# DAC\_PHASE4

# Project Title: Public Health Awareness Campaign Analysis

## Introduction

In this, we are building upon our analysis efforts by utilizing IBM Cognos for data visualization and integrating code, potentially in Python, for advanced data analysis. Our primary objective remains the assessment of the public health awareness campaign's effectiveness and impact. We will design interactive dashboards and reports in IBM Cognos to visually represent campaign reach, awareness levels, and impact metrics, offering valuable insights for stakeholders. Furthermore, we will use code to perform in-depth analysis, including calculating engagement rates, conducting demographic analysis, and running statistical tests, enabling us to provide comprehensive findings that can inform decisions and contribute to the betterment of our communities.

## Import necessary libraries

import pandas as pd import numpy as np import seaborn as sns

import matplotlib.pyplot as plt print('Successfully imported')

This dataset contains the following data:

* Timestamp
* Age
* Gender
* Country
* state: If you live in the United States, which state or territory do you live in?
* self\_employed: Are you self-employed?
* family\_history: Do you have a family history of mental illness?
* treatment: Have you sought treatment for a mental health condition?
* work\_interfere: If you have a mental health condition, do you feel that it interferes with your work?
* no\_employees: How many employees does your company or organization have?
* remote\_work: Do you work remotely (outside of an office) at least 50% of the time?
* tech\_company: Is your employer primarily a tech company/organization?
* benefits: Does your employer provide mental health benefits?
* care\_options: Do you know the options for mental health care your employer provides?
* wellness\_program: Has your employer ever discussed mental health as part of an employee wellness program?
* seek\_help: Does your employer provide resources to learn more about mental health issues and how to seek help?
* anonymity: Is your anonymity protected if you choose to take advantage of mental health or substance abuse treatment resources?
* leave: How easy is it for you to take medical leave for a mental health condition?
* mental\_health\_consequence: Do you think that discussing a mental health issue with your employer would have negative consequences?
* phys\_health\_consequence: Do you think that discussing a physical health issue with your employer would have negative consequences?
* coworkers: Would you be willing to discuss a mental health issue with your coworkers?
* supervisor: Would you be willing to discuss a mental health issue with your direct supervisor(s)?
* mental\_health\_interview: Would you bring up a mental health issue with a potential employer in an interview?
* phys\_health\_interview: Would you bring up a physical health issue with a potential employer in an interview?
* mental\_vs\_physical: Do you feel that your employer takes mental health as seriously as physical health?
* obs\_consequence: Have you heard of or observed negative consequences for coworkers with mental health conditions in your workplace?
* comments: Any additional notes or comments

## Read Dataset

data = pd.read\_csv('/kaggle/input/mental-health-in-tech-survey/survey.csv') data.head()

## Preprocessing and Cleaning dataset

if data.isnull().sum().sum() == 0 :

print ('There is no missing data in our dataset')

else:

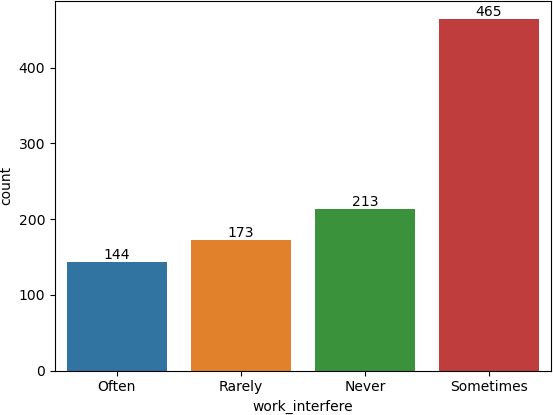
print('There is {} missing data in our dataset '.format(data.isnull().sum().sum()))

There is 1892 missing data in our dataset

frame = pd.concat([data.isnull().sum(), data.nunique(), data.dtypes], axis = 1, sort= False) frame

data['work\_interfere'].unique()

array(['Often', 'Rarely', 'Never', 'Sometimes', nan], dtype=object) ax = sns.countplot(data = data , x = 'work\_interfere');

#Add the value of each parametr on the Plot ax.bar\_label(ax.containers[0]);

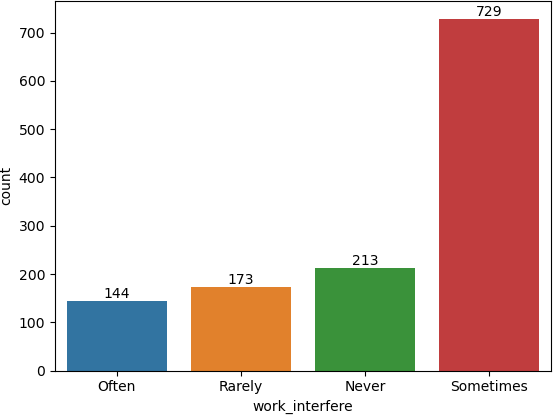
#For filling nan values i used SimpleImputer but you can use fillnan function too from sklearn.impute import SimpleImputer

data = data.drop(columns=['state', 'comments', 'Timestamp', ]) # Fill in missing values in work\_interfere column

data['work\_interfere'] = SimpleImputer(strategy = 'most\_frequent').fit\_transform(data['work\_interfere'].values.reshape(-1,1))

data['self\_employed'] = SimpleImputer(strategy = 'most\_frequent').fit\_transform(data['self\_employed'].values.reshape(-1,1))

data.head()

ax = sns.countplot(data=data, x='work\_interfere'); ax.bar\_label(ax.containers[0]);

