UNLOCKING THE MINDS: ANALYZING MENTAL HEALTH WITH NLP

Major Project documentation submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

Submitted By

DIVYA SRI KUSURI	**(21UK1A6799)**
LIKHITHA RAGAM	**(21UK1A6783)**
SAIRAM SANKOJI	**(21UK1A6785)**
PURNACHAND KONGALA	**(22UK5A6709)**

Under the guidance of

K. SOUMYA

Asst. Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

VAAGDEVI ENGINEERING COLLEGE WARANGAL

CERTIFICATE

This is to certify that the Major project report entitled "UNLOCKING THE MINDS: ANALYZING MENTAL HEALTH WITHNLP" is being submitted by **DIVYA SRI KUSURI(21UK1A6799),LIKHITHA RAGAM(21UK1A6783),SAIRAM SANKOJI(21UK1A6785),PURNACHAND KONGALA(21UK5A6702)**,partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024-2025.

Project Guide

K. SOUMYA

(Asst. professor)

Head of the Department

Dr. K. SHARMILAREDDY

(Professor)

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. Prasad Rao**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this major project in the institute.

We extend our heartfelt thanks to **Dr. K. SHARMILAREDDY,** Head of the Department of CSD, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the mini project.

We express heartfelt thanks to the guide **K. SOUMYA**, Asst. Professor, Department of CSD for her constant support and giving necessary guidance for completion of this major project.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experiencing throughout thesis.

DIVYA SRI KUSURI **(21UK1A6799)**

LIKHITHA RAGAM **(21UK1A6783)**

SAIRAM SANKOJI **(21UK1A6785)**

PURNACHAND KONGALA **(22UK5A6709)**

Unlocking the Minds: Analysing Mental Health with NLP

Mental health is a critical aspect of overall well-being, yet it remains a complex and often stigmatized issue. With advancements in Natural Language Processing (NLP), there is a promising opportunity to gain deeper insights into mental health conditions through the analysis of text data. This project aims to utilize NLP techniques to unlock valuable insights from various sources of textual data related to mental health

Leveraging NLP techniques to analyse textual data related to mental health holds immense potential for gaining insights into the complexities of mental health conditions. By unlocking the linguistic patterns and sentiments expressed in textual data, this project aims to contribute to the advancement of mental health research, support, and intervention efforts.

Scenario 1:

Project Description: "Unlocking the Minds: Analysing Mental Health with NLP" harnesses Natural Language Processing (NLP) techniques to extract insights from textual data pertaining to mental health. Through the analysis of linguistic patterns and sentiments expressed in various sources of text, this project seeks to deepen understanding of mental health conditions, contributing to research, support, and intervention efforts. Scenarios: Social Media Sentiment Analysis: Researchers use NLP algorithms to analyse social media posts related to mental health. By examining the language and sentiments expressed in these posts, they identify trends, common struggles, and support networks within online communities. This insight can inform mental health advocacy campaigns, help target interventions, and provide valuable resources for individuals seeking support in online spaces.

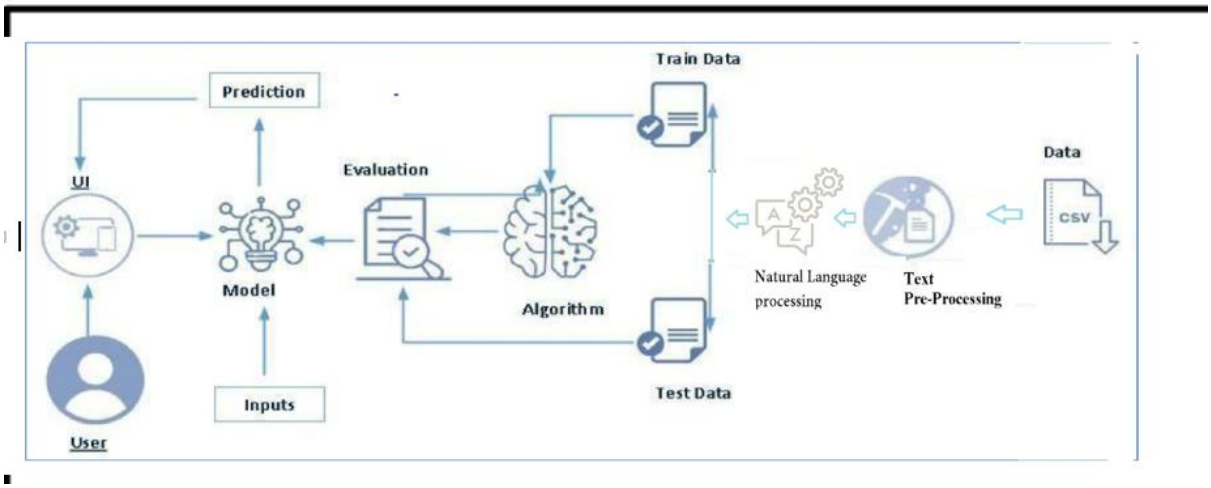
Scenario 2:

Therapist-Patient Communication Analysis: In a clinical setting, therapists utilize NLP tools to analyse transcripts of therapy sessions. By examining language patterns, emotions, and topics discussed, therapists gain deeper insights into patients' mental health struggles, progress, and treatment needs. This analysis can inform personalized treatment plans, identify areas for therapeutic focus, and enhance the effectiveness of therapeutic interventions.

Scenario 3:

Academic Research Paper Analysis: Researchers in the field of psychology and psychiatry employ NLP techniques to analyse academic research papers on mental health topics. By extracting key concepts, trends, and findings from a large corpus of scholarly literature, researchers gain a comprehensive understanding of current knowledge gaps, emerging areas of research, and effective treatment approaches. This analysis can guide future research directions, inform evidence-based practice, and contribute to the advancement of mental health science.

