**PUBLIC HEALTH AWARENESS**

**TEAM MEMBER**

REGISTER NUMBER: 412621243006

NAME: AMAL VIJAY G

# **Phase-4 Document Submission**

Mental health is a critical public health issue, with millions of people worldwide affected by mental health conditions. However, many people do not seek treatment for their mental health problems, due to several barriers, such as stigma, lack of awareness, and financial constraints.

Machine learning and data analytics can address these challenges and improve public health awareness of mental health.

**Dataset Link:** <https://www.kaggle.com/datasets/osmi/mental-health-in-tech-survey>



**Exploratory Data Analysis (EDA)**

EDA is the first step in understanding your data. It helps you identify patterns, relationships, and outliers in your dataset. Here are some things you can do:

**Data cleaning:**

Start by cleaning your data to handle missing values, duplicates, and inconsistencies.

**Summary statistics:**

Calculate basic statistics such as mean, median, standard deviation, and quartiles for numerical variables. For categorical variables, count the frequency of each category.

**Data visualization:**

Create visualizations such as histograms, box plots, and bar charts to understand the distribution of your data. Scatter plots can help you find relationships between variables. Libraries like Pandas, Matplotlib, and Seaborn can be very helpful.

**Correlation analysis:**

Determine the correlation between variables to identify potential relationships. You can use tools like correlation matrices or heatmaps.

**Python program:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import os

os.listdir("/kaggle/input/")

df = pd.read\_csv("/kaggle/input/mental-health-in-tech-survey/survey.csv")

# To see the shape of the dataframe:

df.shape

Data Cleaning: Missing Values

df.info()

df.isna().sum()

self\_employed\_percent = (df["self\_employed"].isnull().sum()/len(df["self\_employed"]))\*100

work\_interfere\_percent = (df["work\_interfere"].isnull().sum()/len(df["work\_interfere"]))\*100

print(f"The percentage of missing values in self\_employed column is {round(self\_employed\_percent, 2)}%")

print(f"The percentage of missing values in work\_interfere column is {round(work\_interfere\_percent, 2)}%")

df["self\_employed"] = df["self\_employed"].fillna(df["self\_employed"].mode()[0])

df["work\_interfere"] = df["work\_interfere"].fillna(df["work\_interfere"].mode()[0])

df.head()

df.drop(["state", "comments"], axis=1, inplace=True)

df.isna().sum()

### **Univariate Analyses**

df.columns

plt.figure(figsize=(17,5))

ax = sns.countplot(x='Country', data=df)

ax.set\_xticklabels(

ax.get\_xticklabels(),

rotation=45,

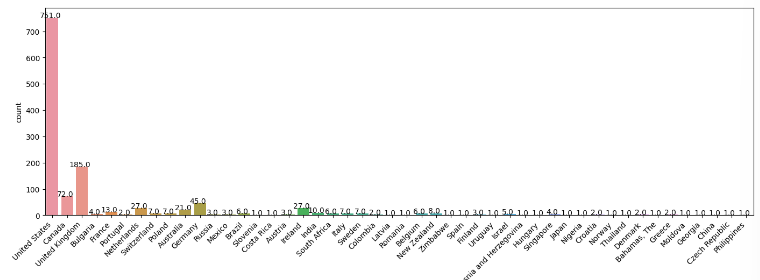
horizontalalignment='right'

)

None

for p in ax.patches:

ax.annotate(p.get\_height(), (p.get\_x()+0.25, p.get\_height()+0.01), ha='center')



### **Attribute 2: Age**

min\_age = df["Age"].min()

max\_age = df["Age"].max()

mean\_age = df["Age"].mean()

median\_age = df["Age"].median()

print(f"Min: {min\_age}, \nMax: {max\_age}, \nMean: {mean\_age}, \nMedian: {median\_age}")

df["Age"].unique()

negative\_age = (df["Age"]<0).sum()

over\_age = (df["Age"]>80).sum()

print(f"Number of negative age entries: {negative\_age}\nNumber of overage: {over\_age}")

