable for classification tasks, such as:
 - Support Vector Machines (SVM)
 - Random Forest
 - Logistic Regression
 - Neural Networks
- Compare the performance of different models and select the one with the best prediction accuracy and interpretability.

Evaluation and Results:

- Use appropriate metrics like accuracy, precision, recall, and F1-score to evaluate model performance.
- Employ cross-validation and other techniques to ensure generalizability of results.
- Interpret the model's predictions to understand which features contribute most to the risk assessment.

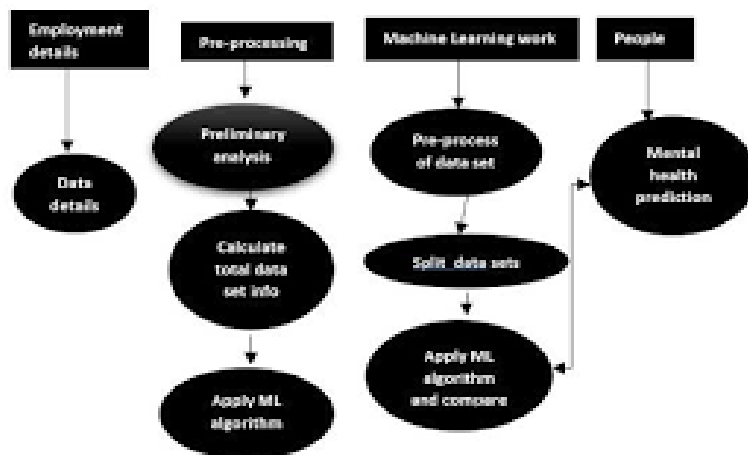
Ethical Considerations:

- Address privacy concerns surrounding data collection and usage.
- Ensure model fairness and avoid biases against specific demographics.
- Emphasize that the model is for prediction, not diagnosis, and encourage seeking professional help when needed.

Potential Applications:

- Early intervention programs for high-risk individuals.
- Personalized mental health resources and support.
- Public health initiatives to address mental health challenges.

Technical Architecture:



Project flow:

1. **Phase 1: Project Definition & Preparation Define Scope & Goals:** Clearly define the target mental health condition(s), target population, and specific objectives (e.g., prediction accuracy, interpretability).

2. **Ethical Considerations:** Establish ethical guidelines for data collection, privacy, and responsible use of predictions.
3. **Team Formation:** Assemble a team with expertise in mental health, data science, and machine learning.
4. **Data Acquisition:** Identify and source appropriate data (anonymized and ethically obtained), considering privacy regulations.

Phase 2: Data Preparation & Exploration

5. **Data Cleaning & Preprocessing:** Handle missing values, outliers, and inconsistencies in the data.
6. **Feature Engineering:** Create meaningful features from raw data relevant to mental health prediction.
7. **Exploratory Data Analysis:** Understand the data distribution, correlations, and potential risks of bias.
8. **Data Splitting:** Divide the data into training, validation, and testing sets for model evaluation.

Phase 3: Model Development & Training

9. **Model Selection:** Choose appropriate supervised learning algorithms based on data characteristics and task requirements.
10. **Model Training:** Train the models on the training data, adjusting hyperparameters for optimal performance.
11. **Model Evaluation:** Use validation data to evaluate model performance using metrics like accuracy, precision, recall, and F1-score.
12. **Model Comparison & Selection:** Compare different models and select the one with the best performance and interpretability.

Phase 4: Model Refinement & Optimization

13. **Feature Importance Analysis:** Understand which features contribute most to the model's predictions.
14. **Hyperparameter Tuning:** Further optimize model performance by adjusting hyperparameters based on validation results.
15. **Explainable AI:** Consider using explainable AI techniques to interpret the model's decision-making process.

Phase 5: Deployment & Evaluation

16. **Model Deployment:** Deploy the final model in a production environment accessible to authorized users.
17. **Real-World Testing:** Monitor the model's performance on real-world data and collect feedback from users.

18. Ethical Review & Continuous Improvement: Regularly review the ethical implications of the system and make necessary adjustments.

Prior knowledge:

1.CNN

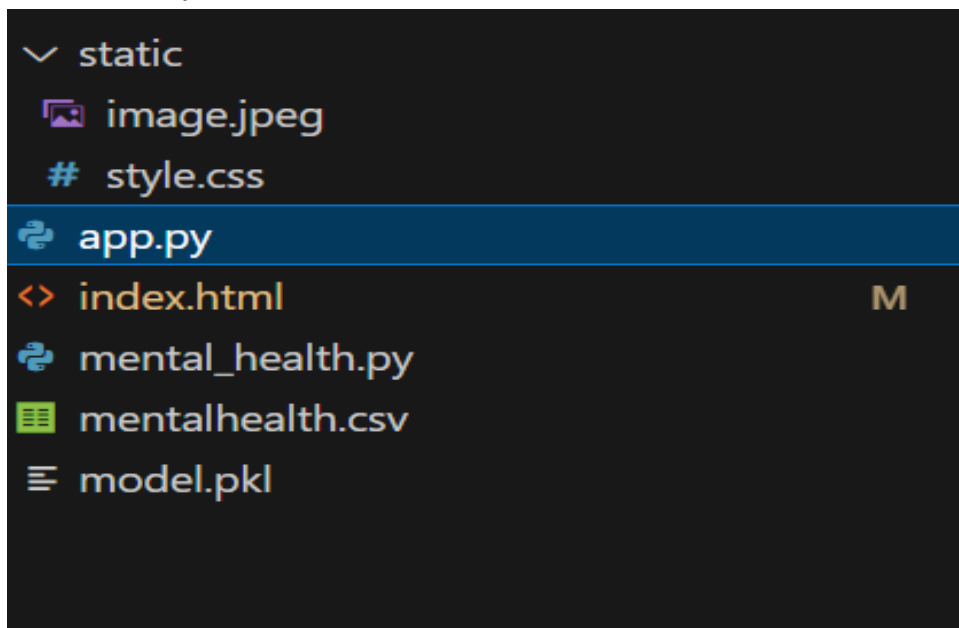
2.Logistic Regression

● Flask: Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications. Link: https://www.youtube.com/watch?v=lj4I_CvBnt0

Build python code:

Project Structure:

Create a Project folder which contains files as shown below:



- The Dataset folder contains the training and testing images for training our model.
- For building a Flask Application we need HTML pages stored in the templates folder, CSS for styling the pages stored in the static folder and a python script app.py for server side scripting
- The IBM folder consists of a trained model notebook on IBM Cloud.

Milestone 1:

Data Collection There are many popular open sources for collecting the data.

Eg: kaggle.com, UCI repository, etc.

Activity 1: Download the dataset

Collect various types of data from people based on respective names as shown in project structure.

In this project, we have collected names, age, family history of the respective people

Dataset: C:\Users\saiku\OneDrive\Desktop\AIML\mentalhealth.csv

Activity 2: Create training and testing dataset

To build a ML model we have to split training and testing data into two separate folders. But in this project dataset folder training and testing folders are not present. So, in this case we have to separate the data into train & test folders.

```
from sklearn.model_selection import train_test_split
```

The input size is

(1258, 2)

Milestone 2: Data Preprocessing

In this milestone we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although perform some geometric transformations of images like rotation, scaling, translation, etc.

Activity 1:

Importing the libraries

```

import numpy as np
import pandas as pd

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from mlxtend.classifier import StackingClassifier
from sklearn.model_selection import train_test_split

import warnings

import pickle

warnings.filterwarnings("ignore")

```

Activity 2:

Replace mismatched data with correct data.

```

male_str = ["male", "m", "male-ish", "maile", "mal", "male (cis)", "make",
"male ", "man","msle", "mail", "malr","cis man", "Cis Male", "cis male"]

trans_str = ["trans-female", "something kinda male?", "queer/she/they", "non-
binary","nah", "all", "enby", "fluid", "genderqueer", "androgynous", "agender",
"male leaning androgynous", "guy (-ish) ^_^", "trans woman", "neuter",
"female (trans)", "queer", "ostensibly male, unsure what that really means"]

female_str = ["cis female", "f", "female", "woman", "femake", "female
","cis-female/femme", "female (cis)", "femail"]

for (row, col) in data.iterrows():

    if str.lower(col.Gender) in male_str:

        data['Gender'].replace(to_replace=col.Gender, value='male',
inplace=True)

    if str.lower(col.Gender) in female_str:

```

```
data['Gender'].replace(to_replace=col.Gender, value='female',  
inplace=True)
```

```
if str.lower(col.Gender) in trans_str:  
    data['Gender'].replace(to_replace=col.Gender, value='trans',  
inplace=True)
```

Activity 3: Apply mapping functionality to Train set and Test set

Let us apply mapping functionality to the Train set and Test set by using the following code. For Training set using `flow_from_directory` function. This function will return batches of images from the subdirectories Arguments:

- **directory:** Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- **batch_size:** Size of the batches of data which is 64.
- **target_size:** Size to resize images after they are read from disk.
- **class_mode:-** 'int': means that the labels are encoded as integers (e.g. for `sparse_categorical_crossentropy` loss).- 'categorical' means that the labels are encoded as a categorical vector (e.g. for `categorical_crossentropy` loss).- 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for `binary_crossentropy`).- None (no labels).

```
X = data[1:,:-1]
```

```
y = data[1:,:-1]
```

```
y = y.astype('int')
```

```
X = X.astype('int')
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,  
random_state=0)
```

Milestone 3:

Applying

KNeighboursclassifier, RandomForestClassifier, StackingClassifier, GuassianNB, LogisticRegression

```
clf1 = KNeighborsClassifier(n_neighbors=1)
clf2 = RandomForestClassifier(random_state=1)
clf3 = GaussianNB()
lr = LogisticRegression()
stack = StackingClassifier(classifiers=[clf1, clf2, clf3],
                           meta_classifier=lr)
stack.fit(X_train, y_train)
```

Activity 3: Configure the Learning Process and save the model.