Technical Architecture



Project Flow

- User interacts with the UI to enter the input.
- Entered input is analyzed 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 or 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
- Testing model with multiple evaluation metrics
- Model Deployment
- Save the best model

Integrate with Web Framework

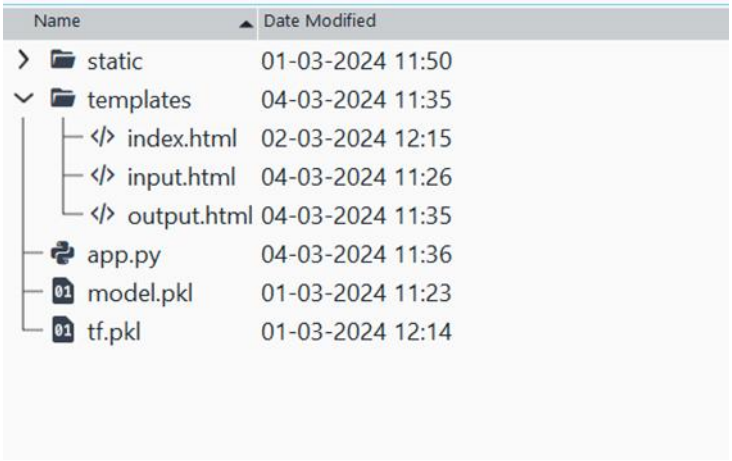
Prior Knowledge

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
- 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/
- NLP:-https://www.javatpoint.com/nlp
- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Technical Architecture

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- model.pkl is our saved model. Further we will use this model for flask integration.
- tf.pkl is our TFIDF converter file.

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.

Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

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/code/swathiunnikrishnan/mental-health-analysis-using-nlp/input

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing 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.

Mental Health Analysis using NLP | Kaggle..

Explore and run machine learning code with Kaggle Notebooks | Using data from Mental Health Corpus..

https://www.kaggle.com/code/swathiunnikrishnan/mental-health-analysis-using-nlp/input

Importing the libraries

Import the necessary libraries as shown in the image.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk # spacy
import re # removing the special characters
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download("punkt")
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
```

Read the Dataset

Activity 3: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. 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('mental_health.csv')
df
```

	text	label
0	dear american teens question dutch person hear...	0
1	nothing look forward lifei dont many reasons k...	1
2	music recommendations im looking expand playli...	0
3	im done trying feel betterthe reason im still ...	1
4	worried year old girl subject domestic physic...	1
...	...	...
27972	posting everyday people stop caring religion ...	0
27973	okay definetly need hear guys opinion ive pret...	0
27974	cant get dog think ill kill myselfthe last thi...	1
27975	whats point princess bridei really think like ...	1
27976	got nudes person might might know snapchat do ...	0

27977 rows × 2 columns

Data Preparation

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling Duplicate Values
- Handling Imbalance Data

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Handling missing values

Let’s find the shape of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.dtypes function is used

For checking the null values, df.isnull() function is used. To sum those null values we use df.isnull().sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
df.shape
```

```
(27977, 2)
```

```
df.dtypes
```

```
text     object
label     int64
dtype: object
```

```
7]: df.isnull().sum()
```

```
7]: text     0
    label    0
    dtype: int64
```

Handling Duplicate Values

Handling duplicate data is a common challenge in data processing Once duplicates are identified, deduplication techniques can be applied to remove redundant entries.

To drop the duplicate values we use drop_duplicated().

```
df.duplicated().sum()
```

```
5
```

```
df = df.drop_duplicates()
print('Number of Duplicates:', len(df[df.duplicated()]))
```

```
Number of Duplicates: 0
```

Handling Imbalance Data

Handling imbalanced data is crucial for ensuring the effectiveness and fairness of machine learning models, especially in tasks such as analyzing mental health with NLP where certain classes or categories may be significantly underrepresented. but in this dataset no need to perform any analysis as we are using data as it is

```
df['label'].value_counts()
```

```
0    14139
1    13838
Name: label, dtype: int64
```

Exploratory Data Analysis

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')
```

	text	label
count	27977	27977.000000
unique	27972	NaN
top	real suppleroot hours up day far	NaN
freq	3	NaN
mean	NaN	0.494621
std	NaN	0.499980
min	NaN	0.000000
25%	NaN	0.000000
50%	NaN	0.000000
75%	NaN	1.000000
max	NaN	1.000000

Visual analysis

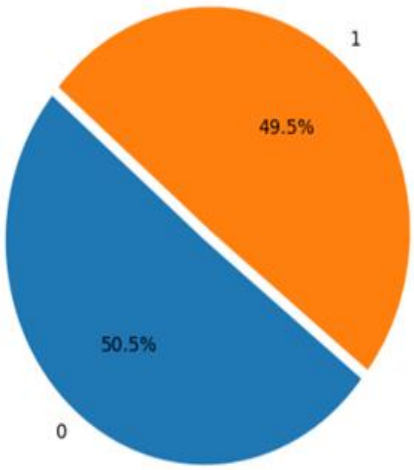
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two pie chart

```
plt.figure(figsize=(5, 6))  # Set the size of the plot
plt.pie(df['label'].value_counts(),labels = df['label'].value_counts().index,autopct='%1.1f%%',
        explode=[0, 0.05] , startangle=140)
```

```
([<matplotlib.patches.Wedge at 0x2730bc743a0>,
  <matplotlib.patches.Wedge at 0x2730bc74af0>],
 [Text(-0.6929627276044373, -0.8542848811438832, '0'),
  Text(0.7244611170241917, 0.8931159442760277, '1')],
 [Text(-0.3779796696024203, -0.46597357153302715, '50.5%'),
  Text(0.4094780226658475, 0.5048046641560157, '49.5%')])
```



Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we don’t have 2 numerical columns in our dataset.

Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we don’t have multiple features in our dataset.

Text pre-processing (python packages)

Text pre-processing is a crucial step in Natural Language Processing (NLP) and Information Retrieval (IR) tasks. The goal is to convert raw text into a more meaningful and manageable representation for further analysis.

In Python, there are several packages that provide support for text pre-processing operations. Some of the most common ones are:

NLTK (Natural Language Toolkit)-It is one of the most widely used NLP libraries in Python. It provides tools for tokenization, stemming, lemmatization, stop-word removal, and more.

```
def remove_stopwords(sentence):

    # List of stopwords
    stopwords = ["a", "about", "above", "after", "again", "against", "all", "am", "an", "and", "any",
                 "are", "as", "at", "be", "because", "been", "before", "being", "below", "between",
                 "both", "but", "by", "could", "did", "do", "does", "doing", "down", "during", "each",
                 "few", "for", "from", "further", "had", "has", "have", "having", "he", "he'd", "he'll",
                 "he's", "her", "here", "here's", "hers", "herself", "him", "himself", "his", "how", "how's",
                 "i", "i'd", "i'll", "i'm", "i've", "if", "in", "into", "is", "it", "it's", "its", "itself",
                 "let's", "me", "more", "most", "my", "myself", "nor", "of", "on", "once", "only", "or",
                 "other", "ought", "our", "ours", "ourselves", "out", "over", "own", "same", "she",
                 "she'd", "she'll", "she's", "should", "so", "some", "such", "than", "that", "that's",
                 "the", "their", "theirs", "them", "themselves", "then", "there", "there's", "these",
                 "they", "they'd", "they'll", "they're", "they've", "this", "those", "through", "to",
                 "too", "under", "until", "up", "very", "was", "we", "we'd", "we'll", "we're", "we've",
                 "were", "what", "what's", "when", "when's", "where", "where's", "which", "while", "who",
                 "who's", "whom", "why", "why's", "with", "would", "you", "you'd", "you'll", "you're",
                 "you've", "your", "yours", "yourself", "yourselves" ]

    # Sentence converted to lowercase-only
    sentence = sentence.lower()

    words = sentence.split()
    no_words = [w for w in words if w not in stopwords]
    sentence = " ".join(no_words)

    return sentence
```

```
df['text1'] = (df['text'].apply(remove_stopwords))
```

Removing Stopwords–Removing stopwords is a common preprocessing step in natural language processing (NLP) tasks, including analyzing mental health with NLP. Stopwords are common words that occur frequently in a language but typically do not carry much semantic meaning. Examples of stopwords include "the," "is," "and," "in," etc

```
msg=df['text1'].str.replace('[^a-zA-Z0-9]+'," ")
msg
```

This will replace all occurrences of alphanumeric characters, plus signs, and double quotes in the 'text1' column with spaces.

```
|: lemmatizer = WordNetLemmatizer()
   data=[]
```

WordNet Lemmatizer is a linguistic tool used to normalize words to their base or dictionary form, known as the lemma. It operates by mapping words to their root forms, helping to reduce inflected forms to a common base for analysis or comparison.

```
def preprocess_text(text):
    # Tokenize the text
    tokens = word_tokenize(text)

    # Filter tokens with length greater than 2
    filtered_tokens = [token for token in tokens if len(token) > 3]

    # Stem each token
    lemmed_tokens = [lemmatizer.lemmatize(token.lower()) for token in filtered_tokens]

    # Join stemmed tokens into a single string
    preprocessed_text = " ".join(lemmed_tokens)

    return preprocessed_text

# Apply text preprocessing to each row of the DataFrame
df['preprocessed_text'] = msg.apply(preprocess_text)
```

This code we are creating a function that will preprocess the text in the 'text' column of the DataFrame by tokenizing each sentence, lemmatizing the tokens, and then joining the lemmatized tokens back into a single string

```
from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer()
data_vec=tf.fit_transform(df['preprocessed_text'])
print(data_vec)
```

TFIDF Vectorizer: TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure used in natural language processing and information retrieval to evaluate the importance of a word in a document relative to a collection of documents

```
df['text_length'] = df['preprocessed_text'].apply(lambda x: len(str(x).split()))
```

```
C:\Users\indra\AppData\Local\Temp\ipykernel_12384\987295424.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
rsus-a-copy
  df['text_length'] = df['preprocessed_text'].apply(lambda x: len(str(x).split()))
```

This code will add a new column 'text_length' to the DataFrame 'df', containing the length of each text in the 'text' column.

After Text Pre-Processing:-

After text pre-processing, the data will typically look much cleaner and more manageable. The specific form it takes will depend on the specific operations performed during pre-processing and here is b

f is passed with dropping the target variable. And on y target variable is passed. 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.

```
y=df['label'].values
y
```

```
array([0, 1, 0, ..., 1, 1, 0], dtype=int64)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data_vec,y,test_size=0.3,random_state=1)
```

Model Building

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.

Support Vector Classifier

A function named Support Vector Classifier is created and train and test data are passed as the parameters. Inside the function, Support Vector Classifier algorithm is initialized and training data is passed to the model with the. fit () function. Test data is predicted with. predict () function and saved in a new variable. To Evaluate the model Classification report is used.

