Mental Health Prediction Using ML

Project Description:

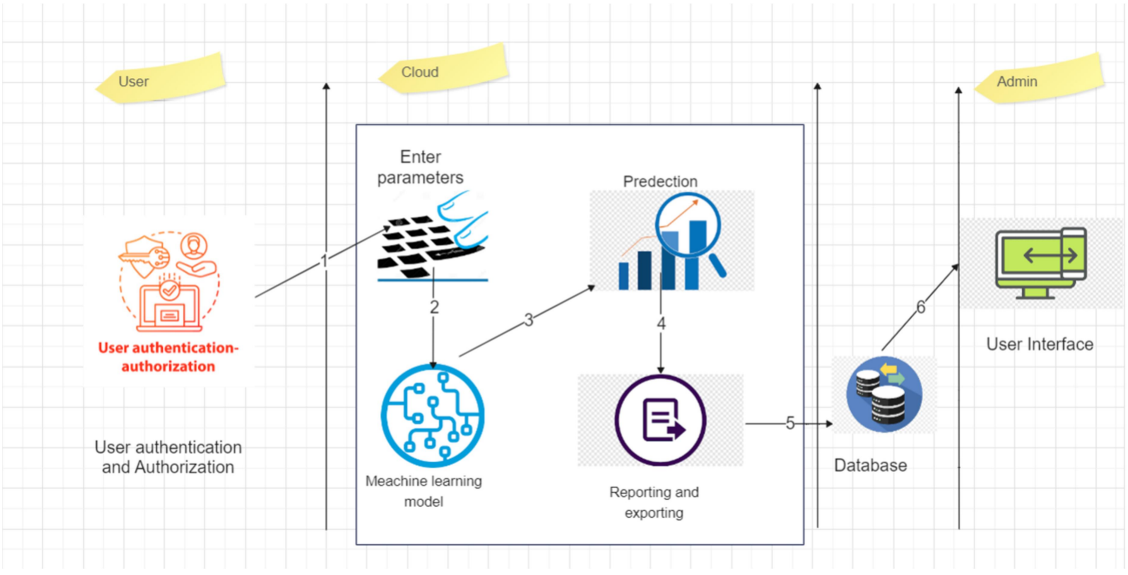
Mental Health First Aid teaches participants how to notice and support an individual who may be experiencing a mental health or substance use concern or crisis and connect them with the appropriate employee resources.

Employers can offer robust benefit 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.Organisations 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-workers 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, GradientBoost 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.

Technical Architecture:



Pre requisites:

To complete this project, you will require the following softwares, concepts and packages

- **Anaconda navigator:**
  - Refer the link below to download anaconda navigator
  - Link : https://youtu.be/1ra4zH2G4o0
- **Python packages:**

- Type “pip install pandas” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type ”pip install matplotlib” and click enter.
- Type ”pip install scipy” and click enter.
- Type ”pip install pickle-mixin” and click enter.
- Type ”pip install seaborn” and click enter.
- Type “pip install Flask” and click enter.

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
  - Regression and classification
    - Logistic regression: https://www.javatpoint.com/logistic-regression-in-machine-learning
    - Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classificati on-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/
    - AdaBoost: https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/
    - Gradient Boost: https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/
    - 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 Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning. • Gain a broad understanding about data.
- Have knowledge on pre-processing the data/transformation techniques and some visualisation concepts before building the model
- Learn how to build a machine learning model and tune it for better performance •

Know how to evaluate the model and deploy it using flask

Project Flow:

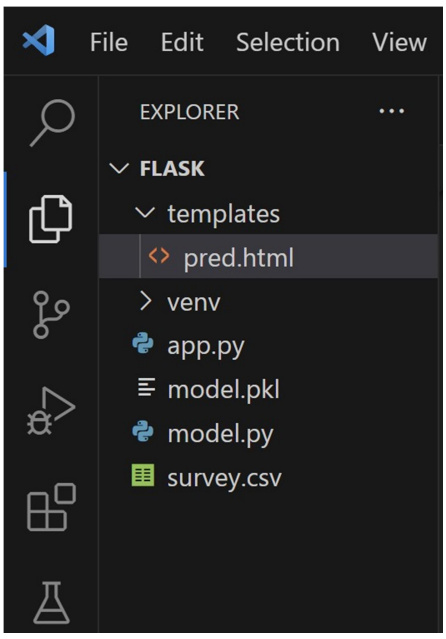
- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- The predictions made by the model is showcased on the UI

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

- Data collection
  - Collect the dataset or create the dataset
- Data pre-processing
  - Removing unnecessary columns
  - Checking for null values
- Visualising and analysing data
  - Univariate analysis
  - Bivariate analysis
  - Descriptive analysis
- Model building
  - Handling categorical values
  - Dividing data into train and test sets
  - Import the model building libraries
  - Comparing accuracy of various models
  - Hyperparameter tuning of the selected model
  - Evaluating performance of models
  - Save the model
- Application Building
  - Create an HTML file
  - Build python code

Project Structure:

Create the Project folder which contains files as shown below



- We are building a flask application which needs the HTML pages to be stored in the templates folder and a python script app.py for scripting.
- Notebook folder contains model training file MentalHealth.ipynb
- model.pkl is our saved model. We will use this model for flask integration.

Milestone 1: Data Collection

Activity 1: Download the dataset

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

In this project we have used survey.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

Load the dataset using read_csv() function:

```
[3] import pandas as pd
    import numpy as np
    np.set_printoptions(suppress=True)
```

DATA COLLECTION AND PRE-PROCESSING

```
[4] df = pd.read_csv('/content/survey.csv')
    df.head()
```

Inside the read_csv() function, specify the path to your dataset.
To observe the first 5 rows of our data, we use the head() method and to observe the last 5 rows of the data, we use the tail() method.

```
df = pd.read_csv('/content/survey.csv')
df.head()
```

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

5 rows × 27 columns

We can use the ‘shape’ attribute of the dataframe to know the shape of our dataset:

```
[5]  df.shape

     (1259, 27)
```

From the above figure, we can say that our dataset has 1259 rows and 27 columns

Next, we will have to see the information pertaining to each of the 27 columns. For that, we will be using info() function:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Timestamp                  1259 non-null   object
 1   Age                        1259 non-null   int64
 2   Gender                     1259 non-null   object
 3   Country                    1259 non-null   object
 4   state                      744 non-null    object
 5   self_employed              1241 non-null   object
 6   family_history             1259 non-null   object
 7   treatment                  1259 non-null   object
 8   work_interfere             995 non-null    object
 9   no_employees               1259 non-null   object
 10  remote_work                1259 non-null   object
 11  tech_company               1259 non-null   object
 12  benefits                   1259 non-null   object
 13  care_options               1259 non-null   object
 14  wellness_program           1259 non-null   object
 15  seek_help                  1259 non-null   object
 16  anonymity                  1259 non-null   object
 17  leave                      1259 non-null   object
 18  mental_health_consequence  1259 non-null   object
 19  phys_health_consequence    1259 non-null   object
 20  coworkers                  1259 non-null   object
 21  supervisor                 1259 non-null   object
 22  mental_health_interview    1259 non-null   object
 23  phys_health_interview      1259 non-null   object
 24  mental_vs_physical         1259 non-null   object
 25  obs_consequence            1259 non-null   object
 26  comments                   164 non-null    object
dtypes: int64(1), object(26)
memory usage: 265.7+ KB
```

Milestone 2: Data Pre-processing

We need to pre-process the collected data before gaining insights and building our model.

We need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

Activity 1: Removing unnecessary columns:

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

Here we are not going to remove columns .

Activity 2: Handling Null values and dealing with wrongly entered data

To check for null values, .isnull() function is used along with .sum() function to the dataframe.

```
df.isnull().sum()
```

```
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
```

We observe that 2 columns - self_employed and work_interfere contain null values. Let us fill the self_employed column with ‘No’ and work_interfere column with ‘N/A’ in place of null values. In order to do this, we will make use of .fillna() function

```
[8]  df['state'].fillna(df['state'].mode()[0],inplace=True)
     df['self_employed'].fillna(df['self_employed'].mode()[0],inplace=True)
     df['work_interfere'].fillna("N/A",inplace=True)
     df['comments'].fillna(df['comments'].mode()[0],inplace=True)
```

Now our dataset is free of null values.

Let us now handle data that might have been entered wrongly. Consider the Age column of our dataset.

If we observe the Age column, it is seen that some impractical values have been entered. So, let us remove the rows with ages based on the median values .

