DAC_PHASE5

DOCUMENTATION AND SUBMISSION

Date: 01-11-2023
Team ID:716
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.

Objective:

The objective of a public health awareness campaign analysis using data analytics is to leverage data-driven insights to assess the effectiveness of the campaign in reaching its target audience, raising awareness, and promoting positive health behaviors. Through data analytics, the campaign aims to identify key performance indicators, assess the impact on public health outcomes, and refine strategies for more impactful and efficient future initiatives.

Design Thinking Process:

1.Data collection

Data Sources: Utilize external sources, such as Kaggle, for obtaining datasets containing mental health survey.

Tools :Jupyter notebook,IBM cognos tool

2.Import necessary libraries

import pandas as

pd import numpy

as np import

seaborn as sns

import matplotlib.pyplot as

pltprint('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?

* 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

*seek_help: Does your employer provide resources to learn more about mental health issues and howto seek help?

* anonymity: Is your anonymity protected if you choose to take advantage of mental health orsubstance 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 employerwould have negative consequences?

* phys_health_consequence: Do you think that discussing a physical health issue with your employerwould 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 aninterview?

* phys_health_interview: Would you bring up a physical health issue with a potential employer in aninterview?

* mental_vs_physical: Do you feel that your employer takes mental health as seriously as physicalhealth?

* obs_consequence: Have you heard of or observed negative consequences for coworkers with mentalhealth conditions in your workplace?

* comments: Any additional notes or comments

* treatment: Have you sought treatment for a mental health condions.

3.Read Dataset

```
data = pd.read_csv('/kaggle/input/mental-health-in-tech-

survey/survey.csv')data.head()
```

4 .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

Plotax.bar_label(ax.containers[0]);
```

```
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]);
```

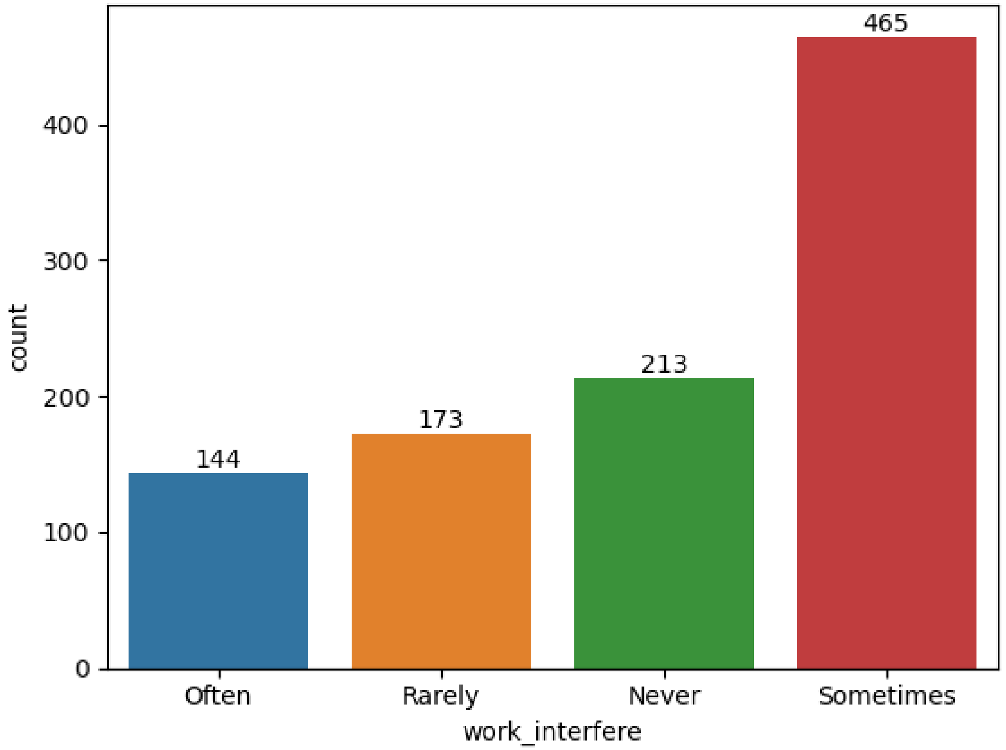


Fig 1 :Bar plot for work interfere

In this diagram x axis is work interfere and y axis is count this diagram shows sometimes is high.