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report,accuracy_score
sv=SVC()
sv.fit(x_train,y_train)
y_pred=sv.predict(x_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.94      0.91      4271
           1       0.93      0.88      0.91      4121

    accuracy                           0.91      8392
   macro avg       0.91      0.91      0.91      8392
weighted avg       0.91      0.91      0.91      8392
```

```
: sv_acc=accuracy_score(y_test,y_pred)
  sv_acc
```

```
: 0.9100333651096282
```

elow fig after cleaning the text

```
df = df.drop(['text', 'text1'], axis=1)
```

```
df.head()
```

	label	preprocessed_text	text_length
0	0	dear american teen question dutch person heard...	19
1	1	nothing look forward lifei dont many reason ke...	18
2	0	music recommendation looking expand playlist u...	48
3	1	done trying feel betterthe reason still alive ...	79
4	1	worried year girl subject domestic physicalmen...	269

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

Here x and y variables are created. On x variable, d

Decision Tree Classifier

A function named Decision Tree Classifier is created and train and test data are passed as the parameters. Inside the function, Decision Tree Classifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. To Evaluate the model Classification report is used.

```
from sklearn.tree import DecisionTreeClassifier
Dt=DecisionTreeClassifier()
Dt.fit(x_train,y_train)
y_pred=Dt.predict(x_test)
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.82      0.82      4271
           1       0.81      0.81      0.81      4121

    accuracy                           0.81      8392
   macro avg       0.81      0.81      0.81      8392
weighted avg       0.81      0.81      0.81      8392

0.8142278360343184
```

```
dt_acc=accuracy_score(y_test,y_pred)
dt_acc
```

```
0.8142278360343184
```

Random Forest Classifier

A function named Random Forest Classifier is created and train and test data are passed as the parameters. Inside the function, Random Forest Classifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. To Evaluate the model Classification report is used.

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
y_pred=rf.predict(x_test)
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.90      0.89      4271
           1       0.89      0.87      0.88      4121

    accuracy                           0.88      8392
   macro avg       0.88      0.88      0.88      8392
weighted avg       0.88      0.88      0.88      8392

0.8841754051477597
```

```
Rf_acc=accuracy_score(y_test,y_pred)
Rf_acc
```

```
0.8841754051477597
```

Adaboost Classifier

A function named Adaboost Classifier is created and train and test data are passed as the parameters. Inside the function, Adaboost Classifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. To Evaluate the model Classification report is used.

```
from sklearn.ensemble import AdaBoostClassifier
ab= AdaBoostClassifier()
ab.fit(x_train,y_train)
y_pred=ab.predict(x_test)
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.91      0.87      4271
           1       0.89      0.81      0.85      4121

    accuracy                           0.86      8392
   macro avg       0.86      0.86      0.86      8392
weighted avg       0.86      0.86      0.86      8392

0.8617731172545281
```

```
ab_acc=accuracy_score(y_test,y_pred)
ab_acc
```

```
0.8617731172545281
```

Gradient Boosting Classifier

A function named Gradient Boosting Classifier is created and train and test data are passed as the parameters. Inside the function Gradient Boosting Classifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with. predict () function and saved in a new variable. To Evaluate the model Classification report is used.

```
from sklearn.ensemble import GradientBoostingClassifier
gb=GradientBoostingClassifier()
gb.fit(x_train,y_train)
y_pred=gb.predict(x_test)
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      4271
           1       0.90      0.81      0.85      4121

    accuracy                           0.86      8392
   macro avg       0.86      0.86      0.86      8392
weighted avg       0.86      0.86      0.86      8392

0.8604623450905624
```

```
gb_acc=accuracy_score(y_test,y_pred)
gb_acc
```

```
0.8604623450905624
```

Logistic Regression

A function named Logistic Regression is created and train and test data are passed as the parameters. Inside the function Logistic Regression algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. To Evaluate the model Classification report is used.

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.94      0.91      4271
           1       0.93      0.88      0.90      4121

    accuracy                           0.91      8392
   macro avg       0.91      0.91      0.91      8392
weighted avg       0.91      0.91      0.91      8392

0.9072926596758818
```

```
lr_acc=accuracy_score(y_test,y_pred)
lr_acc
```

```
0.9072926596758818
```

Performance testing and Hyper Parameter Tunning

Testing the model

Compare the models

For comparing the models we create data frame with accuracy of models

Testing the model

Since form above comparison Support Vector Machine is performing well we will test for that model.

Activity 2: Comparing model accuracy before & after applying hyperparameter tuning (Hyperparameter tuning is optional. For this project it is not required.)

Note :- We are Getting Good Accuracy with Support Vector Machine so, no need of Hyperparameter Tuning

```
model = pd.DataFrame({'Model':['Support Vector Machine','Decision Tree','RandomForest',
                               'AdaBoost','GradientBoosting','Logistic Regression'],
                      'Score':[sv_acc,dt_acc,Rf_acc,ab_acc,gb_acc,lr_acc],
                       })
model
```

	Model	Score
0	Support Vector Machine	0.910033
1	Decision Tree	0.814228
2	RandomForest	0.884175
3	AdaBoost	0.861773
4	GradientBoosting	0.860462
5	Logistic Regression	0.907293

```
y_new=sv.predict(tf.transform(["I'm overwhelmed with anxiety about the future."]))
if y_new==1:
    print("Positive")
if y_new==0:
    print("Negative")
```

```
Positive
```

```
y_new=sv.predict(tf.transform(["I had a great time with friends last night."]))
if y_new==1:
    print("Positive")
if y_new==0:    print("Negative")
```

```
Negative
```

Model Deployment

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.