```
[49] q1 = df.Age.quantile(0.25)
     q3 = df.Age.quantile(0.75)
     print(q1)
     print(q3)

     27.0
     36.0

[50] IQR =q3-q1

[51] upper_limit = q3+1.5*IQR
     upper_limit

     49.5

[52] lower_limit = q1-1.5*IQR
     lower_limit

     13.5

[53] df.median()

     <ipython-input-53-6d467abf240d>:1: FutureWarning: The default value of numeric_only in DataFrame.m
       df.median()
     Age    31.0
     dtype: float64

[54] df['Age'] = np.where(df['Age']>upper_limit,31,df['Age'])
```

Next, consider the Gender column of our dataset.

It is observed that different names are used for the same category of gender. Let us group them into 3 major categories- Male, Female and Non-Binary using the .replace() function.

```
Converting Gender column into 3 major categories Male,Female,Non-Binary

[119] df['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)

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

      df['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)

[61] df.head()
```

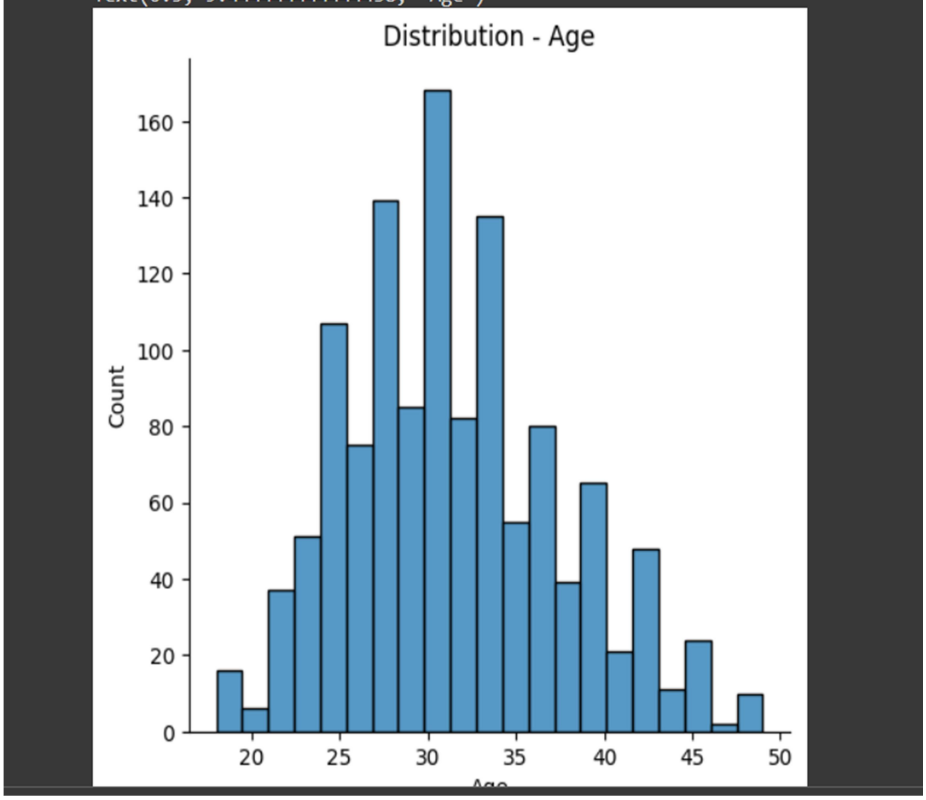
In simple words, univariate analysis is understanding the data with a single feature.

- Seaborn package provides a function called distplot, which helps us to find the distribution of specific features in our dataset. Let us observe the distribution of age.

DATA VISUALISATION

```
[64] import matplotlib.pyplot as plt
     sns.displot(df["Age"])
     plt.title("Distribution - Age")
     plt.xlabel("Age")
```

```
Text(0.5, 9.444444444444438, 'Age')
```

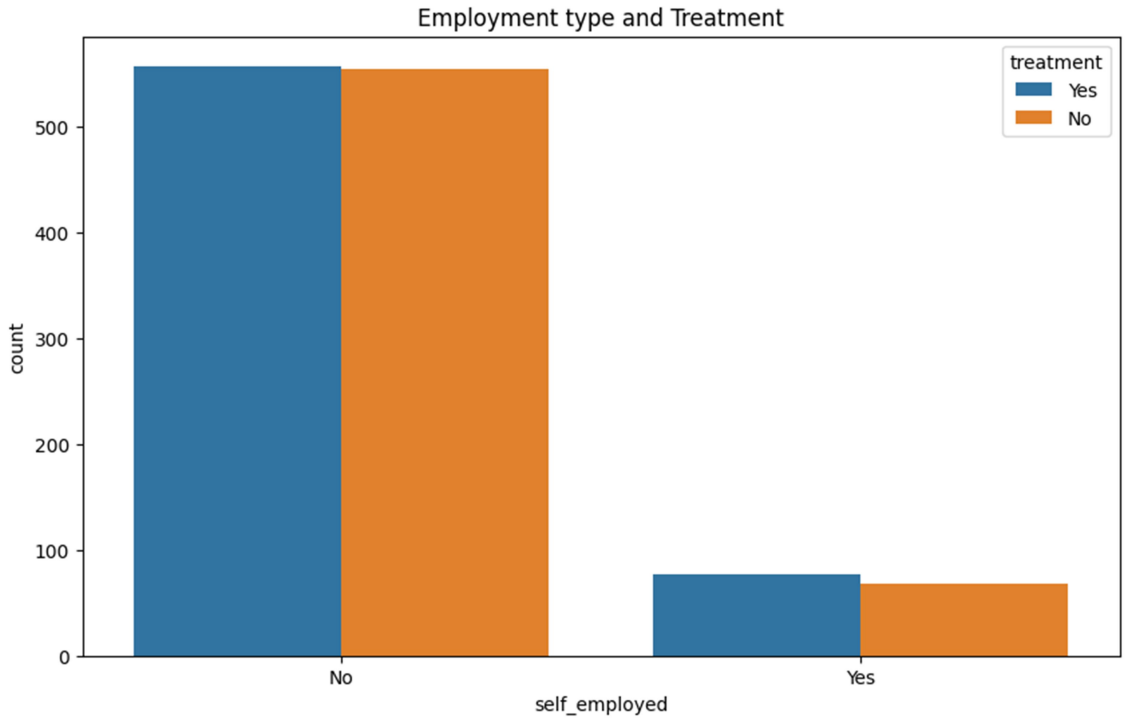


Activity 2: Bivariate analysis

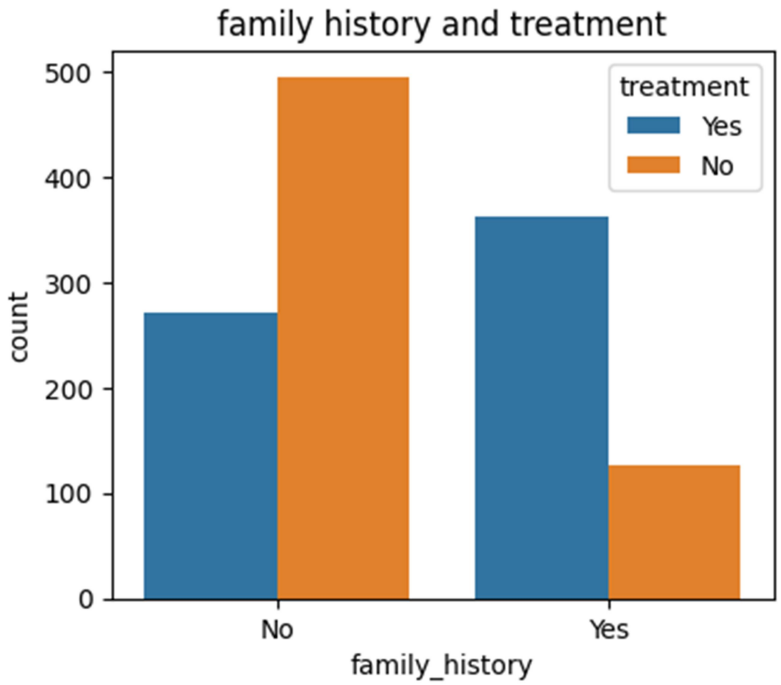
We use bivariate analysis to find the relation between two features. Here we are visualising the relationship of various features with respect to treatment, which is our target variable.

- Employment type and treatment
  We observe that though there is a vast difference between people who are self employed or not, the number of people who seek treatment in both the categories is more or less similar.

