1. INTRODUCTION

Overview

The increase of mental health problems and the need for effective medical health care have led to an investigation of machine learning that can be applied in mental health problems. This report presents a recent systematic review of machine learning and deep learning approaches in predicting mental health problems. Furthermore, we will discuss the challenges, limitations, and future directions for the application of machine learning in the mental health field. We collect research articles and studies that are related in predicting mental health problems by searching reliable databases. Then, we categorize the collected research articles based on the mental health problems such as schizophrenia, bipolar disorder, anxiety and depression, posttraumatic stress disorder, and mental health problems among children. Discussing the findings, we reflect on the challenges and limitations faced by the researchers on machine learning in mental health problems. Additionally, we provide concrete recommendations on the potential future research and development of applying machine learning and deep learning in the mental health field. The main purpose of the Mental Health Prediction system is to predict whether a person needs to seekmental health treatment or not based on inputs provided by them.

Purpose

Mental Health First Aid teaches participants how to notice and support an individual who may be experiencing a mental health or substance use concern or crisis and connect them with the appropriate employee resources. Employers can offer robust benefits packages to support employees who go through mental health issues. That includes Employee Assistance Programs, Wellness programs that focus on mental and physical health, Health and Disability Insurance, or flexible working schedules or time off policies. Organizations that incorporate mental health awareness help to create a healthy and productive work environment that reduces the stigma associated with mental illness, increases the organizations' mental health literacy, and teaches the skills to safely and responsibly respond to a co- worker's mental health concern. The main purpose of the Mental Health Prediction system is to predict whether a person needs to seek mental health treatment or not based on inputs provided by them. We will be using classification algorithms such as Logistic Regression, KNN, Decision tree, Random Forest, AdaBoost, Gradient Boost, and XGBoost. We will train and test the data with these algorithms. From this, the best model is selected and saved in pkl format. We will also be deploying our model locally using Flask.

2. LITERATURE SURVEY

Existing problem

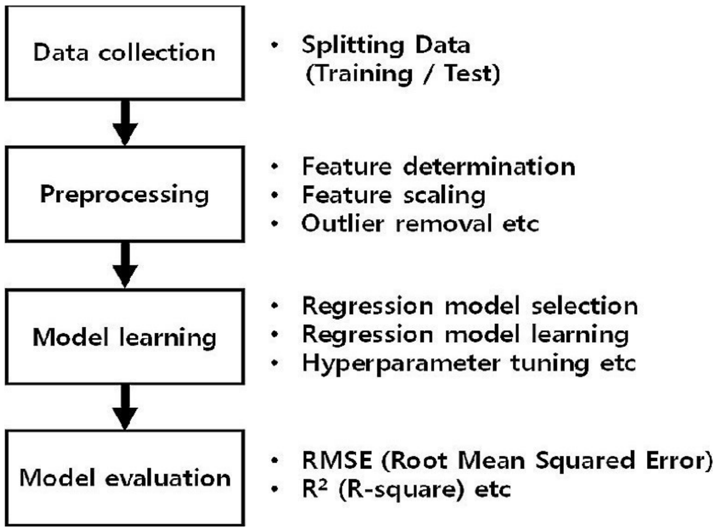
Mental illness is a health problem that undoubtedly impacts emotions, reasoning, and social interaction of a person. These issues have shown that mental illness gives serious consequences across societies and demands new strategies for prevention and intervention. To accomplish these strategies, early detection of mental health is an essential procedure. Mental illness is usually diagnosed based on the individual self-report that requires questionnaires designed for thedetection of the specific patterns of feeling or social interactions. With proper care and treatment,many individuals will hopefully be able to recover from mental illness or emotional disorder.

Proposed solution

Machine learning is a technique that aims to construct systems that can improve through experience by using advanced statistical and probabilistic techniques. It is believed to be a significantly useful tool to help in predicting mental health. It is allowing many researchers to acquire important information from the data, provide personalized experiences, and develop automated intelligent systems. The widely used algorithms in the field of machine learning such as support vector machine, random forest, and artificial neural networks have been utilized to forecast and categorize the future event

3. THEORITICAL ANALYSIS

Block diagram


Data collection
Splitting Data (Training / Test)
Preprocessing
Feature determination
Feature scaling
Outlier removal etc
Model learning
Regression model selection
Regression model learning
Hyperparameter tuning etc
Model evaluation
RMSE (Root Mean Squared Error)
R² (R-square) etc

Hardware / Software Designing

Recommended System Requirements

- Processors: Intel® Core™ i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores, 2 threads per core), 8 GB of DRAM Intel® Xeon® processor E5-2698 v3 at 2.30 GHz (2 sockets,16 cores each, 1 thread per core), 64 GB of DRAM Intel® Xeon Phi™ processor 7210 at 1.30 GHz (1 socket, 64 cores, 4 threads per core), 32 GB of DRAM, 16 GB of MCDRAM (flat mode enabled)
- Disk space: 2 to 3 GB
- Operating systems: Windows® 10, macOS*, and Linux*

Minimum System Requirements

- Processors: Intel Atom® processor or Intel® Core™ i3 processor
- Disk space: 1 GB
- Operating systems: Windows* 7 or later, macOS, and Linux
- Python* versions: 3.9

Software requirements

Anaconda navigator

Anaconda is an open-source distribution for python and R. It is used for data science, machine learning, deep learning, etc. With the availability of more than 300 libraries for data science, it becomes fairly optimal for any programmer to work on anaconda for data science Pycharm: PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

4. EXPERIMENTAL INVESTIGATIONS

Logistic Regression

This type of statistical model (also known as logit model) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure.

Decision Tree Classifier

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle highdimensional data with good accuracy.

K-Neighbors Classifier

K-Nearest Neighbors, or KNN for short, is one of the simplest machine learning algorithms and is used in a wide array of institutions. KNN is a non-parametric, lazy learning algorithm. When we say a technique is non-parametric, it means that it does not make any assumptions about the underlying data. In other words, it makes its selection based off of the proximity to other data points regardless of what feature the numerical values represent. Being a lazy learning algorithm implies that there is little. Therefore, we can immediately classify new data points as they present themselves. To no training phase.

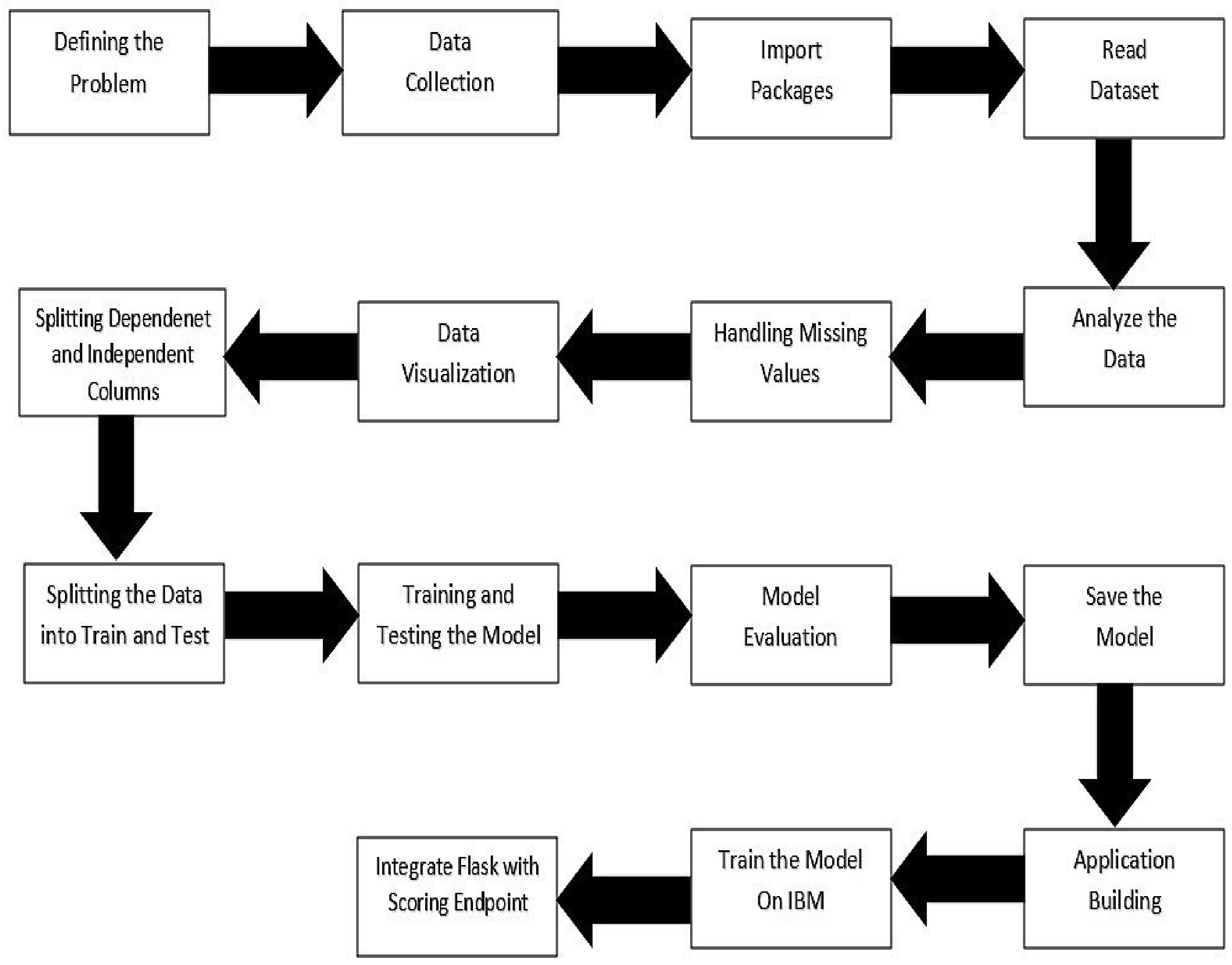
XGB Classifier

The XGBoost stands for extreme Gradient Boosting, which is a boosting algorithm based on gradient boosted decision trees algorithm. XGBoost applies a better regularizationtechnique to reduce overfitting, and it is one of the differences from the gradient boosting. The 'xgboost' is an open- source library that provides machine learning algorithms under the gradient boosting methods. The xgboost. XGBClassifier is a scikit-learn API compatibleclass for classification.