```
import pickle
import warnings
pickle.dump(sv,open("model.pkl" , "wb"))
```

```
import joblib
joblib.dump(tf, 'tf.pkl')
```

```
['tf.pkl']
```

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

Building Html Pages:

For this project create two HTML files namely

- Index.html
- Input.html
- Output.html

and save them in the templates folder.

Build Python code

```
import numpy as np
import pickle
import pandas
import os
from flask import Flask, request, render_template
import re
import nltk
import joblib
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
```

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.

```
app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))
tfidf_vectorizer = joblib.load(open('tf.pkl', 'rb'))
```

Render HTML page:

```
@app.route('/') # rendering the html template
def home():
    return render_template('index.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('/predict',methods=["POST", "GET"]) # rendering the html t
def predict() :
    return render_template("input.html")
```

```
@app.route('/submit',methods=["POST","GET"])# route to show the predictions in a web UI
def submit():
    #  reading the inputs given by the user
    text = request.form['userInput']
    text=re.sub('[^a-zA-Z0-9]+'," ",text)

    tokens = word_tokenize(text)
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in filtered_tokens]
    preprocess_text = ' '.join(lemmatized_tokens)

    print(preprocess_text)
    # Transform the preprocessed text using the TF-IDF vectorizer
    text_vectorized = tfidf_vectorizer.transform([preprocess_text])

    # Make predictions using the model
    prediction = model.predict(text_vectorized)[0]

    # Map prediction to label
    label = "Positive" if prediction == 1 else "Negative"

    return render_template('output.html', prediction_text=f'After Analysis State of Mind was found: {label}')
```

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 input.html page earlier.

Main Function:

```
if __name__=="__main__":