```
#bivariate analysis
#employment and treatment
plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
plt.subplot(1, 1, 1)  # Adjust the subplot as needed
sns.countplot(data=df, x='self_employed', hue='treatment')
plt.title("Employment type and Treatment")
plt.show()
```

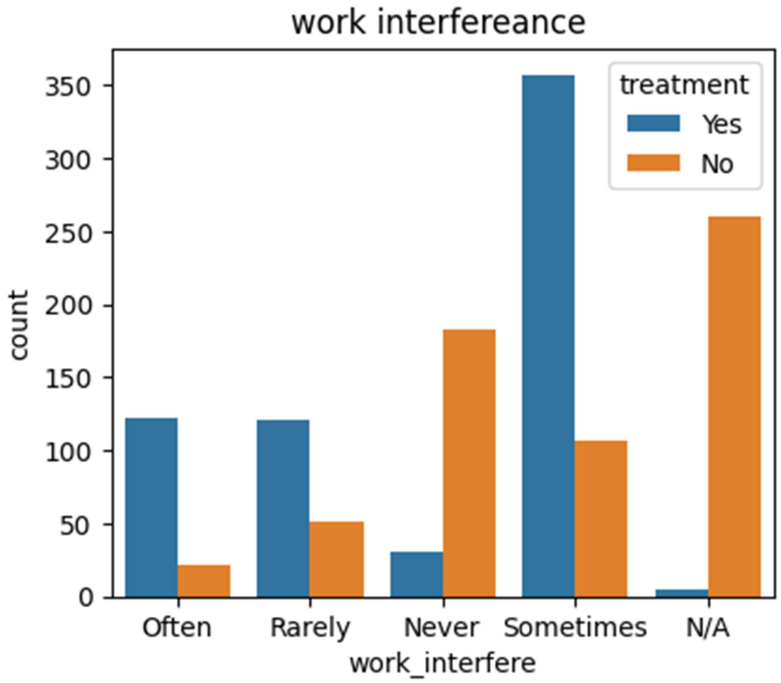


- Family history and treatment



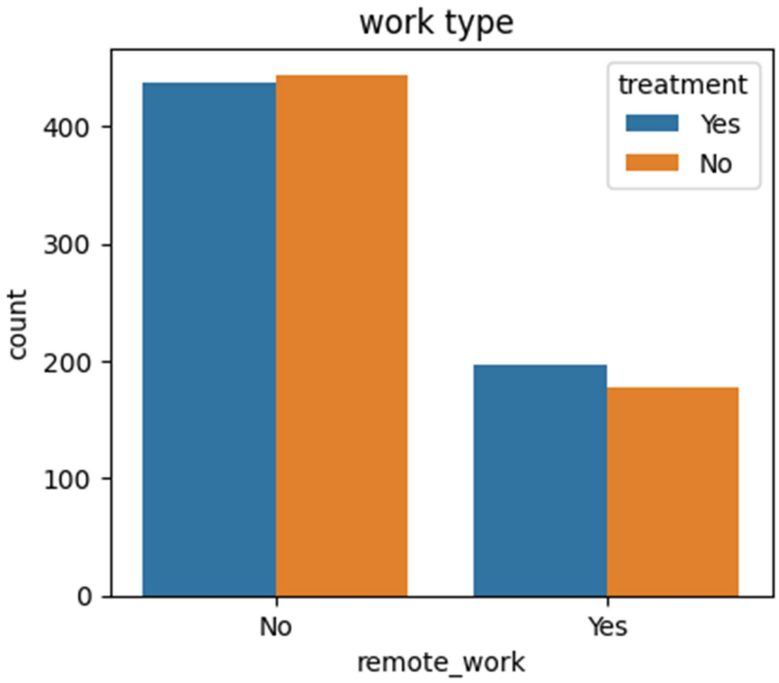
We observe that treatment is directly proportional to family history. Hence this is an important factor.

- Work interference and treatment

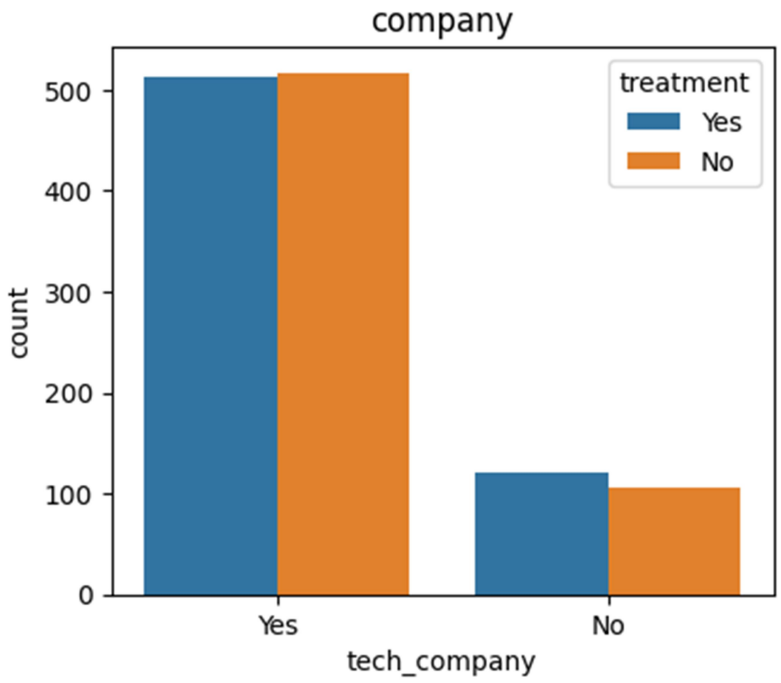


We observe that the people who chose Sometimes were the largest who wanted to get treatment. These group of people are the ones who are reluctant to choose either of the extreme categories.

- Work type and treatment
  We observe that the number of people who seek treatment in both the categories is more or less similar and it does not affect our target variable.

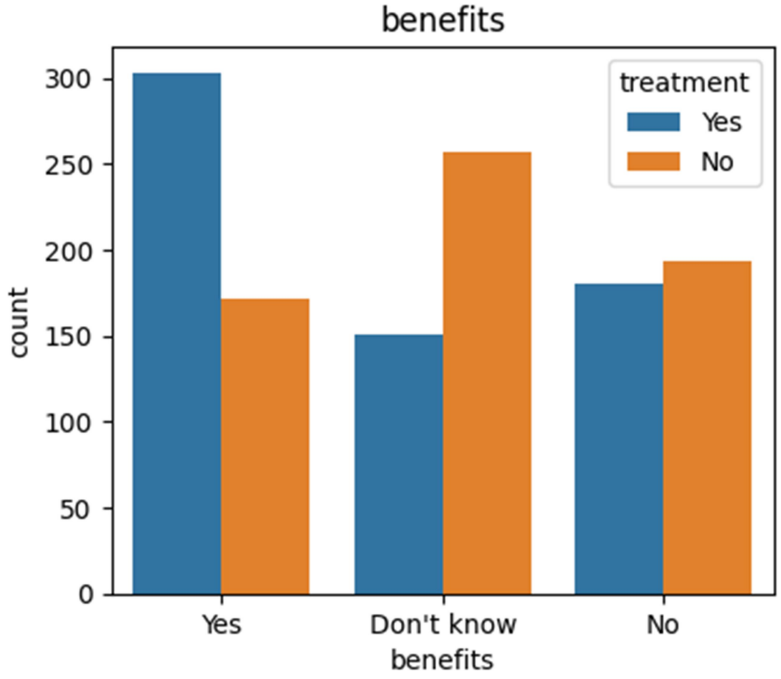


- Company and treatment



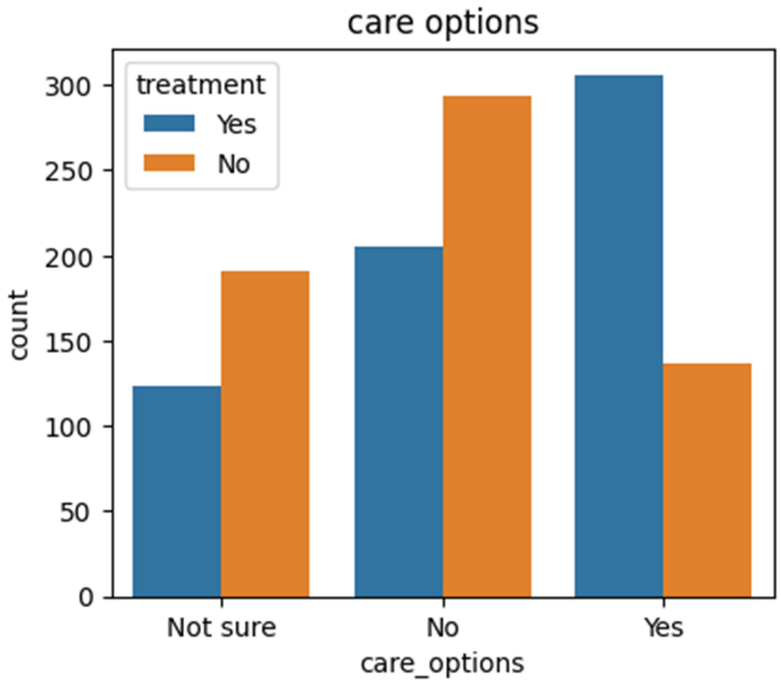
We can conclude that irrespective of the field the company of the people falls in, mental health is a big issue.

● Benefits and treatment



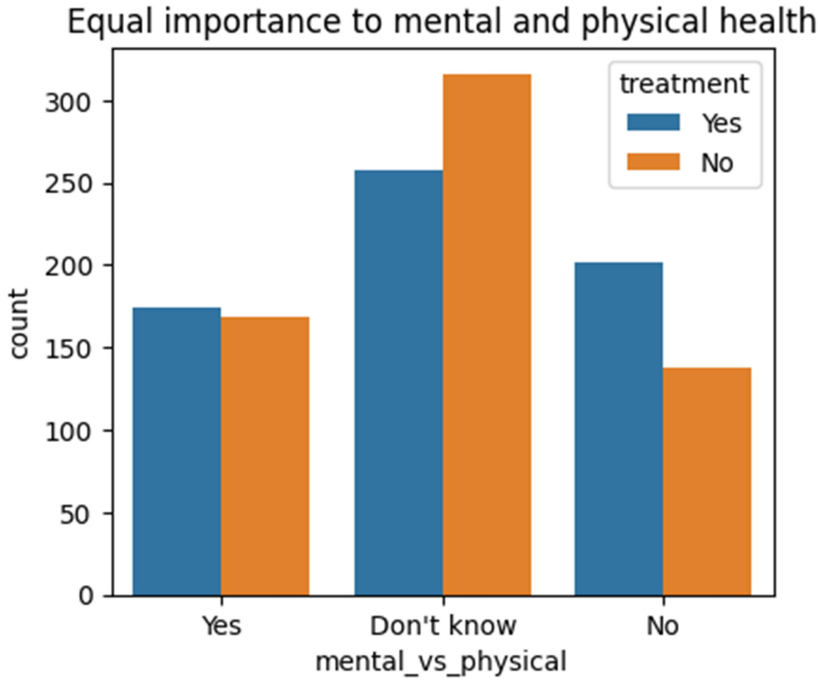
We see that a large group among the people who wanted mental health benefits wanted to seek treatment and also a significant number of people who said No too, wanted to seek treatment.

● Care options and treatment



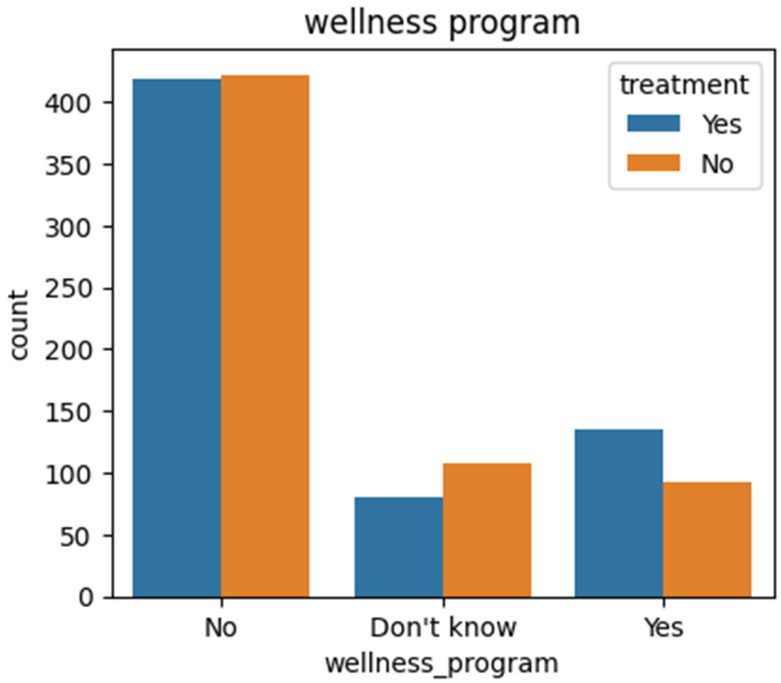
This graph is quite similar to the benefits column.

- Mental vs Physical health



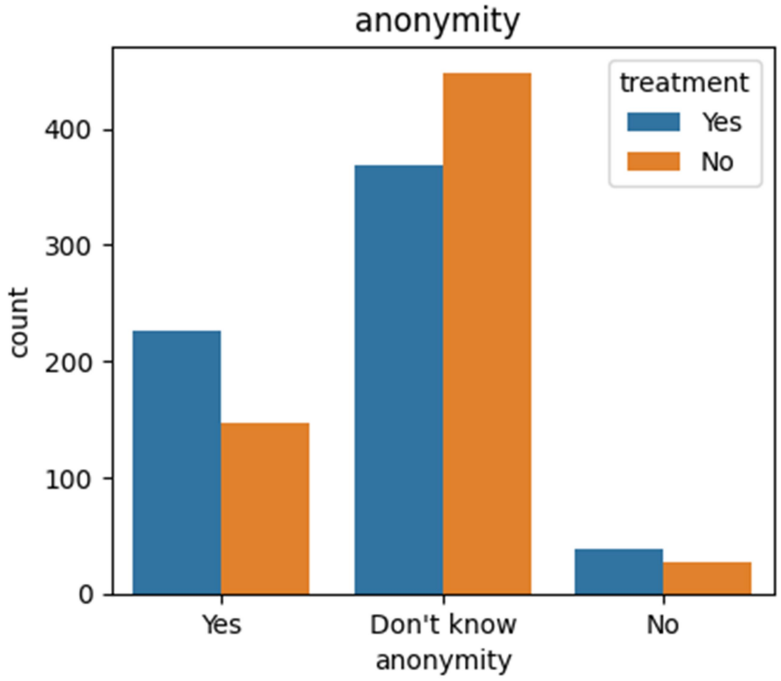
We observe that half of the people are not aware of the importance given to mental health as compared to physical health, whereas almost equal parts of the other halves answered Yes and No.

- Wellness program and treatment



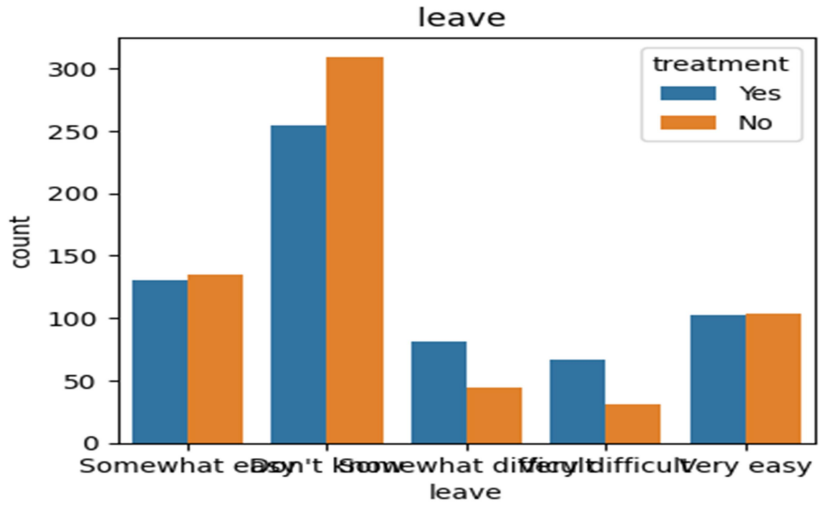
We observe that almost half of the people who said No want to seek treatment, which means their company has to take steps in arranging for the same.