#Check unique data in gender columns print(data['Gender'].unique())

print('')

print('-'\*75)

print('')

#Check number of unique data too.

print('number of unique Gender in our dataset is :', data['Gender'].nunique())

## Output

['Female' 'M' 'Male' 'male' 'female' 'm' 'Male-ish' 'maile' 'Trans-female' 'Cis Female' 'F' 'something kinda male?' 'Cis Male' 'Woman' 'f' 'Mal' 'Male (CIS)' 'queer/she/they' 'non-binary' 'Femake' 'woman' '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' 'cis male' 'A little about you' 'Malr' 'p' 'femail' 'Cis Man' 'ostensibly male, unsure what that really means']

number of unique Gender in our dataset is : 49

#Gender data contains dictation problems, nonsense answers, and too unique Genders. #\_So Let's clean it and organize it into Male, Female, and other categories

data['Gender'].replace(['Male ', 'male', 'M', 'm', 'Male', 'Cis Male',

'Man', 'cis male', 'Mail', 'Male-ish', 'Male (CIS)',

'Cis Man', 'msle', 'Malr', 'Mal', 'maile', 'Make',], 'Male', inplace = True)

data['Gender'].replace(['Female ', 'female', 'F', 'f', 'Woman', 'Female',

'femail', 'Cis Female', 'cis-female/femme', 'Femake', 'Female (cis)', 'woman',], 'Female', inplace = True)

data["Gender"].replace(['Female (trans)', 'queer/she/they', 'non-binary',

'fluid', 'queer', 'Androgyne', 'Trans-female', 'male leaning androgynous', 'Agender', 'A little about you', 'Nah', 'All',

'ostensibly male, unsure what that really means', 'Genderqueer', 'Enby', 'p', 'Neuter', 'something kinda male?', 'Guy (-ish) ^\_^', 'Trans woman',], 'Other', inplace = True)

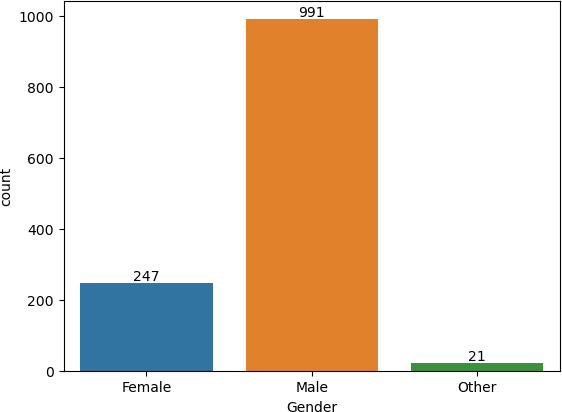
print(data['Gender'].unique())

## Output

['Female' 'Male' 'Other']

#Plot Genders column after cleaning and new categorizing ax = sns.countplot(data=data, x='Gender'); ax.bar\_label(ax.containers[0]);

## Output



#Our data is clean now ? let's see. if data.isnull().sum().sum() == 0: print('There is no missing data')

else:

print('There is {} missing data'.format(data.isnull().sum().sum()))

## Output

There is no missing data

#Let's check duplicated data.

if data.duplicated().sum() == 0: print('There is no duplicated data:')

else:

print('Tehre is {} duplicated data:'.format(data.duplicated().sum())) #If there is duplicated data drop it. data.drop\_duplicates(inplace=True)

print('-'\*50) print(data.duplicated().sum())

## Output

Tehre is 4 duplicated data:

0

#Look unique data in Age column data['Age'].unique()

## Output

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| array([ | 37, | 44, | 32, | 31, | 33, |
|  | 35, | 39, | 42, | 23, | 29, |
|  | 36, | 27, | 46, | 41, | 34, |
|  | 30, | 40, | 38, | 50, | 24, |
|  | 18, | 28, | 26, | 22, | 19, |
|  | 25, | 45, | 21, | -29, | 43, |
|  | 56, | 60, | 54, | 329, | 55, |

99999999999, 48, 20, 57, 58,

47, 62, 51, 65, 49,

-1726, 5, 53, 61, 8,

11, -1, 72])

#We had a lot of nonsense answers in the Age column too

#This filtering will drop entries exceeding 100 years and those indicating negative values. data.drop(data[data['Age']<0].index, inplace = True) data.drop(data[data['Age']>99].index, inplace = True)

print(data['Age'].unique())