```
 #For filling nan values i used SimpleImputer but you can use fillnan function toofrom
 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]);
```

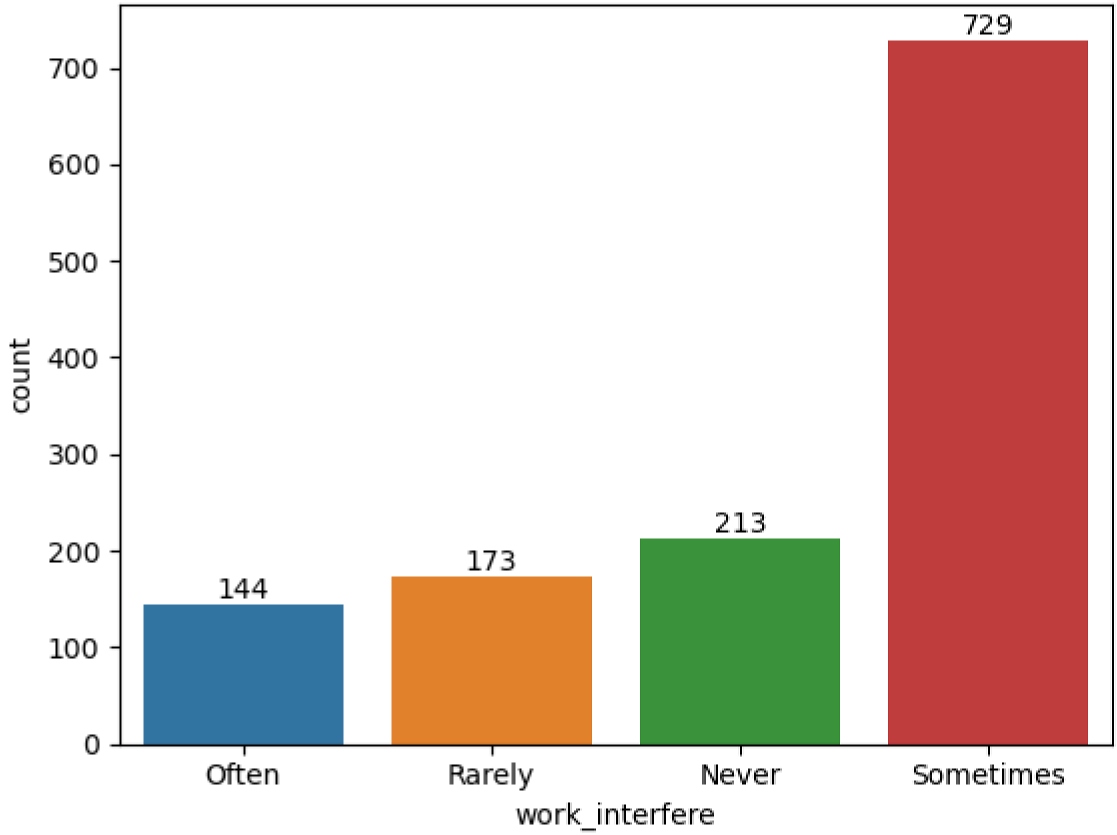


Fig 2 :Bar plot for work interfere

In this diagram x axis is work interfere and y axis is count this diagram shows sometimes is high.