- Anonymity and treatment



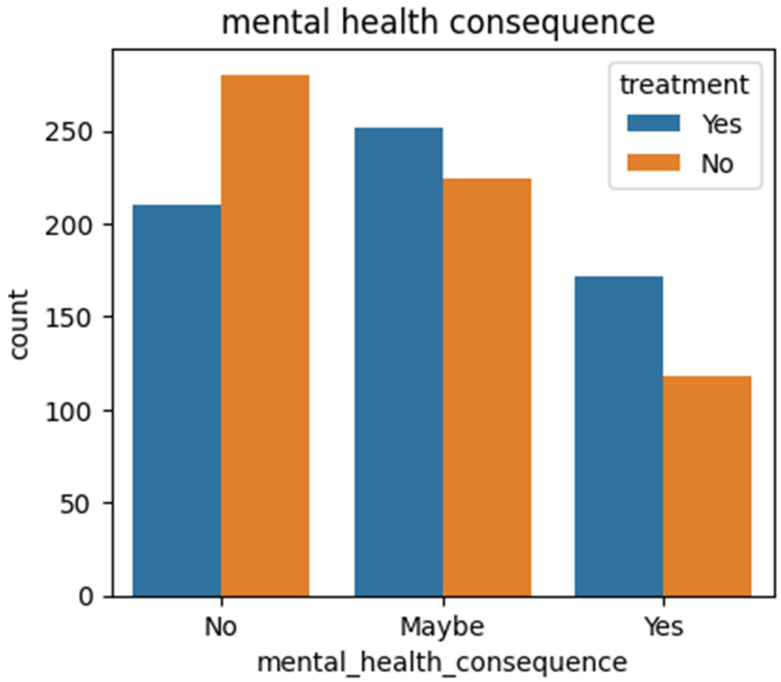
We observe that most people either answered yes or they are not aware if their anonymity will be protected.

- Leave and treatment



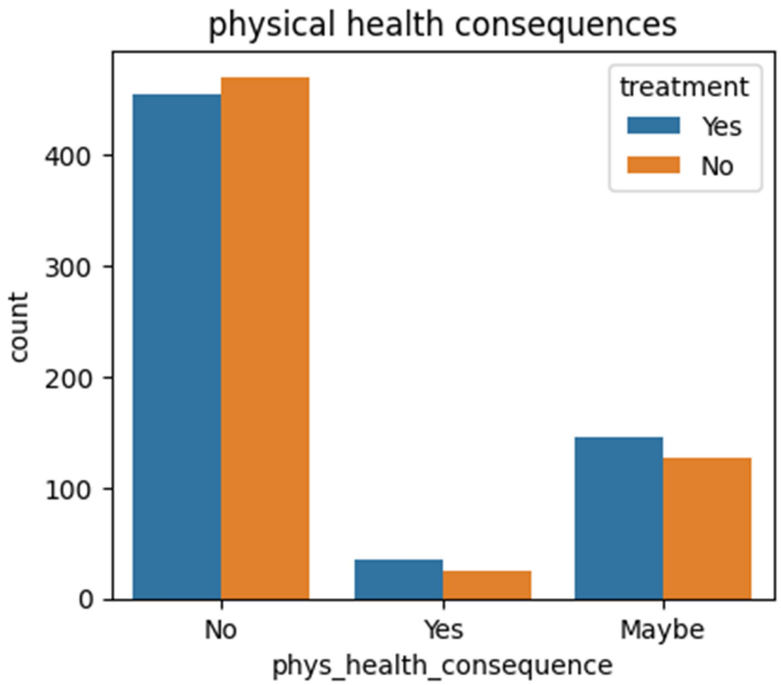
We see that around half of the total people don't know how easy it is to get a leave due to a mental health condition and they are the ones who want to seek treatment the most.

- Mental health consequence and treatment



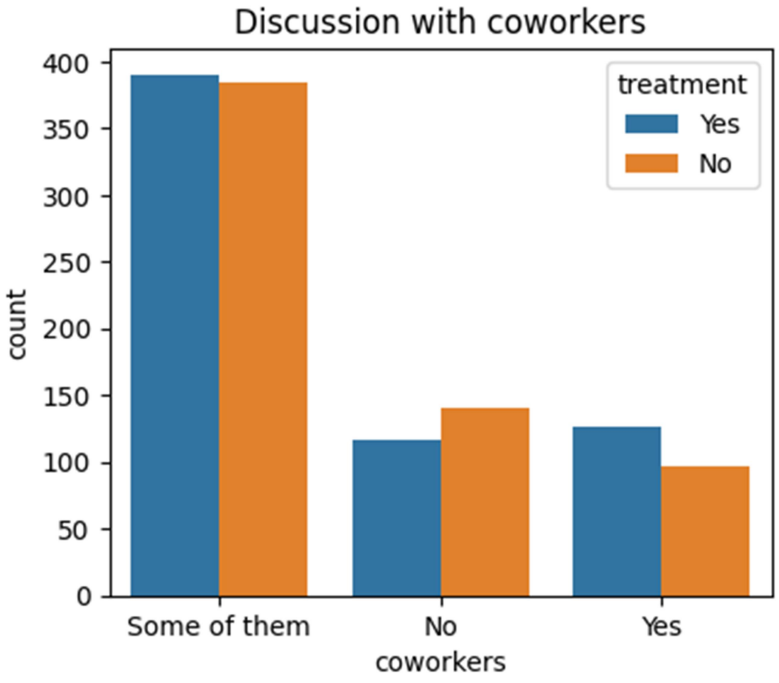
We observe that the majority answered either No or Maybe but around 1/3rd of the people answered yes and they want to seek treatment.

- Physical health consequence and treatment

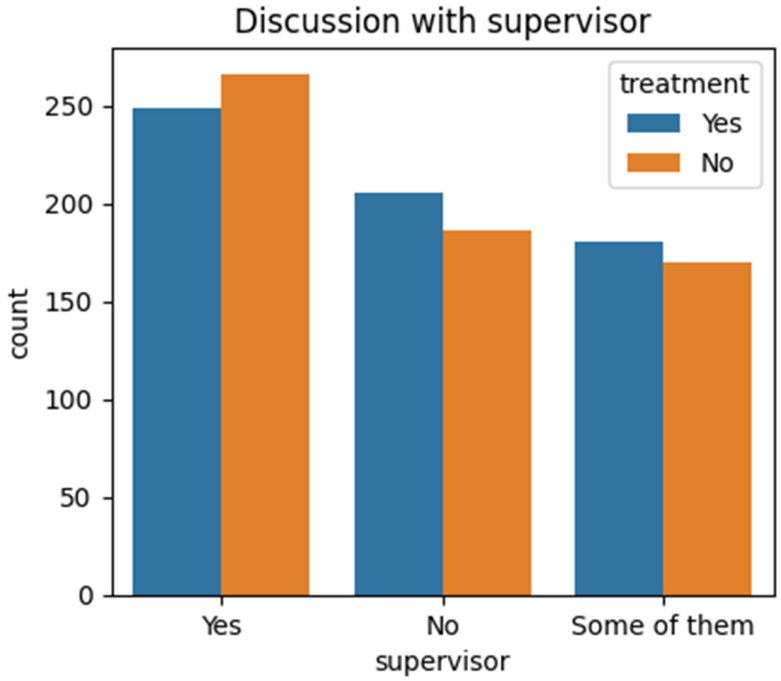


As completely opposed to the above results, a very small number of people answered yes to this question, which means that a major number of people do not face negative consequences by discussing their physical health conditions with their employers.

- Discussion with coworkers and treatment

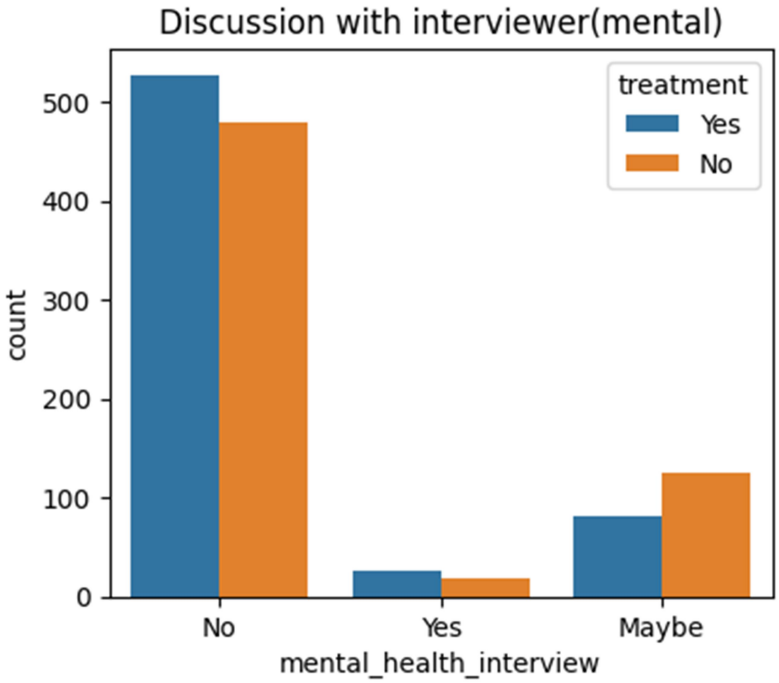


- Discussion with supervisor and treatment

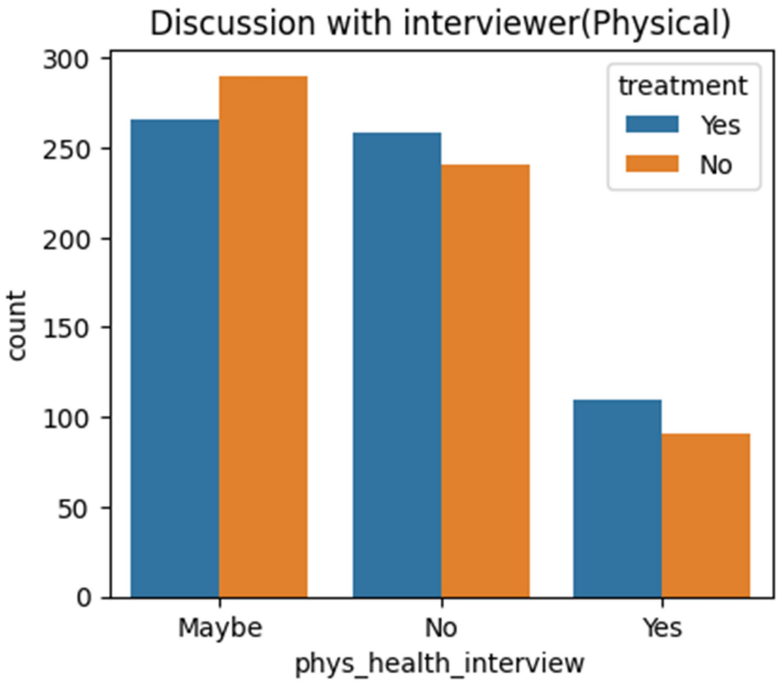


This graph again, is in contrast to the above one. Though majority of people are comfortable with discussing their physical health problems with their supervisor, about 1/3rd of them aren't comfortable with it.

- Mental health discussion during interview and treatment

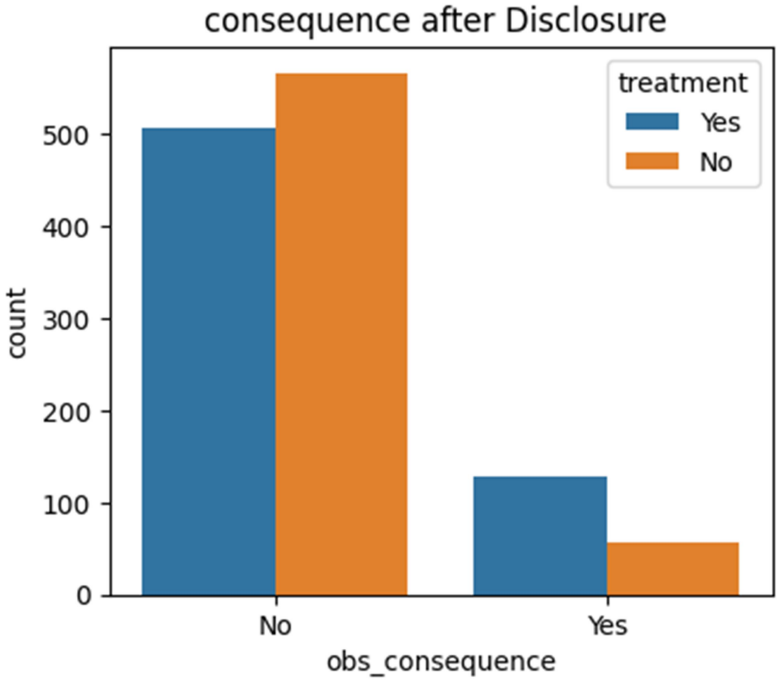


We observe that very few people are comfortable in bringing up their mental health issue in front of a potential employer.



Though this graph is similar to the above one with respect to not being comfortable in bringing up a physical health issue with a potential employer, a good number of people may be willing to or are willing to bring it up.

- Consequence after disclosure and treatment



We observe that majority of people have not faced any negative consequences after discussing their mental health conditions in their workplace