Random Forest Classifier

The Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

5. FLOWCHART



Defining the Problem
Data Collection
Import Packages
Read Dataset
Analyze the Data
Handling Missing Values
Data Visualization
Splitting Dependenet and Independent Columns
Splitting the Data into Train and Test
Training and Testing the Model
Model Evaluation
Save the Model
Application Building
Train the Model On IBM
Integrate Flask with Scoring Endpoint

6. CODE

Mental Health Prediction

The main purpose of the Mental Health Prediction system is to predict whether a person needs to seek Mental health treat -ment or not based on inputs provided by them.

Load Data

```
]: import numpy as np
   import pandas as pd
   import seaborn as sb
   import matplotlib.pyplot as plt
```

```
]: data = pd.read_csv('D:\Dataset\survey.csv')
```

```
]: data.head()
   data.shape
```

```
]: data.tail()
```

```
]: data.info()
```

Data Preprocessing

we need to clean the dataset properly in order to fetch good result,for this we need to follow the below steps.
 1.Removing unnecessary columns.
 2.Handling Null values and dealing with wrongly entered data.

```
data['Country'].value_counts()
```

```
data['Country'].value_counts().plot(kind='bar',figsize=(10,8))
```

Since the countries are not evenly distributed, keeping this column will induce bias in our model. So we will be remov ing country and state columns. We will also remove timestamp and comments columns as they do not contribute to provid ing relevant information.

```
# Removing unwanted features.
data = data.drop(['Country','state','Timestamp','comments'],axis=1)
```

```
# Checking the null values.
data.isnull().sum()
```

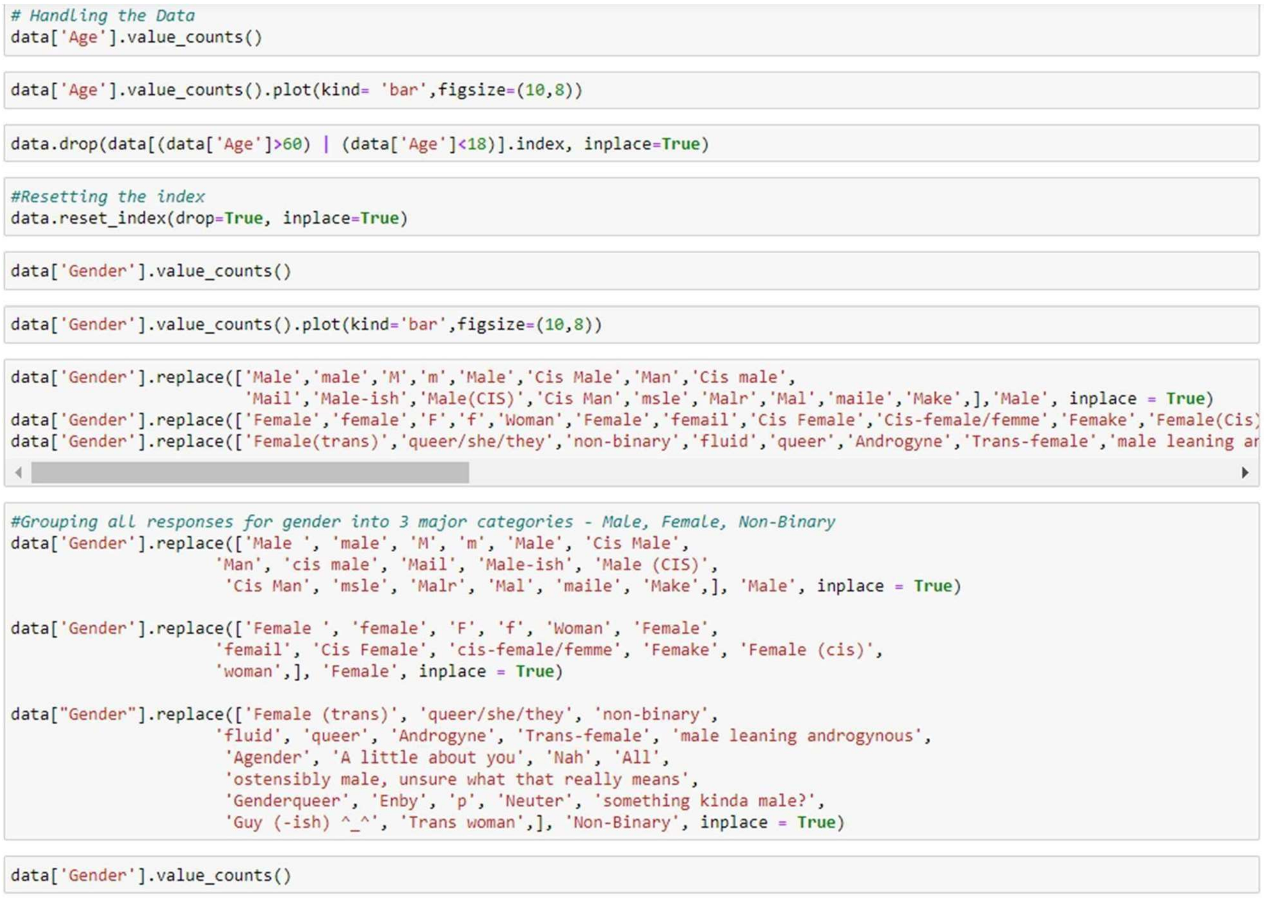
```
# Replacing nan with mode
print(data['self_employed'].value_counts())
data['self_employed'] = data['self_employed'].fillna(data['self_employed'].mode()[0])
```

```
#Replacing null values with No
data['self_employed'].fillna('No', inplace=True)
```

```
print(data['work_interfere'].value_counts())
data['work_interfere'] = data['work_interfere'].fillna(data['work_interfere'].mode()[0])
```

```
#Replacing null values with N/A
data['work_interfere'].fillna('N/A',inplace=True)
```

```
data.isnull().sum()
```



```
# Handling the Data
data['Age'].value_counts()
```

```
data['Age'].value_counts().plot(kind= 'bar',figsize=(10,8))
```

```
data.drop(data[(data['Age']>60) | (data['Age']<18)].index, inplace=True)
```

```
#Resetting the index
data.reset_index(drop=True, inplace=True)
```