The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires a loss function during the model compilation process. Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process

```
pickle.dump(stack, open('model.pkl', 'wb'))
model=pickle.load(open('model.pkl', 'rb'))
```

Milestone 4: Application Building 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 HTMLPages
- Building python code
- Runtheprogramme

Activity1: Building HTML Pages:

For this project create one HTML file namely

- index.html

Let's see how our index.html page looks like: predict section



Predict the probability whether the person wants mental health treatment or not

Here are the instructions to fill the form

1. For Gender: Enter 0 for male, 1 for female and 2 for transgender
2. For Family Hqistory: Enter 0 for No and 1 for Yes

Gender:

Family History:

Predict Probability

Activity 2: Build Python code:

Import the libraries

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

Loading the saved model and initializing the flask app

```
app = Flask(__name__,  
template_folder=r'C:\Users\saiku\OneDrive\Desktop\AIML')
```

```
model=pickle.load(open('model.pkl','rb'))
```

Render HTML pages

```
@app.route('/')  
def hello_world():  
    return render_template("index.html")
```

Once we uploaded the file into the app, then verifying the file uploaded properly or not. Here we will be using declared constructor to route to the HTML page which we have created earlier. In the above example, '/' URL is bound with index.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

```
@app.route('/predict',methods=['POST','GET'])  
def predict():  
    int_features=[int(x) for x in request.form.values()]  
    final=[np.array(int_features)]  
    print(int_features)  
    print(final)  
    prediction=model.predict_proba(final)  
    output='{0:.{1}f}'.format(prediction[0][1], 2)  
  
    if output>str(0.5):  
        return render_template('index.html',pred='You need a  
treatment.\nProbability of mental illness is {}'.format(output))  
    else:  
        return render_template('index.html',pred='You do not need  
treatment.\nProbability of mental illness is {}'.format(output))
```

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 index.html page earlier

Main function:

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

Activity 3: Run the application

- Open Spyder
- Navigate to the folder where your Python script is.
- Now click on the green play button above.
- Click on the predict button from the top right corner, enter the inputs, click on the Classify button, and see the result/prediction on the web.

* Running on http://127.0.0.1:5000

Press CTRL+C to quit

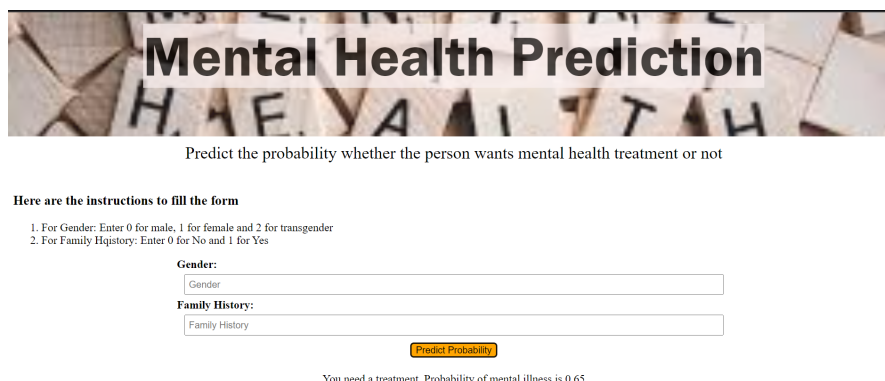
* Restarting with stat

* Debugger is active!

* Debugger PIN: 216-398-786

The home page looks like this. When you click on the Predict button, you'll be redirected to the predict section

Input 1:



The screenshot shows a web application titled "Mental Health Prediction". Below the title is a subtitle: "Predict the probability whether the person wants mental health treatment or not". Underneath, there are instructions: "Here are the instructions to fill the form", followed by two numbered points: "1. For Gender: Enter 0 for male, 1 for female and 2 for transgender" and "2. For Family History: Enter 0 for No and 1 for Yes". The form consists of two input fields: "Gender:" and "Family History:". Below these fields is a yellow button labeled "Predict Probability". At the bottom of the page, a message reads: "You need a treatment. Probability of mental illness is 0.65".

Input 2:



Mental Health Prediction

Predict the probability whether the person wants mental health treatment or not

Here are the instructions to fill the form

1. For Gender: Enter 0 for male, 1 for female and 2 for transgender
2. For Family History: Enter 0 for No and 1 for Yes

Gender:

Family History:

Predict Probability

You do not need treatment. Probability of mental illness is 0.33



Mental Health Prediction

Predict the probability whether the person wants mental health treatment or not

Here are the instructions to fill the form

1. For Gender: Enter 0 for male, 1 for female and 2 for transgender
2. For Family History: Enter 0 for No and 1 for Yes

Gender:

Family History:

Predict Probability

You need a treatment. Probability of mental illness is 0.80