Activity 3: Descriptive analysis

Descriptive analysis is to study the basic statistical features of data. We can achieve it by using the .describe() function. With this describe function we can understand the unique, top and frequent values of categorical features. Also, we can find mean, std, min, max and percentile values of numerical features.

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

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_health_consequ
count	1256	1256.000000	1256.000000	1256.000000	1256.000000	1256.000000	1256.000000	1256.000000	1256.000000	1256.000000	...	1256.000000	1256.00
unique	1244	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
top	2014-08-27 12:44:51	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
freq	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
mean	NaN	31.447452	0.820064	37.776274	13.604299	0.115446	0.390127	0.504777	2.285828	2.784236	...	1.408439	0.85
std	NaN	6.178908	0.423733	13.331870	14.685069	0.319687	0.487973	0.500176	1.595409	1.739288	...	1.508402	0.76
min	NaN	18.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.00
25%	NaN	27.000000	1.000000	42.000000	2.000000	0.000000	0.000000	0.000000	1.000000	1.000000	...	0.000000	0.00
50%	NaN	31.000000	1.000000	45.000000	2.000000	0.000000	0.000000	1.000000	3.000000	3.000000	...	1.000000	1.00
75%	NaN	35.000000	1.000000	45.000000	28.000000	0.000000	1.000000	1.000000	4.000000	4.000000	...	2.000000	1.00
max	NaN	49.000000	2.000000	47.000000	44.000000	1.000000	1.000000	1.000000	4.000000	5.000000	...	4.000000	2.00

11 rows × 27 columns

Milestone 4: Model Building

Activity 1: Handling Categorical Values

As we can see our dataset has categorical data. Before training our model, we must convert the categorical data into a numeric form.

There are multiple encoding techniques to convert the categorical columns into numerical columns. For this project we will be using ordinal encoding for our features and label encoding for our target.

So, for that we first need to divide our data into features and target.