```
#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

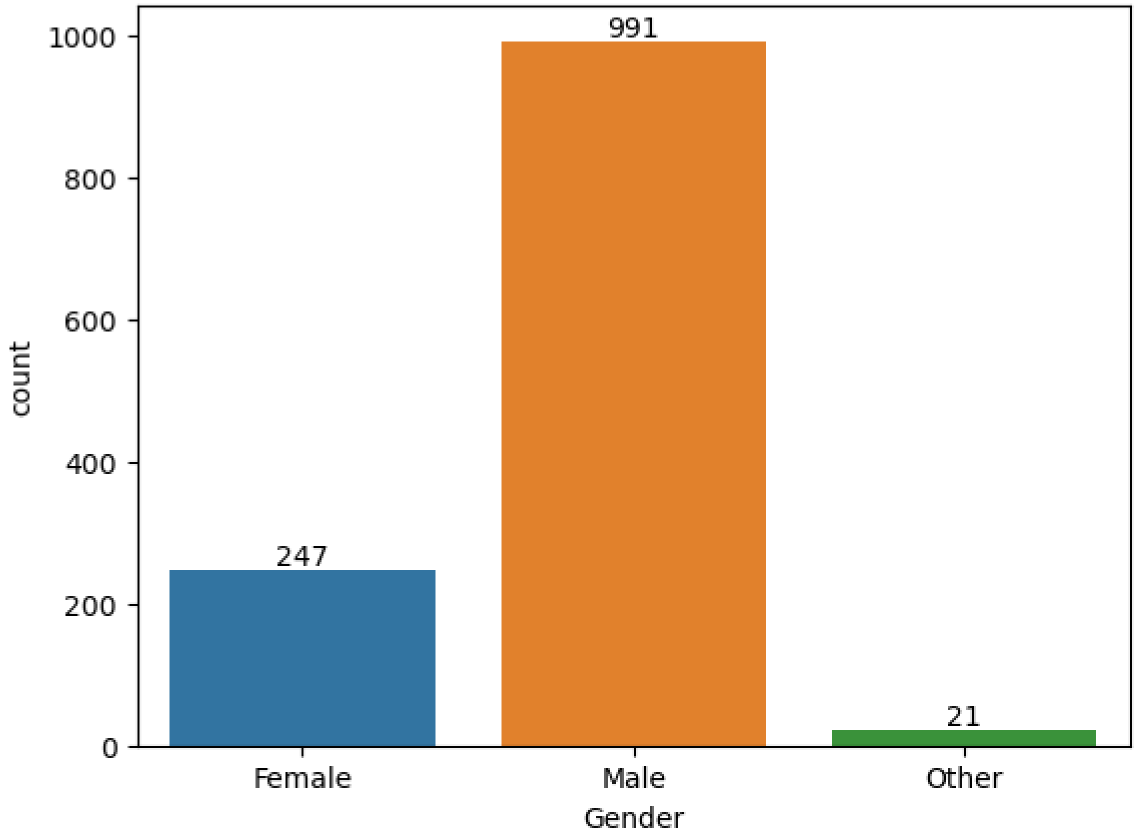


Fig 3 :Bar plot for gender and count

In this diagram x axis is gender and y axis is count this diagram shows male is high.

```
#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]
```

```
#Let's see the 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

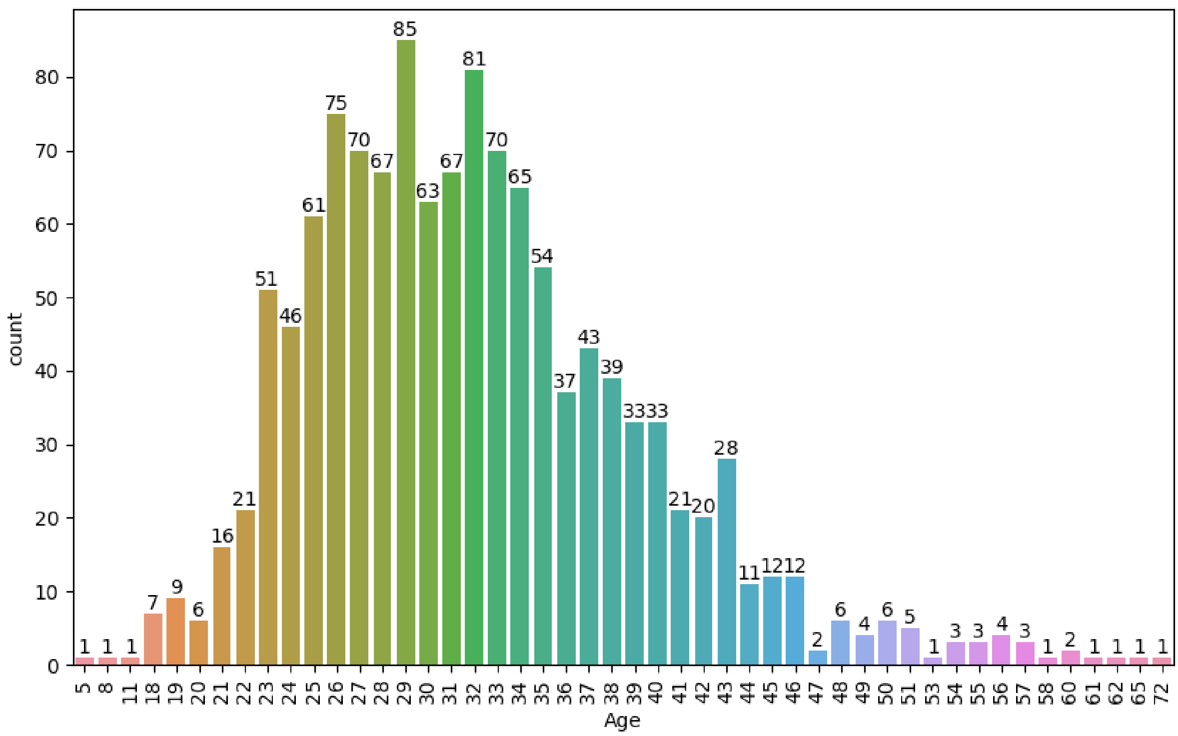


Fig 4 :Range plot for age and count

In this diagram x axis is age and y axis is count

#In this plot moreover on 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>

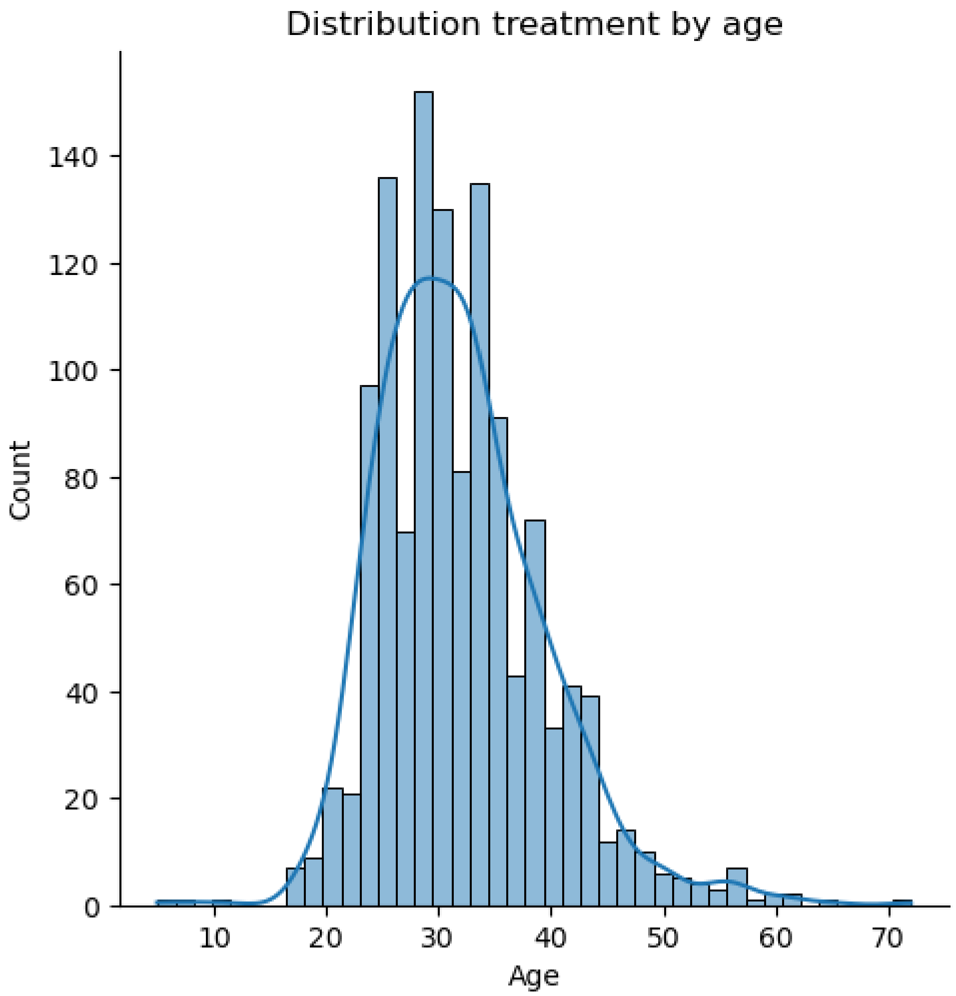


Fig 5 :Bar polt for age and count

In this diagram x axis is age and y axis is count .