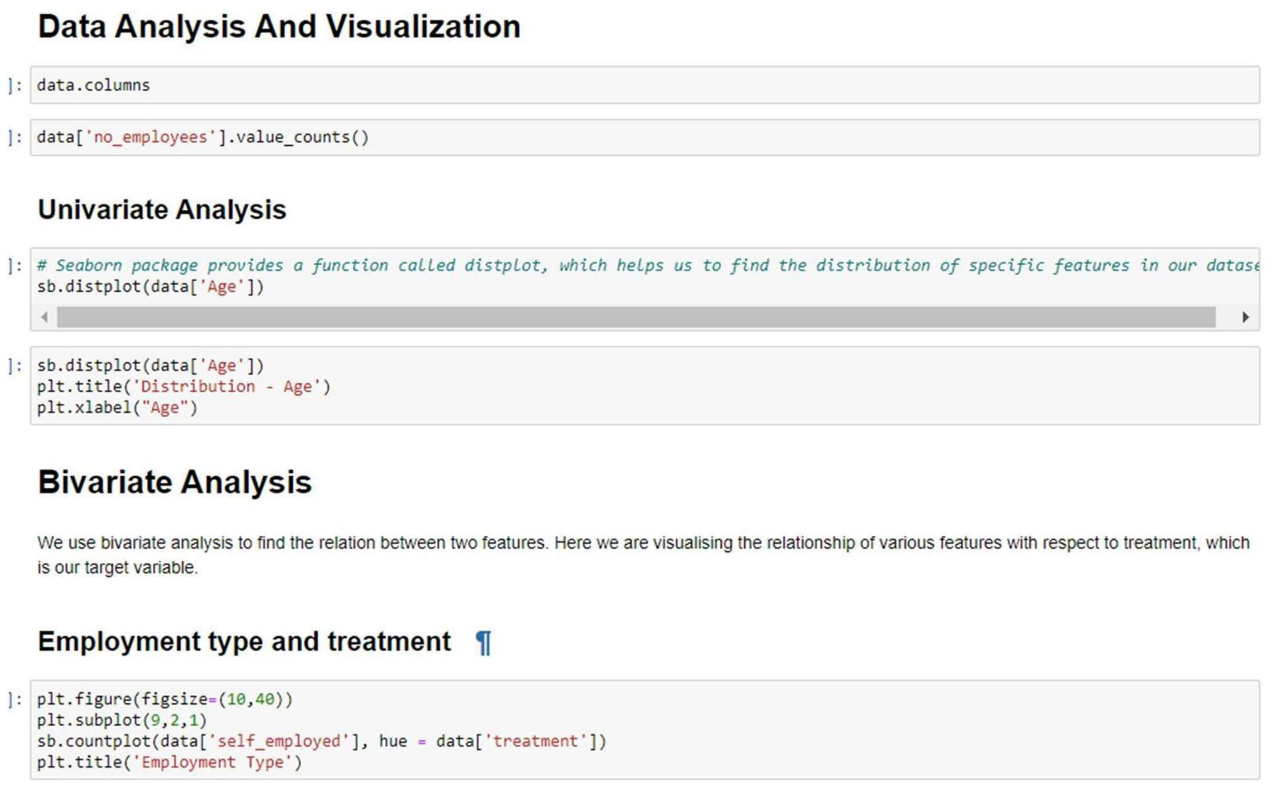
```
data['Gender'].value_counts()
```

```
data['Gender'].value_counts().plot(kind='bar',figsize=(10,8))
```

```
data['Gender'].replace(['Male','male','M','m','Male','Cis Male','Man','Cis male',
                        'Mail','Male-ish','Male(CIS)','Cis Man','msle','Malr','Mal','maile','Make',],'Male', inplace = True)
data['Gender'].replace(['Female','female','F','f','Woman','Female','femail','Cis Female','Cis-female/femme','Femake','Female(Cis)
data['Gender'].replace(['Female(trans)','queer/she/they','non-binary','fluid','queer','Androgyne','Trans-female','male leaning ar
```

```
#Grouping all responses for gender into 3 major categories - Male, Female, Non-Binary
data['Gender'].replace(['Male ', 'male', 'M', 'm', 'Male', 'Cis Male',
                     'Man', 'cis male', 'Mail', 'Male-ish', 'Male (CIS)',
                      'Cis Man', 'msle', 'Malr', 'Mal', 'maile', 'Make',], 'Male', inplace = True)

data['Gender'].replace(['Female ', 'female', 'F', 'f', 'Woman', 'Female',
                     'femail', 'Cis Female', 'cis-female/femme', 'Femake', 'Female (cis)',
                     'woman',], 'Female', inplace = True)

data["Gender"].replace(['Female (trans)', 'queer/she/they', 'non-binary',
                     'fluid', 'queer', 'Androgyne', 'Trans-female', 'male leaning androgynous',
                      'Agender', 'A little about you', 'Nah', 'All',
                      'ostensibly male, unsure what that really means',
                      'Genderqueer', 'Enby', 'p', 'Neuter', 'something kinda male?',
                      'Guy (-ish) ^_^', 'Trans woman',], 'Non-Binary', inplace = True)
```

```
data['Gender'].value_counts()
```



Data Analysis And Visualization

```
]: data.columns
```

```
]: data['no_employees'].value_counts()
```

Univariate Analysis

```
]: # Seaborn package provides a function called distplot, which helps us to find the distribution of specific features in our datase
   sb.distplot(data['Age'])
```

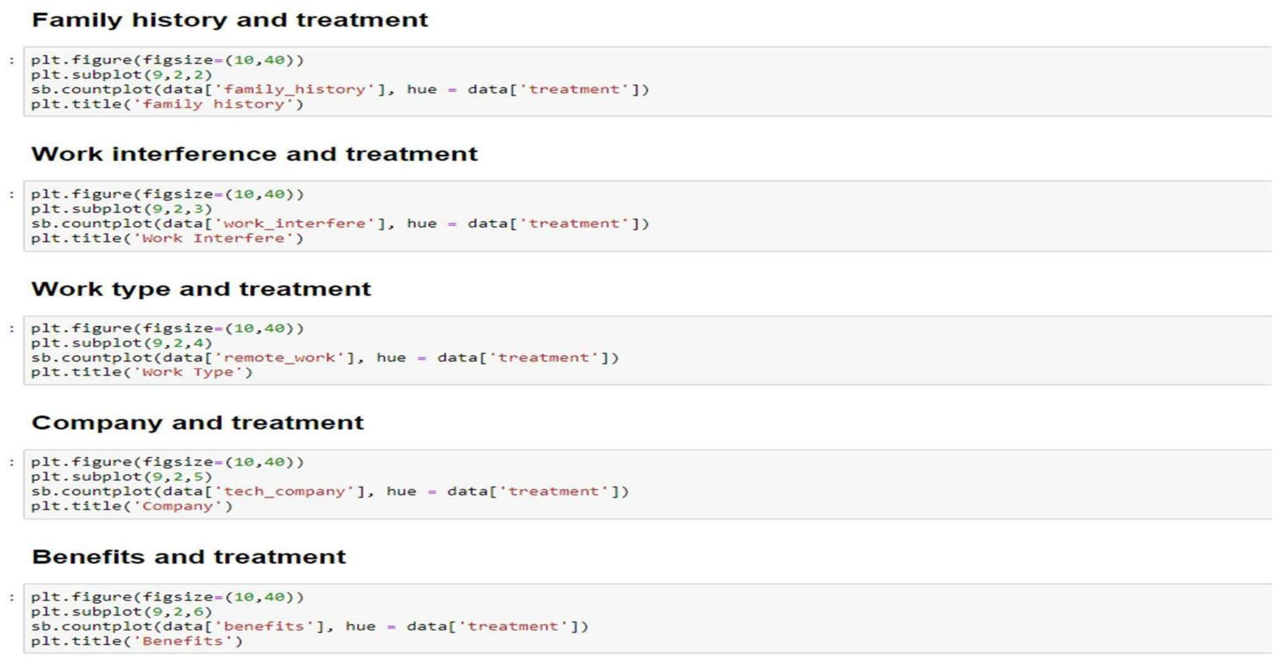
```
]: sb.distplot(data['Age'])
   plt.title('Distribution - Age')
   plt.xlabel("Age")
```

Bivariate Analysis

We use bivariate analysis to find the relation between two features. Here we are visualising the relationship of various features with respect to treatment, which is our target variable.

Employment type and treatment ¶

```
]: plt.figure(figsize=(10,40))
   plt.subplot(9,2,1)
   sb.countplot(data['self_employed'], hue = data['treatment'])
   plt.title('Employment Type')
```



Family history and treatment

```
: plt.figure(figsize=(10,40))
  plt.subplot(9,2,2)
  sb.countplot(data['family_history'], hue = data['treatment'])
  plt.title('family history')
```

Work interference and treatment

```
: plt.figure(figsize=(10,40))
  plt.subplot(9,2,3)
  sb.countplot(data['work_interfere'], hue = data['treatment'])
  plt.title('Work Interfere')
```

Work type and treatment

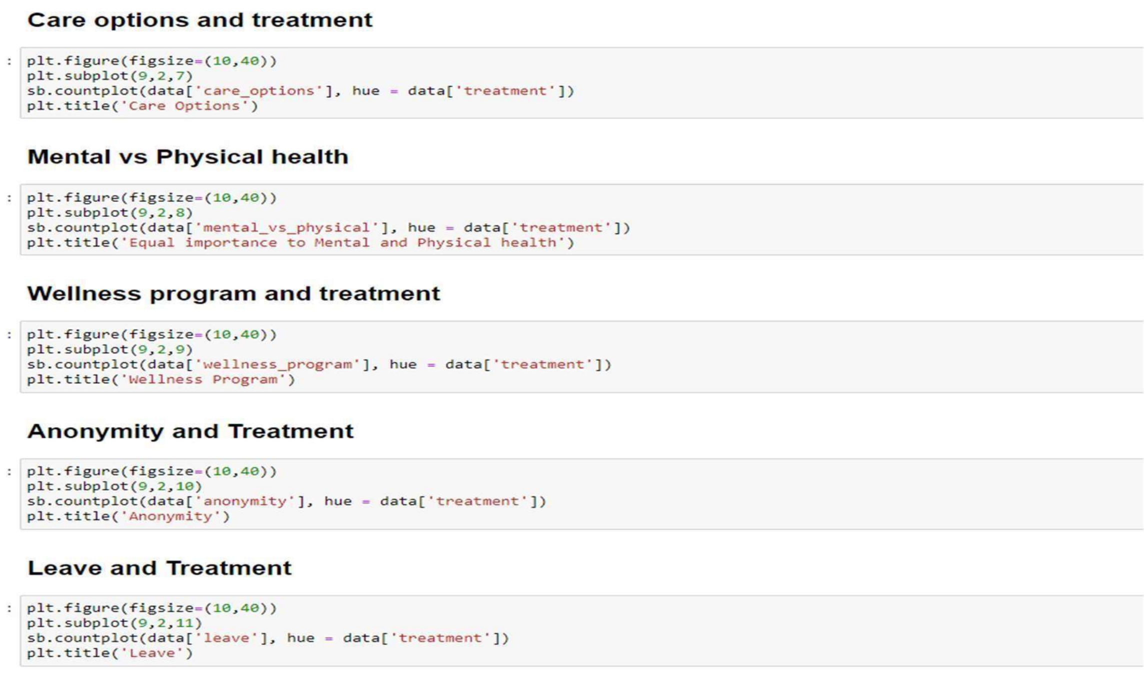
```
: plt.figure(figsize=(10,40))
  plt.subplot(9,2,4)
  sb.countplot(data['remote_work'], hue = data['treatment'])
  plt.title('Work Type')
```