    # app.run(host='0.0.0.0', port=8000,debug=True)    # ru
    port=int(os.environ.get('PORT',5000))
    app.run(debug=True)
```

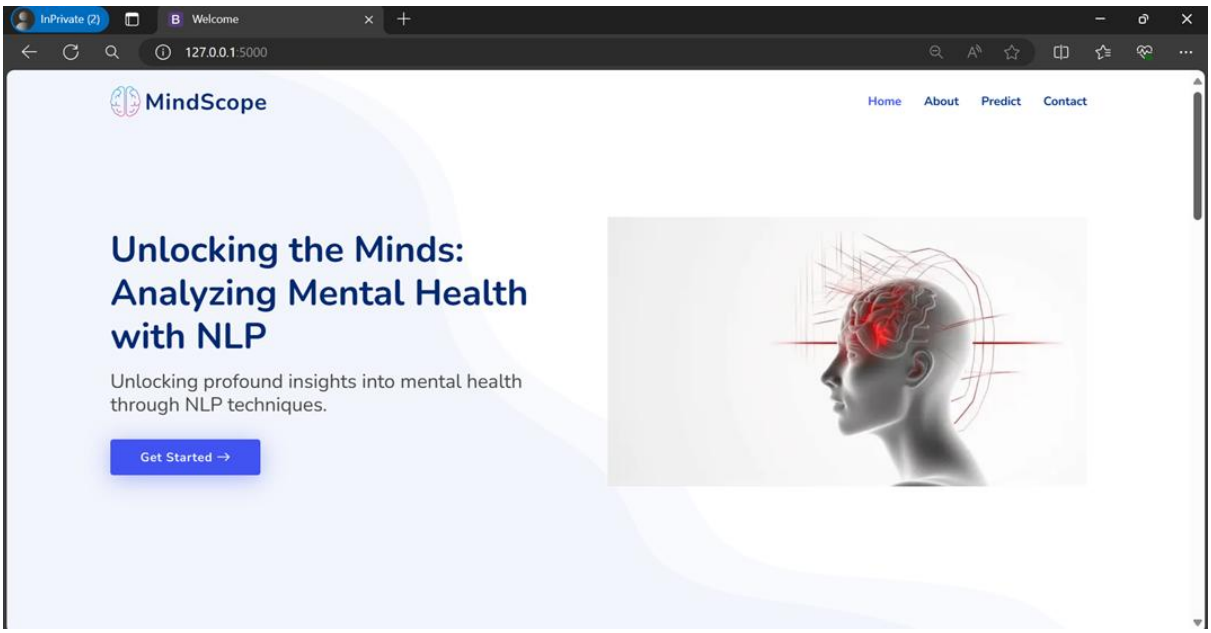
Run the web application

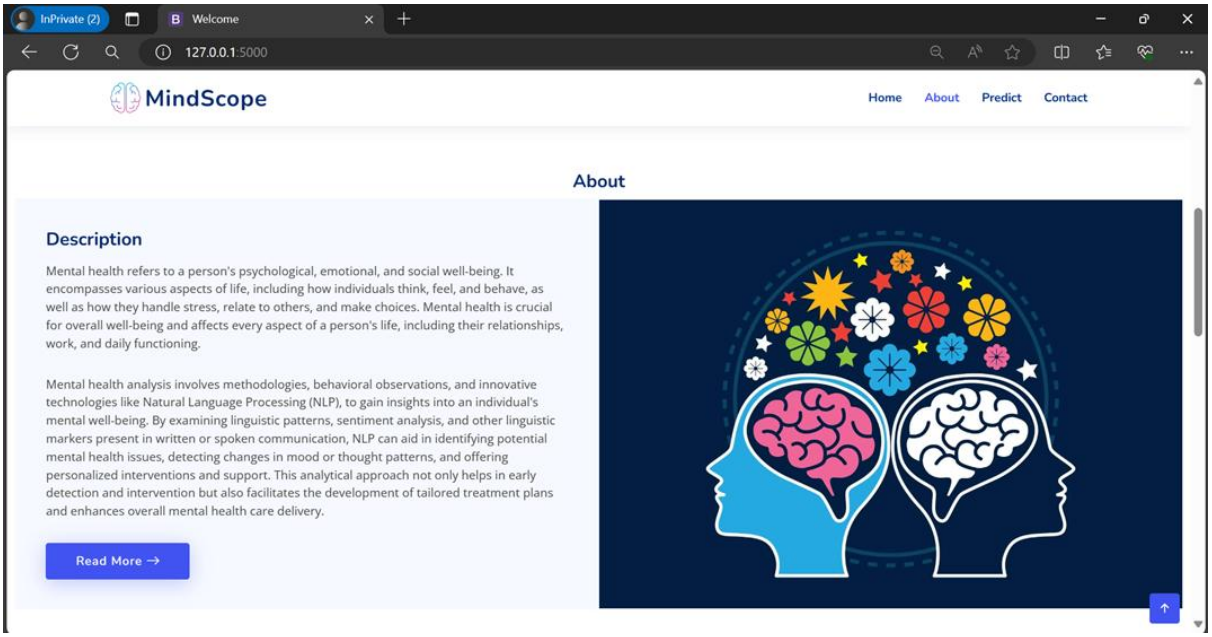
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “app.py” command
- Navigate to the localhost (127.0.0.1:5000) where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

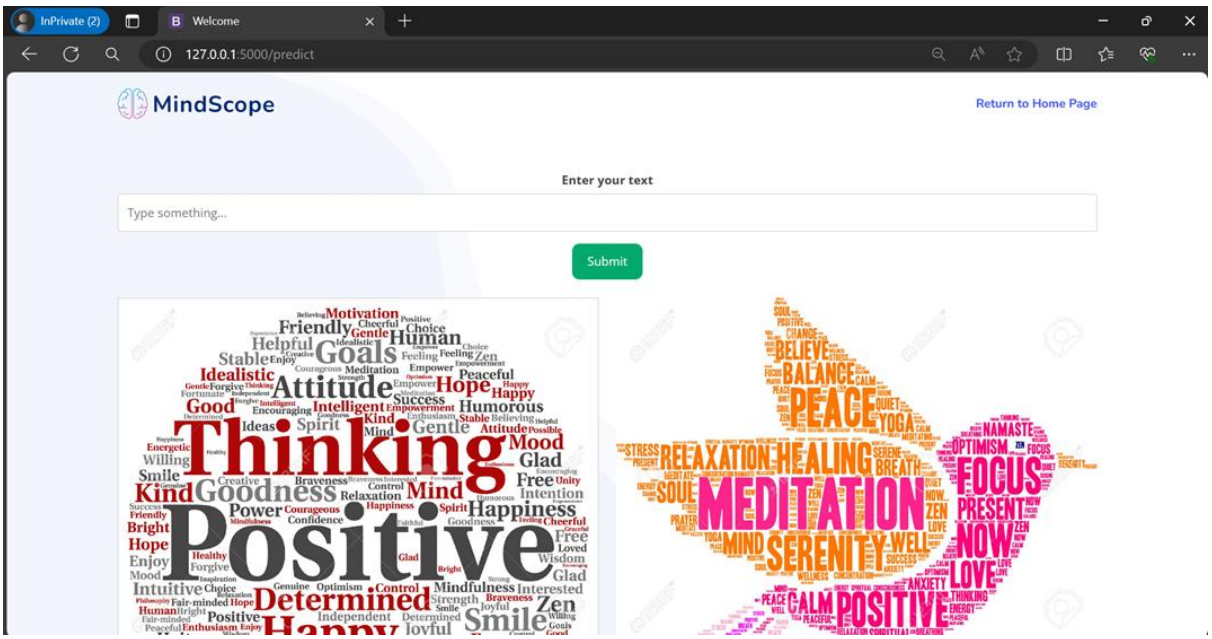
```
 warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
```

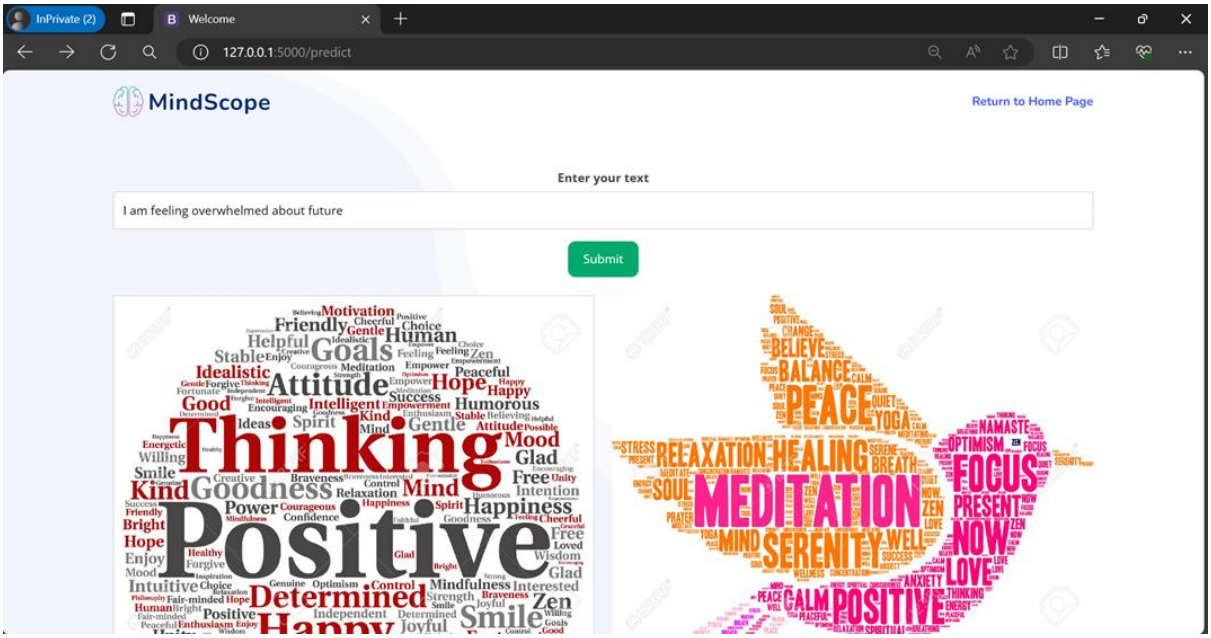
If you set debug as False you can see the port address as below

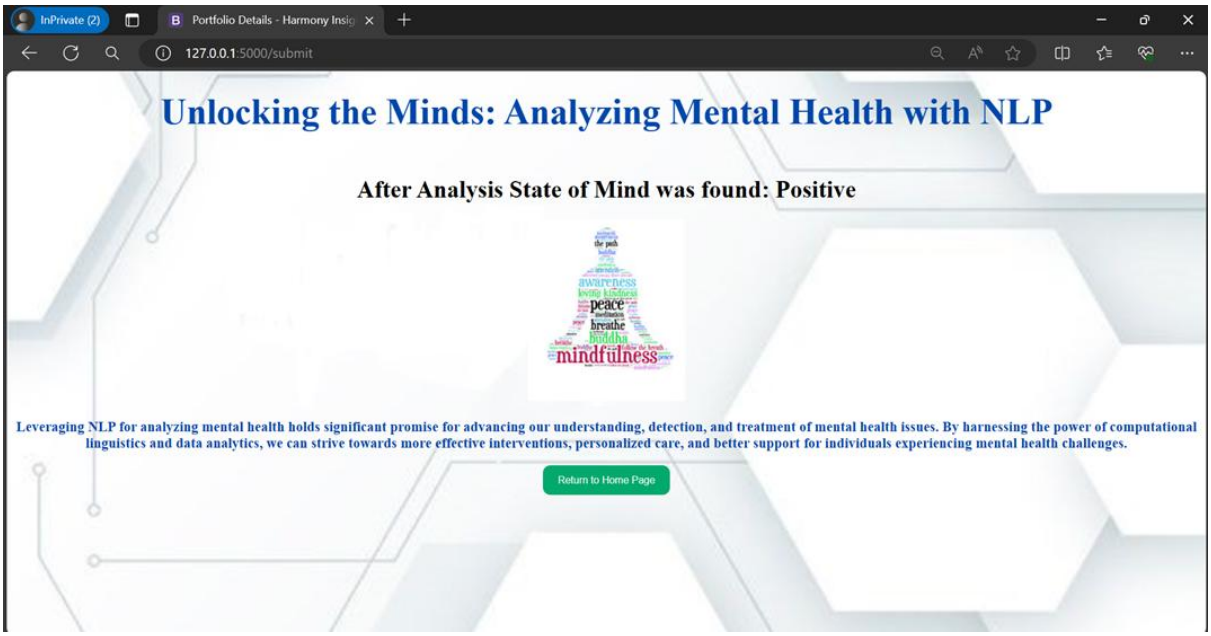
```
WARNING: This is a development server. Do not u
 * Running on http://127.0.0.1:5000
```

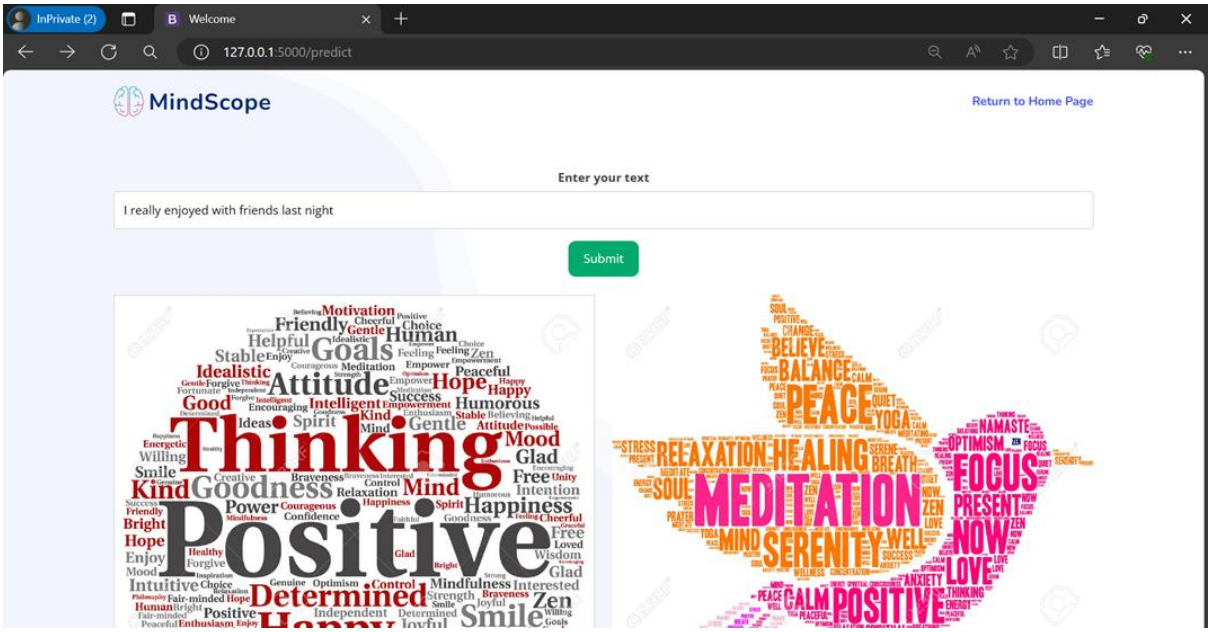


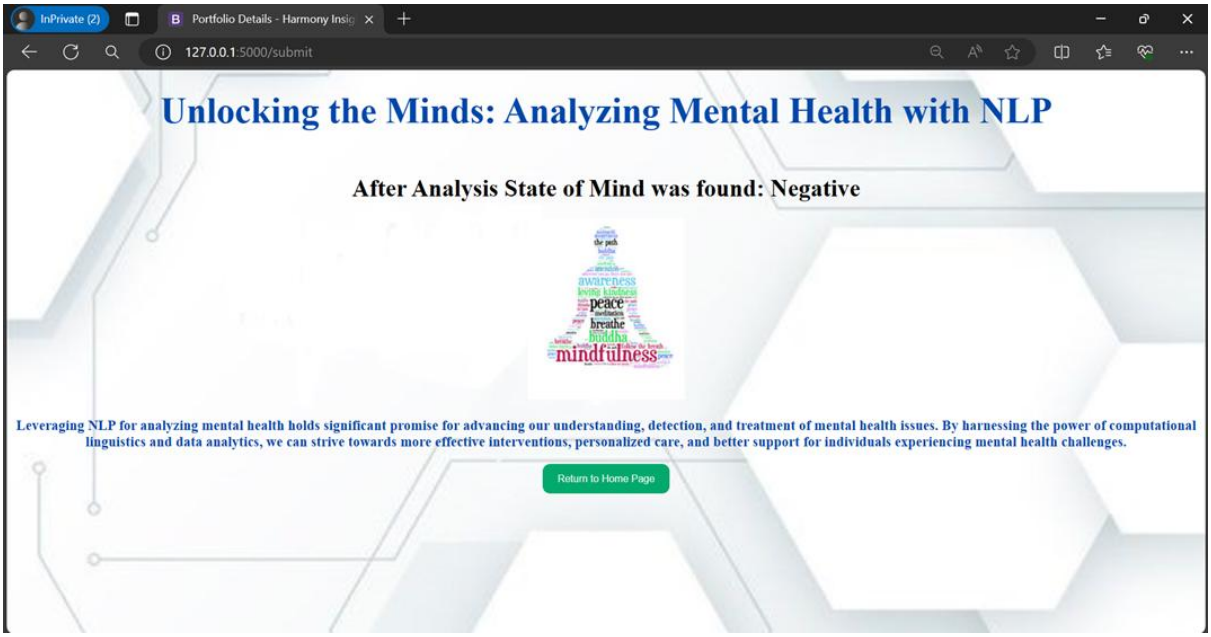

InPrivate (2)
Welcome
127.0.0.1:5000
MindScope
Home
About
Predict
Contact
About
Description
Mental health refers to a person's psychological, emotional, and social well-being. It encompasses various aspects of life, including how individuals think, feel, and behave, as well as how they handle stress, relate to others, and make choices. Mental health is crucial for overall well-being and affects every aspect of a person's life, including their relationships, work, and daily functioning.