#In this plot We can see 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');

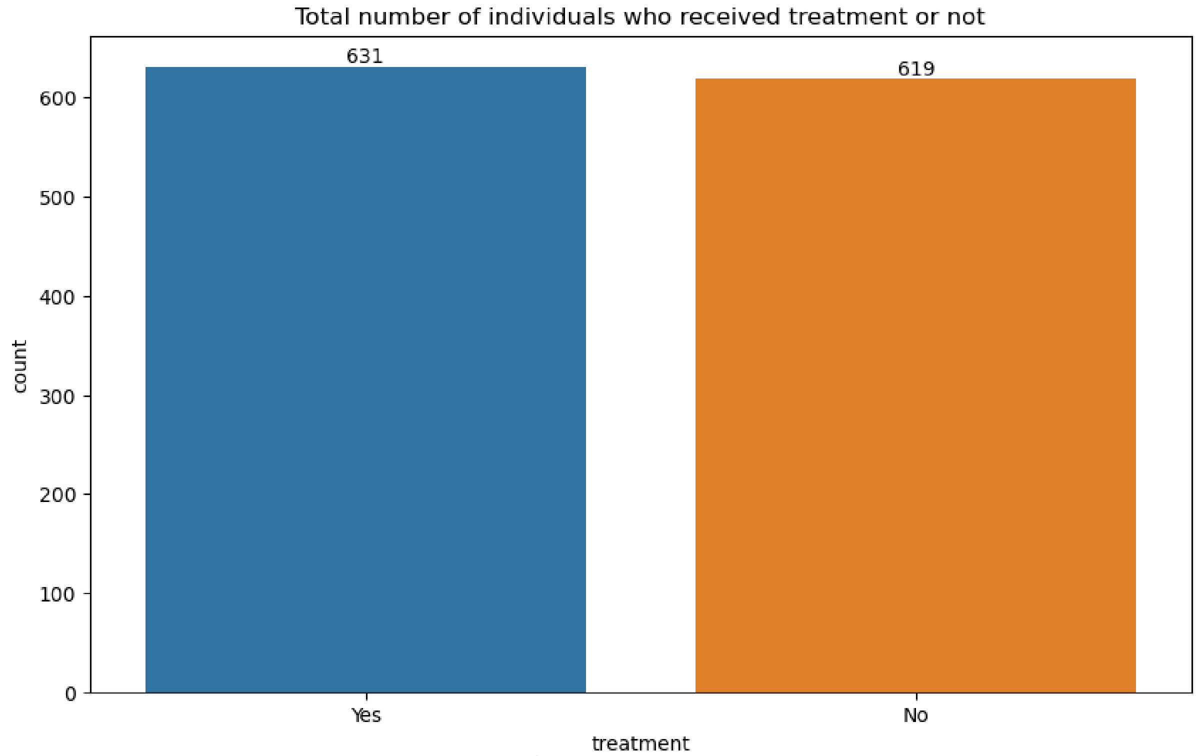


Fig 6 :Bar polt for treatment and count

In this diagram x axis is treatment and y axis is count.

data.info()

Output

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1250 entries, 0 to 1258

Data columns (total 24 columns):

#	Column	Non-Null Count	Dtype
---	-------	----------------	------
0	Age	1250 non-null	int64
1	Gender	1250 non-null	object
2	Country	1250 non-null	object
3	self_employed	1250 non-null	object
4	family_history	1250 non-null	object
5	treatment	1250 non-null	object
6	work_interfere	1250 non-null	object
7	no_employees	1250 non-null	object
8	remote_work	1250 non-null	object
9	tech_company	1250 non-null	object