Company and treatment

```
: plt.figure(figsize=(10,40))
  plt.subplot(9,2,5)
  sb.countplot(data['tech_company'], hue = data['treatment'])
  plt.title('Company')
```

Benefits and treatment

```
: plt.figure(figsize=(10,40))
  plt.subplot(9,2,6)
  sb.countplot(data['benefits'], hue = data['treatment'])
  plt.title('Benefits')
```



Care options and treatment

```
plt.figure(figsize=(10,40))
plt.subplot(9,2,7)
sb.countplot(data['care_options'], hue = data['treatment'])
plt.title('Care Options')
```

Mental vs Physical health

```
plt.figure(figsize=(10,40))
plt.subplot(9,2,8)
sb.countplot(data['mental_vs_physical'], hue = data['treatment'])
plt.title('Equal importance to Mental and Physical health')
```

Wellness program and treatment

```
plt.figure(figsize=(10,40))
plt.subplot(9,2,9)
sb.countplot(data['wellness_program'], hue = data['treatment'])
plt.title('Wellness Program')
```

Anonymity and Treatment

```
plt.figure(figsize=(10,40))
plt.subplot(9,2,10)
sb.countplot(data['anonymity'], hue = data['treatment'])
plt.title('Anonymity')
```

Leave and Treatment

```
plt.figure(figsize=(10,40))
plt.subplot(9,2,11)
sb.countplot(data['leave'], hue = data['treatment'])
plt.title('Leave')
```

Mental health consequence and treatment

```
plt.figure(figsize=(10,40))
plt.subplot(9,2,12)
sb.countplot(data['mental_health_consequence'], hue = data['treatment'])
plt.title('mental Health consequence')
```

Physical health consequence and treatment

```
plt.figure(figsize=(10,40))
plt.subplot(9,2,13)
sb.countplot(data['phys_health_consequence'], hue = data['treatment'])
plt.title('Physical Health Consequence')
```

Discussion with coworkers and treatment

```
plt.figure(figsize=(10,40))
plt.subplot(9,2,14)
sb.countplot(data['coworkers'], hue = data['treatment'])
plt.title('Discussion with Coworkers')
```

Discussion with supervisor and treatment

```
plt.figure(figsize=(10,40))
plt.subplot(9,2,15)
sb.countplot(data['supervisor'], hue = data['treatment'])
plt.title('Discussion with Supervisor')
```

Mental health discussion during interview and treatment

```
plt.figure(figsize=(10,40))
plt.subplot(9,2,16)
sb.countplot(data['mental_health_interview'], hue = data['treatment'])
plt.title('Discussion with Interviewer(mental)')
```

Physical health discussion during interview and treatment

```
# plt.figure(figsize=(10,40))
plt.subplot(9,2,17)
sb.countplot(data['phys_health_interview'], hue = data['treatment'])
plt.title('Discussion with Interviewer(Physical)')
```

The consequence after disclosure and treatment

```
plt.figure(figsize=(10,40))
plt.subplot(9,2,18)
sb.countplot(data['obs_consequence'], hue = data['treatment'])
plt.title('Consequence after Disclosure')
```

Descriptive Analysis

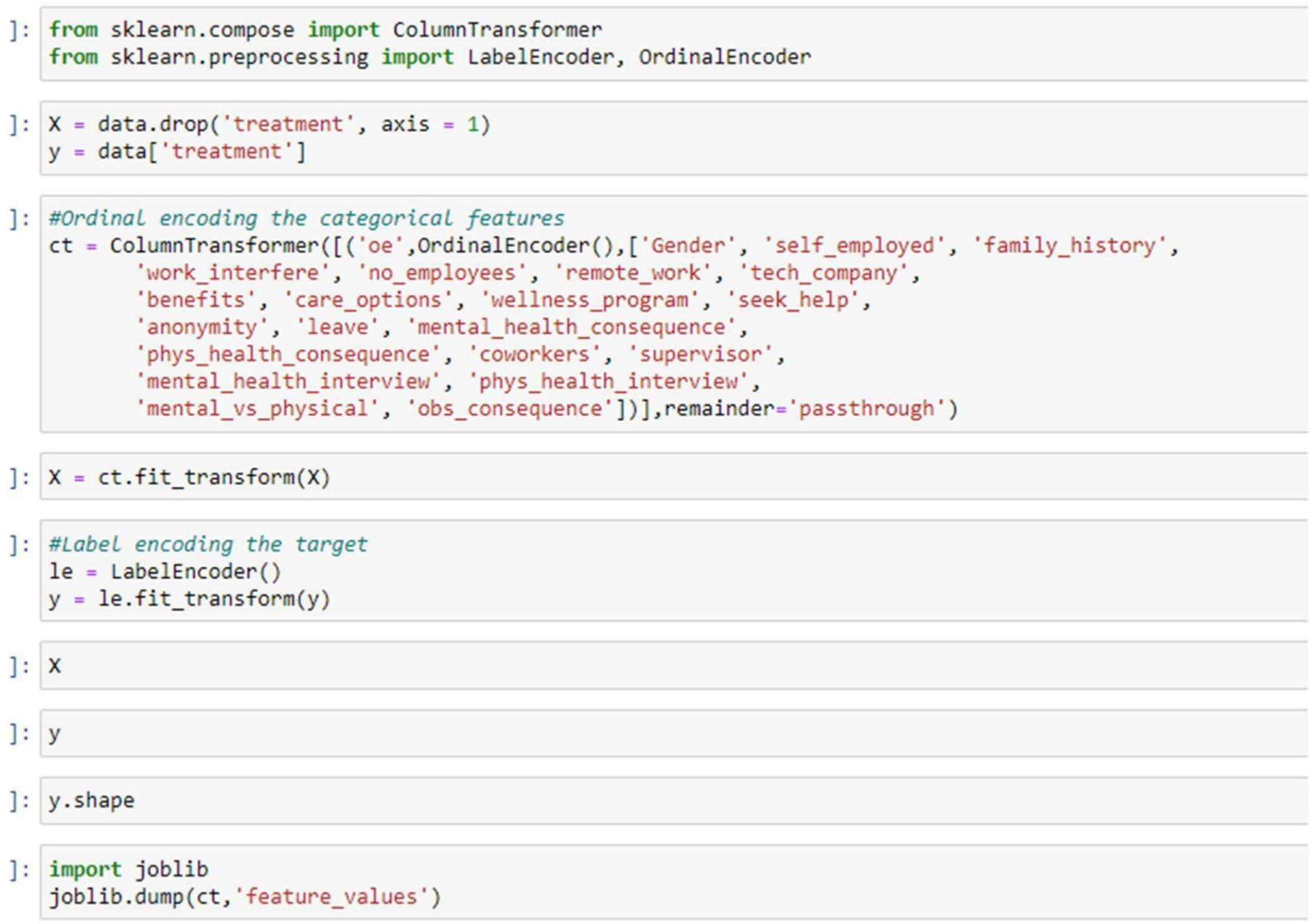
```
data.describe(include='all')
```

Model Building

Encoding

```
obj_cols = ['Gender', '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']
```

Handling Categorical Values



```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
```

```
X = data.drop('treatment', axis = 1)
y = data['treatment']
```

```
#Ordinal encoding the categorical features
ct = ColumnTransformer([('oe',OrdinalEncoder(),['Gender', 'self_employed', 'family_history',
       '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'])],remainder='passthrough')
```

```
X = ct.fit_transform(X)
```

```
#Label encoding the target
le = LabelEncoder()
y = le.fit_transform(y)
```

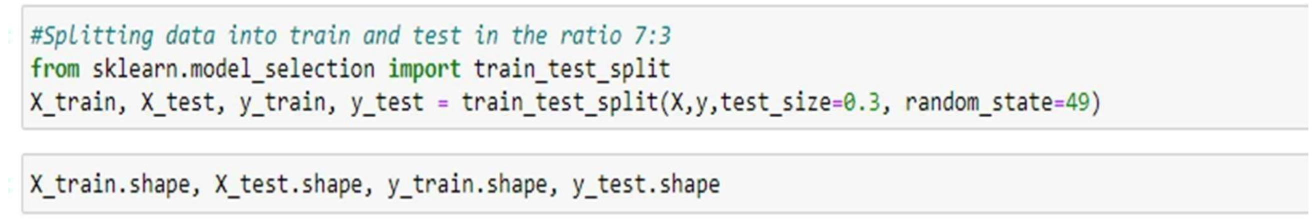
```
X
```

```
y
```

```
y.shape
```

```
import joblib
joblib.dump(ct,'feature_values')
```