Mental health analysis involves methodologies, behavioral observations, and innovative technologies like Natural Language Processing (NLP), to gain insights into an individual's mental well-being. By examining linguistic patterns, sentiment analysis, and other linguistic markers present in written or spoken communication, NLP can aid in identifying potential mental health issues, detecting changes in mood or thought patterns, and offering personalized interventions and support. This analytical approach not only helps in early detection and intervention but also facilitates the development of tailored treatment plans and enhances overall mental health care delivery.
Read More →


InPrivate (2)
Welcome
127.0.0.1:5000/predict
MindScope
Return to Home Page
Enter your text
Type something...
Submit
Thinking
Positive
Attitude
Goals
Motivation
Friendly
Human
Kind
Goodness
Happiness
Determined
Smile
Mind
Hope
Intelligent
Idealistic
Zen
MEDITATION
HEALING
RELAXATION
PEACE
SERENITY
FOCUS
POSITIVE
CALM
NOW
LOVE
MIND
SOUL
BALANCE
BELIEVE
NAMASTE
OPTIMISM
PRESENT
YOGA
BREATH


InPrivate (2)
Welcome
127.0.0.1:5000/predict
MindScope
Return to Home Page
Enter your text
I am feeling overwhelmed about future
Submit
Thinking
Positive
Attitude
Goodness
Determined
Happy
Smile
Goals
Motivation
Friendly
Human
Idealistic
Hope
Happiness
Mind
Mood
MEDITATION
PEACE
BALANCE
HEALING
RELAXATION
SERENITY
POSITIVE
FOCUS
PRESENT
NOW
LOVE
CALM
YOGA
NAMASTE
OPTIMISM
BREATH


InPrivate (2)
Portfolio Details - Harmony Insig
127.0.0.1:5000/submit
Unlocking the Minds: Analyzing Mental Health with NLP
After Analysis State of Mind was found: Positive
awareness
peace
breathe
Buddha
mindfulness
Leveraging NLP for analyzing mental health holds significant promise for advancing our understanding, detection, and treatment of mental health issues. By harnessing the power of computational linguistics and data analytics, we can strive towards more effective interventions, personalized care, and better support for individuals experiencing mental health challenges.
Return to Home Page





CONCLUSION:

The intersection of Natural Language Processing (NLP) and mental health offers a transformative opportunity to understand and address mental well-being more effectively. NLP's ability to process vast amounts of textual data enables early detection, continuous

monitoring, and personalized interventions for mental health challenges. Through the analysis of patterns in language, sentiment, and even conversational nuances, NLP tools can provide insights that were once difficult to obtain.