```
 10  benefits                  1250 non-null  object
 11  care_options              1250 non-null  object
 12  wellness_program          1250 non-null  object
 13  seek_help                 1250 non-null  object
 14  anonymity                 1250 non-null  object
 15  leave                     1250 non-null  object
 16  mental_health_consequence  1250 non-null  object
 17  phys_health_consequence    1250 non-null  object
 18  coworkers                 1250 non-null  object
 19  supervisor                1250 non-null  object
...
 22  mental_vs_physical        1250 non-null  object
 23  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
---  -------                    --------------  -----
 0   Age                        1250 non-null   int64
 1   Gender                     1250 non-null   int64
 2   Country                    1250 non-null   int64
 3   self_employed              1250 non-null   int64
 4   family_history             1250 non-null   int64
 5   treatment                  1250 non-null   int64
 6   work_interfere             1250 non-null   int64
 7   no_employees               1250 non-null   int64
 8   remote_work                1250 non-null   int64
 9   tech_company               1250 non-null   int64
 10  benefits                   1250 non-null   int64
 11  care_options               1250 non-null   int64
 12  wellness_program           1250 non-null   int64
 13  seek_help                  1250 non-null   int64
 14  anonymity                  1250 non-null   int64
 15  leave                      1250 non-null   int64
 16  mental_health_consequence  1250 non-null   int64
 17  phys_health_consequence    1250 non-null   int64
 18  coworkers                  1250 non-null   int64
 19  supervisor                 1250 non-null   int64
...
 22  mental_vs_physical         1250 non-null   int64
 23  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

A Support Vector Machine (SVM) is a type of supervised machine learning algorithm used for classification and regression tasks. Its primary purpose is to find the optimal hyperplane that best separates data points into different classes or predicts a continuous target variable.