Splitting Data into Train and Test



```
#Splitting data into train and test in the ratio 7:3
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=49)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Comparing Accuracy Of Various Models

Comparing Accuracy Of Various Models

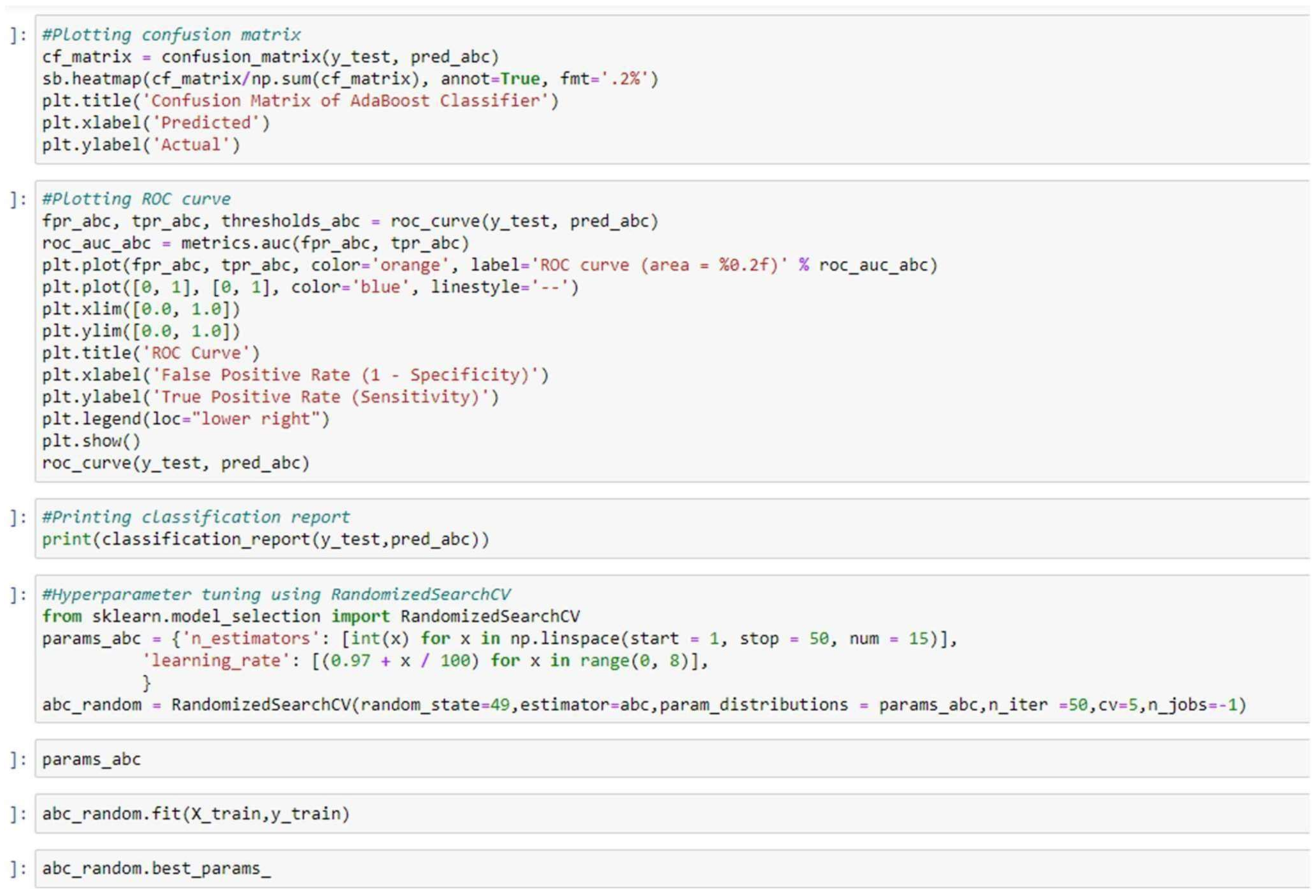
```python
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from xgboost.sklearn import XGBClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score, roc_curve, confusion_matrix, classification_report, auc
```

```python
#Creating a dictionary of all models
model_dict = {}

model_dict['Logistic regression']= LogisticRegression(solver='liblinear',random_state=49)
model_dict['KNN Classifier'] = KNeighborsClassifier()
model_dict['Decision Tree Classifier'] = DecisionTreeClassifier(random_state=49)
model_dict['Random Forest Classifier'] = RandomForestClassifier(random_state=49)
model_dict['AdaBoost Classifier'] = AdaBoostClassifier(random_state=49)
model_dict['Gradient Boosting Classifier'] = GradientBoostingClassifier(random_state=49)
model_dict['XGB Classifier'] = XGBClassifier(random_state=49)
```

```python
#function to print accuracy of all models
def model_test(X_train, X_test, y_train, y_test,model,model_name):
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    print('======================================{}======================================='.format(model_name))
    print('Score is : {}'.format(accuracy))

    print()
```

```python
for model_name,model in model_dict.items():
    model_test(X_train, X_test, y_train, y_test, model, model_name)
```

```python
#Fitting data to AdaBoost classifier
abc = AdaBoostClassifier(random_state=99)
abc.fit(X_train,y_train)
pred_abc = abc.predict(X_test)
print('Accuracy of AdaBoost=',accuracy_score(y_test,pred_abc))
```



```
]: #Plotting confusion matrix
   cf_matrix = confusion_matrix(y_test, pred_abc)
   sb.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%')
   plt.title('Confusion Matrix of AdaBoost Classifier')
   plt.xlabel('Predicted')
   plt.ylabel('Actual')
```

```
]: #Plotting ROC curve
   fpr_abc, tpr_abc, thresholds_abc = roc_curve(y_test, pred_abc)
   roc_auc_abc = metrics.auc(fpr_abc, tpr_abc)
   plt.plot(fpr_abc, tpr_abc, color='orange', label='ROC curve (area = %0.2f)' % roc_auc_abc)
   plt.plot([0, 1], [0, 1], color='blue', linestyle='--')
   plt.xlim([0.0, 1.0])
   plt.ylim([0.0, 1.0])
   plt.title('ROC Curve')
   plt.xlabel('False Positive Rate (1 - Specificity)')
   plt.ylabel('True Positive Rate (Sensitivity)')
   plt.legend(loc="lower right")
   plt.show()
   roc_curve(y_test, pred_abc)
```

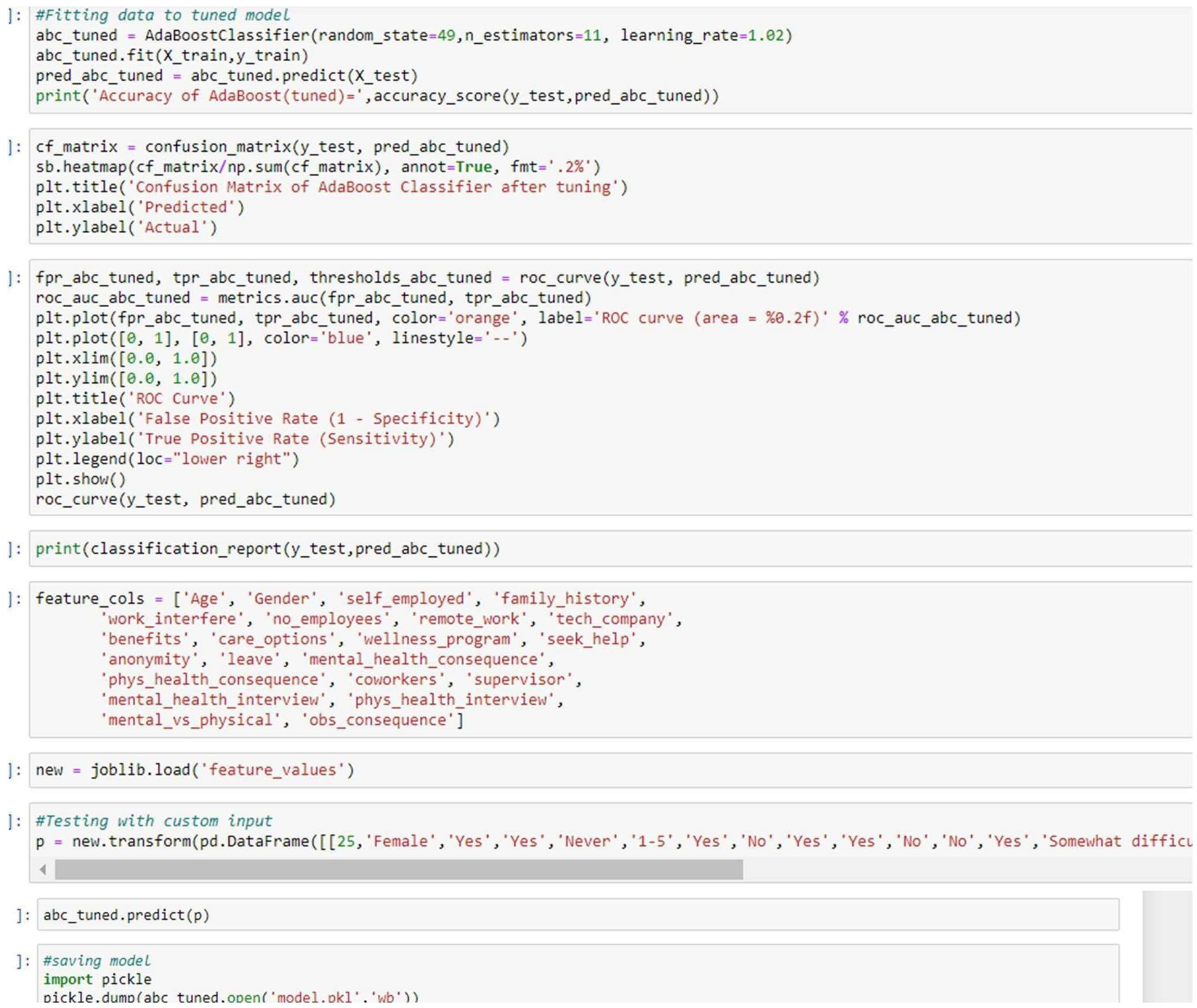
```
]: #Printing classification report
   print(classification_report(y_test,pred_abc))
```