## Output

[37 44 32 31 33 35 39 42 23 29 36 27 46 41 34 30 40 38 50 24 18 28 26 22

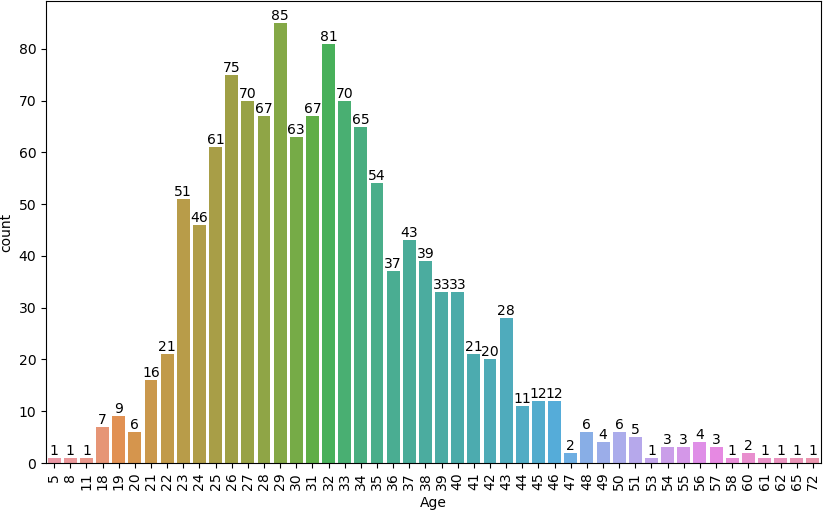
19 25 45 21 43 56 60 54 55 48 20 57 58 47 62 51 65 49 5 53 61 8 11 72]

## Age distribution in this dataset

plt.figure(figsize = (10,6))

age\_range\_plot = sns.countplot(data = data, x = 'Age'); age\_range\_plot.bar\_label(age\_range\_plot.containers[0]); plt.xticks(rotation=90);

## Output



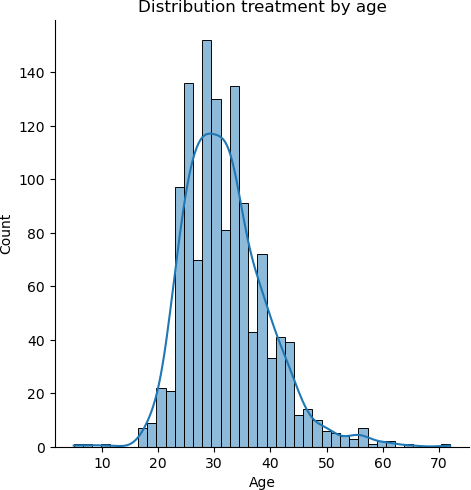
## Age distribution we can see treatment distribution by age

plt.figure(figsize=(10, 6));

sns.displot(data['Age'], kde = 'treatment'); plt.title('Distribution treatment by age');

## Output

<Figure size 1000x600 with 0 Axes>

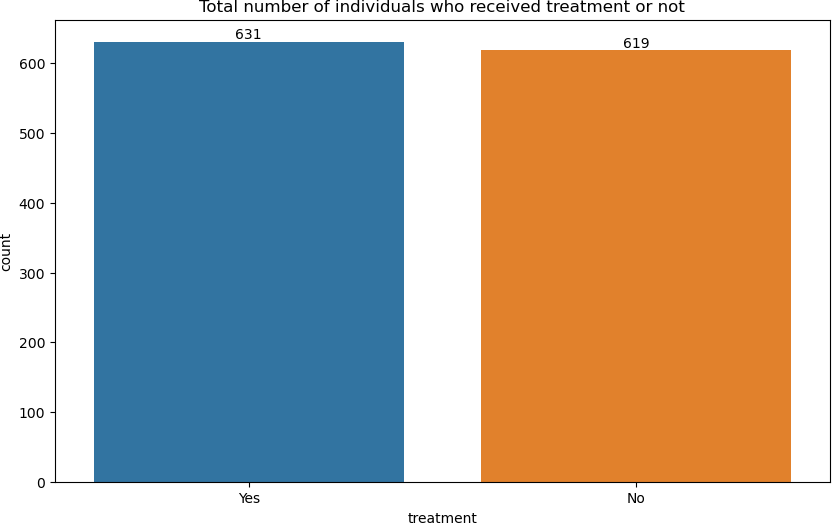


Total number of individuals who received treatment or not

plt.figure(figsize = (10,6));

treat = sns.countplot(data = data, x = 'treatment'); treat.bar\_label(treat.containers[0]);

plt.title('Total number of individuals who received treatment or not');



data.info()

## Output

<class 'pandas.core.frame.DataFrame'> Int64Index: 1250 entries, 0 to 1258 Data columns (total 24 columns):

# Column Non-Null Count Dtype

1. Age 1250 non-null int64
2. Gender 1250 non-null object
3. Country 1250 non-null object
4. self\_employed 1250 non-null object
5. family\_history 1250 non-null object
6. treatment 1250 non-null object
7. work\_interfere 1250 non-null object
8. no\_employees 1250 non-null object
9. remote\_work 1250 non-null object
10. tech\_company 1250 non-null object
11. benefits 1250 non-null object
12. care\_options 1250 non-null object
13. wellness\_program 1250 non-null object
14. seek\_help 1250 non-null object
15. anonymity 1250 non-null object
16. leave 1250 non-null object
17. mental\_health\_consequence 1250 non-null object
18. phys\_health\_consequence 1250 non-null object
19. coworkers 1250 non-null object
20. supervisor 1250 non-null object

...