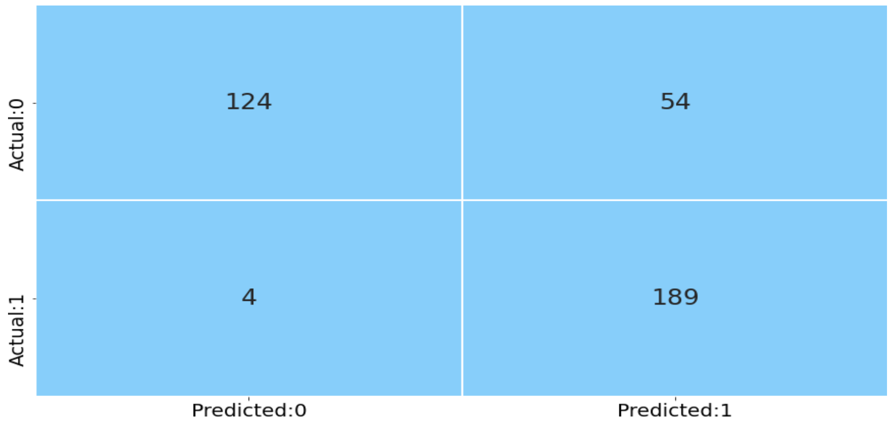
```
svclassifier = SVC(kernel = 'linear')

# 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)

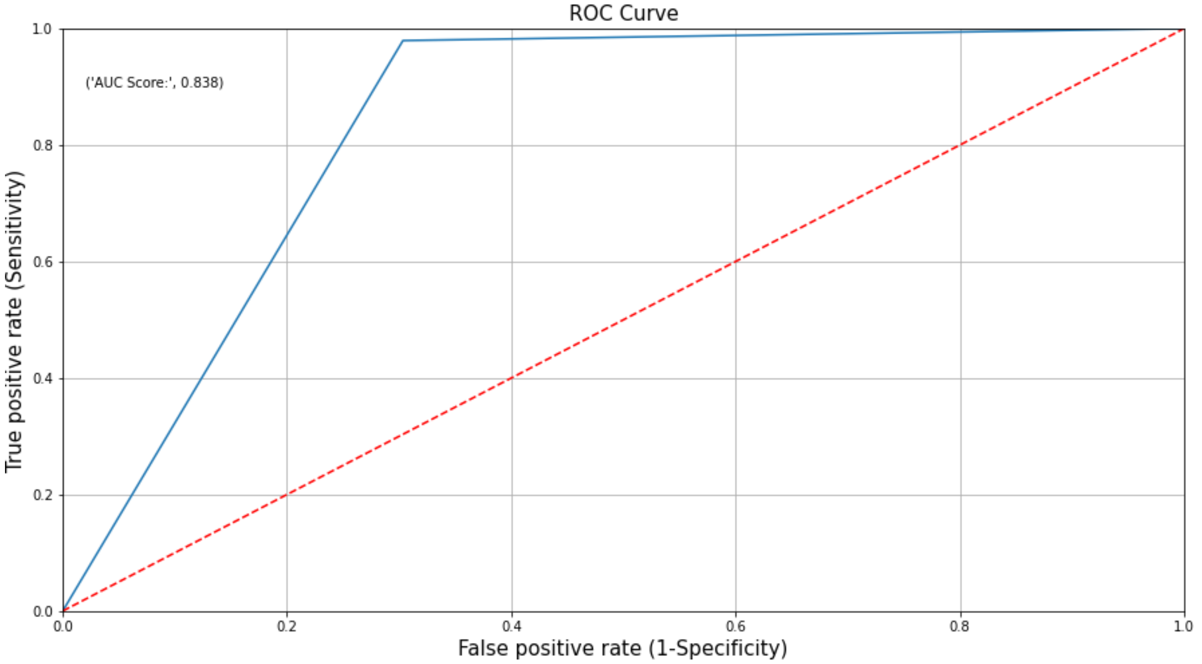
print the kappa value

print(kappa_value)

0.6833632537743901

plot_roc(svc_model)

Output

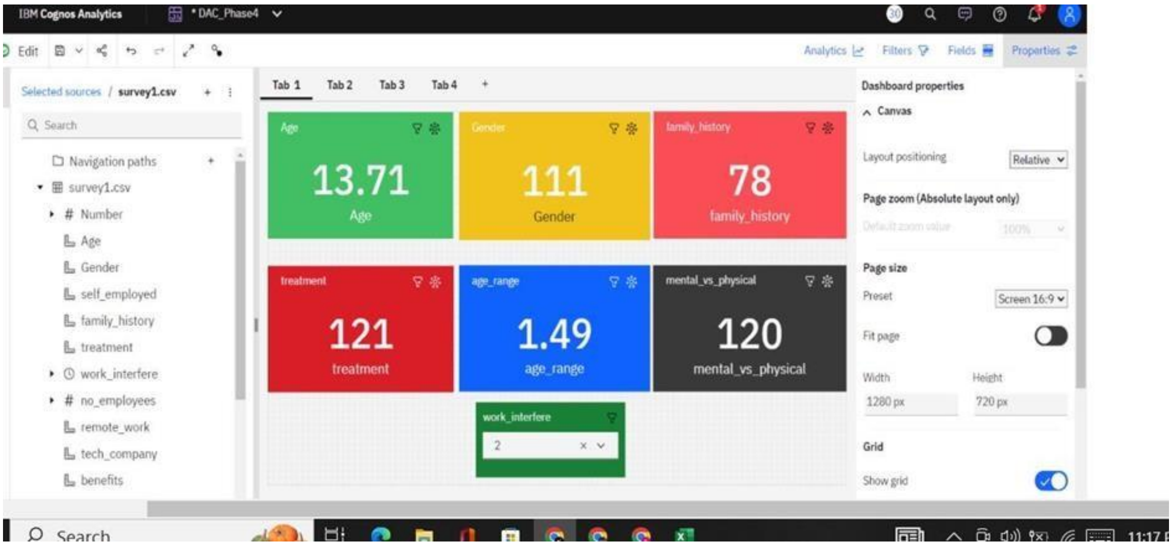


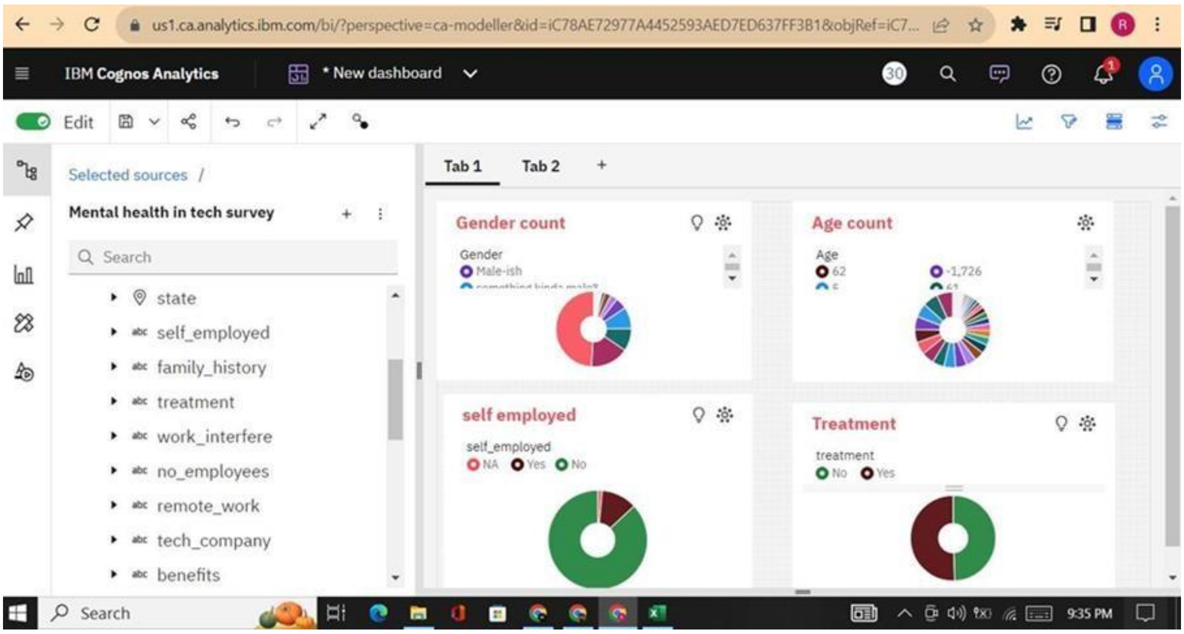
Accuracy 0.84

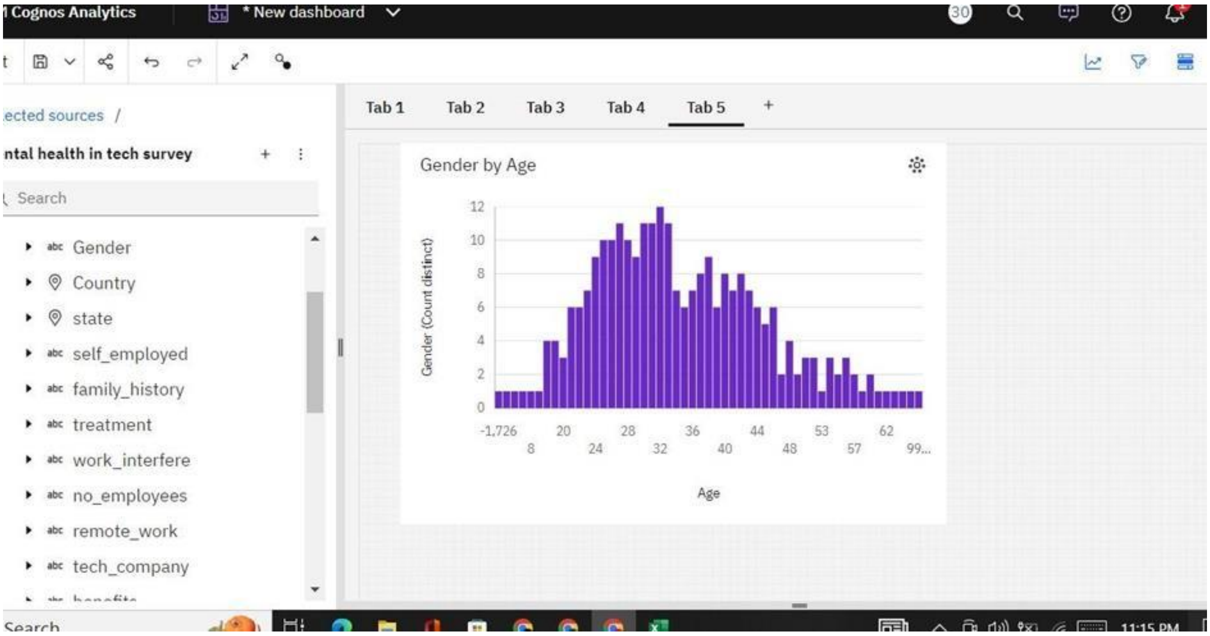
Fig 7: Support Vector Machine

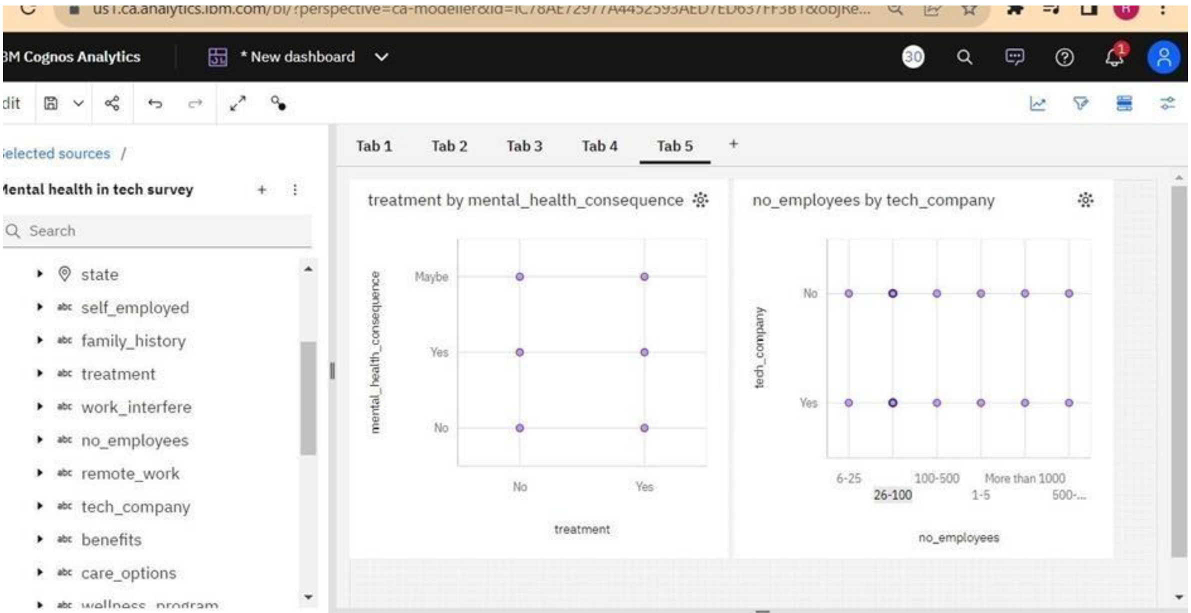
In this diagram x axis is false positive rate and y axis is true positive rate

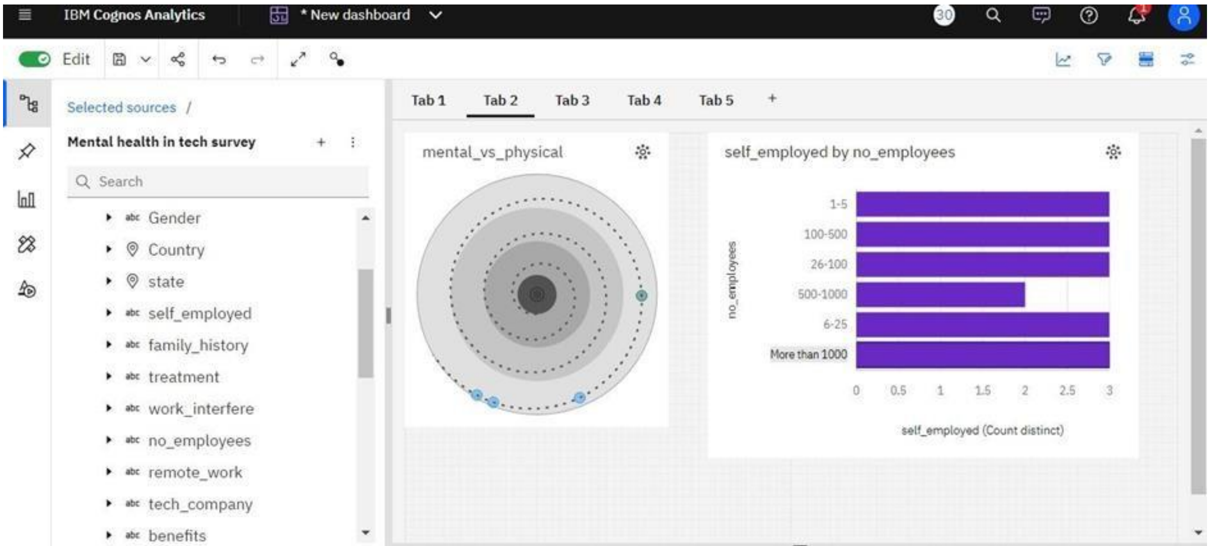
5.Visualizations in IBM cognos tool

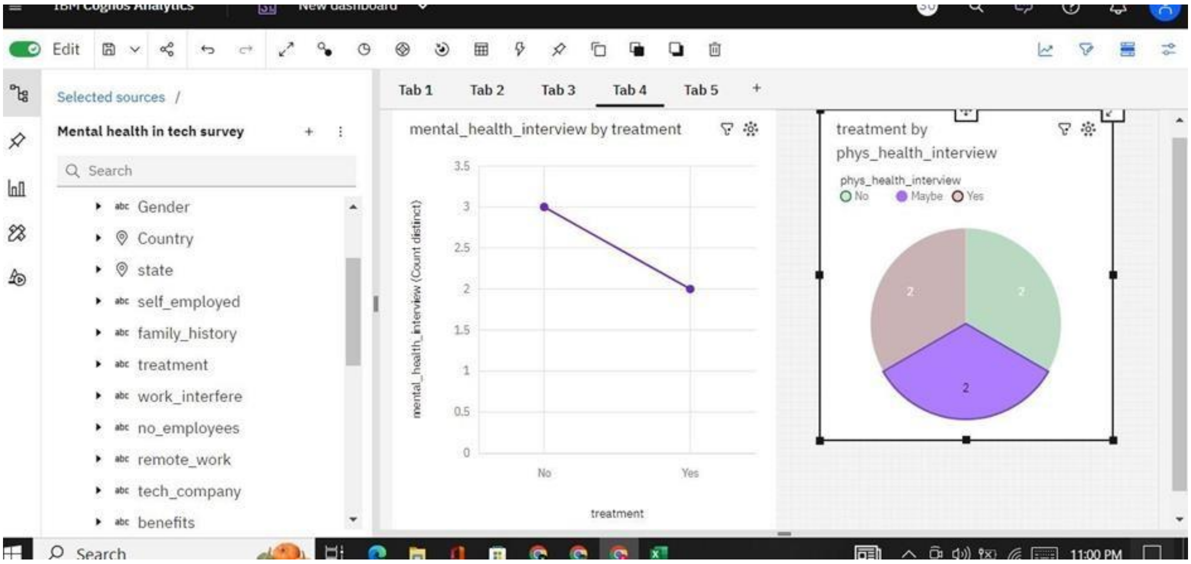

IBM Cognos Analytics
* DAC_Phase4
Edit
Analytics
Filters
Fields
Properties
Selected sources / survey1.csv
Search
Navigation paths
survey1.csv
Number
Age
Gender
self_employed
family_history