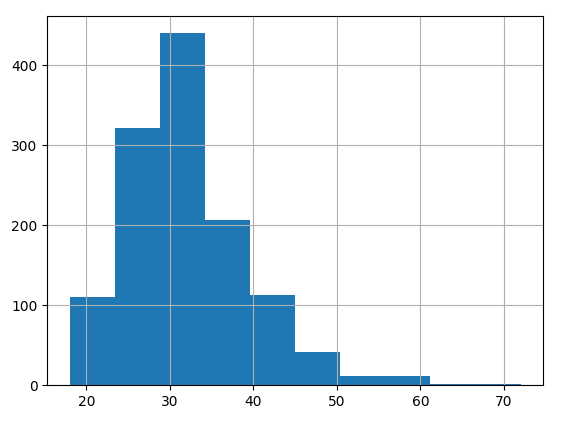
df.loc[df.Age<0, ["Age"]] = df["Age"].median()

df.loc[df.Age>80, ["Age"]] = df["Age"].median()

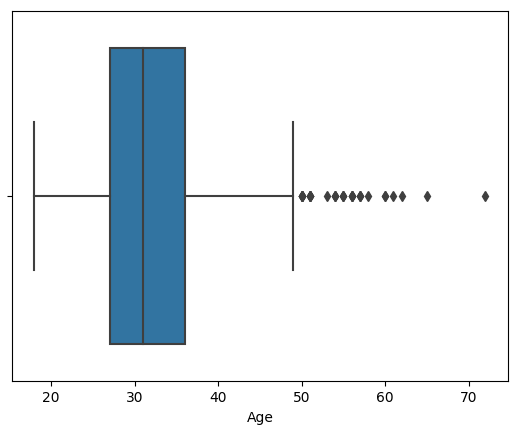
df["Age"].unique()

df.loc[df.Age<18, ["Age"]] = df["Age"].median()

df["Age"].hist()



sns.boxplot(x=df["Age"])



### **New Results for Mean. Also, calculate for S.Deviation & Variance**

import statistics

variance\_age = df["Age"].var()

standard\_dev\_age = statistics.stdev(df["Age"])

print(f"Mean: {round(mean\_age, 2)}"

f"\nVariance: {round(variance\_age, 2)}"

f"\nStandard Deviation: {round(standard\_dev\_age, 2)}")

### **Five (5) Other Attributes: No. of Employees, Family History, Remote Work, Self-Employed & Tech Company**

plt.figure(figsize=(7,4))

ax = sns.countplot(x='no\_employees', data=df)

ax.set\_xticklabels(ax.get\_xticklabels(),

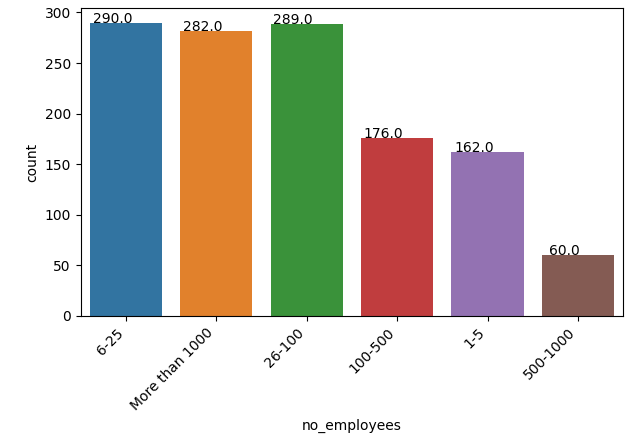
rotation=45,

horizontalalignment='right')

# Then we also display the values for each bar above it;

for p in ax.patches:

ax.annotate(p.get\_height(), (p.get\_x()+0.25, p.get\_height()+0.01), ha='center')



plt.figure(figsize=(5,5))