1. mental\_vs\_physical 1250 non-null object
2. obs\_consequence 1250 non-null object dtypes: int64(1), object(23)

#Use LabelEncoder to change the Dtypes to 'int' from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

#Make the dataset include all the columns we need to change their dtypes

columns\_to\_encode = ['Gender', 'Country', 'self\_employed','family\_history', 'treatment', 'work\_interfere','no\_employees',

'remote\_work', 'tech\_company','benefits','care\_options', 'wellness\_program', 'seek\_help', 'anonymity', 'leave', 'mental\_health\_consequence',

'phys\_health\_consequence',

'coworkers', 'supervisor', 'mental\_health\_interview','phys\_health\_interview', 'mental\_vs\_physical', 'obs\_consequence']

#Write a Loop for fitting LabelEncoder on columns\_to\_encode for columns in columns\_to\_encode:

data[columns] = le.fit\_transform(data[columns]) data.info()

## Output

<class 'pandas.core.frame.DataFrame'> Int64Index: 1250 entries, 0 to 1258 Data columns (total 24 columns):

# Column Non-Null Count Dtype

1. Age 1250 non-null int64
2. Gender 1250 non-null int64
3. Country 1250 non-null int64
4. self\_employed 1250 non-null int64
5. family\_history 1250 non-null int64
6. treatment 1250 non-null int64
7. work\_interfere 1250 non-null int64
8. no\_employees 1250 non-null int64
9. remote\_work 1250 non-null int64
10. tech\_company 1250 non-null int64
11. benefits 1250 non-null int64
12. care\_options 1250 non-null int64
13. wellness\_program 1250 non-null int64
14. seek\_help 1250 non-null int64
15. anonymity 1250 non-null int64
16. leave 1250 non-null int64
17. mental\_health\_consequence 1250 non-null int64
18. phys\_health\_consequence 1250 non-null int64
19. coworkers 1250 non-null int64
20. supervisor 1250 non-null int64

...

1. mental\_vs\_physical 1250 non-null int64
2. obs\_consequence 1250 non-null int64 dtypes: int64(24)

data.describe()

from sklearn.preprocessing import MaxAbsScaler, StandardScaler data['Age'] = MaxAbsScaler().fit\_transform(data[['Age']]) data['Country'] = StandardScaler().fit\_transform(data[['Country']])

data['work\_interfere'] = StandardScaler().fit\_transform(data[['work\_interfere']]) data['no\_employees'] = StandardScaler().fit\_transform(data[['no\_employees']]) data['leave'] = StandardScaler().fit\_transform(data[['leave']])

data.describe()

## Split the data to train and test

from sklearn.model\_selection import train\_test\_split #I wanna work on 'treatment' column.

X = data.drop(columns = ['treatment']) y = data['treatment']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25) print(X\_train.shape, y\_train.shape)

print('-'\*30)

print(X\_test.shape, y\_test.shape) print('\_'\*30)

## Output

(937, 23) (937,)

(313, 23) (313,)

from sklearn.pipeline import Pipeline from sklearn.decomposition import PCA

from sklearn.ensemble import RandomForestClassifier as RFC from sklearn.neighbors import KNeighborsClassifier as KNN from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA

from sklearn.tree import DecisionTreeClassifier as DT

## Support Vector Machine

svclassifier = SVC(kernel = 'linear')

*# fit the model*

svc\_model=svclassifier.fit(X\_train, y\_train)

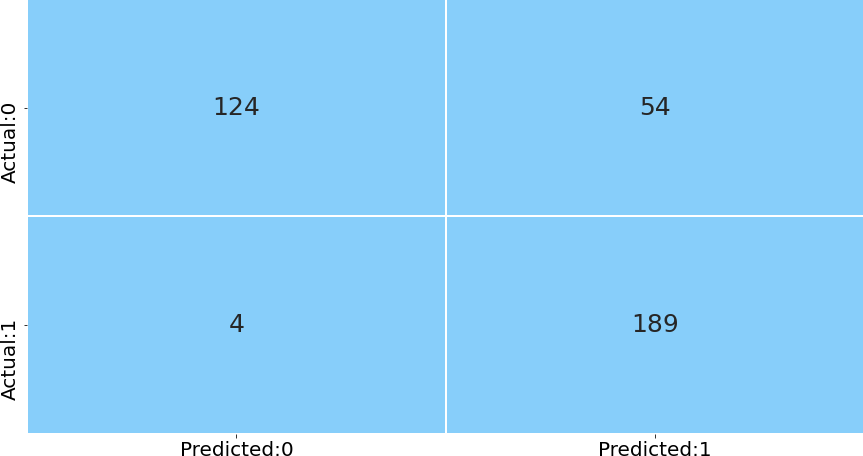
*# predict the values*

y\_pred = svclassifier.predict(X\_test)

linkcode

*# call the function to plot the confusion matrix*

plot\_confusion\_matrix(svc\_model)



test\_report = get\_test\_report(svc\_model)

# print the performace measures print(test\_report)

precision recall f1-score support

|  |  |  |  |
| --- | --- | --- | --- |
| 0 0.97 | 0.70 | 0.81 | 178 |
| 1 0.78 | 0.98 | 0.87 | 193 |