```
[83] from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()

     df.Gender=le.fit_transform(df.Gender)
     df.head()
```

Here X contains our features and y contains our target.

X AND Y SPLITS

```
df.treatment=le.fit_transform(df.treatment)
y=df['treatment']
y
```

```
0       1
1       0
2       0
3       1
4       0
       ..
1254    1
1255    1
1256    1
1257    0
1258    1
Name: treatment, Length: 1256, dtype: int64
```

```
[90] x=df.drop(columns = ['treatment'],axis=1)
```

```
[91] x.head()
```

```
[92] x_new=x.drop(columns = ['Timestamp'],axis=1)
```

```
[93] x_new = x_new.drop(columns= ['state'],axis =1)
     x_new = x_new.drop(columns= ['Country'],axis =1)
     x_new = x_new.drop(columns= ['comments'],axis =1)
```

The next step is to apply respective encoding techniques on features and target

.

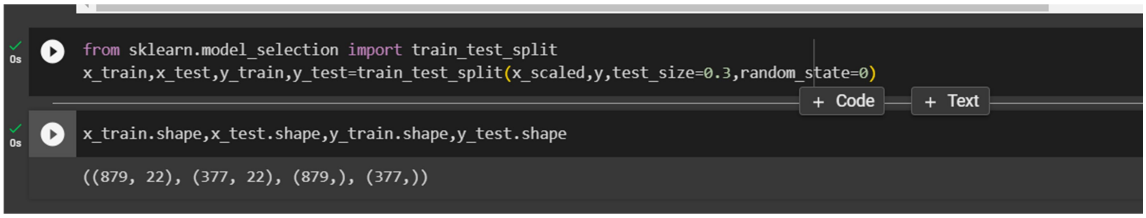
LABEL ENCODING

```
[85] df.Country=le.fit_transform(df.Country)
     df.state=le.fit_transform(df.state)
     df.self_employed=le.fit_transform(df.self_employed)
     df.family_history=le.fit_transform(df.family_history)
     df.work_interfere=le.fit_transform(df.work_interfere)
     df.no_employees=le.fit_transform(df.no_employees)
     df.leave=le.fit_transform(df.leave)
     df.mental_health_consequence=le.fit_transform(df.mental_health_consequence)
     df.phys_health_consequence=le.fit_transform(df.phys_health_consequence)
     df.coworkers=le.fit_transform(df.coworkers)
     df.supervisor=le.fit_transform(df.supervisor)
     df.mental_health_interview=le.fit_transform(df.mental_health_interview)
     df.phys_health_interview=le.fit_transform(df.phys_health_interview)
     df.obs_consequence=le.fit_transform(df.obs_consequence)
     df.mental_vs_physical=le.fit_transform(df.mental_vs_physical)
     df.mental_vs_physical=le.fit_transform(df.mental_vs_physical)
```