```
]: #Hyperparameter tuning using RandomizedSearchCV
   from sklearn.model_selection import RandomizedSearchCV
   params_abc = {'n_estimators': [int(x) for x in np.linspace(start = 1, stop = 50, num = 15)],
             'learning_rate': [(0.97 + x / 100) for x in range(0, 8)],
             }
   abc_random = RandomizedSearchCV(random_state=49,estimator=abc,param_distributions = params_abc,n_iter =50,cv=5,n_jobs=-1)
```

```
]: params_abc
```

```
]: abc_random.fit(X_train,y_train)
```

```
]: abc_random.best_params_
```



```
#Fitting data to tuned model
abc_tuned = AdaBoostClassifier(random_state=49,n_estimators=11, learning_rate=1.02)
abc_tuned.fit(X_train,y_train)
pred_abc_tuned = abc_tuned.predict(X_test)
print('Accuracy of AdaBoost(tuned)=',accuracy_score(y_test,pred_abc_tuned))
```

```
cf_matrix = confusion_matrix(y_test, pred_abc_tuned)
sb.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%')
plt.title('Confusion Matrix of AdaBoost Classifier after tuning')
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

```
fpr_abc_tuned, tpr_abc_tuned, thresholds_abc_tuned = roc_curve(y_test, pred_abc_tuned)
roc_auc_abc_tuned = metrics.auc(fpr_abc_tuned, tpr_abc_tuned)
plt.plot(fpr_abc_tuned, tpr_abc_tuned, color='orange', label='ROC curve (area = %0.2f)' % roc_auc_abc_tuned)
plt.plot([0, 1], [0, 1], color='blue', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC Curve')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.legend(loc="lower right")
plt.show()
roc_curve(y_test, pred_abc_tuned)
```

```
print(classification_report(y_test,pred_abc_tuned))
```

```
feature_cols = ['Age', 'Gender', 'self_employed', 'family_history',
       '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']
```

```
new = joblib.load('feature_values')
```

```
#Testing with custom input
p = new.transform(pd.DataFrame([[25,'Female','Yes','Yes','Never','1-5','Yes','No','Yes','Yes','No','No','Yes','Somewhat difficu
```

```
abc_tuned.predict(p)
```

```
#saving model
import pickle
pickle.dump(abc_tuned,open('model.pkl','wb'))
```

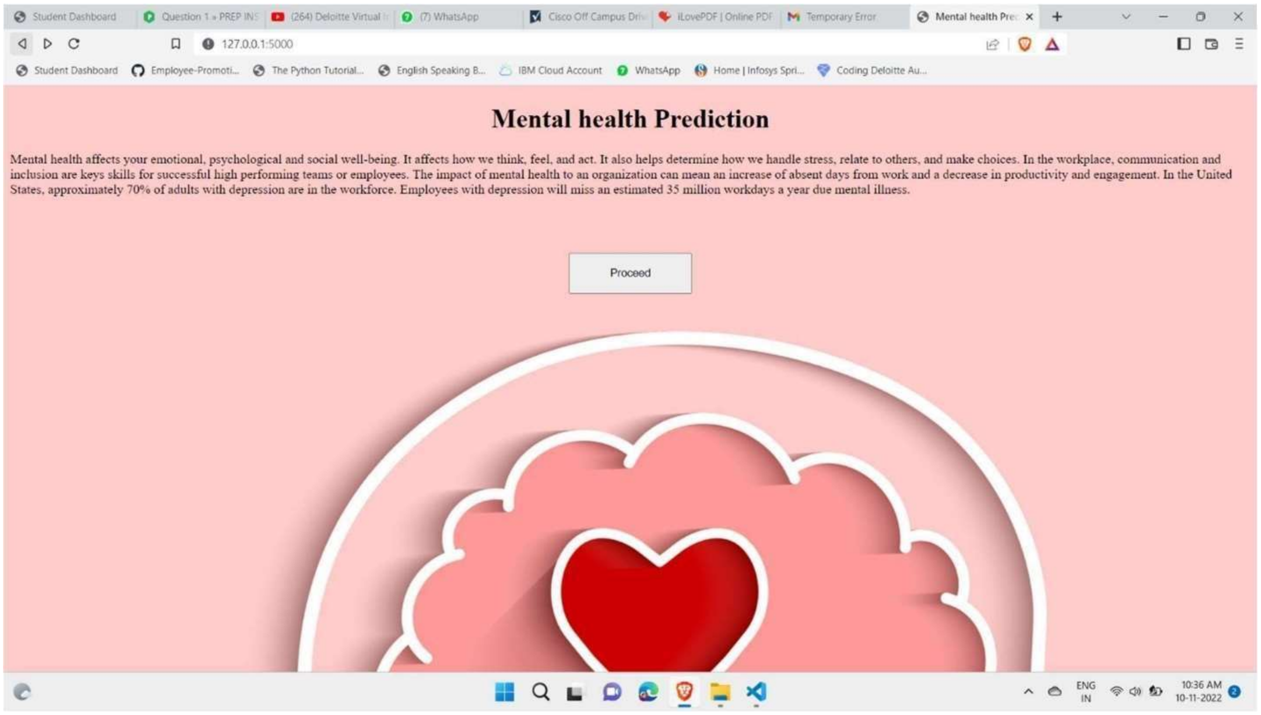
\

FLASK APP

```
from flask import Flask, render_template, request
import pickle, joblib
import pandas as pd

app = Flask(__name__)

model = pickle.load(open("model.pkl","rb"))
ct = joblib.load('feature_values')

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

@app.route('/pred')
def predict():
    return render_template("index.html")

@app.route('/out', methods =["POST"])
def output():
    age = request.form["age"]
    gender = request.form["gender"]
    self_employed = request.form["self_employed"]
    family_history = request.form["family_history"]
    work_interfere = request.form["work_interfere"]
    no_employees = request.form["no_employees"]
    remote_work = request.form["remote_work"]
    tech_company = request.form["tech_company"]
    benefits = request.form["benefits"]
    care_options = request.form["care_options"]
    wellness_program = request.form["wellness_program"]
    seek_help = request.form["seek_help"]
    anonymity = request.form["anonymity"]
    leave = request.form["leave"]
    mental_health_consequence = request.form["mental_health_consequence"]
    phys_health_consequence = request.form["phys_health_consequence"]
    coworkers = request.form["coworkers"]
    supervisor = request.form["supervisor"]
    mental_health_interview = request.form["mental_health_interview"]
    phys_health_interview = request.form["phys_health_interview"]
    mental_vs_physical = request.form["mental_vs_physical"]
    obs_consequence = request.form["obs_consequence"]
    data = 
[[age,gender,self_employed,family_history,work_interfere,no_employees,remote_work,
             tech_company,benefits,care_options,wellness_program,seek_help,anonymity,le
ave,
             mental_health_consequence,phys_health_consequence,coworkers,supervisor,
             mental_health_interview,phys_health_interview,mental_vs_physical,obs_conse
quence]]


    feature_cols = ['Age', 'Gender', 'self_employed', 'family_history',
       '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']

    pred = model.predict(ct.transform(pd.DataFrame(data,columns=feature_cols)))
    pred = pred[0]
    if pred:
        return render_template("output.html",y="This person requires mental health 
treatment ")
    else:
        return render_template("output.html",y="This person doesn't require mental 
health treatment ")



if___name___== '_main_':
    app.run(debug = True)
```

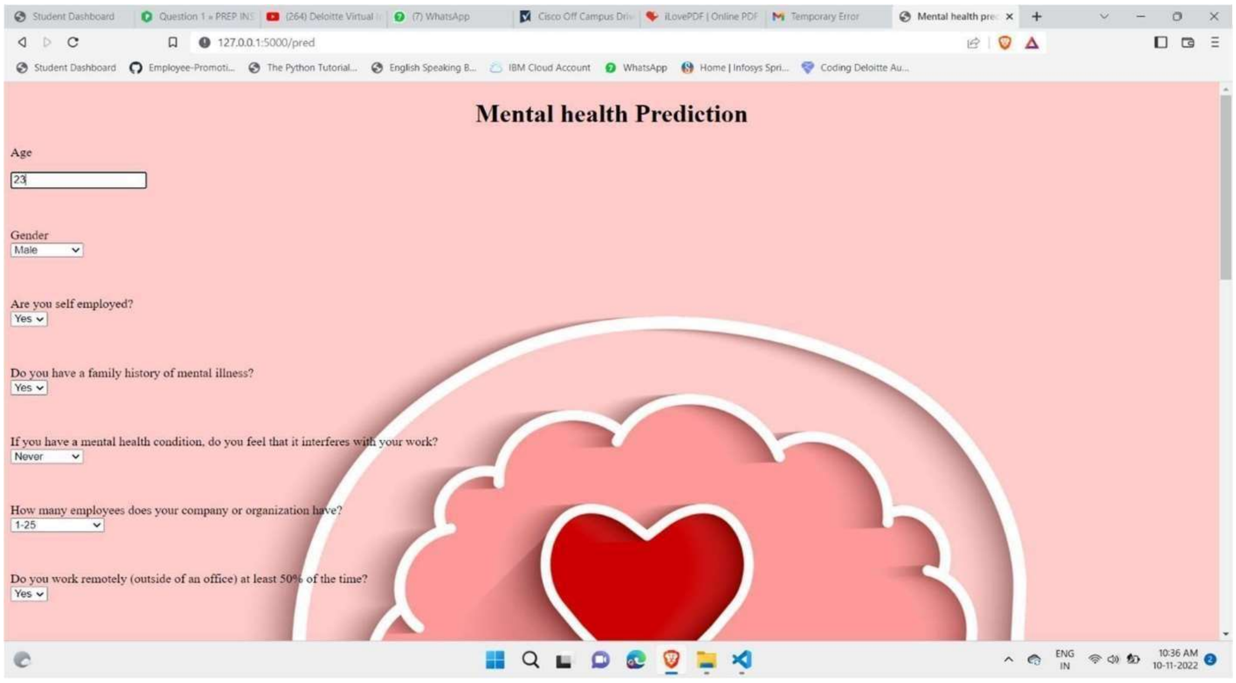
7. RESULT

This study identified five machine learning techniques i.e. k nearest neighbor classifier, logistic regression, decision tree, and stacking, and random forest. And we assessed their accuracy in identifying mental health issues.