## Output

accuracy 0.84 371

macro avg 0.87 0.84 0.84 371

weighted avg 0.87 0.84 0.84 371

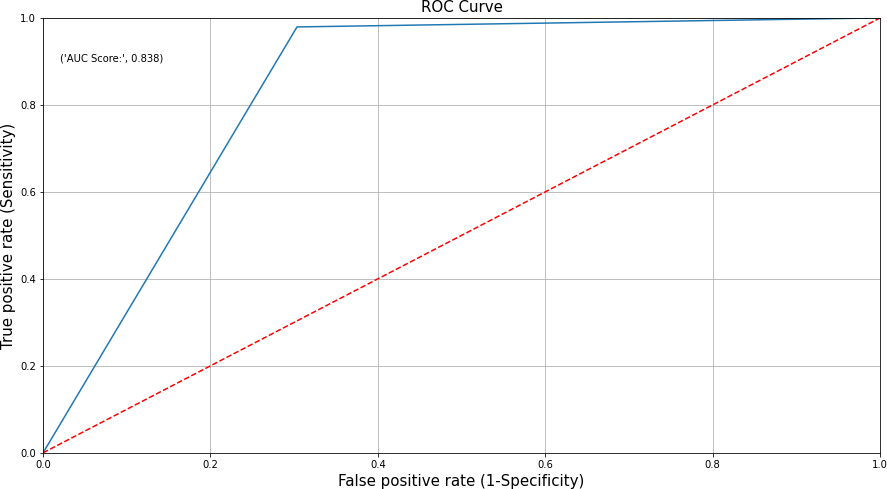
## Input

kappa\_value = kappa\_score(svc\_model)

# print the kappa value print(kappa\_value) 0.6833632537743901

plot\_roc(svc\_model)

