1. INTRODUCTION

- Work often constitutes a significant portion of our lives, where we not only earna living but also form connections. A fulfilling job can positively impact **mental health** and **overall well-being**.

- Mental health is integral to our overall well-being, shaping our thoughts, emotions, and behaviors in our daily lives. Its importance in the tech workspace is paramount, influencing our fundamental sense of wellness.

- Good mental health brings about a multitude of benefits, positively impacting various aspects of an individual's life. Some of the key advantages include:

  - **Enhanced Emotional Well-being.**
  - **Improved Relationships.**
  - **Increased Resilience.**
  - **Higher Productivity and Performance.**
  - **Better Physical Health.**

1.1 PROBLEM STATEMENT

- Daily life involves natural fluctuations, influenced by various factors. However, when the ratio of these fluctuations becomes imbalanced, it can lead to mental and physical burnout, disrupting life's equilibrium and potentially resulting in **Anxiety**& **Depression**.
- We are trying to analyze what influences most for the mental health issues people face in a Tech workspace with the help of the data collected by OSMI.

OSMI, **Open Sourcing Mental Illness** is a non-profit organization focused on changing how mental health is addressed in the tech community. Founded by **Ed Finkler**, a developer and advocate for mental health awareness

- OSMI aims to raise awareness, provide support, and create an open dialogue about mental health issues in the tech industry.

- They perform surveys to **measure attitudes** towards mental health in the tech workplace.

- OSMI encourages open conversations and provides educational materials to help employers and employees better understand and manage mental health issues in the workplace.

Millions of people around the world are affected by one or more mental disorders that interfere with their thinking and behavior. A timely detection of these issues is challenging but crucial since it could open the possibility of offering help to people before the illness gets worse. One alternative to accomplish this is to monitor how people express themselves, which is for example what and how they write, or even a step further, what emotions they express in their social media communications. In this study, we analyze two computational representations that aim to model the presence and changes of the emotions expressed by social media users. In our evaluation, we use recent public data sets for the mental disorder: Depression. The obtained results suggest that the presence and variability of emotions, captured by the proposed representations, allow for highlighting important information about social media users suffering from depression.

1.2 INTRODUCTION TO MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data.

Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learnin g algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they learn about data to make predictions: supervised and unsupervised learning.

Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation.

This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

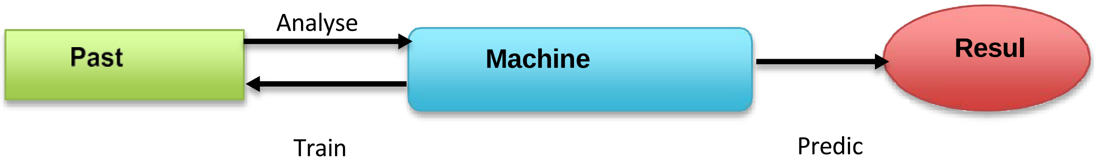


Fig. 1 Process of Machine Learning

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is y = f(X). The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression**,** multi-class classification**,** Decision Trees and support vector machines etc**.** Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification.

PROJECT ANALYSIS

2. PROJECT ANALYSIS

2.1 EXISTING SYSTEM:

1. Data Visualization (limited to column charts): Utilizing column charts, one can visually represent data distributions, trends, and comparisons.

2. Exploratory Analysis on Limited Constraints: This entails conducting preliminary investigations of the data to discover patterns, spot anomalies, identify relationships, and formulate hypotheses. However, this analysis is limited by certain constraints, possibly pertaining to available resources, time, or specific research objectives.

3. Prediction Methods: This refers to employing various techniques to forecast future outcomes based on historical data patterns. Common prediction methods include regression analysis, time series forecasting, machine learning algorithms such as decision trees, random forests etc. These methods utilize pre- processed data to train predictive models and make informed predictions.

2.2 DISADVANTAGES OF THE EXISTING SYSTEM:

The disadvantages of the present system includes:

- Data visualization on limited columns
- Exploratory analysis on limited constraints

2.3 PROPOSED SYSTEM

1. **Data Acquisition:** the data set is collected from OSMI (Open Sourcing Mental Illness). OSMI is a non- profitable organization, which conducted a survey in 2014. It describes the attitudes towards mental health and frequency of mental health disorders in the tech workplace.
2. **Data Pre-Profiling Analysis**: Before diving deep into analysis, it's essential to conduct an initial examination of the data. This involves identifying data types, checking for missing values, outliers, and understanding the overall structure of the data.
3. **Column Profiling**: This step focuses on analyzing individual columns or variables within the dataset. It includes examining the distribution of values, identifying unique values, and understanding the range and variability of each column.
4. **Data Preprocessing:** Data preprocessing involves cleaning and transforming the data to make it suitable for analysis. This may include handling missing values, encoding categorical variables, scaling numerical features, and other data transformations.
5. **Data Visualizations:** Visualizations are powerful tools for understanding data and communicating insights. The mentioned types of visualizations (pie charts, bar graphs, line graphs, column charts) can be used to represent different aspects of the data and relationships between variables.
6. **Exploratory Analysis on Important Constraints**: This is the core of the analysis, where the focus is on exploring relationships and patterns in the data related to specific constraints or variables of interest. The mentioned constraints such as age, gender, family history, etc., are crucial factors that could impact the analysis outcomes.
7. **Analysis Conclusions:** After conducting exploratory analysis, conclusions are drawn based on the insights gained from the data. These conclusions may include identifying trends, correlations, associations, or making predictions based on the analyzed constraints.
8. **Machine learning**: After the exploratory data analysis machine learning algorithms such as decision trees, random forests, and neural networks. These methods utilize pre-processed data to train predictive models and make informed predictions.
9. **Flask Deployment:** Model is deployed using flask.

Expanding on the mentioned constraints:

- Age: Analysis how age correlates with various factors under consideration. For example, understanding if there's a relationship between age and treatment effectiveness.
- Density Distribution: Examining the distribution of data across different density levels and its impact on the analysis.
- Gender vs. Treatment: Investigating if there's any difference in treatment outcomes between different genders.
- Work Interference vs. Treatment: Exploring how work interference levels relate to the effectiveness of treatments.
- Family History vs. Treatment: Analyzing if family history of a certain condition influences treatment outcomes.
- Employee Count and Countries: Understanding how the number of employees and geographic location relate to the variables being analyzed.
- Frequency vs. Work Interference/Attitude vs. Frequency/Attitude vs. Age: Examining these relationships to uncover insights into how attitudes, frequency of occurrences, and work interference are interrelated and how they vary with age.

Overall, through a systematic approach encompassing data acquisition, preprocessing, exploratory analysis, and visualization, meaningful insights can be derived from the data to inform decision-making processes.

2.4 SCOPE OF THE SYSTEM:-

1. **Data Collection**: This phase involves gathering relevant data from diverse sources such as surveys, databases, or APIs. It's crucial to ensure the data collected is comprehensive and representative of the study's scope.

2. **Descriptive Analysis**: After collecting the data, the next step is to conduct descriptive analysis. This involves summarizing and exploring the data set to understand its basic characteristics, such as measures of central tendency, dispersion, and distribution of variables.

3. **Identifying Risk Factors**: In this stage, researchers aim to identify potential risk factors or variables that may influence the outcomes being studied. This involves analyzing correlations and patterns within the data to pinpoint factors that could contribute to certain outcomes or phenomena.

4. **Demographic Analysis**: Understanding the demographic characteristics of the sample population is essential for contextualizing the findings. Demographic analysis involves examining variables such as age, gender, education level, income, etc., to gain insights into how these factors may impact the study outcomes.

5. **Comparative Analysis**: Comparative analysis involves comparing different groups or categories within the data set to identify disparities, trends, or significant differences. This could include comparing treatment outcomes between different demographic groups, geographic regions, or other relevant factors.

6. **Recommendations and Interventions**: Based on the findings from the analysis, recommendations and interventions can be proposed to address identified issues or leverage opportunities. These recommendations may include policy changes, interventions, or strategies aimed at improving out mitigating risks.

7. **Ethical Considerations**: Throughout the entire process, ethical considerations must be taken into account to ensure the rights and well-being of participants are protected. This involves obtaining informed consent, maintaining confidentiality, and adhering to ethical guidelines and regulations.

8. **Continuous Improvement:** Data analysis is an iterative process, and continuous improvement is essential for refining methodologies, enhancing data quality, and updating recommendations based on new evidence or insights. This involves ongoing monitoring, evaluation, and adaptation to ensure the effectiveness and relevance of interventions or strategies over time.

LITERATURE REVIEW

3. LITERATURE REVIEW

Paper: International Journal of Science and Research Archive (IJSRA)

Author: Madhurima Paul and Swapan Das

Date: 01 August 2023

One of the key uses of machine learning in the field of mental health is the detection and diagnosis of mental health problems in individuals. Moreover, it entails creating risk frameworksto forecast people's propensity for mental health problems, which can aid with early intervention

Another recent study indicated that care for mental health is based primarily on self- assessment as mental illnesses are the outcomes of patients' behaviors. The study also demonstrated that predictive models could be used to identify a patient who requires relatively higher care and concern

Moreover, models have been used to forecast mental health issues of technical workers. The most important factors for predicting mental health disorders, according to these studies, include employees' prior mental health concerns and their family history of mental disease.

In the context of the IT industry, patterns of employee stress and the main stressors have been examined to assist organizations in understanding their employees' mental health and identifying any contributing variables, this study intends to provide forecasts on employee risk levels and mental health. We anticipate that these insights will increase employers' understanding and lead to workplace mental health treatments that will enhance their mental health.

SYSTEM REQUIREMENTS

4. SYSTEM REQUIREMENTS

Requirements are the basic constrains that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Technical requirements
    A. Hardware requirements
    B. Software requirements

4.1 FUNCTIONAL REQUIREMENTS:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like:

- Sk-learn
- pandas
- numpy
- matplotlib
- seaborn

4.2 NON-FUNCTIONAL REQUIREMENTS:

Process of functional steps:

- Problem Define
- Collecting Data
- Data Preprofiling
- Data Preprocessing
- Exploratory Data analysis
- Feature Selection
- Prediction Model
- Analysis & Evaluation of Results
- Drawing inferences

4.3 TECHNICAL REQUIREMENTS:

- **Software Requirements:**

Operating System	: Windows
Tool	: Anaconda with Jupyter Notebook, Visual Studio Code
Framework	: Flask

- **Hardware requirements:**

Processor	: Pentium IV/III
Hard disk	: minimum 80 GB
RAM	: minimum 2 G

SOFTWARE DESCRIPTION

5. SOFTWARE DESCRIPTION

5.1 ANACONDA:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system Conda. Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, new environments can be created that include any version of Python packaged with conda.

ANACONDA NAVIGATOR:

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command - line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda is created by Continuum Analytic, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

Navigator is an easy, point-and-click way to work with packages and environments with needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them –all inside Navigator.

The following applications are available by default in Navigator:

- Jupyter Lab
- Jupyter Notebook
- Spyder
- PyCharm
- VS Code
- Glue viz
- Orange 3 App
- R studio
- Anaconda Prompt (Windows only)
- Anaconda Power Shell (Windows only)

Fig.2. Anaconda Navigator (1)

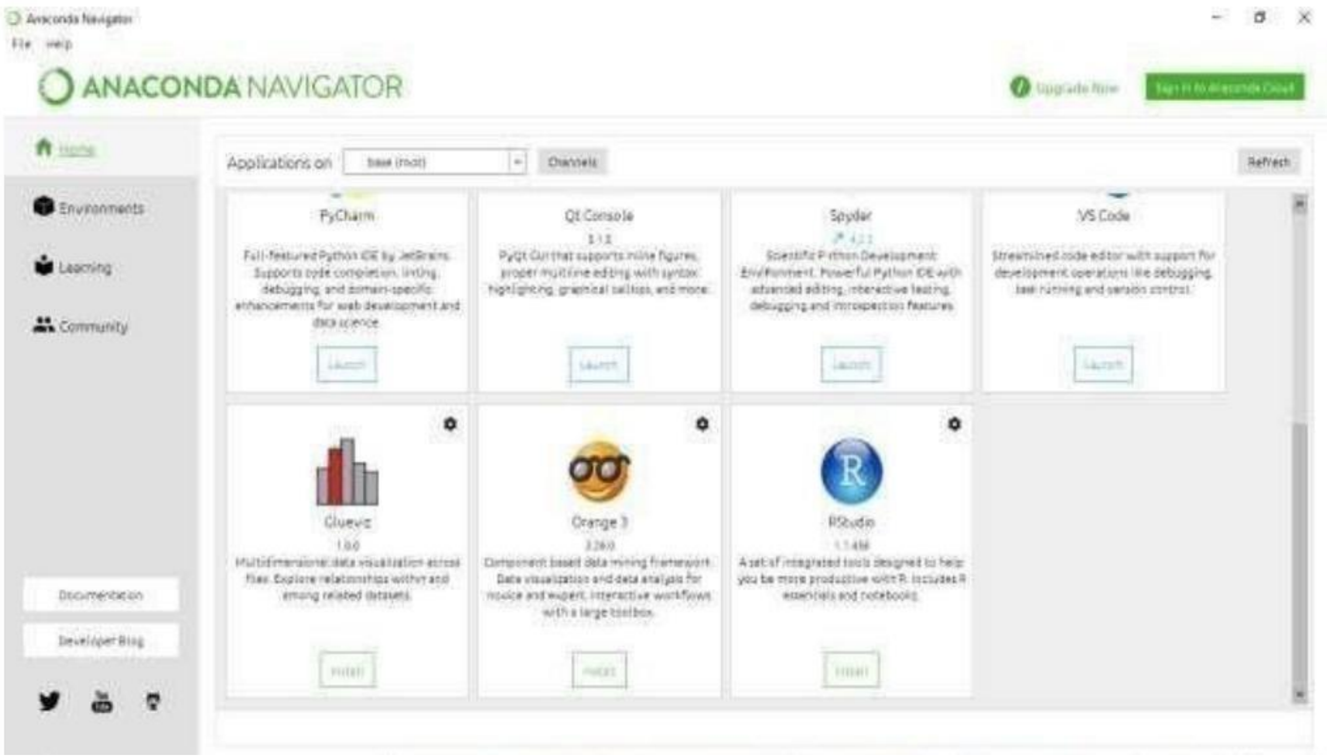


Fig.3. Anaconda Navigator (2)

Conda:

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi- language projects. The Conda package and environment manager is included in all versions of Anaconda, Mini conda, and Anaconda Repository.

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for you. It consists of many software which will help you to build your machine learning project and deep learning project. These Software have great graphical user interface and can also use it to run your python script. These are the software carried by anaconda navigator.

5.2 JUPYTER NOTEBOOK:

This website acts as meta documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to ―develop open -source software, open-standards, and services for interactive computing across dozens of programming languages‖. It was spun off from Python in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations,figures, links, etc…). Notebook documents are both human-readable documentscontaining the analysis description and the results (figures, tables, etc.)as well asexecutable documents which can be run to perform data analysis.

Installation: The easiest way to install the *Jupyter Notebook App* is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called **Anaconda.**

5.3WORKING PROCESS:

- Download and install anaconda and get the most useful package for machine learning python.
- Load a data set and understand its structure using statistical summaries and data visualization.
- Machine Learning models, pick the best and build confidence that the accuracy is reliable.

5.4.PYTHON:

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

5.5FLASK:

Flask is a lightweight web framework for Python. It's designed to make getting started with web development in Python quick and easy, while still providing the flexibility to build complex web applications. Flask is often referred to as a "micro-framework" because it doesn't impose any dependencies or project structure on developers, allowing them to have full control over the design and architecture of theirapplications.

Key features of Flask include:

Routing: Flask allows developers to define URL routes that map to specific functions in their Python code. This makes it easy to create different endpoints for handling different types of requests.

Templates: Flask includes a built-in tempting engine called Jinja2, which allows developers to generate HTML dynamically by combining static content with data from their Python code.

HTTP request handling: Flask provides simple and intuitive ways to access and manipulate incoming HTTP requests, including accessing request data, handling file uploads, and setting response headers.

Session management: Flask includes support for managing user sessions, which allows developers to store user-specific data across multiple requests.

Extension ecosystem: While Flask itself is minimalist, it has a large ecosystem of extensions developed by the community that add additional features and functionality, such as database integration, authentication, and form validation.

Development server: Flask includes a built-in development server that makes it easy to test and debug applications locally before deploying them to production servers.Overall, Flask is a popular choice for web development in Python due to its simplicity, flexibility, and ease of use. It's suitable for building a wide range of web applications, from small personal projects to large-scale enterprise applications.