Mental health Prediction

Mental health affects your emotional, psychological and social well-being. It affects how we think, feel, and act. It also helps determine how we handle stress, relate to others, and make choices. In the workplace, communication and inclusion are keys skills for successful high performing teams or employees. The impact of mental health to an organization can mean an increase of absent days from work and a decrease in productivity and engagement. In the United States, approximately 70% of adults with depression are in the workforce. Employees with depression will miss an estimated 35 million workdays a year due mental illness.



Mental health Prediction

Age
23

Gender
Male

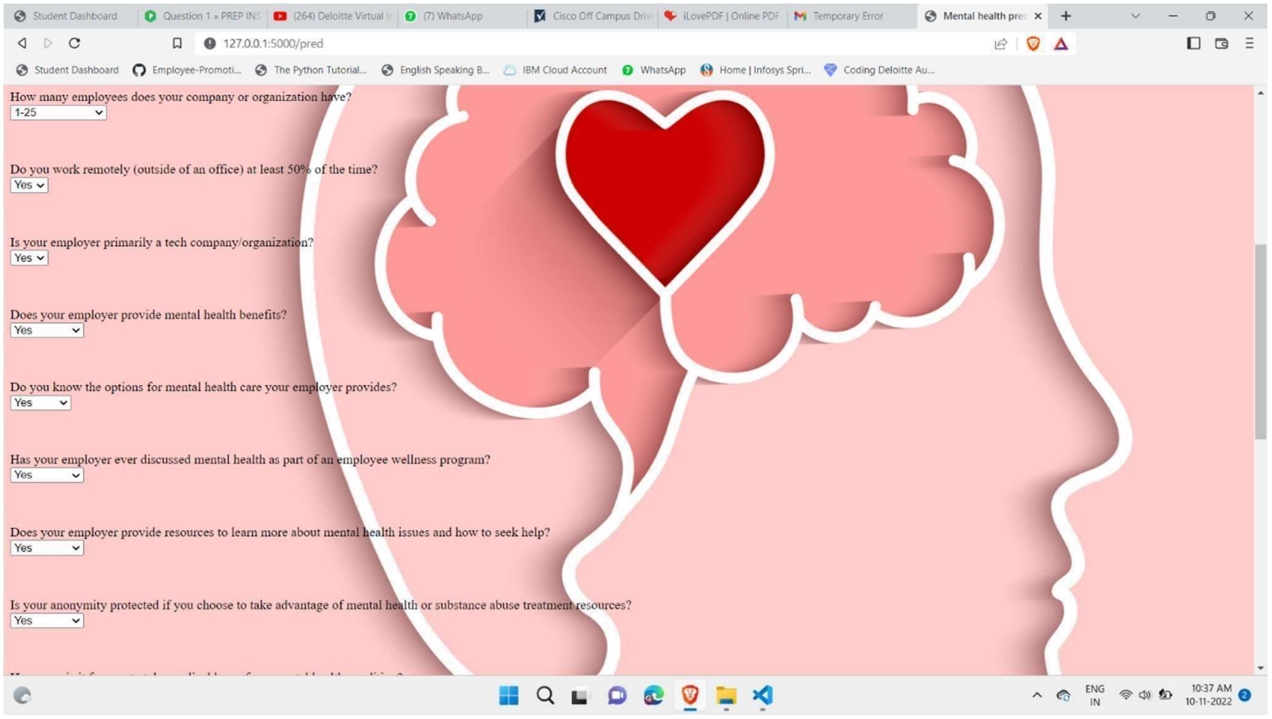
Are you self employed?
Yes

Do you have a family history of mental illness?
Yes

If you have a mental health condition, do you feel that it interferes with your work?
Never

How many employees does your company or organization have?
1-25

Do you work remotely (outside of an office) at least 50% of the time?
Yes



How many employees does your company or organization have?
1-25

Do you work remotely (outside of an office) at least 50% of the time?
Yes

Is your employer primarily a tech company/organization?
Yes

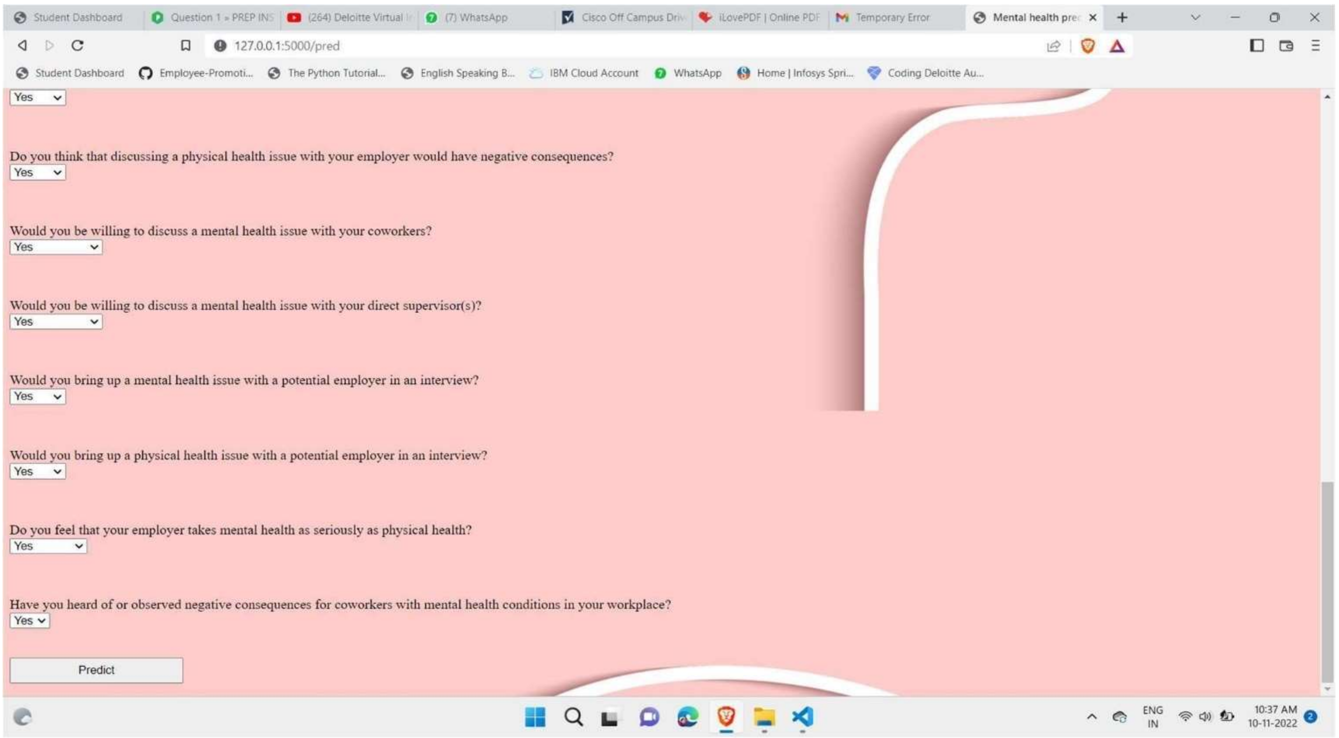
Does your employer provide mental health benefits?
Yes

Do you know the options for mental health care your employer provides?
Yes

Has your employer ever discussed mental health as part of an employee wellness program?
Yes

Does your employer provide resources to learn more about mental health issues and how to seek help?
Yes

Is your anonymity protected if you choose to take advantage of mental health or substance abuse treatment resources?
Yes


Yes

Do you think that discussing a physical health issue with your employer would have negative consequences?
Yes