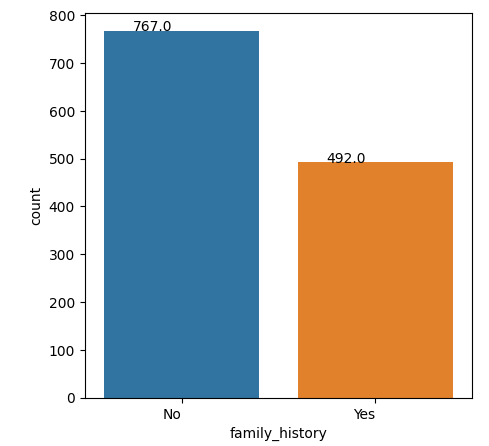
ax = sns.countplot(x='family\_history', data=df)

ax.set\_xticklabels(ax.get\_xticklabels(),

horizontalalignment='right')

for p in ax.patches:

ax.annotate(p.get\_height(), (p.get\_x()+0.25, p.get\_height()+0.01), ha='center')



plt.figure(figsize=(5,5))

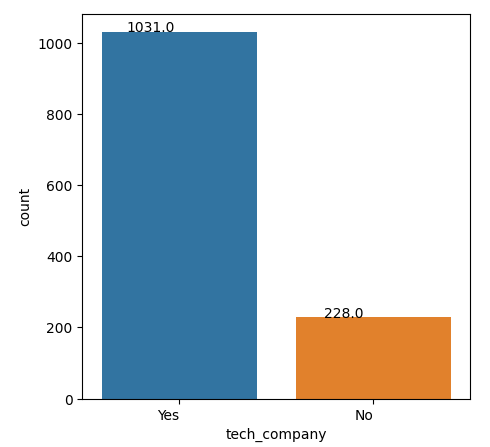
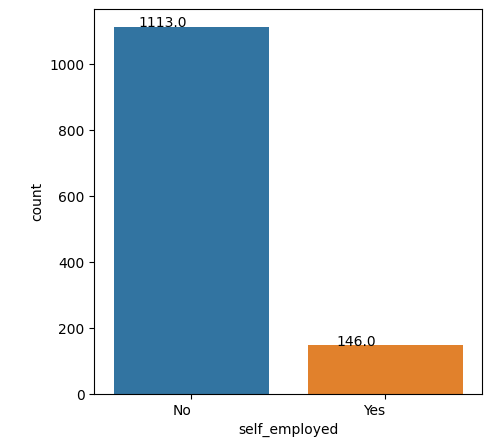
ax = sns.countplot(x='self\_employed', data=df)

ax.set\_xticklabels(ax.get\_xticklabels(),

horizontalalignment='right')

for p in ax.patches:

ax.annotate(p.get\_height(), (p.get\_x()+0.25, p.get\_height()+0.01), ha='center')



plt.figure(figsize=(5,5))

ax = sns.countplot(x='tech\_company', data=df)

ax.set\_xticklabels(ax.get\_xticklabels(),

horizontalalignment='right')

for p in ax.patches:

ax.annotate(p.get\_height(), (p.get\_x()+0.25, p.get\_height()+0.01), ha='center')

### **Data Manipulation: Changing from Categorical to Numerical**

print(f'Family History Unique Entries: {df["family\_history"].unique()}')

df["family\_history\_num"] = df["family\_history"].map({"No": 0, "Yes": 1})

df[["family\_history", "family\_history\_num"]].head()

df1 = pd.DataFrame() # Empty dataframe

df1["Year"] = df["Year"]

df1["self\_employed\_num"] = df["self\_employed\_num"]

df1["treatment\_num"] = df["treatment\_num"]

df1["remote\_work\_num"] = df["remote\_work\_num"]

df1["benefits\_num"] = df["benefits\_num"]

df1["wellness\_programs\_num"] = df["wellness\_programs\_num"]

df1["seek\_help\_num"] = df["seek\_help\_num"]

df1["anonymity\_num"] = df["anonymity\_num"]

df1["mental\_health\_consequence\_num"] = df["mental\_health\_consequence\_num"]

df1["phys\_health\_consequence\_num"] = df["phys\_health\_consequence\_num"]

plt.figure(figsize=(12, 10))

sns.heatmap(df1.corr(), annot=True)