SYSTEM DESIGN

6. SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE

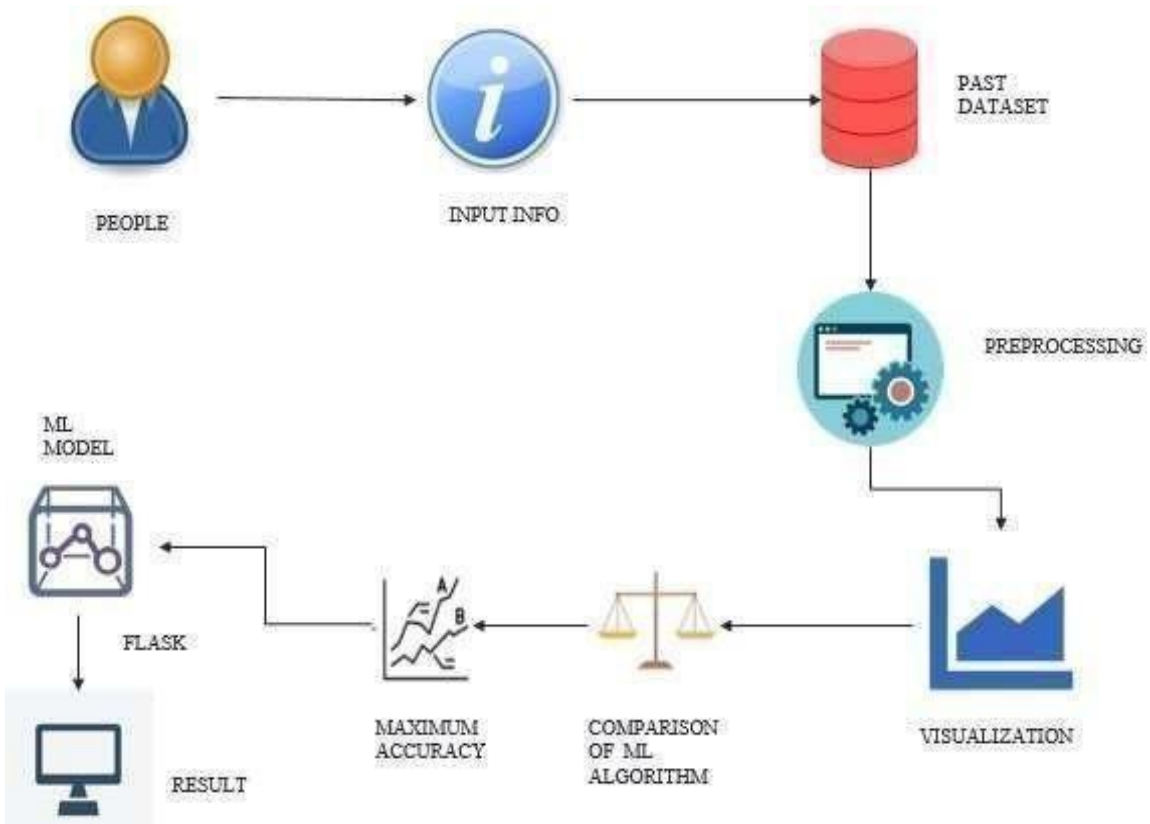


Fig.4. System Architecture

6.2 WORKFLOW DIAGRAM

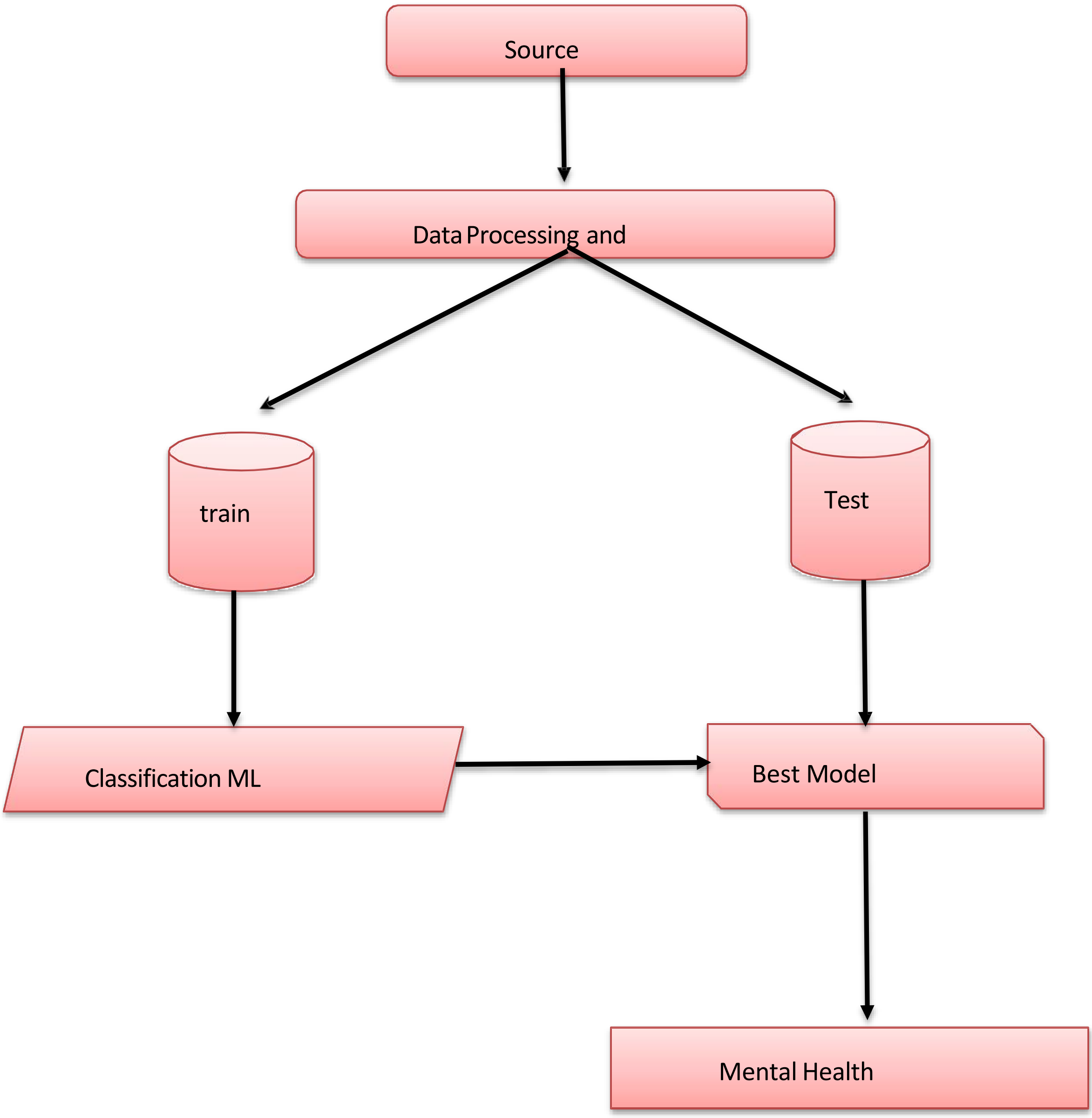


Fig.5. Work Flow Diagram

JUPYTER NOTEBOOK CODING

7. JUPYTER NOTEBOOK CODING

7.1 CODING

Importing Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
pd.set_option('display.float_format', lambda x : '%.2f' %x)
```

Data Acquisition

- This data set is obtained from a survey in 2014.
- It describes the attitudes towards mental health and frequency of mental health disorders in the tech workplace.

Records	Features	Dataset Size
1259	27	296 KB

Id	Features	Description
01	**Timestamp**	Time the survey was submitted.
02	**Age**	The age of the person.
03	**Gender**	The gender of the person.
04	**Country**	The country name where person belongs to.
05	**state**	The state name where person belongs to.
06	**self_employed**	Is the person self employed or not.
07	**family_history**	Does the person's family history had mental illness or not?
08	**treatment**	Have you sought treatment for a mental health condition?

Id	Features	Description
09	**work_interfere**	If you have a mental health condition, do you feel that it interferes with your work?
10	**no_employees**	How many employees does your company or organization have?
11	**remote_work**	Do you work remotely (outside of an office) at least 50% of the time?
12	**tech_company**	Is your employer primarily a tech company/organization?
13	**benefits**	Does your employer provide mental health benefits?
14	**care_options**	Do you know the options for mental health care your employer provides?
15	**wellness_program**	Has your employer ever discussed mental health as part of an employee wellness program?
16	**seek_help**	Does your employer provide resources to learn more about mental health issues and how to seek help?
17	**anonymity**	Is your anonymity protected if you choose to take advantage of mental health or substance abuse treatment resources?
18	**leave**	How easy is it for you to take medical leave for a mental health condition?
19	**mental_health_consequence**	Do you think that discussing a mental health issue with your employer would have negative consequences?
20	**phy_health_consequence**	Do you think that discussing a physical health issue with your employer would have negative consequences?

Id	Features	Description
21	**coworkers**	Would you be willing to discuss a mental health issue with your coworkers?
22	**supervisor**	Would you be willing to discuss a mental health issue with your direct supervisor(s)?
23	**mental_health_interview**	Would you bring up a mental health issue with a potential employer in an interview?
24	**phs_health_interivew**	Would you bring up a physical health issue with a potential employer in an interview?
25	**mental_vs_physical**	Do you feel that your employer takes mental health as seriously as physical health?
26	**obs_consequence**	Have you heard of or observed negative consequences for coworkers with mental health conditions in your workplace?
27	**comments**	Any additional notes or comments.

Data=pd.read_csv("C://Users//VENKAT//Downloads//surv

ey. csv")data.head()

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employ
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	Often	6-25
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	Rarely	More than 1000
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Rarely	6-25
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	Never	100-500

[5 rows x 27 columns]

data.shape

+(1259, 27)

- We have a total of **1259** rows & **27** columns in the dataset.data.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

Data PreProfiling

data.describe()

	Age
count	1259.00
mean	79428148.31
std	2818299442.98
min	-1726.00
25%	27.00
50%	31.00
75%	36.00
max	99999999999.0

Incorrect values in Age:

- Max Age of a person cannot be **99999999999.00**.
- And also No person would have an age of negative **-1726**.
- Since most countries designate **18 years** as the legal age to commence work, let's examine the Age field for any records below this threshold.

data[data['Age'] <18] In [10]:
Out [10]:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_er
143	2014-08-27 12:39:14	-29	Male	United States	MN	No	No	No	NaN	More 1000
715	2014-08-28 10:07:53	-1726	male	United Kingdom	NaN	No	No	Yes	Sometimes	26-1C
734	2014-08-28 10:35:55	5	Male	United States	OH	No	No	No	NaN	100-5
989	2014-08-29 09:10:58	8	A little about you	Bahamas, The	IL	Yes	Yes	Yes	Often	1-5
1090	2014-08-29 17:26:15	11	male	United States	OH	Yes	No	No	Never	1-5
1127	2014-08-30 20:55:11	-1	p	United States	AL	Yes	Yes	Yes	Often	1-5

6 rows × 27 columns

- Similarly, let's assess the Age field for individuals within a maximum age rangeof approximately **75 years.**

data[data['Age'] >75] In [11]:

Out [11]:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfe
364	2014-08-27 15:05:21	329	Male	United States	OH	No	No	Yes	Often
390	2014-08-27 15:24:47	99999999999	All	Zimbabwe	NaN	Yes	Yes	Yes	Often

2 rows x 27 columns

- Based on the above output, it appears that these individuals may prefer not to disclose their personal information and have provided potentially random or unspecified values.
- In this scenario, we can address this issue similarly to how we handle **missing values**, by replacing them with the **mean or median** according to the data.

data.shape[0]-data.count() In [12]:

Out [12]:

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

- Percentage of missing values.
- 100*((data.shape[0]-data.count())/data.shape[0])

Timestamp	0.00
Age	0.00
Gender	0.00
Country	0.00
state	40.91
self_employed	1.43
family_history	0.00
treatment	0.00
work_interfere	20.97
no_employees	0.00
remote_work	0.00
tech_company	0.00
benefits	0.00
care_options	0.00
wellness_program	0.00
seek_help	0.00
anonymity	0.00
leave	0.00
mental_health_consequence	0.00
phys_health_consequence	0.00
coworkers	0.00
supervisor	0.00
mental_health_interview	0.00
phys_health_interview	0.00
mental_vs_physical	0.00
obs_consequence	0.00
comments	86.97

dtype: float64

- We have missing values in the following columns **state, self_employed, work_interfere and comments.**

In [14]:

```
data_missing = pd.DataFrame(index = data.columns.values)
```

```
data_missing['Null'] = data.isnull().sum().values
```
In [15]:

```
data_missing
```
Out [15]:

	Null
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

```
data_missing['Null percentage'] = 100*((data.shape[0]-data.count())/data.shape[0])
data_missing
```
In [16]:
Out [16]:

	Null	Null percentage
Timestamp	0	0.00
Age	0	0.00
Gender	0	0.00
Country	0	0.00
state	515	40.91

self_employed	18	1.43
family_history	0	0.00
treatment	0	0.00
work_interfere	264	20.97
no_employees	0	0.00
remote_work	0	0.00
tech_company	0	0.00
benefits	0	0.00
care_options	0	0.00
wellness_program	0	0.00
seek_help	0	0.00
Leave	0	0.00
mental_health_consequence	0	0.00
phys_health_consequence	0	0.00
coworkers	0	0.00
supervisor	0	0.00
mental_health_interview	0	0.00
phys_health_interview	0	0.00
mental_vs_physical	0	0.00
obs_consequence	0	0.00
comments	1095	86.97

In [17]:

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

Out [17]:

```
United States        751
United Kingdom       185
Canada               72
Germany              45
Ireland              27
Netherlands          27
Australia            21
France               13
India                10
New Zealand          8
Poland               7
Switzerland          7
Sweden               7
Italy                7
South Africa         6
```

Belgium	6
Brazil	6
Israel	5
Singapore	4
Bulgaria	4
Austria	3
Finland	3
Mexico	3
Russia	3
Denmark	2
Greece	2
Colombia	2
Croatia	2
Portugal	2
Moldova	1
Georgia	1
Bahamas, The	1
China	1
Thailand	1
Czech Republic	1
Norway	1
Romania	1
Nigeria	1
Japan	1
Hungary	1
Bosnia and Herzegovina	1
Uruguay	1
Spain	1
Zimbabwe	1
Latvia	1
Costa Rica	1
Slovenia	1
Philippines	1

Name: Country, dtype: int64

data[data['Country'] == 'United States']['state'].value_counts().head() In [18]:

Out [18]:

CA	138
WA	70
NY	56
TN	45
TX	44

Name: state, dtype: int64

data['state'].mode() In [19]:

0 CA Out [19]:
dtype: object

data['state'].value_counts()[:10] In [20]:

Out [20]:

CA	138
WA	70
NY	57
TN	45
TX	44
OH	30
IL	29
OR	29
PA	29
IN	27

Name: state, dtype: int64

data[data['Country']! = 'United States'] In [21]:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_emp
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Rarely	6-25
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100
7	2014-08-27 11:32:05	39	M	Canada	NaN	NaN	No	No	Never	1-5
9	2014-08-27 11:32:43	23	Male	Canada	NaN	NaN	No	No	Never	26-100
11	2014-08-27 11:32:49	29	male	Bulgaria	NaN	NaN	No	No	Never	100-50(
...	...	...	...	...	...	...	...	...	...	...
1244	2015-05-05 15:16:25	32	female	United Kingdom	NaN	No	No	No	NaN	More th 1000
1245	2015-05-06 10:14:50	22	Male	Australia	NaN	No	Yes	Yes	Often	100-50(
1247	2015-05-07 10:08:50	36	male	Finland	NaN	No	No	Yes	Often	6-25
1251	2015-08-17 09:38:35	36	Male	South Africa	NaN	No	Yes	Yes	Often	100-50(
1254	2015-09-12 11:17:21	26	male	United Kingdom	NaN	No	No	Yes	NaN	26-100

[508 rows x 27 columns]

#Percentage of data outside of the United States

100 * (data[data['Country']! = 'United States'].shape[0] / data.shape[0]) In [22]:

40.3494837172359 Out [22]:

We face a situation where **40%** of the data is from outside the United States, and the 'state' field has **40%** missing values.

- It's important to note that the mode for the 'state' field is California, a location within theUnited States.
- Therefore, it wouldn't be appropriate to replace missing values in the 'state' column fordifferent countries with 'California'.

```
data['self_employed'].value_counts ()                                    In [23]:

No     1095                                                              Out [23]:
Yes     146
Name: self_employed, dtype: int64

data['work_interfere'].value_counts ()                                   In [24]:

Sometimes    465                                                         Out [24]:
Never        213
Rarely         3
Often        144
Name: work_interfere, dtype: int64

data['Gender'].value_counts ()                                           In [25]:

Male                              615                                    Out [25]:
male                              206
Female                            121
M                                 116
female                             62
F                                  38
m                                  34
f                                  15
Make                                4
Male                                3
Woman                               3
Cis Male                            2
Man                                 2
Female (trans)                      2
Female                              2
Trans woman                         1
msle                                1
male leaning androgynous            1
Neuter                              1
cis male                            1
queer                               1
Female (cis)                        1
Mail                                1
cis-female/femme                    1
A little about you                  1
Malr                                1
p                                   1
femail                              1
Cis Man                             1
Guy (-ish)                          1
Enby                                1
Agender                             1
Androgyne                           1
Male-ish                            1
```

```
Trans-female                        1
Cis Female                          1
something kinda  male?              1
Mal                                 1
Male (CIS)                          1
queer/she/they                      1
non-binary                          1
Femake                              1
woman                               1
Nah                                 1
All                                 1
fluid                               1
Genderqueer                         1
ostensibly male, unsure what that really means      1
Name: Gender, dtype: int64
```

- To facilitate a clearer analysis, we will categorize the 'Gender' field into **'Male'**,' **'Female',** and **'Trans'**.

Corrections needed:

1. Inconsistencies in the Age field.
2. Missing Values:
   - state - Delete the field. (40% values are missing).
   - self_employed - Replace with Mode. (1.4% values are missing).
   - work_interfere - Replace with Mode. (21% values are missing).
   - comments - Delete the field. (87% values are missing).
3. Duplicate - not allowed.
4. Gender - Categorize into Male, Female & Others - for the better understanding of our analysis.
5. Type Casting of Datetime format.

Data PreProcessing

- **Performing the above mentioned corrections for the Missing values**

In [26]:

```
data['self_employed'].mode()[0]
```

Out [26]:

```
'No'
```

```
data['self_employed'] = data['self_employed'].replace(np.nan, data['self_employed'].mode()[0])
data['work_interfere'] = data['work_interfere'].replace(np.nan, data['work_interfere'].mode()[0])
data.drop(['state','comments'], axis=1, inplace = True)
```

- **Lets verify the integrity of missing values again.**

data_missing['Null percentage'] = 100*((data.shape[0]-data.count())/data.shape[0])

In [29]:

data_missing

Out [29]:

	Null	Null percentage
Timestamp	0	0.00
Age	0	0.00
Gender	0	0.00
Country	0	0.00
state	515	NaN
self_employed	18	0.00
family_history	0	0.00
treatment	0	0.00
work_interfere	264	0.00
no_employees	0	0.00
remote_work	0	0.00
tech_company	0	0.00
benefits	0	0.00
care_options	0	0.00
wellness_program	0	0.00
seek_help	0	0.00
anonymity	0	0.00
leave	0	0.00
mental_health_consequence	0	0.00
phys_health_consequence	0	0.00
coworkers	0	0.00
supervisor	0	0.00

	Null	Null percentage
mental_health_interview	0	0.00
phys_health_interview	0	0.00
mental_vs_physical	0	0.00
obs_consequence	0	0.00
comments	1095	NaN

100*((data.shape[0]-data.count())/data.shape[0])

In [30]:

Out [30]:

```
Timestamp          0.00
Age                0.00
Gender             0.00
Country            0.00
self_employed      0.00
```

```
family_history                0.00
treatment                     0.00
work_interfere                0.00
no_employees                  0.00
remote_work                   0.00
tech_company                  0.00
benefits                      0.00
care_options                  0.00
wellness_program              0.00
seek_help                     0.00
anonymity                     0.00
leave                         0.00
mental_health_consequence     0.00
phys_health_consequence       0.00
coworkers                     0.00
supervisor                    0.00
mental_health_interview       0.00
phys_health_interview         0.00
mental_vs_physical            0.00
obs_consequence               0.00
dtype: float64
```

- **Let’s check for any duplicated values**

data.duplicated().any() In [31]:

False Out [31]:

print('Contains any Duplicated rows ?', data.duplicated().any()) In [32]:

Contains any Duplicated rows ? False Out [32]:

- **Typecasting of Timestamp**

data['Timestamp'].unique() In [33]:

```
array(['2014-08-27 11:29:31', '2014-08-27 11:29:37',
       '2014-08-27 11:29:44', ..., '2015-11-07 12:36:58',
       '2015-11-30 21:25:06', '2016-02-01 23:04:31'], dtype=object)
```
Out [33]:

data['Timestamp'] = pd.to_datetime(data['Timestamp']) In [34]:

data.info () In [36]:

<class 'pandas.core.frame.DataFrame'> Out [36]:

```
RangeIndex:1259 entries, 0 to 1258
Data columns (total 25 columns):
 #   Column                         Non-Null Count  Dtype
```

0	Timestamp	1259 non-null	datetime64[ns]
1	Age	1259 non-null	int64
2	Gender	1259 non-null	object
3	Country	1259 non-null	object
4	self_employed	1259 non-null	object
5	family_history	1259 non-null	object
6	treatment	1259 non-null	object
7	work_interfere	1259 non-null	object
8	no_employees	1259 non-null	object
9	remote_work	1259 non-null	object
10	tech_company	1259 non-null	object
11	benefits	1259 non-null	object
12	care_options	1259 non-null	object
13	wellness_program	1259 non-null	object
14	seek_help	1259 non-null	object
15	anonymity	1259 non-null	object
16	leave	1259 non-null	object
17	mental_health_consequence	1259 non-null	object
18	phys_health_consequence	1259 non-null	object
19	coworkers	1259 non-null	object
20	supervisor	1259 non-null	object
21	mental_health_interview	1259 non-null	object
22	phys_health_interview	1259 non-null	object
23	mental_vs_physical	1259 non-null	object
24	obs_consequence	1259 non-null	object

dtypes: datetime64[ns](1), int64(1),
object(23)memory usage: 246.0+ KB

- **We need to address the concerns present in the Age & Gender fields.**

#unique values in the Gender field

In [37]:

```
print('Unique Genders present in the data :', data['Gender'].nunique())
print('Unique Genders present in the data :', set(data['Gender']))
```

Unique Genders present in the data : 49

Unique Genders present in the data : {'msle', 'Male', 'Cis Male', 'Cis Man', 'fluid', 'Male ', 'f', 'woman', 'male', 'Female', 'queer', 'Guy (-ish) ^_^', 'Nah', 'Agender', 'male leaning androgynous',
'maile', 'Female (trans)', 'Cis Female', 'All', 'Trans woman', 'Mal', 'queer/she/they',
'Enby', 'F', 'p', 'femail', 'A little about you', 'Male-ish', 'Genderqueer', 'Make', 'female',
'Androgyne', 'Female (cis)', 'Woman', 'Femake', 'Mail', 'm', 'cis male', 'M', 'Man',
'Trans-female', 'cis-female/femme','Female ', 'Male (CIS)', 'ostensibly male, unsure what that really means', 'non-binary', 'Neuter','Malr', 'something kinda male?'}

In [38]:

```
data['Gender'].str.lower().unique()

array(['female', 'm', 'male', 'male-ish', 'maile', 'trans-female',
    'cis female', 'f', 'something kinda male?', 'cis male', 'woman','mal', 'male (cis)',
    'queer/she/they', 'non-binary', 'femake',
    'make', 'nah', 'all', 'enby', 'fluid', 'genderqueer', 'female ', 'androgyne', 'agender',
    'cis-female/femme', 'guy (-ish) ^_^', 'male leaning androgynous', 'male ', 'man',
    'trans woman', 'msle','neuter', 'female (trans)', 'queer', 'female (cis)', 'mail',
    'a little about you', 'malr', 'p', 'femail', 'cis man',
    'ostensibly male, unsure what that really means'], dtype=object)

unique_gender = data['Gender'].str.lower().unique()

# Stratas of Gender category
male_str = ["male", "m", "male-ish", "maile", "mal", "male (cis)", "make",
        "male ", "man","msle", "mail", "malr","cis man", "Cis Male","cis male"]
trans_str = ["trans-female", "something kinda male?", "queer/she/they","non-
        binary","nah", "all", "enby", "fluid", "genderqueer",
        "androgyne", "agender", "male leaning androgynous", "guy (-ish) ^_^","trans woman",
        "neuter", "female (trans)", "queer",
        "ostensibly male, unsure what that really means"]
female_str = ["cis female", "f", "female", "woman", "femake", "female ","cis-
        female/femme", "female (cis)", "femail"]

# Iterate over rows and replace the inconsistent data with right data
for (row, col) in data.iterrows():
  if str.lower(col['Gender']) in male_str: data['Gender'].replace(to_replace=col['Gender'],
    value='male', inplace=True)

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

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

# Remove rest of the unnecessary text
stk_list = ['A little about you', 'p']
data = data[~data['Gender'].isin(stk_list)]

# Display the unique value of Gender feature
print(data['Gender'].uni

que())['female' 'male'

'trans']
```

- Substituting the outliers in the Age field with the **median** value, as the median is not influenced by the outliers.

```
data['Age'].median()                                         In [40]:
```

```
31.0                                                         Out [40]:
```

```
data['Age'][data['Age'] > 75] = data['Age'].median()         In [41]:
data['Age'][data['Age'] < 18] = data['Age'].median()         In [42]:
```

```
data['Age'].describe()                                       Out [42]:
```

```
count    1257.00
mean     32.07
std      7.27
min      18.00
25%      27.00
50%      31.00
75%      36.00
max      72.00
Name: Age, dtype: float64
```

```
data.info ()                                                 In [43]:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1257 entries, 0 to 1258
Data columns (total 25 columns):                             Out [43]:
 #   Column                     Non-Null Count  Dtype
--  -------                     --------------  ------
 0   Timestamp                  1257 non-null   datetime64[ns]
 1   Age                        1257 non-null   int64
 2   Gender                     1257 non-null   object
 3   Country                    1257 non-null   object
 4   self_employed              1257 non-null   object
 5   family_history             1257 non-null   object
 6   treatment                  1257 non-null   object
 7   work_interfere             1257 non-null   object
 8   no_employees               1257 non-null   object
 9   remote_work                1257 non-null   object
 10  tech_company               1257 non-null   object
 11  benefits                   1257 non-null   object
 12  care_options               1257 non-null   object
 13  wellness_program           1257 non-null   object
 14  seek_help                  1257 non-null   object
 15  anonymity                  1257 non-null   object
 16  leave                      1257 non-null   object
 17  mental_health_consequence  1257 non-null   object
 18  phys_health_consequence    1257 non-null   object
 19  coworkers                  1257 non-null   object
 20  supervisor                 1257 non-null   object
 21  mental_health_interview    1257 non-null   object
 22  phys_health_interview      1257 non-null   object
 23  mental_vs_physical         1257 non-null   object
 24  obs_consequence            1257 non-null   object
dtypes: datetime64[ns](1), int64(1),
```

object(23)memory usage: 255.3+ KB

7.2 Exploratory Data Analysis

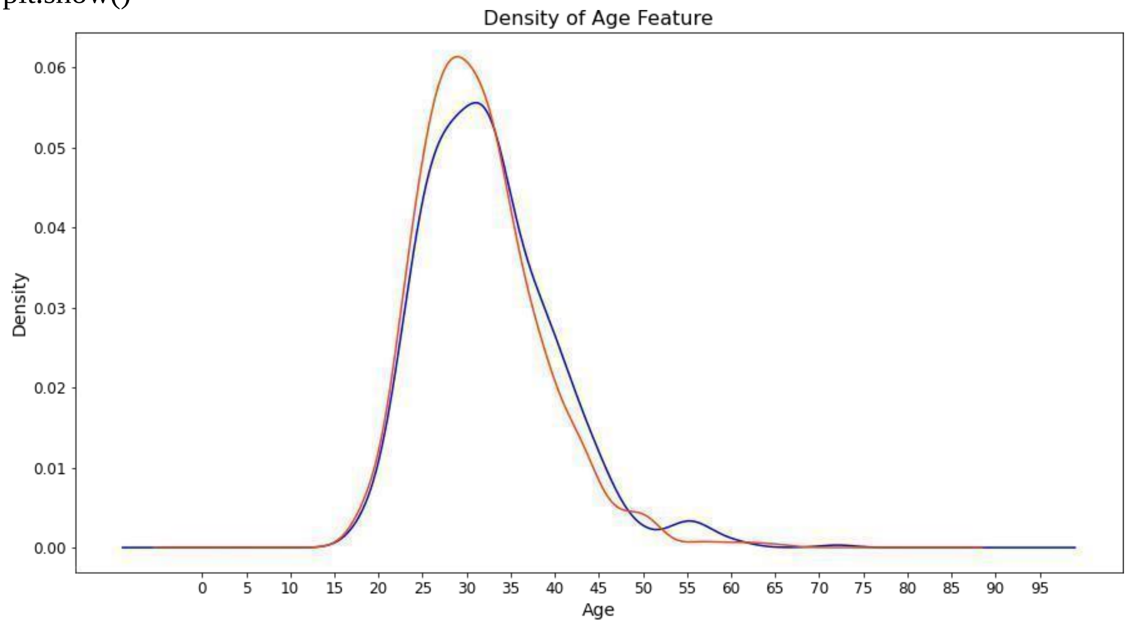
A Series of questions to uncover patterns, trends, anomalies, relationships, and key insights without making any formal assumptions about the data.

Q. Which Age group are more conscious about their mental health?

```
#Lets try to plot Age vs Treatment
data['treatment'].value_counts()
```

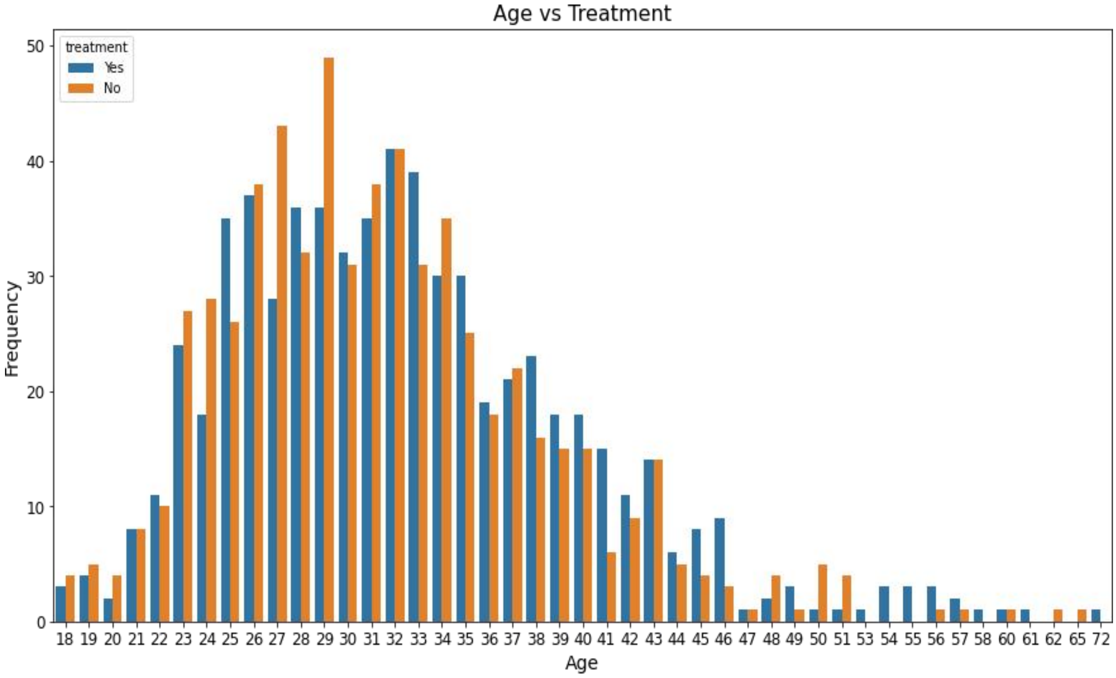
```
Yes   635
No    622
Name: treatment, dtype: int64
```

```
figure = plt.figure(figsize=[15, 8])
data[data['treatment'] ==
'Yes']['Age'].plot.kde(color='blue')    data[data['treatment']
== 'No']['Age'].plot.kde(color='orangered')

plt.xticks(ticks=np.arange(0, 100, 5),
size=12)plt.yticks(size=12)
plt.xlabel(xlabel='Age', size=14)
plt.ylabel(ylabel='Density', size=14)
plt.title(label='Density of Age Feature',
size=16)

plt.show()
```




```
fig = plt.figure(figsize=(15, 8))

sns.countplot(x='Age',  hue='treatment',

data=data)
```

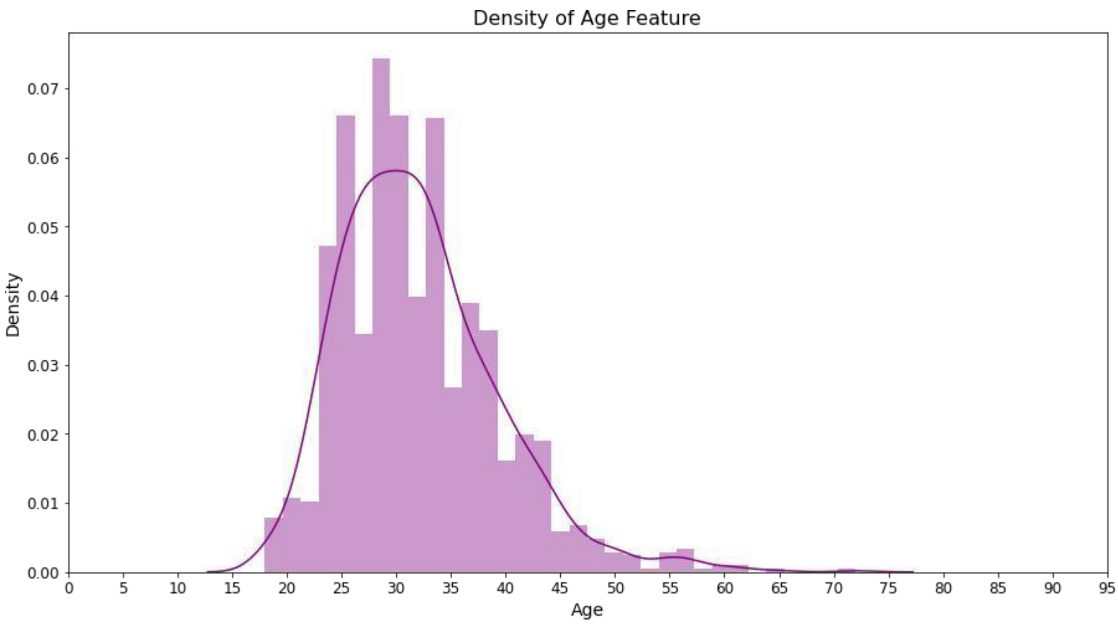
```
plt.title(label='Age vs Treatment',
size=16)plt.xlabel(xlabel='Age',
size=14)
plt.ylabel(ylabel='Frequency',
size=14) plt.xticks(size=12)
plt.yticks(size=12)
#plt.grid(b=True)

Plt.show ()
```



- Analyzing the proportions, it suggests that individuals over the age of **30** are addressingtheir mental health concerns.

Q. What is the Density distribution Age field?

```
figure = plt.figure(figsize=[15, 8])

sns.distplot(data['Age'], kde=True, color = 'purple')
plt.xticks(ticks=np.arange(0, 100, 5),
size=12)plt.yticks(size=12)
plt.xlabel(xlabel='Age', size=14)
plt.ylabel(ylabel='Density', size=14)
plt.title(label='Density of Age Feature',
size=16)
plt.show()
```

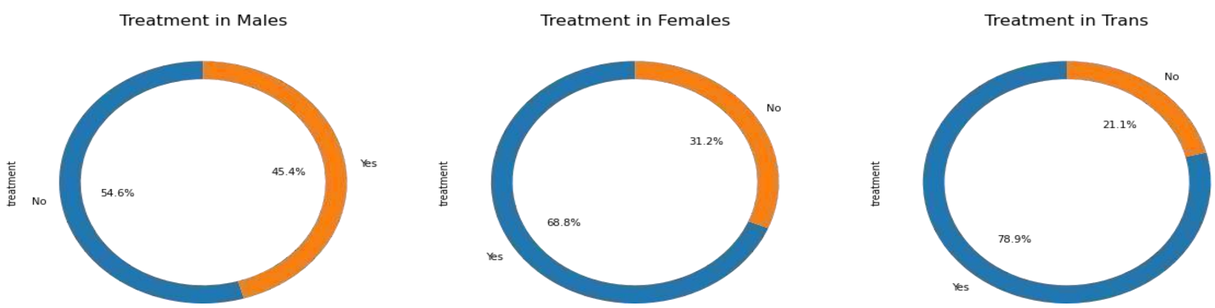


Observation:

- The data shows a prominent peak occurring between the **mid-20s** to about **mid-30s**, indicating that the majority of individuals fall within this age range.

Q. What is the association between Gender & Treatment?

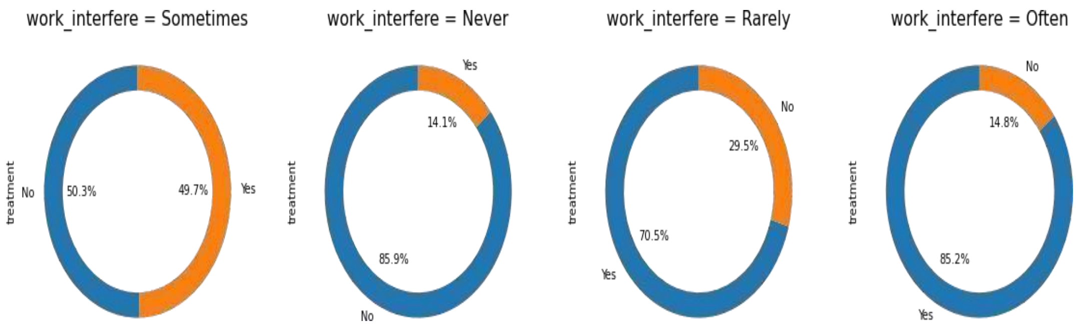
```
figure = plt.figure(figsize=[20, 10])

    plt.subplot(1,3,1)
    data['treatment'][data['Gender'] == 'male'].value_counts().plot(kind='pie', autopct='
    %1.1f%%', wedgeprops = dict(width = 0.15), startangle=90)
    plt.title(label='Treatment in Males', size=16)

    plt.subplot(1,3,2)
    data['treatment'][data['Gender'] ==
    'female'].value_counts().plot.pie(autopct='%1.1f%%',wedgeprops = dict(width = 0.15),
    startangle=90)
    plt.title(label='Treatment in Females', size=16)
    plt.subplot(1,3,3)
    data['treatment'][data['Gender'] == 'trans'].value_counts().plot.pie(autopct='%1.1f%%',
    wedgeprops = dict(width = 0.15), startangle=90)
    plt.title(label='Treatment in Trans', size=16)

    plt.show()
```



- Based on the data, it appears that individuals identifying as **Trans** and **Females** exhibit a higher tendency to seek treatment for mental health issues in comparison to **Males**.

Q. What is the association between Treatment & Work_interference?

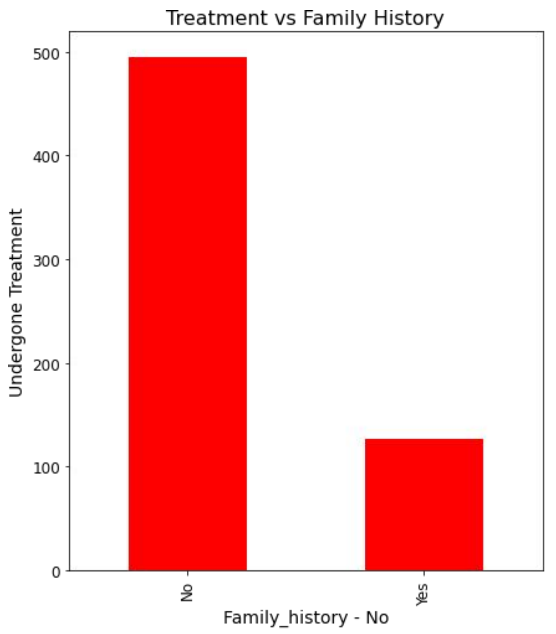
```
data['treatment'].value_counts()

Yes        635
No         622
Name: treatment, dtype: int64

data['work_interfere'].value_counts()

Sometimes    729
Never        213
Rarely       173
Often        142
Name: work_interfere, dtype: int64

figure = plt.figure(figsize=[20,

10])

plt.subplot(1,4,1)
data['treatment'][data['work_interfere'] ==
    'Sometimes'].value_counts().plot(kind='pie', autopct='%1.1f%%',
    wedgeprops = dict(width = 0.20), startangle=90)
plt.title(label='work_interfere = Sometimes', size=16)

plt.subplot(1,4,2)
data['treatment'][data['work_interfere']
==
    'Never'].value_counts().plot.pie(autopct='%1.1f%%',wedgeprops = dict(width
    = 0.20), startangle=90)
    plt.title(label='work_interfere = Never', size=16)

plt.subplot(1,4,3)
data['treatment'][data['work_interfere'] == 'Rarely'].value_counts().plot(kind='pie',
    autopct='%1.1f%%', wedgeprops = dict(width = 0.20), startangle=90)
plt.title(label='work_interfere = Rarely', size=16)

plt.subplot(1,4,4)
data['treatment'][data['work_interfere']
==

    'Often'].value_counts().plot.pie(autopct='%1.1f%%',wedgeprops = dict(width
    = 0.20), startangle=90)
plt.title(label='work_interfere = Often',

size=16)plt.show()
```

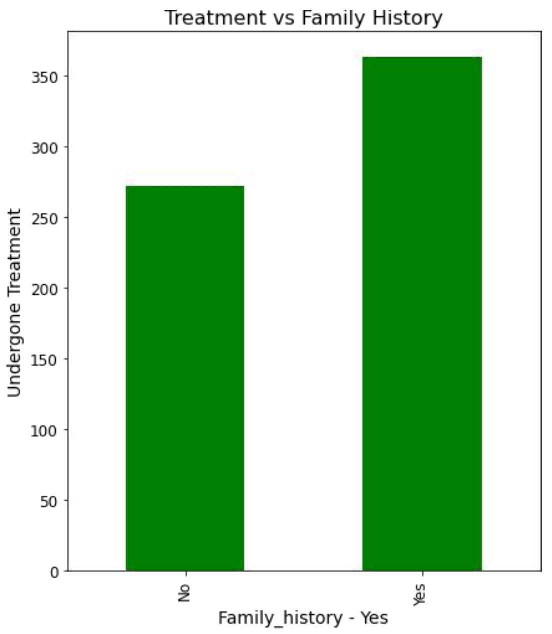


- We can observe that employees who are more **'Often'** & **'Rarely'** interfered during work are likely to have Mental health issues and hence are seeking Treatment.

Q. Do individuals show a greater willingness to seek treatment for mental health issues ifthere is a family history of such conditions?