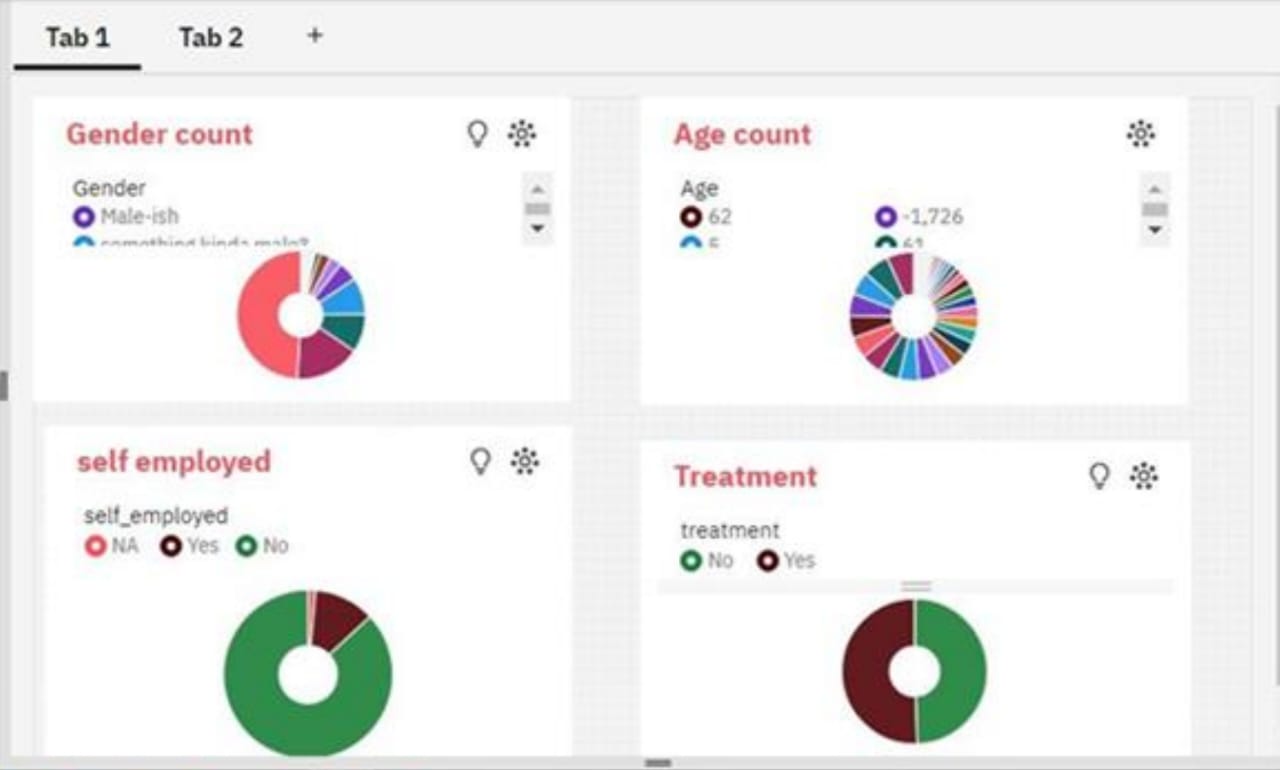
## Output

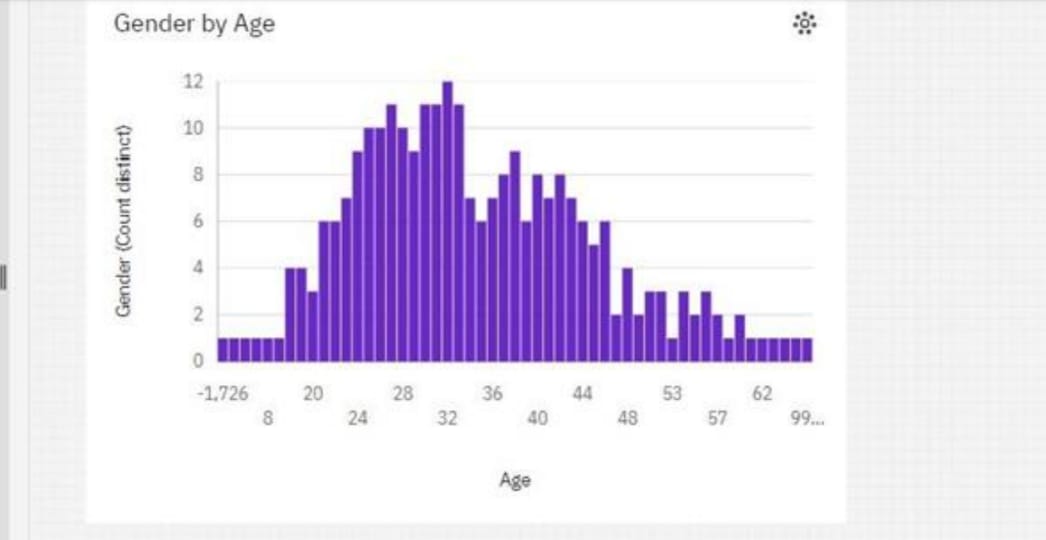


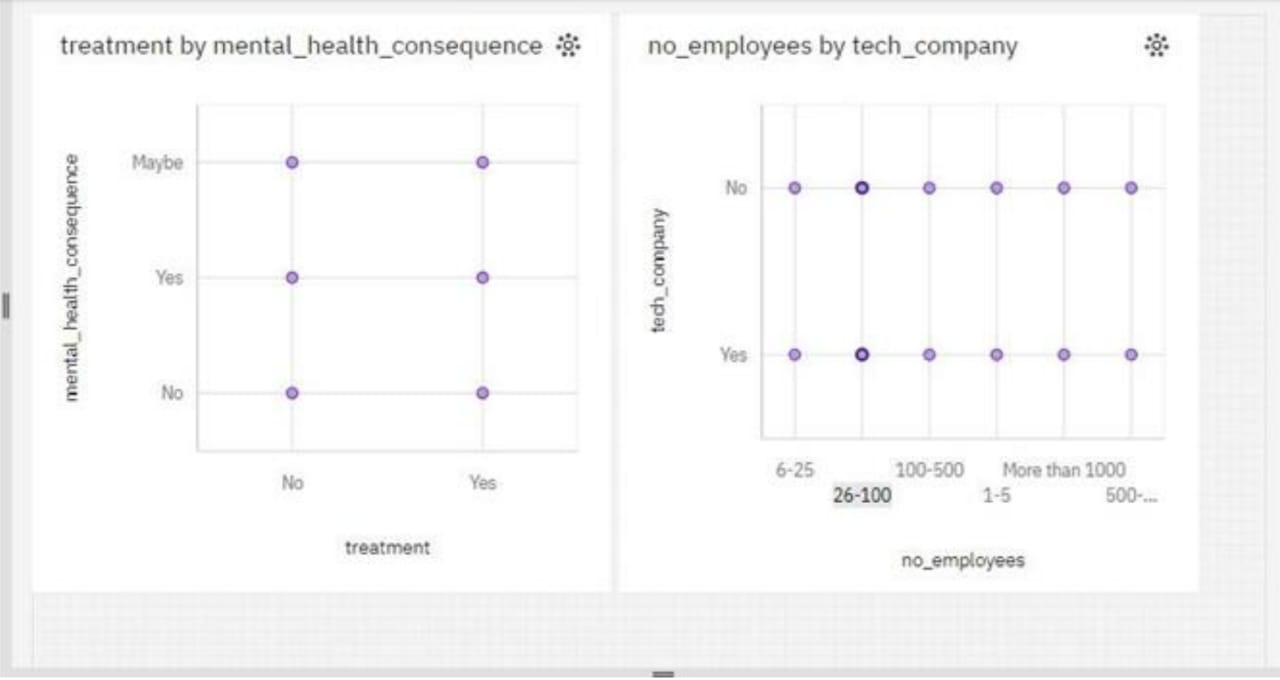