```
[86] df.remote_work=le.fit_transform(df.remote_work)
     df.benefits=le.fit_transform(df.benefits)
     df.care_options=le.fit_transform(df.care_options)
     df.wellness_program=le.fit_transform(df.wellness_program)
     df.seek_help=le.fit_transform(df.seek_help)
```

Activity 2: Splitting data into train and test

For splitting the data into train and test sets, we are using the train_test_split() function from sklearn. As parameters, we are passing X, y, test_size, random_state.

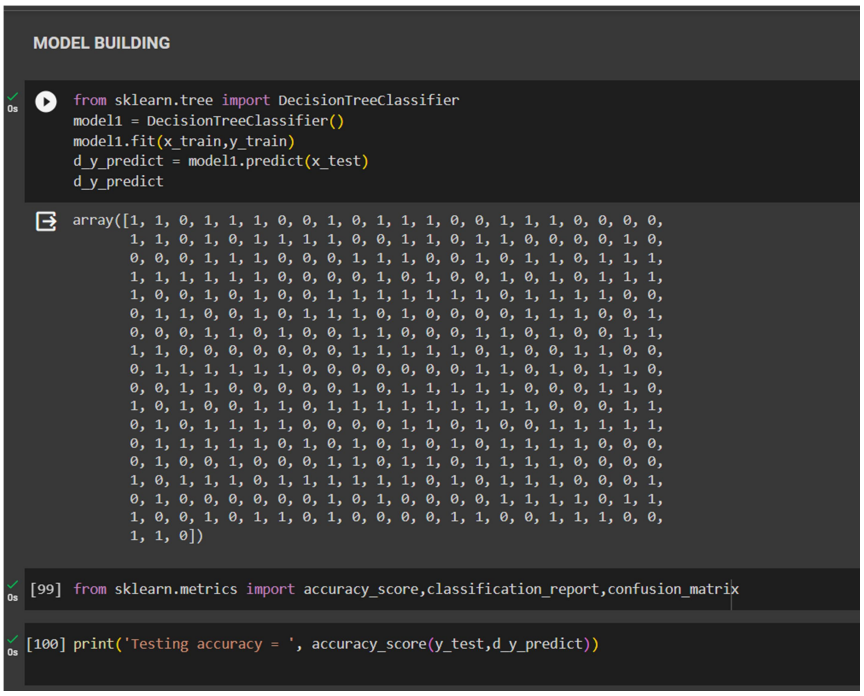


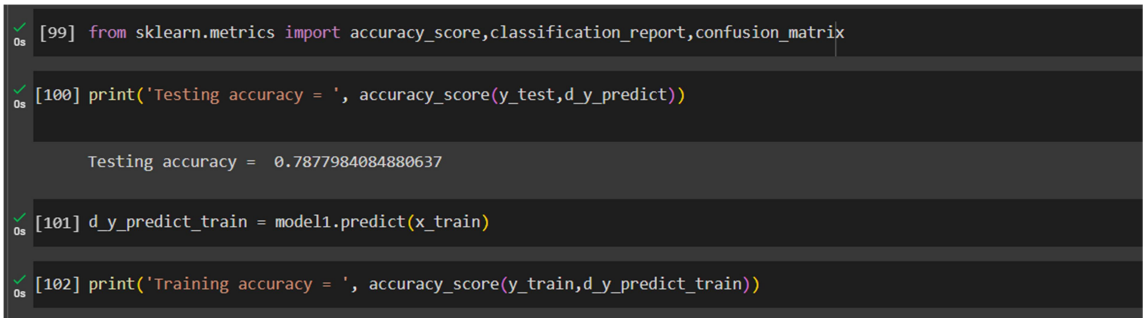
Activity 3: Comparing accuracy of various models

We will be considering multiple models to train our data and choose the one that performs the best. So, we need to import the necessary libraries and create a dictionary of our models.

And also , we will define a function known as model_test() that accepts 6 parameters - X_train, X_test, y_train, y_test, model, model_name. We will obtain y_pred by using .predict() function and compute the accuracy score for every model by iterating through the dictionary.

Model 1:





Model 2:

```
[103] from sklearn.ensemble import RandomForestClassifier ,AdaBoostClassifier ,GradientBoostingClassifier
      model2 = RandomForestClassifier(random_state=99)
      model2.fit(x_train,y_train)
      l_y_predict = model2.predict(x_test)

[104] l_y_predict_train = model2.predict(x_train)

      print('Testing accuracy = ', accuracy_score(y_test,l_
      print('Training accuracy = ', accuracy_score(y_train,l_y_predict_train))

      Testing accuracy =  0.843501326259947
      Training accuracy =  1.0
```

ndarray: l_y_predict_train
ndarray with shape (879,)

Model 3:

```
[106] from sklearn.linear_model import LogisticRegression
      model3 = LogisticRegression()
      model3.fit(x_train,y_train)
      lr_y_predict = model3.predict(x_test)

[107] lr_y_predict_train = model3.predict(x_train)

      print('Testing accuracy = ', accuracy_score(y_test,lr_y_predict))
      print('Training accuracy = ', accuracy_score(y_train,lr_y_predict_train))

      Testing accuracy =  0.8037135278514589
      Training accuracy =  0.820250284414107
```

Loading...

Model 4:

```
+ Code

[109] model4 = AdaBoostClassifier()
      model4.fit(x_train,y_train)
      a_y_predict = model4.predict(x_test)
      a_y_predict_train = model4.predict(x_train)
      print('Testing accuracy = ', accuracy_score(y_test,a_y_predict))
      print('Training accuracy = ',accuracy_score(y_train,a_y_predict_train))

      Testing accuracy =  0.8381962864721485
      Training accuracy =  0.8418657565415245
```

Model 5:

```
      model5 = GradientBoostingClassifier()
      model5.fit(x_train,y_train)
      g_y_predict = model5.predict(x_test)
      g_y_predict_train = model5.predict(x_train)
      print('Testing accuracy = ', accuracy_score(y_test,g_y_predict))
      print('Training accuracy = ',accuracy_score(y_train,g_y_predict_train))

      Testing accuracy =  0.8328912466843501
      Training accuracy =  0.8896473265073948
```

Model 6:

```
[111] from sklearn.svm import SVC

[112] model6 = SVC()
      model6.fit(x_train,y_train)
      s_y_predict = model6.predict(x_test)
      print('Testing accuracy = ', accuracy_score(y_test,s_y_predict))

      Testing accuracy =  0.7877984084880637
```

From the above results, it is clear that AdaBoost Classifier provides the best accuracy. So let us create a different variable to fit and make predictions using the model.

```
AdaBoostClassifier has best accuracy

[113] 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))

      accuracy of adaboost =  0.8381962864721485
```

Activity 4: Evaluating performance of models

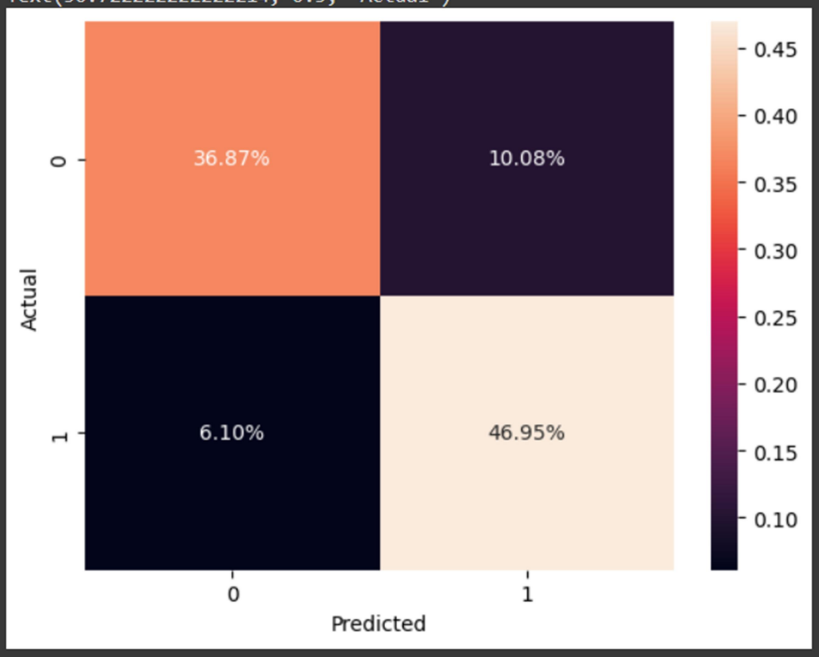
We will compare the confusion matrix and classification report for both models.

In order to obtain these, we will be using the confusion_matrix() and classification_report() functions from sklearn.metrics.

Confusion matrix:

```
[114] cf_matrix = confusion_matrix(y_test,pred_abc)
      sns.heatmap(cf_matrix/np.sum(cf_matrix),annot =True,fmt='.2%')
      plt.xlabel('Predicted')
      plt.ylabel('Actual')
```

```
Text(50.722222222222214, 0.5, 'Actual')
```



Classification report:

CLASSIFICATION REPORT