```
data['family_history'].value_counts()
No     767
Yes    490
Name: family_history, dtype: int64

data.groupby(['treatment','family_history'])['family_history'].count()['No']
family_history
No     495
Yes    127
Name: family_history, dtype: int64

data.groupby(['treatment','family_history'])['family_history'].count()['Yes']
family_history
No     272
Yes    363
Name: family_history, dtype: int64

figure = plt.figure(figsize=[15, 8])
    plt.subplot(1,2,1)
    data.groupby(['treatment','family_history'])['family_history'].count()['No'].plot.bar(color='red')
    plt.xticks(size=12)
    plt.yticks(size=12)
    plt.xlabel(xlabel='Family_history - No', size=14)
    plt.ylabel(ylabel='Undergone Treatment', size=14)
    plt.title(label='Treatment vs Family History', size=16)

    plt.subplot(1,2,2)
    data.groupby(['treatment','family_history'])['family_history'].count()['Yes'].plot.bar(color='gree
n')
    plt.xticks(size=12)
    plt.yticks(size=12)
    plt.xlabel(xlabel='Family_history - Yes', size=14)
    plt.ylabel(ylabel='Undergone Treatment', size=14)
    plt.title(label='Treatment vs Family History', size=16)
    plt.show()
```



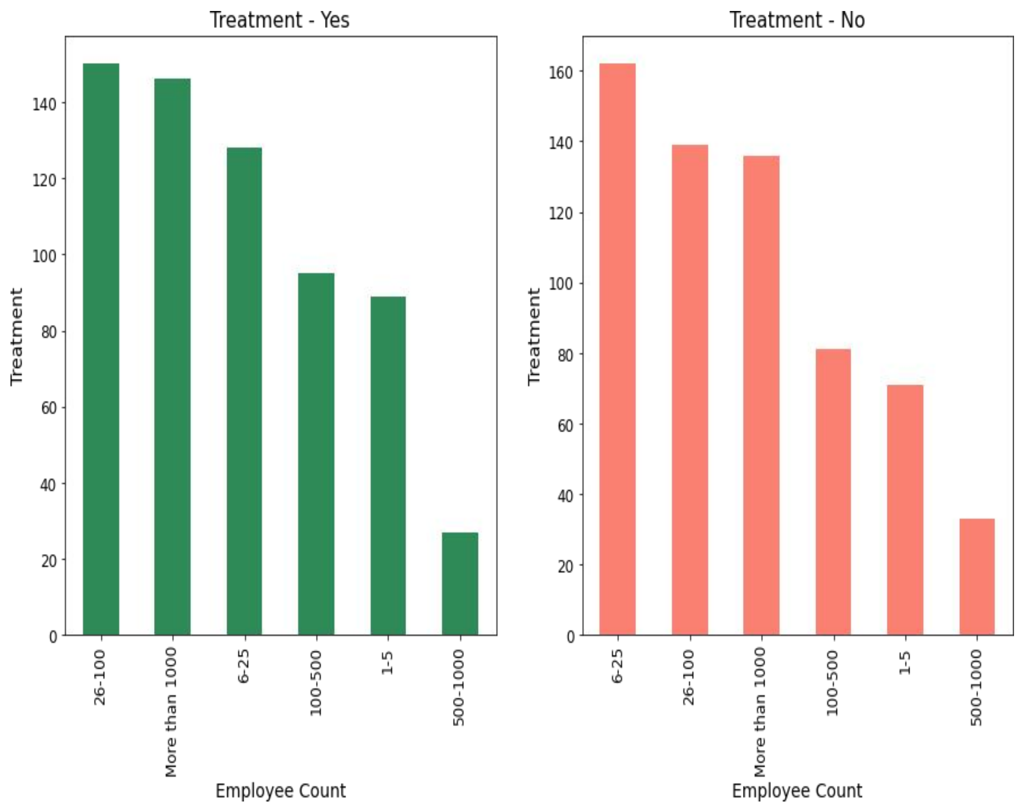


- We observe that employees with a **family history** of mental health issues are **moreinclined** to choose **treatment**.
- In contrast, employees without a family history of mental health issues may have **lowerawareness** and, consequently, a reduced likelihood of seeking treatment.

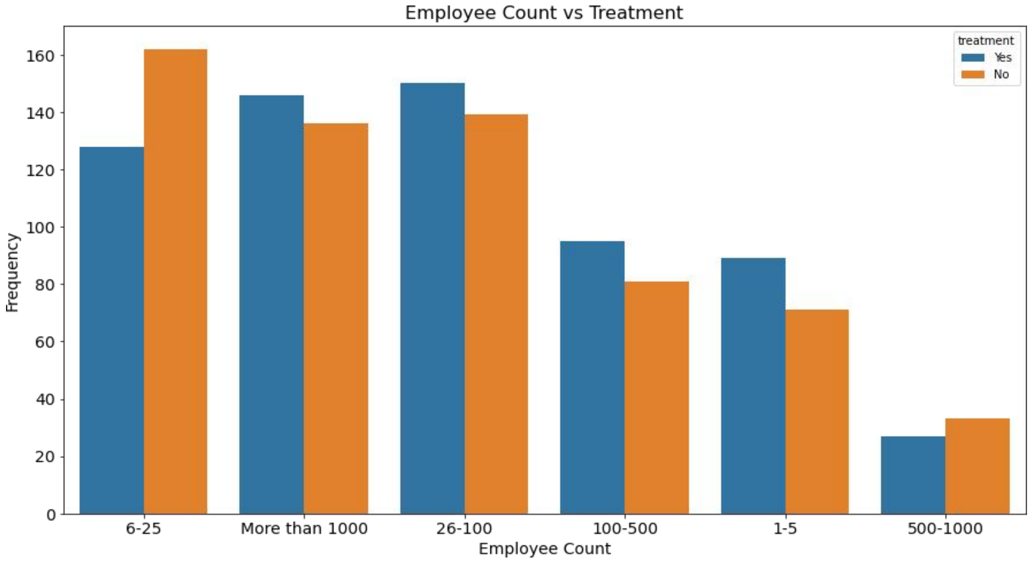
Q. What is the association between treatment and employee count in a company?

```
data['treatment'].value_counts()
Yes    635
No     622
Name: treatment, dtype: int64

data['no_employees'].value_counts()
6-25              290
26-100            289
More than 1000    282
100-500           176
1-5               160
500-1000           60
Name: no_employees, dtype: int64

data[data['treatment'] == 'Yes']['no_employees'].value_counts()
26-100            150
More than 1000    146
6-25              128
100-500            95
1-5                89
500-1000           27
Name: no_employees, dtype: int64

data[data['treatment'] == 'No']['no_employees'].value_counts()
6-25              162
26-100            139
More than 1000    136
```

```
100-500         81
1-5             71
500-1000        33
Name: no_employees, dtype: int64

figure = plt.figure(figsize=(15,8))
plt.subplot(1,2,1)
data[data['treatment']  ==  'Yes']['no_employees'].value_counts().plot.bar(color='seagreen')

    plt.xticks(size=12)
    plt.yticks(size=12)
    plt.xlabel(xlabel='Employee Count', size=14)
    plt.ylabel(ylabel='Treatment', size=14)
    plt.title(label='Treatment - Yes', size=16)
    plt.subplot(1,2,2)

    data[data['treatment'] == 'No']['no_employees'].value_counts().plot.bar(color='salmon')

    plt.xticks(size=12)
    plt.yticks(size=12)
    plt.xlabel(xlabel='Employee Count', size=14)
    plt.ylabel(ylabel='Treatment', size=14)
    plt.title(label='Treatment - No', size=16)

    plt.show()
```



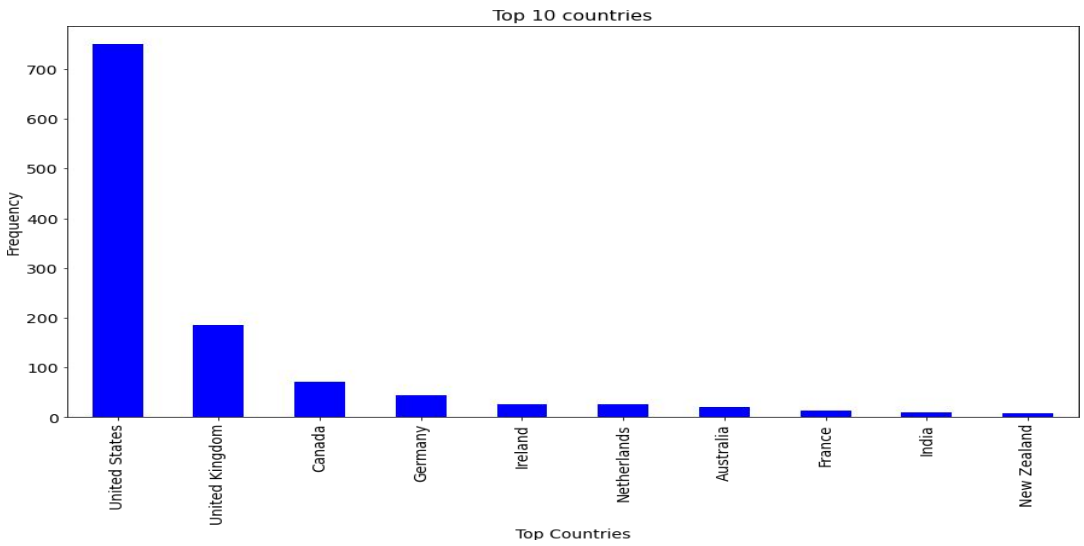
```
figure = plt.figure(figsize=[15, 8])
sns.countplot(x = 'no_employees', hue ='treatment', data=data)

plt.xticks(size=14)
plt.yticks(size=14)
plt.xlabel(xlabel ='Employee Count',
size=14)plt.ylabel(ylabel ='Frequency',
size=14)
plt.title(label ='Employee Count vs Treatment', size=16)

plt.show()
```



- Based on the data, it can be inferred that the highest number of employees who soughtmental health treatment belong to companies sized between **26-100** employees.
- Conversely, the largest number of employees who did not seek treatment come fromcompanies sized between **6-25** employees

Q. Top 10 Countries recorded for mental health treatment?

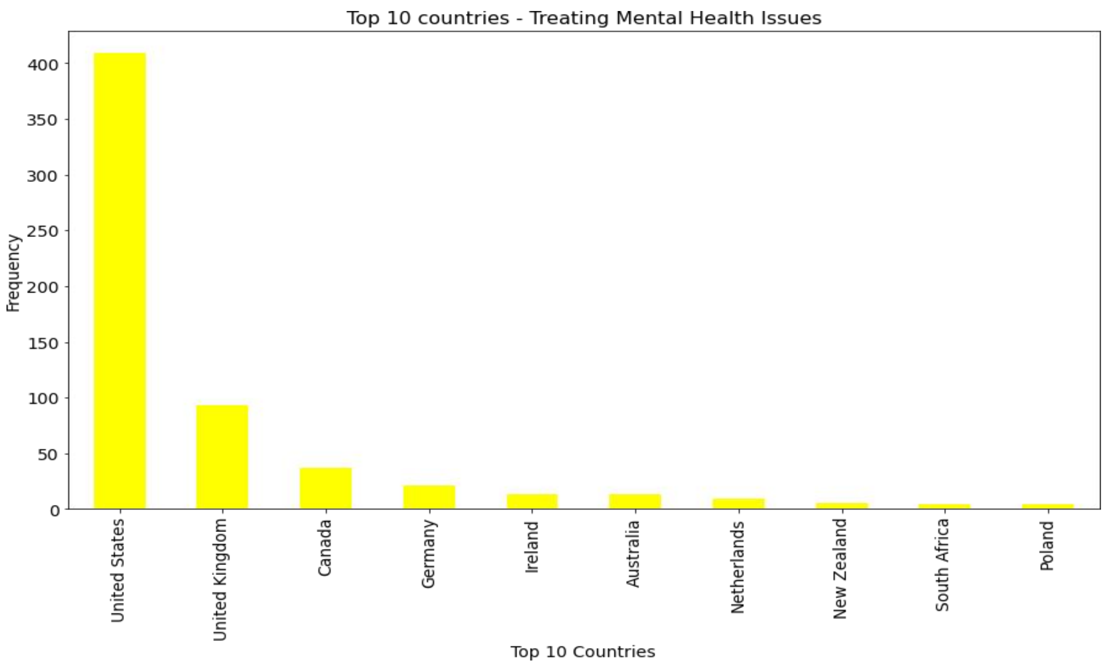
```
fig = plt.figure(figsize=[15,8]) data['Country'].value_counts().head(10).plot.bar(color='purple')
    plt.xticks(rotation='vertical', size=14)
 plt.yticks(size=14)
plt.xlabel(xlabel ='Top Countries',
size=14)plt.ylabel(ylabel
='Frequency', size=14) plt.title(label
='Top 10 countries', size=16)

plt.show()
```



- The majority of the records are from the United States, followed by the United Kingdom and Canada.

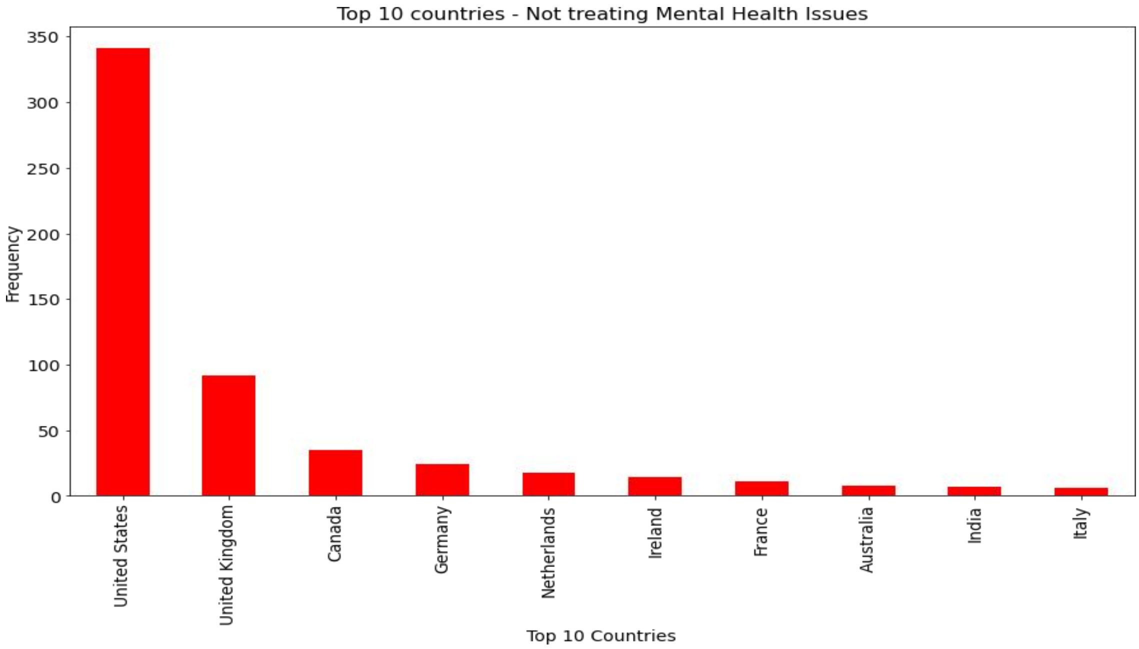
Q. Which countries are actually contributing more for mental health treatment?

```
fig = plt.figure(figsize=[15,8])

data[data['treatment']== 'Yes']['Country'].value_counts().head(10).plot.bar(color='seagreen')
plt.xticks(rotation='vertical', size=14)
plt.yticks(size=14)
plt.xlabel(xlabel ='Top 10 Countries', size=14)
plt.ylabel(ylabel ='Frequency', size=14)

plt.title(label ='Top 10 countries - Treating Mental Health Issues', size=16)
plt.show()
```



- We observe a shift in the lower section of the bar chart, indicating countries where a **larger number** of individuals are seeking treatment for their mental health.