**Accuracy 0.84**

## Dashboard and Report in IBM COGNOS

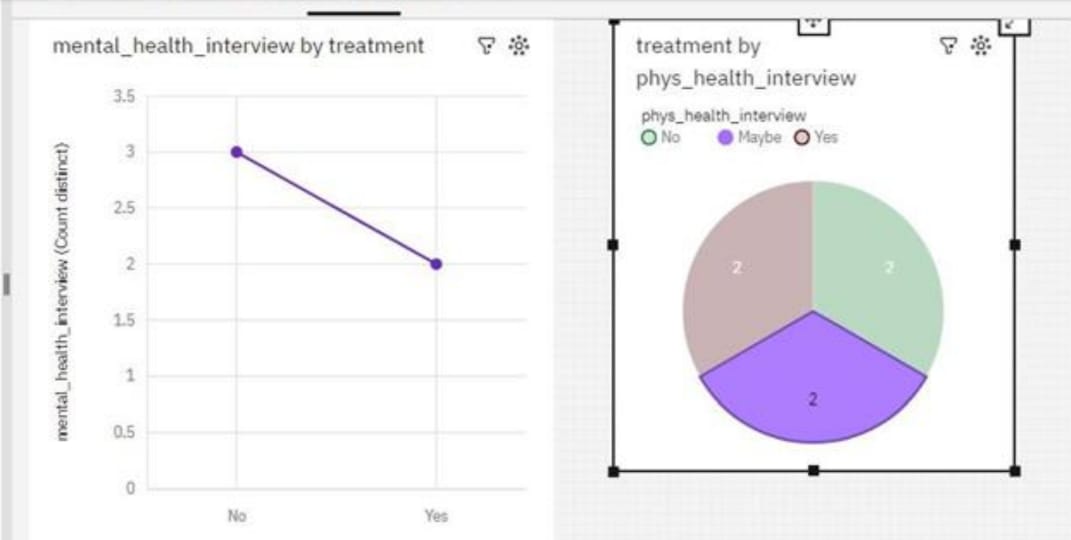
[](../Documents)