Would you be willing to discuss a mental health issue with your coworkers?
Yes

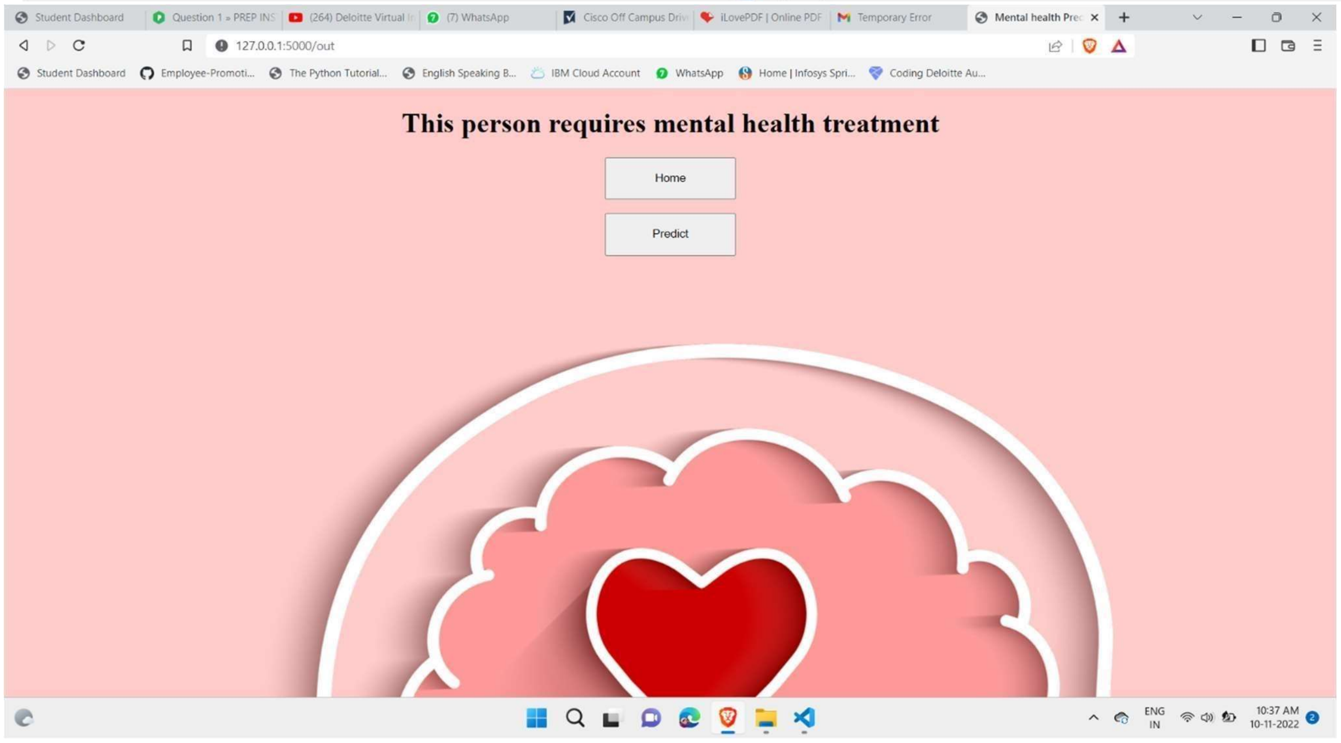
Would you be willing to discuss a mental health issue with your direct supervisor(s)?
Yes

Would you bring up a mental health issue with a potential employer in an interview?
Yes

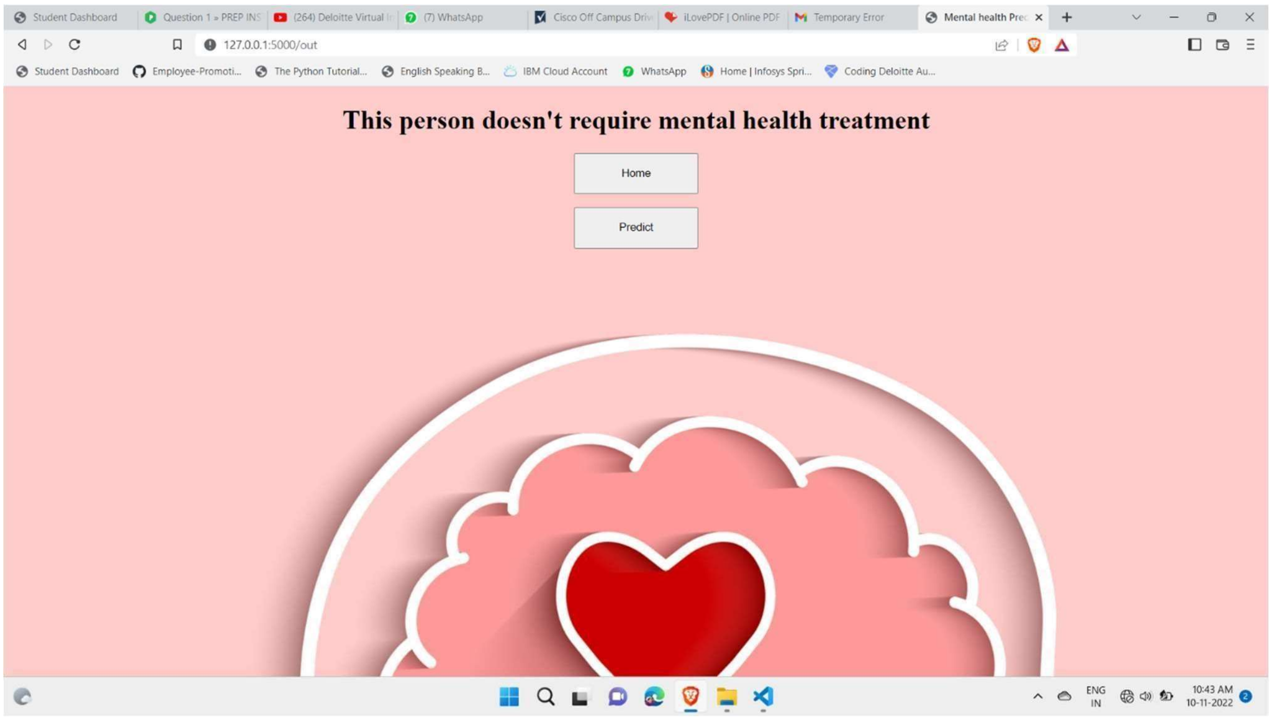
Would you bring up a physical health issue with a potential employer in an interview?
Yes

Do you feel that your employer takes mental health as seriously as physical health?
Yes

Have you heard of or observed negative consequences for coworkers with mental health conditions in your workplace?
Yes


This person requires mental health treatment

After changing the Prediction value’s we can obtain the output based on input.


This person doesn't require mental health treatment

8. ADVANTAGES AND DISADVANTAGES

Advantages

There is an endless number of advantages of Machine learning. We can take a look at the one which arereally helpful.

The advantages of Machine learning tell us how using Machine learning would benefit us.So, let’s have aLook at the advantages of Machine Learning.

- Automation of Everything
- Wide Range of Applications
- Scope of Improvement
- Efficient Handling of Data
- Best for Mental Health

Disadvantages

Similar to the advantages of Machine Learning, we should also know the disadvantages of Machine Learning. If you don't know the cons, you won't know the risks of Machine Learning. So; let’s have a look at these disadvantages.

- Possibility of High Error
- Algorithm Selection
- Data Acquisition
- Time and Space
- Internet Issues

9. CONCLUSION

Many different techniques and algorithms had been introduced and proposed to test and solve the mental health problems. There are still many solutions that can be refined. In addition, there are still many problems to be discovered and tested using a wide variety of settings in machine learning for the mental health domain. As classifying the mental health data is generally a very challenging problem, the features used in the machine learning algorithms will significantly affect the performance of the classification.

The existing studies and research show that machine learning can be a useful tool in helping understand psychiatric disorders. Besides that, it may also help distinguish and classify the mental health problems among patients for further treatment. Newer approaches that use datathat arise from the integration of various sensor modalities present in technologically advanced devices have proven to be a convenient resource to recognize the mood state and responses from patients among others.

It is noticeable that most of the research and studies are still struggling to validate the results because of insufficiency of acceptable validated evidence, especially from the external sources. Besides that, most of the machine learning might not have the same performance across all the problems. The performance of the machine learning models will vary depending on the data samples obtained and the features of the data. Moreover, machine learning models can also be affected by preprocessing activities such as data cleaning and parameter tuning in order to achieve optimal results.

Hence, it is very important for researchers to investigate and analyze the data with various machine learning algorithms to choose the highest accuracy among the machine learning algorithms. Not only has that, challenges and limitations faced by the researchers needed to be managed with proper care to achieve satisfactory results that could improve the clinical practice and decision making.

10. FUTURE SCOPE

We can extend the Mental Health Prediction Project by using different constraints. Various machine learning approaches can be implemented to predict or detect a disease at its early stages so that the treatment for it would be less complex and it would increase the probability of the patient being cured. As a result of these approaches, different types of disease can been detected but With diverse accuracy levels depending on factors such as the used algorithm, feature set, training dataset, and so on.

11. BIBILOGRAPHY

Installation of Anaconda Navigator:

https://www.youtube.com/embed/5mDYijMfSzs

Installation of Pycharm Professionals:

https://www.youtube.com/embed/z73PyNDgVyQ

Installation of Python Packages:

https://www.youtube.com/embed/akj3_wTploUData

Collection:

https://www.kaggle.com/datasets/rishal005/admission-predict

Data Pre-processing:

https://thesmartbridge.com/documents/spsaimldocs/Datapreprocessing.pdf

Handling Null Values:

https://towardsdatascience.com/7-ways-to-handle-missing-values-in-machine-learning-1a6326adf79e

Data Visualization:

https://www.youtube.com/embed/TLdXM0A7SR8

Splitting Dependent and Independent Columns:

https://www.youtube.com/embed/A_V6daPQZIU

Splitting the Data into Train and Test:
https://www.youtube.com/embed/xgDs0scjuuQ

Trainingand Testing the Model:

https://www.youtube.com/embed/yIYKR4sgzI8

Model Evaluation:

https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know- aa97784ff226

Flask Frame Work Reference:

https://www.youtube.com/embed/lj4I_CvBnt0

Flask Reference To Run:
https://www.youtube.com/embed/UbCWoMf80PY

Train The Model On IBM:

Account Creation:

https://www.youtube.com/embed/4y_zD-0Q3F8

Train Model on IBM Watson:

https://www.youtube.com/embed/TysuP3KgSzc

Integrate Flask With Scoring Endpoint:

https://www.youtube.com/embed/ST1ZYLmYw2U