```
fig = plt.figure(figsize=[15,8])
data[data['treatment']== 'No']['Country'].value_counts().head(10).plot.bar(color='lightcoral')
plt.xticks(rotation='vertical', size=14)
plt.yticks(size=14)
plt.xlabel(xlabel ='Top 10 Countries', size=14)
plt.ylabel(ylabel ='Frequency', size=14)
plt.title(label ='Top 10 countries - Not treating Mental Health Issues', size=16)

plt.show()
```



- Presented are the Top 10 countries where individuals are **least** inclined to seek treatment for their mental health concerns.
- The data illustrates that the **United States**, **United Kingdom**, **Canada**, and **Germany** rank highest in both categories: **treating a significant number of individuals** with mental health issues and concurrently having the **highest number of untreated mental health cases**. This paradox within the statistics underscores the contradictory nature of our statement.
- To resolve the aforementioned paradox, let's conduct a comprehensive analysis focusing on the data distribution, specifically examining countries that meet the condition where **Treatment** equals **'Yes'** out of the **total values recorded**.
- **Let's calculate the ratio of observations from countries addressing mental health issues to the total number of countries included in the dataset.**

```
df_yes = data[data['treatment']== 'Yes']['Country'].value_counts().head(10)

df_yes.sort_values(ascending=False)
```

United States	409
United Kingdom	93
Canada	37
Germany	21
Ireland	13
Australia	13

```
Netherlands        9
New Zealand        5
South Africa       4
Poland             4
Name: Country, dtype: int64
```

```
data['Country'].value_counts().head(10)
```

```
United States      750
United Kingdom     185
Canada             72
Germany            45
Ireland            27
Netherlands        27
Australia          21
France             13
India              10
New Zealand         8
Name: Country, dtype: int64
```

```
df_yes.sort_values(ascending=False)  /  data['Country'].value_counts().head(10)
```

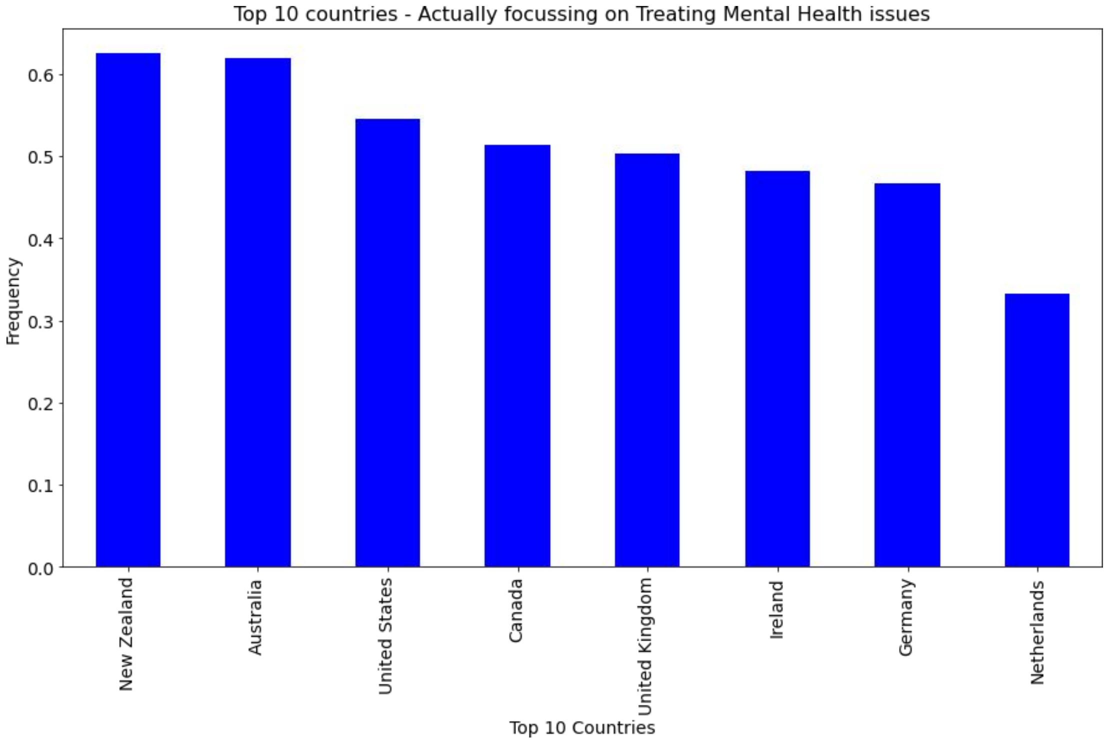
```
Australia       0.62
Canada          0.51
France          NaN
Germany         0.47
India           NaN
Ireland         0.48
Netherlands     0.33
New Zealand 0.62
Poland          NaN
South Africa    NaN
United Kingdom   0.50
United States      0.55
Name: Country, dtype: float64
```

```
df_yt = df_yes.sort_values(ascending=False) / data['Country'].value_counts().head(10)
df_yt.dropna().sort_values(ascending=False)
```

```
New Zealand       0.62
Australia         0.62
United States     0.55
Canada            0.51
United Kingdom    0.50
Ireland           0.48
Germany           0.47
Netherlands       0.33
Name: Country, dtype: float64
```

```
fig = plt.figure(figsize=[15,8])

df_yt.dropna().sort_values(ascending=False).plot.bar(color='blue')

plt.xticks(rotation='vertical', size=14)
```

```
plt.yticks(size=14)
plt.xlabel(xlabel ='Top 10 Countries', size=14)
plt.ylabel(ylabel ='Frequency', size=14)
plt.title(label ='Top 10 countries - Actually focussing on Treating Mental Health issues', size=16)

plt.show()
```



- These countries prioritize influencing a **significant proportion** of their population to address mental health issues, considering the total number of reported issues.
- **New Zealand** & **Australia** top the list, followed by United States & Canada.

Q. What is the contribution of top 3 countries among all in terms of mental health?

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

```
United States          750
United Kingdom          185
Canada                 72
Name: Country, dtype: int64
```

```
list(data['Country'].value_counts()[:3].index)
```

```
'United States', 'United Kingdom', 'Canada']
```

```
data_top3 = data[data['Country'].is
in(list(data['Country'].value_counts()[:3].index))]data_top3.head()
```

	Timestamp	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_employees	remote_work
0	2014-08-27 11:29:31	37	female	United States	No	No	Yes	Often	6-25	No
1	2014-08-27 11:29:37	44	male	United States	No	No	No	Rarely	More than 1000	No
2	2014-08-27 11:29:44	32	male	Canada	No	No	No	Rarely	6-25	No
3	2014-08-27 11:29:46	31	male	United Kingdom	No	Yes	Yes	Often	26-100	No
4	2014-08-27 11:30:22	31	male	United States	No	No	No	Never	100-500	Yes

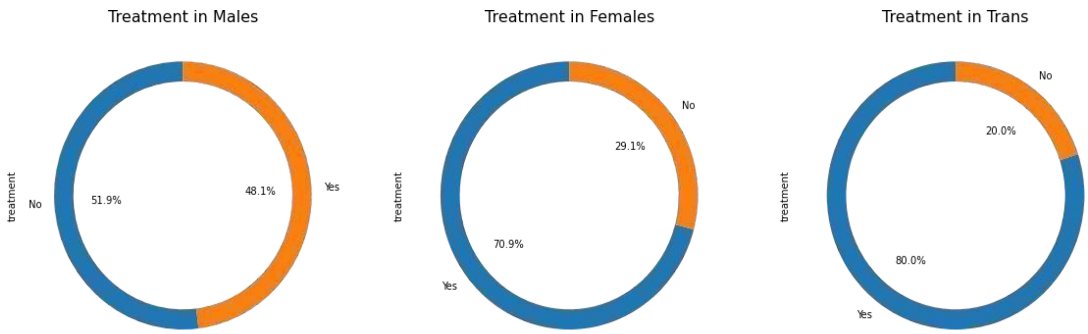
data_top3.shape(1007, 25)

 # Display the results
Print ('The number of people that exist from top 3 countries are: ', data_top3.shape[0])
Print ('Their proportion from total people surveyed is ', np.round(data_top3.shape[0]/data.shape[0], decimals=2))

The number of people that exist from top 3 countries are:
1007
Their proportion from total people surveyed is 0.8

Q. How many people did go for treatment based on gender for the top 3 countries?

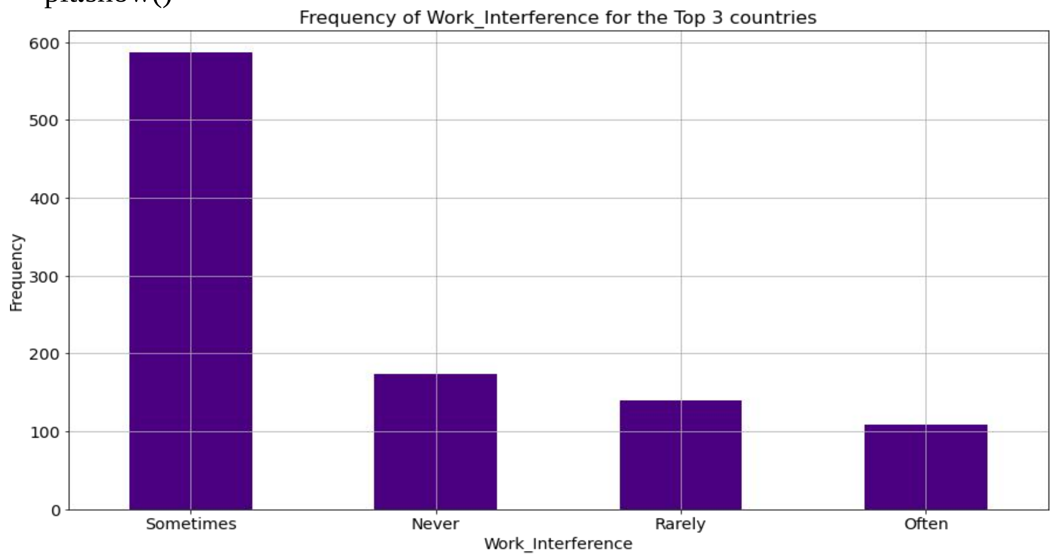
```
fig = lt.figure(figsize=[20,10])

plt.subplot(1,3,1)
data_top3['treatment'][data_top3['Gender'] ==
    'male'].value_counts().plot(kind='pie', autopct='%1.1f%%', wedgeprops =
    dict(width = 0.15), startangle=90)
plt.title(label='Treatment in Males', size=16)

plt.subplot(1,3,2)
data_top3['treatment'][data_top3['Gender'] ==
    'female'].value_counts().plot.pie(autopct='%1.1f%%', wedgeprops = dict(width = 0.15),
    startangle=90)
plt.title(label='Treatment in Females', size=16)

plt.subplot(1,3,3)
data_top3['treatment'][data_top3['Gender'] == 'trans'].value_counts().plot.pie(autopct='%1.1f%%',
    wedgeprops = dict(width = 0.15), startangle=90)
plt.title(label='Treatment

inTrans',size=16)

plt.show()
```



- **48%** of Males, **71%** of Females and **80%** of Trans, have gone through treatment among the top 3 countries.

Q. What is the frequency distribution of work interference among employees for the top 3 countries?

```
data_top3['work_interfere'].value_counts()

Sometimes   586
Never       173
Rarely      139
Often       109
Name: work_interfere, dtype: int64

figure = plt.figure(figsize=(15,8))
data_top3['work_interfere'].value_counts().plot.bar(color='indigo')
plt.xticks(rotation=0, size=14)
plt.yticks(size=14)
plt.xlabel(xlabel= 'Work_Interference', size=14)
plt.ylabel(ylabel='Frequency', size=14)
plt.title(label= 'Frequency of Work_Interference for the Top 3 countries',
size=16) plt.grid(True)

plt.show()
```



- The majority of individuals seeking treatment for their mental health issues experienced interference with their work at times.

Q. Relation between Treatment and Mental Health Consequence?

'**mental_health_consequence -** Do you think that discussing a mental health issue with your employer would have negative consequences?'

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

```
No       490
Maybe    477
Yes      290
Name: mental_health_consequence, dtype: int64
```

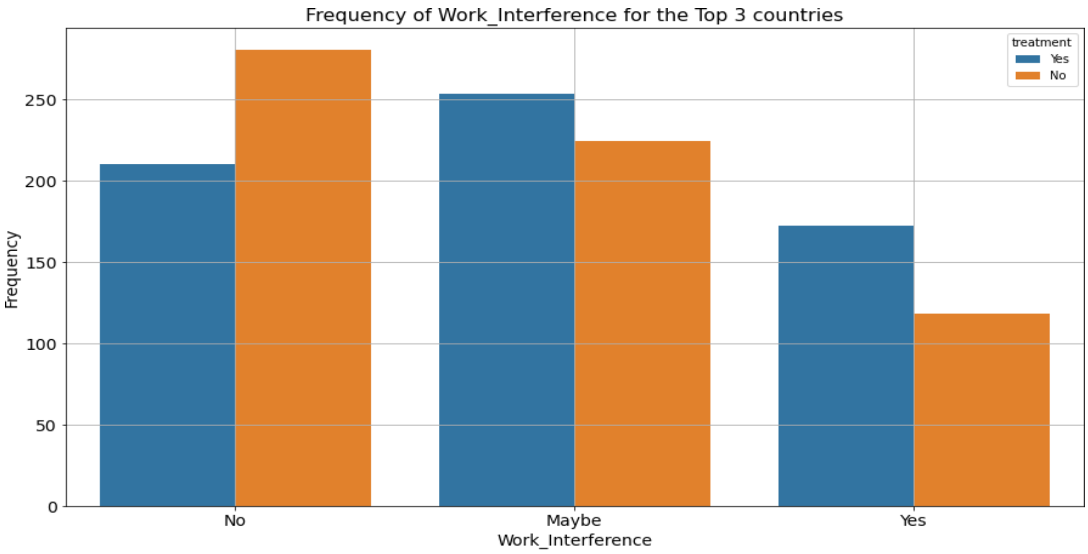
```
data[data['treatment']=='Yes']['mental_health_consequence'].value_counts()
```

```
Maybe    253
No       210
Yes      172
Name: mental_health_consequence, dtype: int64
```

```
data[data['treatment']=='No']['mental_health_consequence'].value_counts()
```

```
No       280
Maybe    224
Yes      118
Name: mental_health_consequence, dtype: int64
```

```
figure = plt.figure(figsize=(15,8))


sns.countplot(data=data, x='mental_health_consequence', hue='treatment')

plt.xticks(rotation=0, size=14)
plt.yticks(size=14)
plt.xlabel(xlabel= 'Work_Interference', size=14)
plt.ylabel(ylabel='Frequency', size=14)
plt.title(label= 'Frequency of Work_Interference for the Top 3 countries', size=16)
plt.grid(True)


plt.show()
```

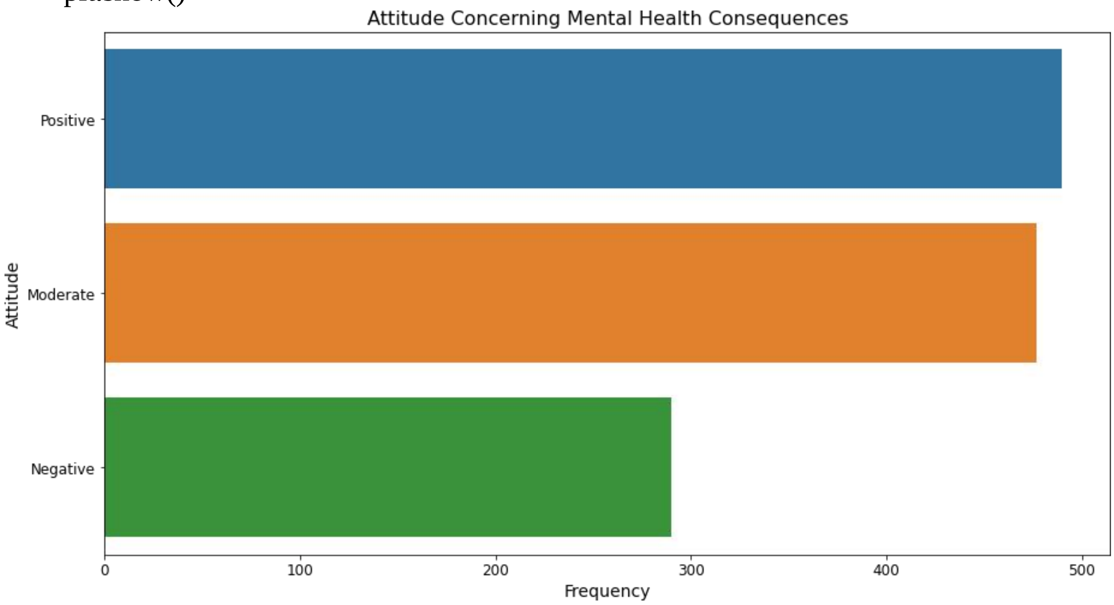


- Individuals who anticipate negative consequences when discussing their mental health issues with their employers show a higher willingness to seek treatment fortheir issues.
- Similarly, Individuals who feel comfortable discussing their mental health issues withtheir employers tend to show a lower willingness to seek treatment for their concerns.

Q. What is the relationship between mental health consequences and the attitude?

```
def attitude(x):
"""A custom function to map values in a feature."""
 if x == 'No':
 return 'Positive'
elif x == 'Yes':
 return 'Negative'
elif x == 'Maybe':
 return 'Moderate'
else:
 return x

data['attitudes'] = data['mental_health_consequence'].apply(attitude)
data['attitudes'].value_counts()

Positive 490
Moderate 477
Negative 290
Name: attitudes, dtype:
int64 figure =
plt.figure(figsize=[15, 8])

sns.countplot(y='attitudes', data=data)
plt.title(label='Attitude Concerning Mental Health Consequences', size=16)
plt.xlabel(xlabel='Frequency', size=14)
plt.ylabel(ylabel='Attitude', size=14)
plt.xticks(size=12)
```

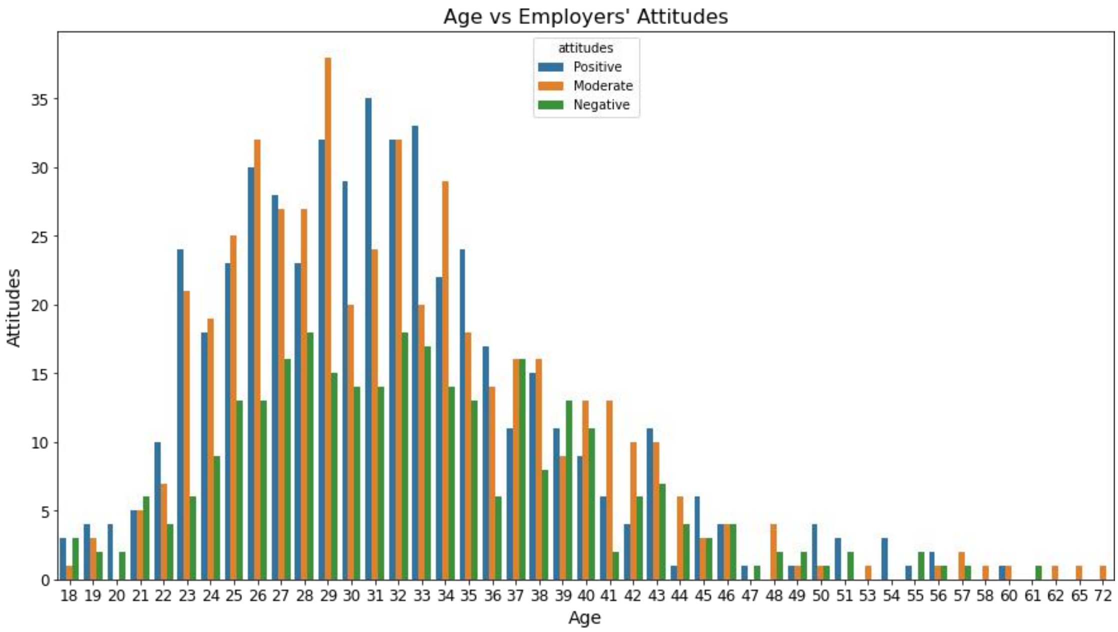
```
plt.yticks(size=12)
#plt.grid(b=True)

plt.show()
```



- The majority of individuals perceive their employers' attitudes to be more positive or moderately supportive rather than negative when addressing their mental health concerns.

Q. How does age relate to various behaviors and/or their awareness of their employer'sattitude toward mental health?