```
[115] print(classification_report(y_test,pred_abc))
```

	precision	recall	f1-score	support
0	0.86	0.79	0.82	177
1	0.82	0.89	0.85	200
accuracy			0.84	377
macro avg	0.84	0.84	0.84	377
weighted avg	0.84	0.84	0.84	377

Activity 6: Saving the model

The final step is saving our model. We can do it by using pickle.dump().

SAVING THE MODEL

```
[116] import pickle
      pickle.dump(abc,open('model.pkl','wb'))
```

Downloading the model

```
[117] from google.colab import files

      files.download('model.pkl')
```

Milestone 5: 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 entered 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

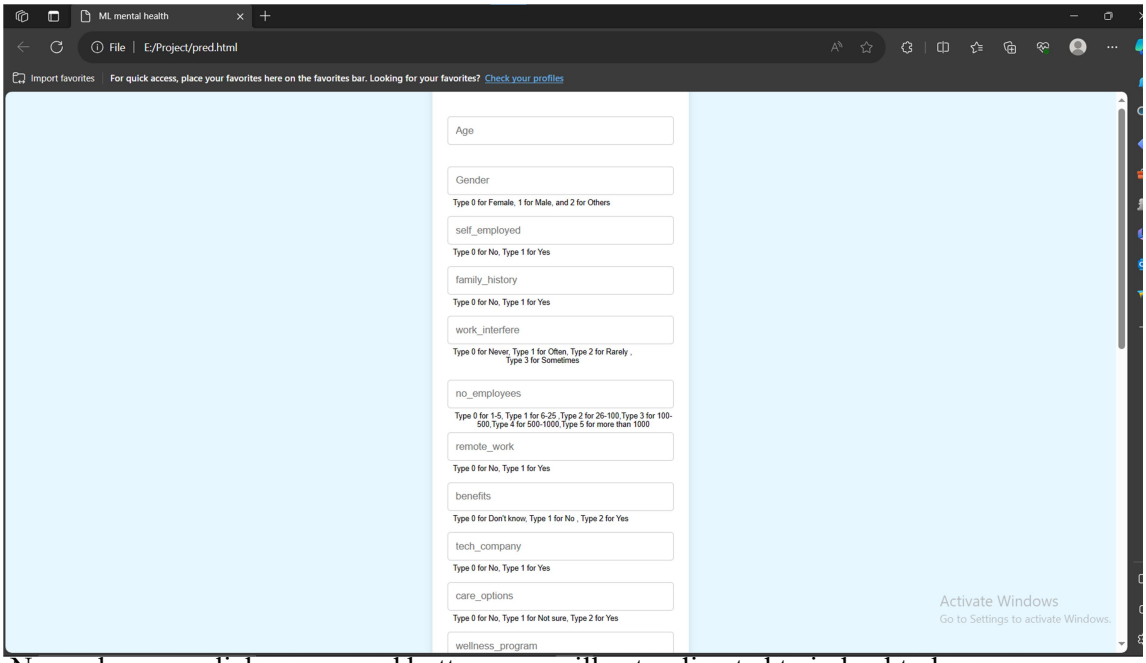
Activity1: Building Html Pages:

For this project create three HTML files namely

- home.html
- index.html
- output.html

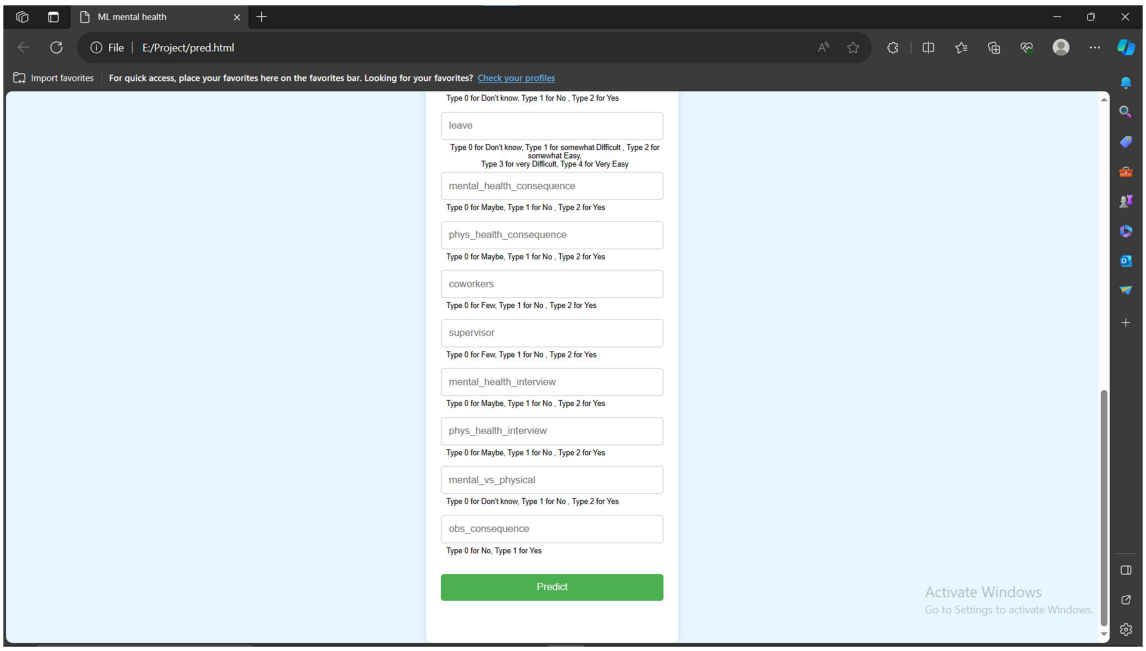
and save them in the templates folder.

Let's see how our home.html page looks like:



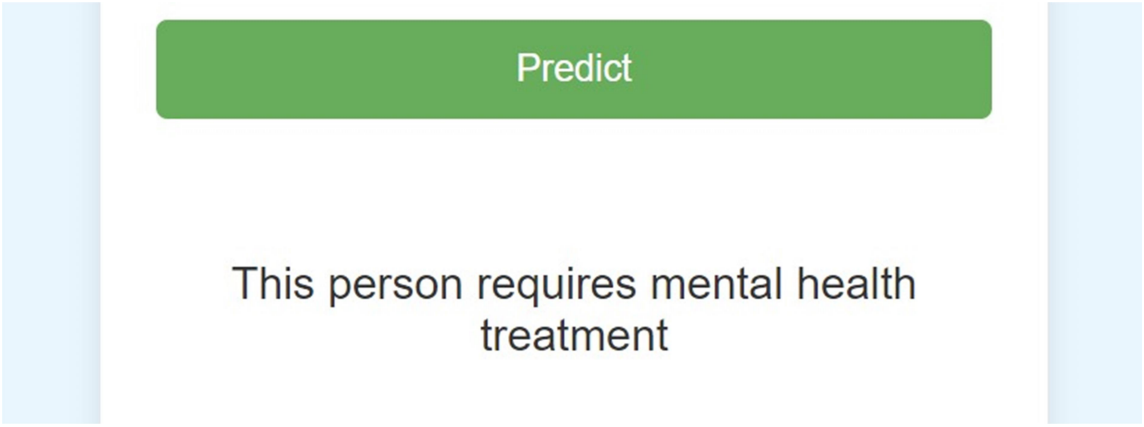
Now when you click on proceed button, you will get redirected to index.html

Let us look how our index.html page looks like:



Now when you click on predict button from left bottom corner you will get redirected to output.html

Let us look how our output.html page looks like:



Activity 2: Build Python code:

Import the required libraries and load model and ct

app.py pred.html model.py

app.py > prediction

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

# Create Flask app
application = Flask(__name__)
model = pickle.load(open("model.pkl", "rb"))

@application.route('/')
def pred():
    return render_template('pred.html')
```

The values entered in can be retrieved using the POST Method.

Retrieves the value from UI:

```python
    # Get form data and convert to float
    float_features = [float(x) for x in request.form.values()]
    final_features = [np.array(float_features)]
```

```python

    # Make prediction
    prediction = model.predict(final_features)

    # Prepare the prediction message
    if prediction[0] == 1:
        prediction_text = "This person requires mental health treatment"
    else:
        prediction_text = "This person doesn't require mental health treatment"

    return render_template('pred.html', prediction_text=prediction_text)

```

Here we are routing our app to output() function. This function retrieves all the values from

model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the output.html page earlier.

Main Function:

```
30  if __name__ == "__main__":
31      application.run(debug=True)
32
```

Activity 3: Run the 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 proceed button, enter the inputs, click on the predict button, and see the result/prediction on the web.

```
(venv) PS D:\flask> python .\app.py
 * 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
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 801-151-542
```