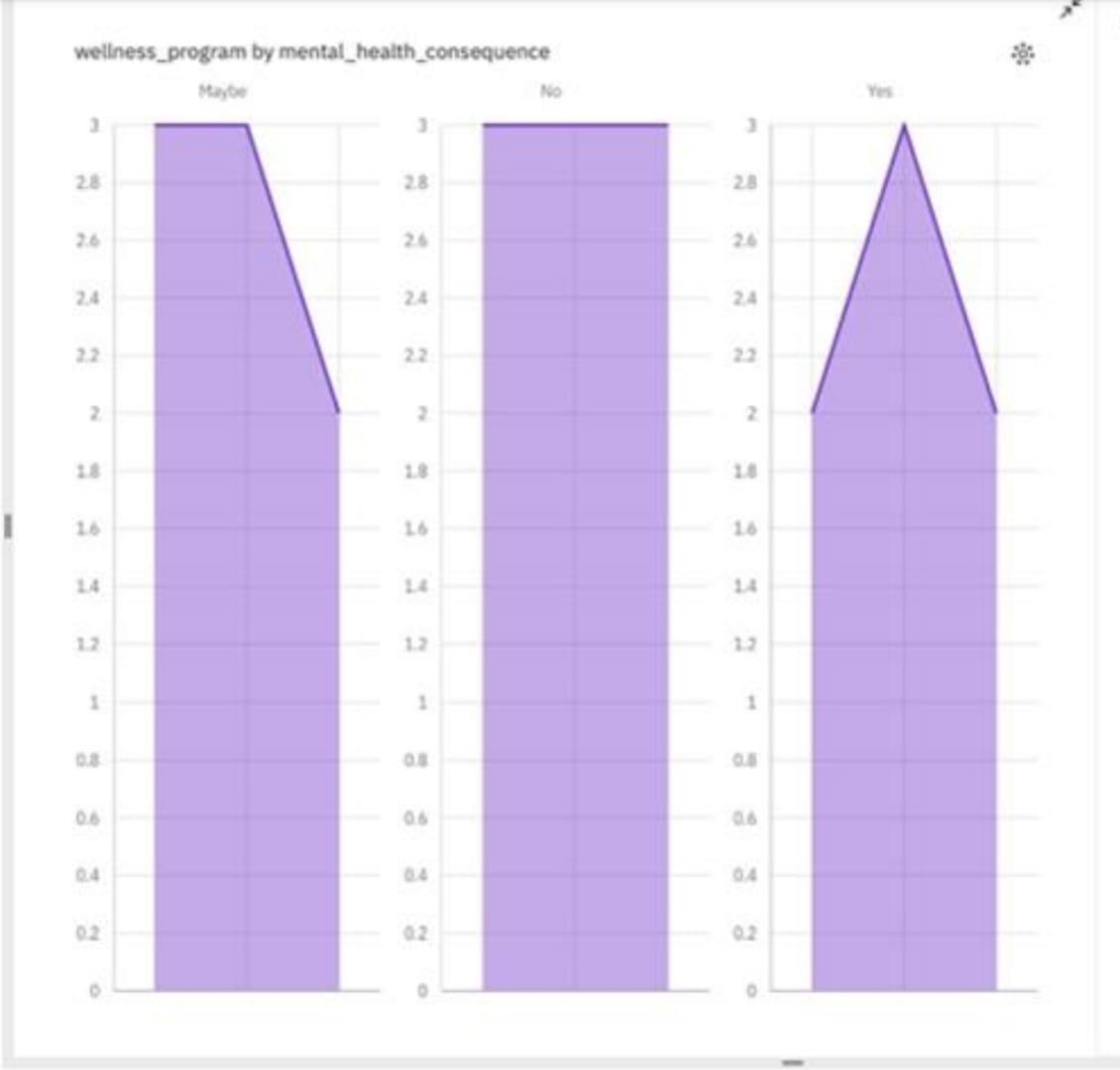




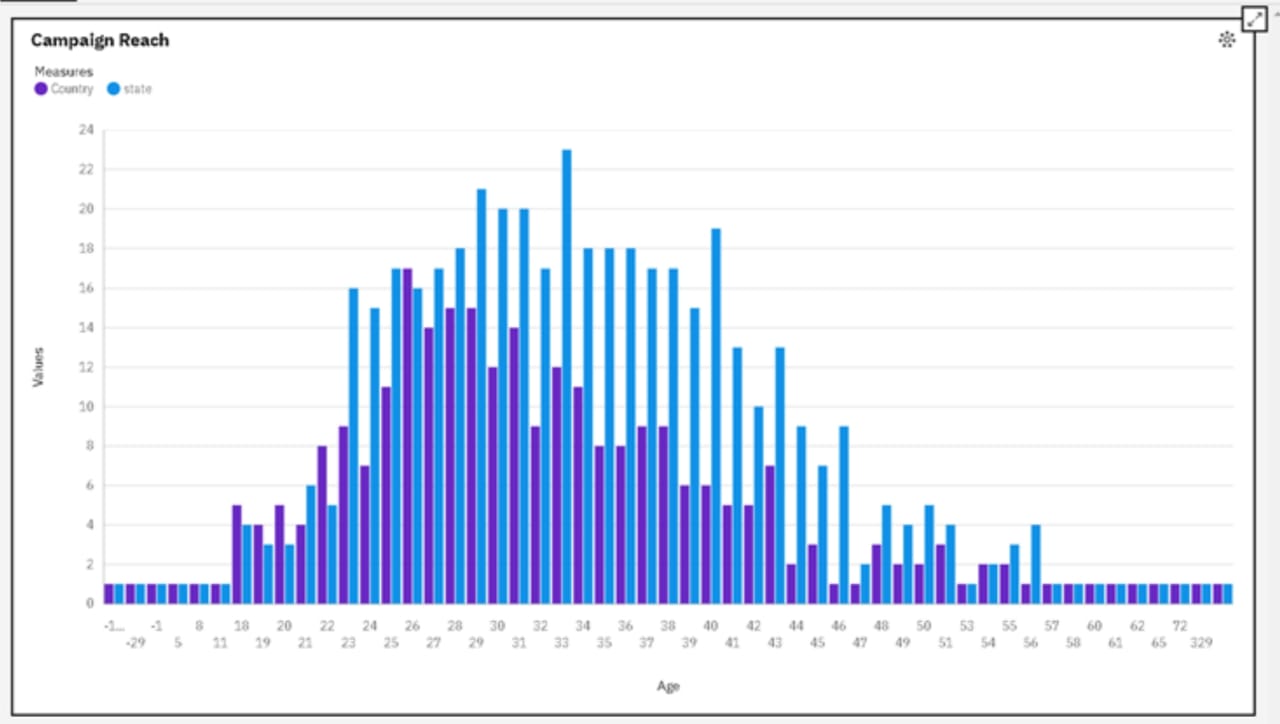




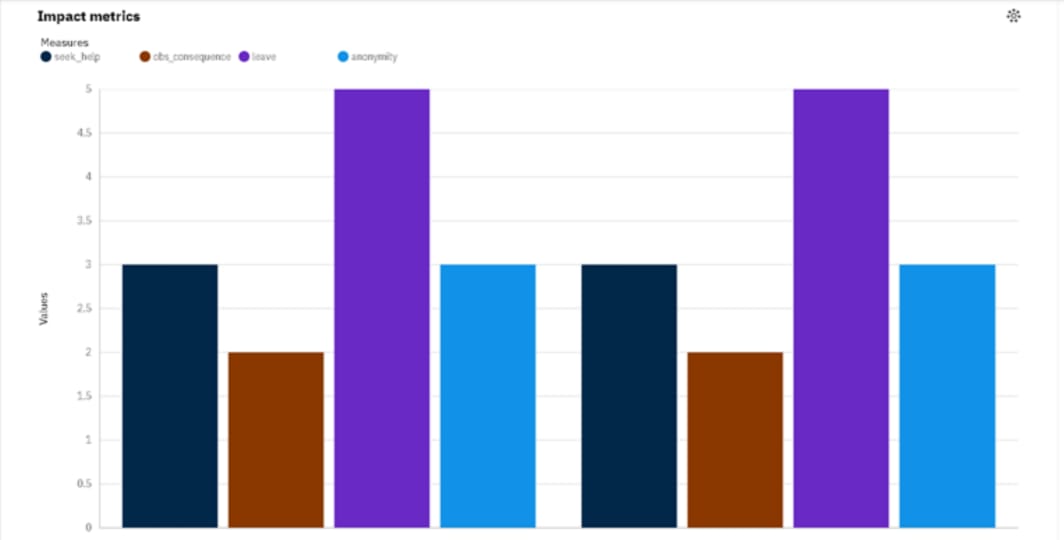




## CAMPAIGN REACH



## Impact metrics

****

## AWARENESS LEVEL

**Conclusion:**

It has focused on the assessment of a public health awareness campaign's effectiveness and impact. We have employed IBM Cognos for data visualization and integrated code, possibly in Python, for advanced data analysis. Through the creation of interactive dashboards and reports, we have effectively visualized campaign reach, awareness levels, and impact metrics, providing actionable insights for campaign organizers, healthcare professionals, and policymakers. Additionally, our use of code for in-depth analysis, including calculating engagement rates, conducting demographic analysis, and running statistical tests, has yielded comprehensive findings. This combined approach equips us to contribute to healthier communities by supporting informed decision-making and promoting public health initiatives.