```
fig = plt.figure(figsize=(15, 8))

sns.countplot(x='Age', hue='attitudes', data=data)

plt.title(label="Age vs Employers' Attitudes", size=16)
plt.xlabel(xlabel='Age', size=14)
plt.ylabel(ylabel='Attitudes', size=14)
plt.xticks(size=12)
plt.yticks(size=12)
#plt.grid(b=True)

plt.show()
```



- This suggests that individuals in their **mid-20s** to **mid-30s** perceive their employers' attitude to be more positive or moderately supportive rather than negative when they discuss their mental health concerns.

Summarization

Conclusion:

- The mental health survey has helped us to understand the mental condition of employees working in tech firms across countries.
- A total of 1259 entries were recorded during the survey out of which 1007 were recorded from the top 3 countries.
- The United States leads the chart in terms of participation in the survey followed by the United Kingdom and Canada.
- 45% OF males, 69% of females, and 79% of trans were found to have sought treatment concerning the overall survey.
- Likewise, data indicates that 48% of males, 71% of females, and 80% of trans individuals have received treatment within the top three countries in the recorded dataset. The following set of parameters are found to be affecting mental health the most and thus requires treatment:
- Age
- Family history,
- Work Interference,
- Number of employees working in a company,
- New Zealand and Australia lead in prioritizing the resolution of employees' mental health issues, encouraging a higher number of individuals to seek treatment, followed by the United States and Canada.

Actionable Insights:

- There should be an **awareness program** about mental health and its effects.

- Implementing an awareness program on mental health and its effects is crucial to encourage greater participation from **males**, considering their **lower** representation among the survey participants.

- Relationship Managers should provide supportive guidance to their employees.

- **Managers** and **Employers** need to maintain **unbiased** attitudes toward both the work and the employees, offering appropriate measures and support for those experiencing mental health challenges.

- Regular **appreciation** at work is beneficial for employee well-being.

7.3 PREDICTION METHOD

- To predict whether the employees is susceptible to have mental health issues & treatment is required, different prediction models were implemented.
- The predictions were obtained by classifying the employees into two classes namely: ‘diagnosed for mental health issue’ and ‘not diagnosed for mental health issue’. The classification was done based on the target variable, treatment (‘Has the employee being diagnosed with a mental health condition’). 70% of the data set was used for training and30% for testing.
- Models tested include the GaussianNB, k-nearest neighbor (KNN) classifier, logistic regression, decision tree classifier, random forest, and Gradient Boosting classifiers. In a supervised learning environment, these classifiers were chosen based on their usefulness as small-data machine learning models and the effective-ness of earlier efforts for similar understanding of the older OSMI data.

Gaussian NB

```
import pandas
as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score
from sklearn.preprocessing import OneHotEncoder

data = pd.read_csv("C://Users//VENKAT//Downloads//survey.csv")

# Assuming 'treatment' is the target variable
X =
data.drop('treatment',
axis=1) y =
data['treatment']

data = data.drop(columns=['Timestamp', 'comments'])

categorical_columns = ['Gender', 'Country', 'state', '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']

data = pd.get_dummies(data,

columns=categorical_columns)

X =

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

 # Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

Gaussian Naive Bayes
classifier gnb =
GaussianNB()
gnb.fit(X_train, y_train)
y_pred =
gnb.predict(X_test)

 # Evaluate the Gaussian Naive Bayes classifier
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, pos_label='Yes')
precision = precision_score(y_test, y_pred, pos_label='Yes',
labels=['No','Yes']) f1 = f1_score(y_test, y_pred, pos_label='Yes',
labels=['No', 'Yes'])

print("Gaussian Naive Bayes
Metrics:") print(f"
Accuracy:
{accuracy}") print(f" Recall:
{recall}")
print(f" Precision:
{precision}")print(f" F1
Score: {f1}")

Gaussian Naive Bayes Metrics:
Accuracy:
0.47883597883597884
Recall: 0.0
Precision: 0.0
F1 Score: 0.0

K Neighbors Classifier
 # Import necessary libraries
import pandas **as** pd
from sklearn.model_selection **import** train_test_split
from sklearn.neighbors **import** KNeighborsClassifier
from sklearn.metrics **import** accuracy_score, recall_score,

```
precision_score, f1_score  data =

pd.read_csv("C://Users//VENKAT//Downloads//survey.csv")

# Assuming 'treatment' is the target variable
X =
data.drop('treatment',
axis=1) y =
data['treatment']

data = data.drop(columns=['Timestamp', 'comments'])categorical_columns =
['Gender', 'Country', 'state', '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']

data = pd.get_dummies(data,

columns=categorical_columns)

X =data.drop('treatment', axis=1)

y = data['treatment']# Split the

dataset into training and testing

sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# KNN classifier
knn = KNeighborsClassifier(n_neighbors=5) # You can adjust the number of neighbors (k) as
needed
knn.fit(X_train,
y_train)y_pred =
knn.predict(X_test)

# Evaluate the KNN classifier
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, pos_label='Yes')
precision = precision_score(y_test, y_pred, pos_label='Yes',
labels=['No','Yes']) f1 = f1_score(y_test, y_pred, pos_label='Yes',
labels=['No',  'Yes'])
print("
KNeighborsClassifier
Metrics:") print(f"
Accuracy:
{accuracy}") print(f"  Recall:
{recall}")
print(f"  Precision:
{precision}")print(f"   F1
```

score: {f1}")

K NeighborsClassifier

Metrics: Accuracy:
0.6481481481481481
Recall: 0.5736040609137056
Precision: 0.6975308641975309
F1 Score: 0.6295264623955432

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score
import pandas as pd


    data = pd.read_csv("C://Users//VENKAT//Downloads//survey.csv")

# Assuming 'treatment' is the target variable
X =
data.drop('treatment',
axis=1) y =
data['treatment']

data = data.drop(columns=['Timestamp', 'comments'])

categorical_columns = ['Gender', 'Country', 'state', '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']

data = pd.get_dummies(data,

columns=categorical_columns)

X =data.drop('treatment',

axis=1)
y = data['treatment']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
y_pred_rf = rf_classifier.predict(X_test)

# Evaluate the Random Forest classifier
accuracy = accuracy_score(y_test,y_pred_rf)
```

```
recall = recall_score(y_test, y_pred_rf, pos_label='Yes')
precision = precision_score(y_test, y_pred_rf, pos_label='Yes',
labels=['No','Yes']) f1 = f1_score(y_test, y_pred_rf, pos_label='Yes',
labels=['No', 'Yes'])
print (" Random Forest
Metrics:")
print (f"  Accuracy:
{accuracy}")
print (f" Recall:

{recall}")print (f"

Precision:

{precision}")
print(f"  F1 Score: {f1}")
    Random Forest Metrics: Accuracy:0.80158730158730 16
    Recall: 0.8527918781725888
  Precision: 0.7850467289719626
  F1 Score: 0.8175182481751824
```

logistic regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score
import pandas as pd

data = pd.read_csv("C://Users//VENKAT//Downloads//survey.csv")

# Assuming 'treatment' is the target variable
X =
data.drop('treatment',
axis=1) y =
data['treatment']

data = data.drop(columns=['Timestamp', 'comments'])

categorical_columns = ['Gender', 'Country', 'state', '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']

data = pd.get_dummies(data,

columns=categorical_columns)

X =
```

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

 # Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

 # Logistic Regression model
logreg_model = LogisticRegression(random_state=42)
logreg_model.fit(X_train, y_train)
y_pred_logreg = logreg_model.predict(X_test)

 # Evaluate the Logistic Regression
accuracy = accuracy_score(y_test,y_pred_logreg)
recall = recall_score(y_test, y_pred_logreg, pos_label='Yes')
precision = precision_score(y_test, y_pred_logreg, pos_label='Yes',
labels=['No','Yes']) f1 = f1_score(y_test,y_pred_logreg, pos_label='Yes',
labels=['No', 'Yes'])

print (" Logistic
RegressionMetrics:")
print (f" Accuracy:
{accuracy}")print (f"
Recall: {recall}")
print (f" Precision: {precision}")

print (f" F1 Score:
{f1}") Logistic
Regression Metrics:
Accuracy:
0.8174603174603174
Recall: 0.868020304568528
Precision: 0.7990654205607477
F1 Score: 0.832116788321168

Decision Tree

from sklearn.tree **import** DecisionTreeClassifier
from sklearn.model_selection **import** train_test_split
from sklearn.metrics **import** accuracy_score, recall_score, precision_score, f1_score
import pandas **as** pd

data = pd.read_csv("C://Users//VENKAT//Downloads//survey.csv")

 # Assuming 'treatment' is the target variable
X =
data.drop('treatment',
axis=1) y =
data['treatment']

data = data.drop(columns=['Timestamp', 'comments'])

```
categorical_columns = ['Gender', 'Country', 'state', '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']

data = pd.get_dummies(data,

columns=categorical_columns) X

=data.drop('treatment', axis=1)
y = data['treatment']
    'Yes']) f1 = f1_score(y_test,y_pred_dt, pos_label='Yes', labels=['No', 'Yes'])

print (" Logistic
RegressionMetrics:")
print (f" Accuracy:
{accuracy}")print (f"
Recall: {recall}")
print (f" Precision:
{precision}")print (f" F1
Score: {f1}")
```

```
 Logistic Regression Metrics:
  Accuracy:
  0.7724867724867724
  Recall: 0.7817258883248731
  Precision: 0.7817258883248731
  F1 Score: 0.7817258883248731
```

Gradient boost classifier

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score
import pandas as pd

data = pd.read_csv("C://Users//VENKAT//Downloads//survey.csv")

# Assuming 'treatment' is the target variable
X =
data.drop('treatment',
axis=1) y =
data['treatment']
```

```
data = data.drop(columns=['Timestamp', 'comments'])

categorical_columns = ['Gender', 'Country', 'state', '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']

data = pd.get_dummies(data,

columns=categorical_columns) X

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

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Gradient Boosting model
gb_model = GradientBoostingClassifier(random_state=42)
gb_model.fit(X_train, y_train)
y_pred_gb = gb_model.predict(X_test)
# Evaluate the Gradient boost classifier
accuracy = accuracy_score(y_test,y_pred_gb)
recall = recall_score(y_test,y_pred_gb, pos_label='Yes')
precision = precision_score(y_test, y_pred_gb, pos_label='Yes',
labels=['No','Yes']) f1 = f1_score(y_test,y_pred_gb, pos_label='Yes',
labels=['No', 'Yes'])

print (" Gradient Boost Classifier
Metrics:")
print (f" Accuracy:
{accuracy}")print (f"
Recall: {recall}")
print (f" Precision:
{precision}")
print (f" F1 Score: {f1}")

 Gradient Boost
  ClassifierMetrics:

 Accuracy:

 0.8227513227513228
  Recall: 0.8984771573604061
  Precision: 0.7901785714285714
  F1 Score: 0.8408551068883611
```

ALL CLASSIFIERS

```
import pandas as pd
from sklearn.model_selection import train_test_split
```

from sklearn.ensemble **import** GradientBoostingClassifier,
RandomForestClassifier **from** sklearn.ensemble **import**

GradientBoostingClassifier, AdaBoostClassifier **from** sklearn.neighbors
import KNeighborsClassifier
from sklearn.naive_bayes **import** GaussianNB
from sklearn.linear_model **import** LogisticRegression
from sklearn.tree **import** DecisionTreeClassifier
from sklearn.discriminant_analysis **import**
LinearDiscriminantAnalysis **as** LDA **from**
sklearn.discriminant_analysis **import**
QuadraticDiscriminantAnalysis **as** QDA **from** sklearn.svm **import**
SVC **from** sklearn.metrics **import** accuracy_score, recall_score,
precision_score,

f1_score data = pd.read_csv("C://Users//VENKAT//Downloads//survey.csv")

 # Assuming 'treatment' is the target variable
X =
data.drop('treatment',
axis=1) y =
data['treatment']

data = data.drop(columns=['Timestamp', 'comments'])
categorical_columns = ['Gender', 'Country', 'state', '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']

data = pd.get_dummies(data,

columns=categorical_columns)

X =data.drop('treatment',

axis=1)
y = data['treatment']

 # Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

 # Define classifiers
classifiers = {
'Gradient
Boosting':
GradientBoostingClassifier(random_state=42),
'Random Forest':
RandomForestClassifier(random_state=42), 'KNN':
KNeighborsClassifier(n_neighbors=5),
'Naive Bayes': GaussianNB(),
'Logistic Regression':
LogisticRegression(random_state=42), 'LDA': LDA(),

```
    'QDA': QDA(),
    'AdaBoost':
    AdaBoostClassifier(random_state=42),
    'Decision Tree':
    DecisionTreeClassifier(random_state=42),
    'SVM': SVC (kernel='linear', C=1.0,
    random_state=42)
}
# Loop over classifiers
for clf_name, clf in classifiers.items():
    # Train the classifier
    clf.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = clf.predict(X_test)

    # Evaluate the Gaussian Naive Bayes classifier
    accuracy = accuracy_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred, pos_label='Yes')
    precision = precision_score(y_test, y_pred, pos_label='Yes',
    labels=['No','Yes']) f1 = f1_score(y_test, y_pred, pos_label='Yes',
    labels=['No', 'Yes'])

    # Print results
    print(f"{clf_name}
    Metrics:")print (f"
    Accuracy:
    {accuracy}")
    print (f" Recall: {recall}")

            print (f" Precision:

    {precision}")
    print (f" F1 Score:
    {f1}")print ()
```

Gradient Boosting
Metrics:Accuracy:
0.8227513227513228
Recall: 0.8984771573604061
Precision: 0.7901785714285714
F1 Score: 0.8408551068883611

Random Forest
Metrics:Accuracy:
0.8015873015873016
Recall: 0.8527918781725888
Precision: 0.7850467289719626
F1 Score: 0.8175182481751824

KNN Metrics:

Accuracy: 0.6481481481481481
Recall: 0.5736040609137056
Precision: 0.6975308641975309
F1 Score: 0.6295264623955432

Naive Bayes Metrics:
Accuracy: 0.47883597883597884
Recall: 0.0
Precision: 0.0
F1 Score: 0.0

Logistic Regression
Metrics:Accuracy:
0.8174603174603174
Recall: 0.868020304568528
Precision: 0.7990654205607477
F1 Score: 0.832116788321168

LDA Metrics:
Accuracy: 0.8227513227513228
Recall: 0.9035532994923858
Precision: 0.7876106194690266
F1 Score: 0.8416075650118203

QDA Metrics:
Accuracy: 0.46825396825396826
Recall: 0.09137055837563451
Precision: 0.45
F1 Score: 0.15189873417721517

AdaBoost Metrics:
Accuracy: 0.8095238095238095
Recall: 0.8730964467005076
Precision: 0.7853881278538812
F1 Score: 0.8269230769230769

Decision Tree Metrics:
Accuracy: 0.7724867724867724
Recall: 0.7817258883248731
Precision: 0.7817258883248731
F1 Score: 0.7817258883248731

SVM Metrics:
Accuracy: 0.828042328042328
Recall: 0.9187817258883249
Precision: 0.7869565217391304
F1 Score: 0.8477751756440282

Separate showcase of accuracy

```
import pandas as pd
import numpy as np
```

```
data = pd.read_csv("C://Users//VENKAT//Downloads//survey.csv")

# Assuming 'treatment' is the target variable
X =
data.drop('treatment',
axis=1) y =
data['treatment']

data = data.drop(columns=['Timestamp', 'comments'])

categorical_columns = ['Gender', 'Country', 'state', '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']

data = pd.get_dummies(data,

columns=categorical_columns)

X =

data.drop('treatment',
axis=1)y =
data['treatment']
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

from sklearn.ensemble import

RandomForestClassifiermodel=

RandomForestClassifier(random_state=42),

# Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
print(rf_classifier.score(X_train, y_train))
1.0

testing_accuracy =
rf_classifier.score(X_test,y_test)
print("Testing Accuracy:",
testing_accuracy)

Testing Accuracy: 0.8015873015873016

import numpy as np
from sklearn.preprocessing import StandardScaler # Import the scaler

new_employee_data = pd.DataFrame([[28, # Age
```

```
'Female', # Gender 'United States', # Country
'California', # State
'No', # Self-employed 'No', # Family history
'Sometimes', # Work interference '100-500', # No. of employees
'Yes', # Remotework
'Yes', # Tech company
'Yes', # Mental health benefits
'Yes', # Care options 'Yes', # Wellness program 'Yes', # Seekhelp
'Yes', # Anonymity


'Somewhat easy', # Leave
'No', # Mental health consequence'No', # Physical health consequence 'Yes', # Coworkers 'Yes', # Supervisor
'Maybe', # Mental health interview

'No', # Physical health interview
'Yes', # Mental vs. physical 'No', # Observed consequences
'Additional comments go here', # Comments
]],
columns=['Age', 'Gender', 'Country', 'State', 'Self_Employed',
    'Family_History', 'Work_Interfere', 'No_of_Employees',
    'Remote_Work', 'Tech_Company', 'Mental_Health_Benefits',
    'Care_Options', 'Wellness_Program', 'Seek_Help', 'Anonymity',
    'Leave', 'Mental_Health_Consequence',
    'Physical_Health_Consequence', 'Coworkers', 'Supervisor',
    'Mental_Health_Interview', 'Physical_Health_Interview',
    'Mental_vs_Physical', 'Obs_Consequence', 'Comments'])

new_employee_data_encoded = pd.get_dummies(new_employee_data, columns=['Gender','Country', 'State'], drop_first=True)

missing_columns = set(X_train.columns) - set(new_employee_data_encoded.columns)

# Add missing columns to new_employee_data_encoded and set their values to 0
for col in missing_columns:
  new_employee_data_encoded[
```

```
col] =0

# Reorder columns to match the order in the training data
new_employee_data_encoded = new_employee_data_encoded[X_train.columns]

# Fill NaN values with zeros
new_employee_data_encoded = new_employee_data_encoded.fillna(0)

# Make predictions
prediction = gb_model.predict(new_employee_data_encoded)

# Map numerical prediction back to "yes" or "no"
prediction_label = "YES! MEET YOUR MEDICAL EXPERT." if prediction[0] == 1 else
"NO NOT REQUIRED YOUR HEALTH IS TOTALLY FINE"


# Print the prediction
print(f"The employee is predicted to need treatment: {prediction_label}")

The employee is predicted to need treatment: NO NOT REQUIRED YOUR HEALTH
ISTOTALLY FINE

import pickle
pickle.dump(gb_model,
open('pickle.pkl','wb'))
gb_model=pickle.load(open('pickle.pk

l','rb'))

print(accuracy_score(y_test,y_pred_

gb)) 0.8227513227513228


import joblib

joblib.dump(gb_mod

el,'pickle.pkl')

['pickle.pkl']
```

7.4 FLASK DEPLOYMENT

Source code

App.py

```
from flask import Flask, render_template, requestimport
pickle
import pandas as pd
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

```
app = Flask(__name__)

import pickle
# Load the model
gb_model=pickle.load(open('pickle.pkl','rb'))

import joblib
feature_transform = joblib.load('feature_transform.pkl')
label_encoders = joblib.load('label_encoders.pkl')



# Home route
@app.route('/')def
home():
    return render_template('index.html')

# Prediction route
@app.route('/predict', methods=['POST'])def
predict():
    # Get values from the form
    Age = float(request.form['Age'])
    Gender = request.form['Gender']
    Country = request.form['Country']state
    = request.form['state']
    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']

    # Handle encoding for 'self_employed' column
    self_employed_column = 'self_employed'
    self_employed_encoded_value = None
```

```
if self_employed_column in label_encoders:
    self_employed_encoded_value = label_encoders[self_employed_column]

self_employed_encoded = 1 if self_employed == 'Yes' else 0 if self_employed ==
'No' else None

# Handle encoding for 'family_history' column
family_history_column = 'family_history'
family_history_encoded_value = None

if family_history_column in label_encoders:
    family_history_encoded_value = label_encoders[family_history_column]

    # Assuming family_history is a binary column
family_history_encoded = 1 if family_history == 'Yes' else 0 if family_history ==
'No' else None

# Handle encoding for 'work_interfere' column
work_interfere_column = 'work_interfere'
work_interfere_encoded_value = None

if work_interfere_column in label_encoders:
    work_interfere_encoded_value = label_encoders[work_interfere_column]

    # Assuming 'work_interfere' is an ordinal variable
work_interfere_mapping = {'Never': 0, 'Rarely': 1, 'Sometimes': 2, 'Often': 3}
work_interfere_encoded = work_interfere_mapping.get(work_interfere, None)

# Handle encoding for 'no_employees' column
no_employees_column = 'no_employees'
no_employees_encoded_value = label_encoders.get(no_employees_column, None)

if no_employees_encoded_value is not None:try:
no_employees_encoded =
    no_employees_encoded_value.transform([no_e
    mployees])[0]
    except ValueError:
    # Handle the case where an unseen label is encountered
        print(f"Unseen label '{no_employees}' in column '{no_employees_column}'.
Using a default value.")
        no_employees_encoded = None

else:
    print(f"Label encoder not found for column '{no_employees_column}'.")
    no_employees_encoded = None


# Handle encoding for 'remote_work' column
remote_work_column = 'remote_work'
remote_work_encoded_value = None
```

```
    if remote_work_column in label_encoders:
        remote_work_encoded_value = label_encoders[remote_work_column]

        # Assuming remote_work is a binary column
    remote_work_encoded = 1 if remote_work == 'Yes' else 0 if remote_work == 'No'
else None

    # Handle encoding for 'tech_company' column
    tech_company_column = 'tech_company'
    tech_company_encoded_value = None

    if tech_company_column in label_encoders:
        tech_company_encoded_value = label_encoders[tech_company_column]

    # Assuming tech_company is a binary column
    tech_company_encoded = 1 if tech_company == 'Yes' else 0 if tech_company ==
'No' else None


    categorical_columns = ['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']

    encoded_values = {}

    for column in categorical_columns:
        column_encoded_value = label_encoders.get(column, None)
        if column_encoded_value is not None:
        # Assuming binary encoding for these columns
            encoded_values[column] = 1 if request.form[column] == 'Yes' else 0 if
request.form[column] == 'No' else None
        else:
            print(f"Label encoder not found for column '{column}'.")


        # Create a DataFrame with the entered values
    employee_data =pd.DataFrame({
        'Age':Age,
        'Gender': label_encoders['Gender'].transform([Gender])[0] if 'Gender' in
label_encoders else None,
        'Country': label_encoders['Country'].transform([Country])[0] if 'Country' in
label_encoders else None,
        'state': state,
        'self_employed': self_employed,
        'family_history': family_history,
        'work_interfere': work_interfere,
        'no_employees': no_employees,
```

```
        'remote_work': remote_work,
        'tech_company': tech_company,
        'benefits': encoded_values['benefits'],
        'care_options': encoded_values['care_options'],
        'wellness_program': encoded_values['wellness_program'],
        'seek_help': encoded_values['seek_help'],
        'anonymity': encoded_values['anonymity'],
        'leave': encoded_values['leave'],
        'mental_health_consequence': encoded_values['mental_health_consequence'],
        'phys_health_consequence': encoded_values['phys_health_consequence'],
        'coworkers': encoded_values['coworkers'],
        'supervisor': encoded_values['supervisor'],
        'mental_health_interview': encoded_values['mental_health_interview'],
        'phys_health_interview': encoded_values['phys_health_interview'],
        'mental_vs_physical': encoded_values['mental_vs_physical'],
        'obs_consequence': encoded_values['obs_consequence']
    }, index=[0])


    categorical_columns= ['Age', 'Gender', 'Country','state', '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']
                        # Convert categorical variables using label encodersfor column in
                        categorical_columns:
        column_encoded_value = label_encoders.get(column, None)
        if column_encoded_value is not None and column in employee_data.columns:
            try:
                not_null_indices = employee_data[column].notnull()
                employee_data.loc[not_null_indices, column] =
column_encoded_value.transform(     employee_data.loc[n
                ot_null_indices, column]
                )
            except ValueError as e:
                print(f"Error transforming column '{column}': {e}")#
            Handle the error (e.g., set to None or a default value)
                employee_data[column] = None
        else:
        # Handle the case where the column is not present in the input data or encoder is
None
            employee_data[column] = None

    import numpy as np
    from sklearn.preprocessing import StandardScaler # Import the scaler

    data = pd.read_csv("data/data/survey.csv") #
```

```
Assuming 'treatment' is the target variable
X_train = data.drop(['treatment', 'comments','Timestamp'], axis=1)
y_train = data['treatment']

data = data.drop(columns=['Timestamp', 'comments'])
categorical_columns = ['Gender', 'Country', 'state', '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']

X_train = pd.get_dummies(X_train, columns=categorical_columns)#

Split the dataset into training and testing sets
 categorical_columns_prediction = ['Gender', 'Country', 'state', '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_employee_data = pd.DataFrame({'Age':Age,
                          'Gender': label_encoders['Gender'].transform([Gender])[0] if
'Gender' in label_encoders else None,
                          'Country': label_encoders['Country'].transform([Country])[0]
if 'Country' in label_encoders else None,
                          'state': state,
                          'self_employed': self_employed,
                          'family_history': family_history,
                          'work_interfere': work_interfere,
                          'no_employees': no_employees,
                          'remote_work': remote_work,
                          'tech_company': tech_company,
                          'benefits': encoded_values['benefits'],
                          'care_options':   encoded_values['care_options'],
                          'wellness_program': encoded_values['wellness_program'],
                          'seek_help': encoded_values['seek_help'],
                          'anonymity': encoded_values['anonymity'],
                          'leave': encoded_values['leave'],
                          'mental_health_consequence':
encoded_values['mental_health_consequence'],
                          'phys_health_consequence':
encoded_values['phys_health_consequence'],
                          'coworkers':      encoded_values['coworkers'],
```

```
                                'supervisor':        encoded_values['supervisor'],
                                'mental_health_interview':
encoded_values['mental_health_interview'],
                                'phys_health_interview':
encoded_values['phys_health_interview'],
                                'mental_vs_physical': encoded_values['mental_vs_physical'],
                                'obs_consequence': encoded_values['obs_consequence']},
index=[0])

  employee_data_encoded = pd.get_dummies(employee_data,
columns=categorical_columns_prediction)

  missing_columns = set(X_train.columns) - set(employee_data_encoded.columns)

  for col in missing_columns:
    employee_data_encoded[col] = 0

  employee_data_encoded = employee_data_encoded[X_train.columns]


  employee_data_encoded = employee_data_encoded.fillna(0)
  prediction = gb_model.predict(employee_data_encoded)

    # Map numerical prediction back to "yes" or "no"
  prediction_label = "Take care of your Health" if prediction[0] == 1 else "You need
to consult your Doctor"

  return render_template('result.html',prediction_label=prediction_label)

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

TEMPLATES

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Machine Learning Prediction</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='style.css') }}">

</head>
<body>
<div class=”container”></div>
    <h1>Machine Learning Prediction</h1>
```

```html
<form action="/predict" method="post">
  <label for="Age">Age:</label>
  <input type="number" name="Age" required><br>

  <label for="Gender">Gender:</label>
  <select name="Gender" required>
    <option value="Male">Male</option>
    <option value="Female">Female</option>
    <option value="Trans">Trans</option>
    <!-- Add more gender options as needed -->
  </select><br>
  <label for="Country">Country:</label>
  <select name="Country" required>
    <option value="United States">United States</option>
    <option value="United Kingdom">United Kingdom</option>
    <option value="Canada">Canada</option>
    <option value="Germany">Germany</option>
    <option value="Ireland">Ireland</option>
    <option value="Netherlands">Netherlands</option>
    <option value="Australia">Australia</option>
    <option value="France">France</option>
    <option value="India">India</option>
    <option value="New Zealand">New Zealand</option>
    <option value="Poland">Poland</option>
    <option value="Switzerland">Switzerland</option>
    <option value="Sweden">Sweden</option>
    <option value="Italy">Italy</option>
    <option value="South Africa">South Africa</option>
    <option value="Brazil ">Brazil </option>Belgium
    <option value="Israel">Israel</option>
    <option value="Singapore">Singapore </option>
    <option value="Bulgaria">Bulgaria </option>
    <option value="Austria"> Austria</option>
    <option value="Finland">Finland</option>
    <option value="Mexico">Mexico</option>
    <option value="Russia">Russia</option>
    <option value="Denmark">Denmark</option>
    <option value="Greece">Greece</option>
    <option value="Colombia">Colombia</option>
    <option value="Croatia">Croatia</option>
    <option value="Portugal">Portugal</option>
    <option value="Moldova">Moldova</option>
    <option value="Georgia">Georgia</option>
    <option value="Bahamas, The">Bahamas, The</option>
    <option value="China">China</option>
    <option value="Thailand">Thailand</option>
    <option value="Czech Republic">Czech Republic</option>
    <option value="Norway">Norway</option>
    <option value="Romania">Romania</option>
    <option value="Nigeria">Nigeria</option>
```

```
    <option value="Japan">Japan</option>
    <option value="Hungary">Hungary</option>
    <option value="Bosnia and Herzegovina">
    Bosnia andHerzegovina</option>
    <option value="Uruguay">Uruguay</option>
    <option value="Spain">Spain</option>
    <option value="Zimbabwe">Zimbabwe</option>
    <option value="Latvia">Latvia</option>
    <option value="Costa Rica">Costa Rica</option>
    <option value="Slovenia">Slovenia</option>
    <option value="Philippines">Philippines</option>
     <!-- Add more country options as needed -->
</select><br>

<label for="state">State:</label>
<select name="state" required>
    <option value="CA">CA</option>
    <option value="WA">WA</option>
    <option value="NY">NY</option>
    <option value="TN">TN</option>
    <option value="TX">TX</option>
    <option value="OH">OH</option>
    <option value="IL">IL</option>
    <option value="OR">OR</option>
    <option value="PA">PA</option>
    <option value="IN">IN</option>
    <option value="MI">MI</option>
    <option value="MN">MN</option>
    <option value="MA">MA</option>
    <option value="FL">FL</option>
    <option value="NC">NC</option>
    <option value="VA">VA</option>
    <option value="WI">WI</option>
    <option value="GA">GA</option>
    <option value="MO">MO</option>
    <option value="UT">UT</option>
    <option value="CO">CO</option>
    <option value="MD">MD</option>
    <option value="AL">AL</option>
    <option value="AZ">AZ</option>
    <option value="OK">OK</option>
    <option value="NJ">NJ</option>
    <option value="KY">KY</option>
    <option value="SC">SC</option>
    <option value="IA">IA</option>
    <option value="CT">CT</option>
    <option value="DC">DC</option>
    <option value="NV">NV</option>
    <option value="VT">VT</option>
    <option value="SD">SD</option>
```

```html
    <option value="KS">KS</option>
    <option value="NH">NH</option>
    <option value="WY">WY</option>
    <option value="NM">NM</option>
    <option value="NE">NE</option>
    <option value="WV">WV</option>
    <option value="ID">ID</option>
    <option value="MS">MS</option>
    <option value="RI">RI</option>
    <option value="LA">LA</option>
    <option value="ME">ME</option>
    <!-- Add more state options as needed -->
</select><br>
<label for="self_employed">Self Employed:</label>
<select name="self_employed" required>
    <option value="Yes">Yes</option>
    <option value="No">No</option>
</select><br>

<label for="family_history">Family History of Mental Illness:</label>
<select name="family_history" required>
    <option value="Yes">Yes</option>
    <option value="No">No</option>
</select><br>
<label for="work_interfere">Work Interference:</label>
<select name="work_interfere" required>
    <option value="Never">Never</option>
    <option value="Rarely">Rarely</option>
    <option value="Sometimes">Sometimes</option>
    <option value="Often">Often</option>
</select><br>

<label for="no_employees">no_employees</label>
<input type="number" name="no_employees" required><br>


</select><br>

<label for="remote_work">Remote Work:</label>
<select name="remote_work" required>
    <option value="No">No</option>
    <option value="Yes">Yes</option>
</select><br>

<label for="tech_company">Tech Company:</label>
<select name="tech_company" required>
    <option value="No">No</option>
    <option value="Yes">Yes</option>
</select><br><label for="benefits">Mental Health Benefits:</label>
<select name="benefits" required>
```

```
    <option value="No">No</option>
    <option value="Yes">Yes</option>
</select><br>

<label for="care_options">Mental Health Care Options:</label>
<select name="care_options" required>
    <option value="No">No</option>
    <option value="Not sure">Not sure</option>
    <option value="Yes">Yes</option>
</select><br>

<label for="wellness_program">Wellness Program:</label>
<select name="wellness_program" required>
    <option value="No">No</option>
    <option value="Yes">Yes</option>
</select><br>

<label for="seek_help">Seek Help:</label>
<select name="seek_help" required>
    <option value="0">No</option>
    <option value="1">Yes</option>
</select><br>
<label for="anonymity">Anonymity:</label>
<select name="anonymity" required>
    <option value="0">No</option>
    <option value="1">Yes</option>
</select><br>

<label for="leave">Leave:</label>
<select name="leave" required>
    <option value="0">No</option>
    <option value="1">Yes</option>
</select><br>

<label for="mental_health_consequence">Mental Health
Consequence:</label>
<select name="mental_health_consequence" required>
    <option value="0">No</option>
    <option value="1">Yes</option>
</select><br>

<label for="phys_health_consequence">Physical Health
Consequence:</label>
<select name="phys_health_consequence" required>
    <option value="0">No</option>
    <option value="1">Yes</option>
</select><br>

<label for="coworkers">Coworkers:</label>
<select name="coworkers" required>
```

```
            <option value="0">No</option>
            <option value="1">Yes</option>
        </select><br>

        <label for="supervisor">Supervisor:</label>
        <select name="supervisor" required>
            <option value="0">No</option>
            <option value="1">Yes</option>
        </select><br>

        <label  for="mental_health_interview">Mental   Health  Interview:</label>
        <select   name="mental_health_interview"   required>
            <option value="0">No</option>
            <option value="1">Yes</option>
        </select><br>

        <label   for="phys_health_interview">Physical   Health   Interview:</label>
        <select   name="phys_health_interview"   required>
            <option value="0">No</option>
            <option value="1">Yes</option>
        </select><br>

        <label  for="mental_vs_physical">Mental   vs  Physical:</label>
        <select   name="mental_vs_physical"   required>
            <option value="0">No</option>
            <option value="1">Yes</option>
        </select><br>

        <label   for="obs_consequence">Observation   Consequence:</label>
        <select name="obs_consequence" required>
            <option value="0">No</option>
            <option value="1">Yes</option>
        </select><br>

        <!-- Add similar dropdowns for other attributes -->

        <input type="submit" value="Predict">
    </form>
</div>
 </body>
 </html>
```

Indexstyle.css

```
body {
font-family: Arial, sans-serif;
background-color:aquamarine;
background-image: url('surendra.jpg');
background-size: cover;
background-repeat: no-repeat;
margin: 0;
padding:  0;
```

```
}
.container {
   max-width: 600px; /* Adjust the max-width as needed */
   margin: 20px auto; /* Center the container horizontally */
   padding: 20px;
   background-color: #fff;
   border-radius: 5px;
   box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
   overflow-y: auto; /* Add vertical scrollbar when needed */
}

h1 {
   text-align: center;color:
   white;
}

form {
   max-width: 400px;
   margin: 0 auto;
   background-color: #fff;
   padding: 20px;
   border-radius: 5px;
   box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
label {
   display: block; margin-
   bottom: 5px;font-
   weight: bold;
}

input[type="number"],
select {
   width:    100%;
   padding: 8px;
   margin-bottom:    10px;
   border: 1px solid #ccc;
   border-radius: 4px;
   box-sizing: border-box;
}
input[type="submit"] {
   width: 100%;
   background-color: #4caf50;color:
   white;
   padding: 10px 0;
   border: none; border-
   radius: 4px;cursor:
   pointer;
}

input[type="submit"]:hover {
```

```
   background-color:  #45a049;
}
```

RESULT.HTML

```
 <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Machine Learning Prediction Result</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='resultstyle.css') }}">

</head>
<body>
    <h1>Machine Learning Prediction Result</h1>
    <p>Prediction: {{ prediction_label }}</p>
</body>
</html>
```

RESULTSTYLE.CSS

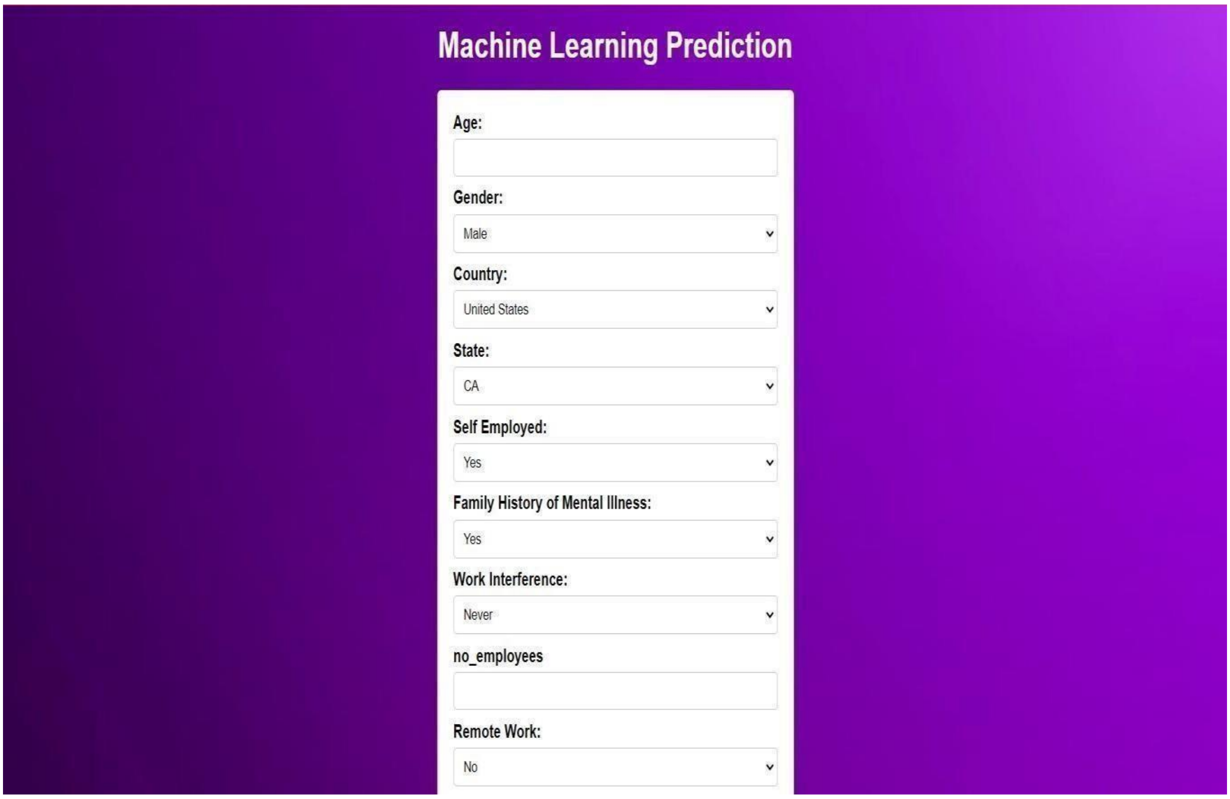
```
body {
   font-family:  Arial,  sans-serif;
   background-color:aquamarine;
   background-image: url('brain.jpg');
   background-size: cover;
   background-repeat: no-repeat;
   margin: 0;
   padding:  0;

}

h1 {
   text-align: center;
   margin-top: 100px;
}
p {
   text-align: center;
   font-size: 24px;
}
```

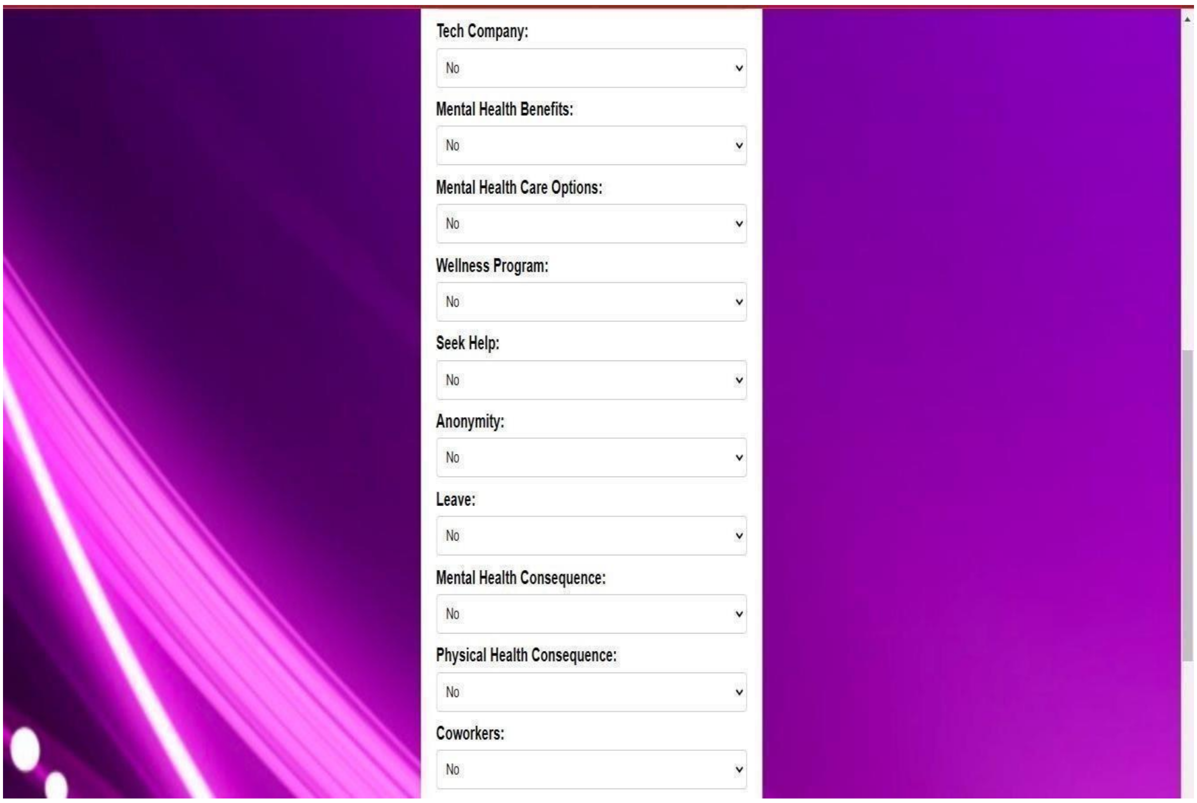
REQUIREMENTS:

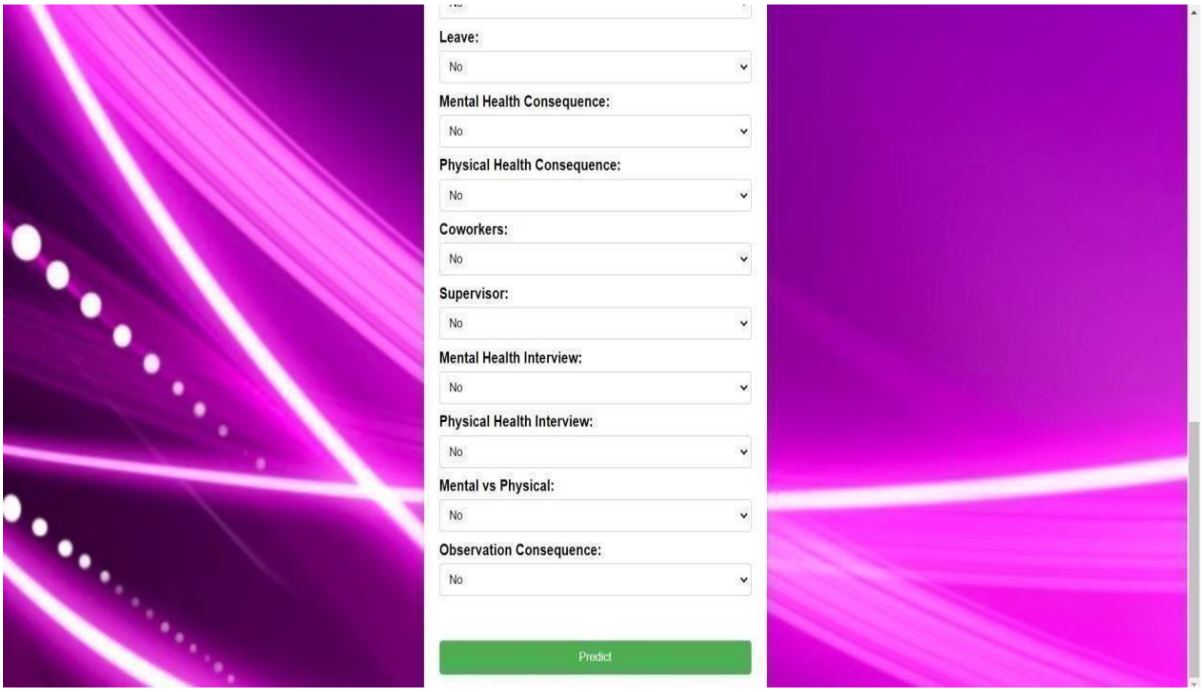
archspec
blinker==1.7.0
boltons
Brotli
certifi==2024.2.2cffi
charset-normalizerclick==8.1.7
Conda
conda-libmamba-solver
conda-package-handling
package-handling
conda_package_streaming
cryptography
Distro Flask==3.0.2
gunicorn==21.2.0
idna
itsdangerous==2.1.2
Jinja2==3.1.3
joblib==1.3.2
jsonpatch
jsonpointer==2.1
Libmambapy
MarkupSafe==2.1.5
menuinst
numpy==1.26.4
packaging
pandas==2.2.1
pickle-mixin==1.0.2platformdirs
pluggy Pycosat
pycparser
PySocks
python-dateutil==2.9.0.post0
pytz==2024.1
Requests
ruamel.yaml
scikit-learn==1.4.1.post1
scipy==1.12.0
setuptools==69.1.1 six==1.16.0
threadpoolctl==3.3.0
tqdm
Truststore
tzdata==2024.1urllib3

SCREENS

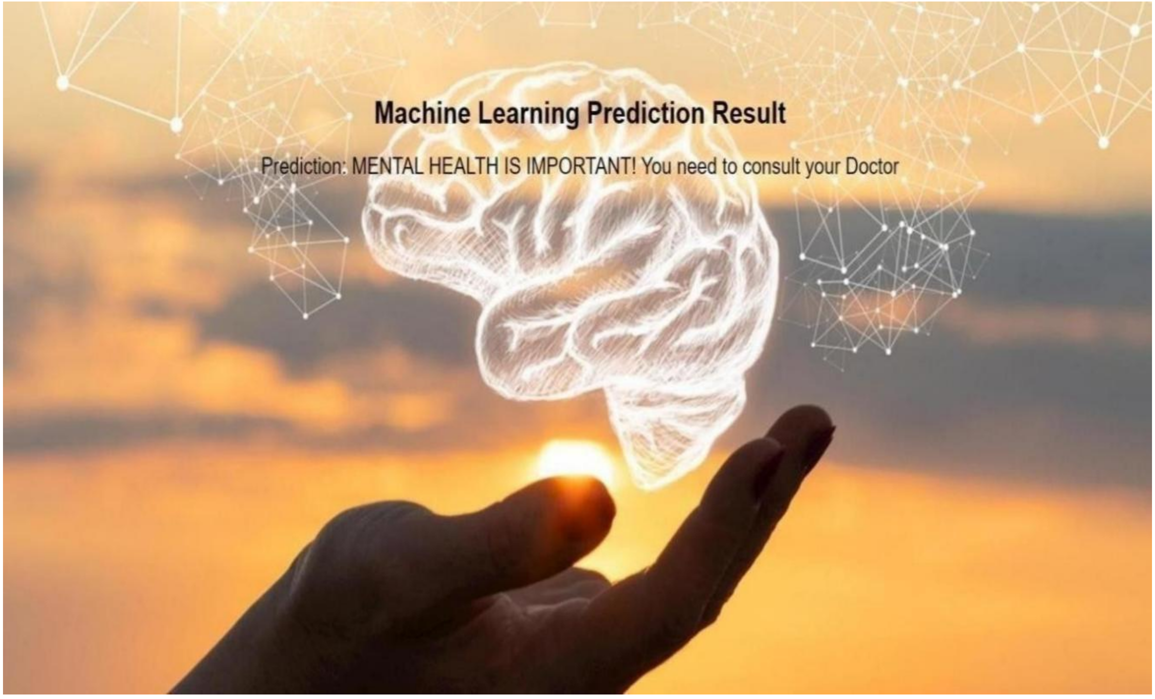
8. SCREENS


Machine Learning Prediction
Age:
Gender:
Male
Country:
United States
State:
CA
Self Employed:
Yes
Family History of Mental Illness:
Yes
Work Interference:
Never
no_employees
Remote Work:
No


Tech Company:
No
Mental Health Benefits:
No
Mental Health Care Options:
No
Wellness Program:
No
Seek Help:
No
Anonymity:
No
Leave:
No
Mental Health Consequence:
No
Physical Health Consequence:
No
Coworkers:
No



RESULT SCREEN



SYSTEM TESTING

9. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type address a specific testing requirement.

TYPES OF TESTS

Unit Testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event-driven and ease more concerned with the basic outcome of screens or fields. These demonstrate that although the components were individually satisfied, as shown by successfully unit testing, the combination of components are correct and consistent. Integration Testing is specifically aimed at exposing the pr oblems that arise from the combination of components. Their are two types of integration testing

TOP-DOWN TESTING:

Modules are integrated by moving downwards through the control hierarchy beginning with main program. The subordinate modules are incorporated into structure in either a breadth first manner or depth first manner.This process is done in five steps:

- Main control module is used as a test driver and steps are substituted or all modules directly tomain program.
- Depending on the integration approach selected subordinate is replaced at a

time with actual modules. Tests are conducted.

- On completion of each set of tests another stub is replaced with the real module Regression testing may be conducted to ensure that new errors have not been introduced. This process continues from step 2 until the entire program structure is reached. In top-down integration strategy decision-making occurs at upper levels in thehierarchy and is encountered first. If major control problems do exist early recognitionis essential.
- If depth first integration is selected a complete function of the software may be implemented and demonstrated.
- Some problems occur when processing at low levels in the hierarchy is required to adequately test upper-level steps to replace low-level modules at the beginning of the top-down testing. So no data flows upward in the program structure.

BOTTOM-UP TESTING:

Begins construction and testing with atomic modules. As modules are integrated from the bottom up, processing requirements for modules subordinate to a given level is always available and need for stubs is eliminated. The following steps implement this strategy.

Low-level modules are combined in to clusters that perform a specific software sub-function. A driver is written to coordinate test case input and output, Cluster is tested. Drivers are removed and moving upward in the program structure combines clusters. Integration moves upward, and there is need for separate test driver's lesions.

If the top levels of program structures are integrated top-down, the number of drivers can be reduced substantially and integration of clusters is greatly simplified.

Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid input : identified classes of valid input must

be accepted

Invalid input : identified classes of invalid must be

rejected.

Functions : identified must be exercised.

Output : identified classes of application outputs must

be exercised. Systems/Procedures : interfacing systems or procedures

must be invoked.

Organization and preparation of functional tests are focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifying Business process flows, data fields, predefined processes, and successive processes must be co nsidered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test:

System testing ensures that the entire integrated software system meets requirements. It test s a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing:

White Box Testing is attesting in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot see into it. The test provides inputs and responds to outputs without considering how the software works.

Regression Testing:

Each time a new module is added as a part of integration as the software changes. Regression testing is an actually that helps to ensure changes that do not introduce unintended behavior as additional errors.

Regression testing may be conducted manually by executing a subset of all test cases or using automated capture playback tools to enable the software engineer to capture the test case and results for subsequent playback and compression. The regression suite contains different classes of test cases. A representative s a m p l e t o tests that will exercise all software functions. Additional tests that focus on software functions that are likely to be affected by the change

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software life cycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Testing Strategies:

Testing is a set of activities that can be planned in advanced and conducted systematically. A strategy for software testing must accommodation low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements as specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

The main objective of software is testing to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed each test step is accomplished through a series of systematic test technique that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened.

Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase. This is an activity to be performed in parallel with the software effort and one that consists of its own phases of analysis, design, implementation, execution and maintenance.

Test objectives:

- All field entries work properly.
- Pages must be activated from the identified link.
- The entry screen, messages, and responses must not be delayed.

Features to be tested:

Verify that the entries are of the correct format No duplicate entries should be allowed

All links should take the user to the correct page Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of integration task is to check that component or software Applications at the company level interact without error.

Test results:

All the test cases mentioned above pass successfully. No defects were encountered.

Acceptance testing:

User acceptance testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test results:

All the test cases mentioned above passed successfully. No defects are encountered.

Implementation:

Implementation is the process of converting a new or revised system design into operation alone. There are three types of implementation:

- Implementation of a computer system to replace a manual system. The problems encountered are converting files, training users, and verifying printouts for integrity.
- Implementation of a new computer system to replace an existing one. This is usually a difficult conversion. If not properly planned there can be many problems.
- Implementation of a modified application to replace an existing one using the same computer. This type of conversion is relatively easy to handle, provided there are no major changes inthe files.

Implementation in Generic tool project is done in all modules. In the first module User level identification is done. In this module every user is identified whether they are genuine one not to access the database and also generates the session for the user. Illegal use of any form is strictly avoided.

In the Table creation module, the tables are created with user specified fields and user can create many tables at a time. They may specify conditions, constraints and calculations in creation of tables. The Generic code maintain the user requirements throughout the project.

In Updating module user can update or delete or insert the new record into the database. This is very important module in Generic code project. User has to specify the filed value in the form then the Generic tool automatically gives whole filed values for that particular record.

In Reporting module user can get the reports from the database in 2Dimentional or 3Dimensional view. User has to select the table and specify the condition then the report will be generated for the user.

CONCLUSIONS

10. CONCLUSION

A worker's performance on the job, communication with co-workers, physical abilities, and daily functioning can all be significantly impacted by poor mental health. Loss of productivity is the result of all of this. Therefore, it is crucial that employees receive the care and assistance they require for their mental health issue. By openly discussing mental health illnesses and offering resources and benefits for mental health at work, employers must treat mental health with the same respect as physical health.

According to the study of the Mental Health in Tech Survey data set, more needs to be done to educate the tech community about mental health issues and to provide assistance for employees who are dealing with mental illnesses. The OSMI provides training to employers and can assist in finding the appropriate services for supporting staff who are suffering with mental health concerns.

From the Project, it is clear that people are well aware of the negative effects of mental health in their workplaces. However, there seems to be a gender bias when it comes to how mental health is viewed with ladies being more reluctant to bring up their issues due to fear of how their issues will be viewed.

As much as more gents seem to have taken advantage of the facilities given to them, they still feel like more could be done evidenced by the fact that most gents feel like their mental issues aren't taken as seriously as they would like.

My recommendations would be as follows:

a) Workplaces should have more gender-focused facilities so that an extra layer could be added when it comes to gender sensitivity.

b) Employers should conduct regular campaigns that ensure anonymity is a priority to make it easier for employees to come out.

c) Employers should make it a priority to ensure that they, on a regular basis, make it known to employees that mental health care is available and is taken seriously just as any other health issue would.

REFERENCES

11. REFERENCES

1. International Journal of Science and Research Archive, 2023, 10(01), 221–233. ISSN ApprovedInternational Journal of Science Fast Publication (ijsra.net)
2. Open Sourcing Mental Illness. OSMI Mental Health in Tech Survey.https://osmihelp.org/research. Accessed October 21, 2020
3. Centers for Disease Control and Prevention. Metal Health in the Workplace. https://www.cdc.gov/workplacehealthpromotion/tools-resources/workplace- health/mentalhealth/index.html. Accessed October 22, 2020
4. World Health Organization. Mental Health and Work. https://www.who.int/mental_health/prevention/guidelines_mental_health_wor k/en/. Accessed October 22, 2020.
5. Social Empowered. Ultimate Guide to Mental Health in the Workplace. https://socalempowered.com/ultimateguide-to-mental-health-in-the- workplace/
6. Man doing human brain puzzle. Character vector created by vectorjuice. https://www.freepik.com/vectors/character Mitravinda, K.M., Nair, D.S. & Srinivasa, G. MentalHealth in Tech: Analysis of Workplace Risk Factors and Impact of COVID -19. SN COMPUT.SCI. 4, 197 (2023). https://doi.org/10.1007/s42979-022-01613-z
7. Warner V, Weissman MM, Mufson L, Wickramaratne PJ. Grandparents, parents, and grandchildren at high risk for depression: a three-generation study. J Am Acad Child Adolesc Psychiatry. 1999;38(3):289–96.
8. Dellabella H. Family history of psychiatric illness increases risk in offspring. HTTP://www.psychiatryadvisor.com/home/bipolar-disorder- advisor/family-history-of- psychiatric-illness-increases-risk-in-offspring/. Accessed 18 Oct 2022; 2019.
9. Phillips L. Challenging the inevitability of inherited mental illness. https://ct.counseling.org/2019/08/challenging-the-inevitability-of- inherited-mental-illness/#.Accessed 18 Oct 2022; 2022
10. Dellabella H. Family history of psychiatric illness increases risk in offspring. https://www.psychiatryadvisor.com/home/bipolar-disorder- advisor/family-history-of- psychiatric-illness-increases-risk-in-offspring/. Accessed 18 Oct 2022; 2019.
11. Agenagnew L, et al. The lifetime prevalence and factors associated with relapse among mentally ill patients at Jimma university medical center, Ethiopia: cross sectional study. J Psych Rehab Mental Health. 2020; 7(3):211–20.

12. Plana-Ripoll O, Pedersen CB, Holtz Y, Benros ME, Dalsgaard S, De Jonge P, Fan CC, Degenhardt L, Ganna A, Greve AN, et al. Exploring comorbidity within mental disorders among a Danish national population. JAMA Psychiatry. 2019; 76(3):259–70.
13. Nealon M. Employers show new concern for workers' mental health. https://www.un.org/en/un-chronicle/pandemic-accelerant-how-covid- 19-advanced-our-mental- health priorities. Accessed 18 Oct 2022; 2021.
14. Greenwood K, Anas J. It's a new era for mental health at work. https://hbr.org/2021/10/its-a- new-era-for-mental-health-at-work. Accessed 18 Oct 2022; 2021.
15. Graveling RA, Crawford JO, Cowie H, Amati C, Vohra S. A review of workplace interventions that promote mental wellbeing in the workplace. Edinburgh: Institute of Occupational Medicine. 2008.