treatment
work_interfere
no_employees
remote_work
tech_company
benefits
Tab 1
Tab 2
Tab 3
Tab 4
Age
13.71
Age
Gender
111
Gender
family_history
78
family_history
treatment
121
treatment
age_range
1.49
age_range
mental_vs_physical
120
mental_vs_physical
work_interfere
2
Dashboard properties
Canvas
Layout positioning
Relative
Page zoom (Absolute layout only)
Page size
Preset
Screen 16:9
Fit page
Width
Height
1280 px
720 px
Grid
Show grid
Search

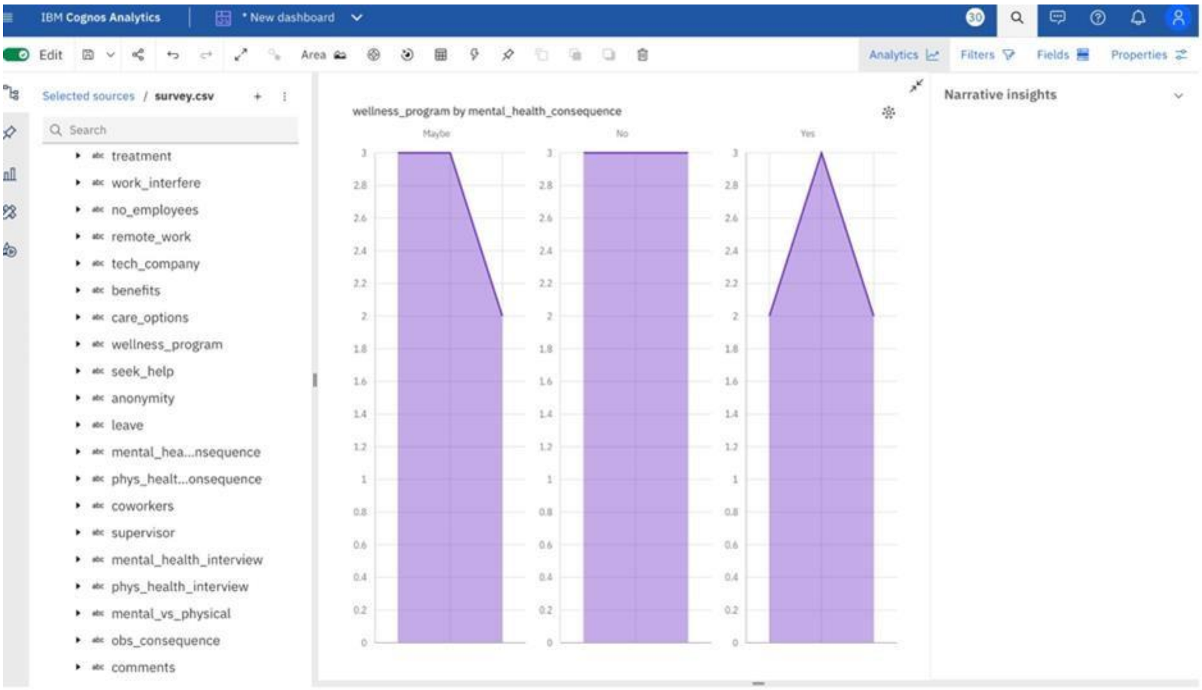

us1.ca.analytics.ibm.com/bi/?perspective=ca-modeller&id=iC78AE72977A4452593AED7ED637FF3B1&objRef=iC7...
IBM Cognos Analytics
* New dashboard
Edit
Selected sources /
Mental health in tech survey
Search
state
self_employed
family_history
treatment
work_interfere
no_employees
remote_work
tech_company
benefits
Tab 1
Tab 2
Gender count
Gender
Male-ish
Age count
Age
62
-1,726
self employed
self_employed
NA
Yes
No
Treatment
treatment
No
Yes
Search
9:35 PM


Cognos Analytics
* New dashboard
ected sources /
ntal health in tech survey
Search
Gender
Country
state
self_employed
family_history
treatment
work_interfere
no_employees
remote_work
tech_company
Tab 1
Tab 2
Tab 3
Tab 4
Tab 5
Gender by Age
Gender (Count distinct)
12
10
8
6
4
2
0
-1,726
8
20
24
28
32
36
40
44
48
53
57
62
99...
Age
Search
11:15 PM


BM Cognos Analytics
* New dashboard
ielected sources /
Mental health in tech survey
Search
state
self_employed
family_history
treatment
work_interfere
no_employees
remote_work
tech_company
benefits
care_options
Tab 1
Tab 2
Tab 3
Tab 4
Tab 5
treatment by mental_health_consequence
mental_health_consequence
Maybe
Yes
No
No
Yes
treatment
no_employees by tech_company
tech_company
No
Yes
6-25
26-100
100-500
1-5
More than 1000
500-...
no_employees

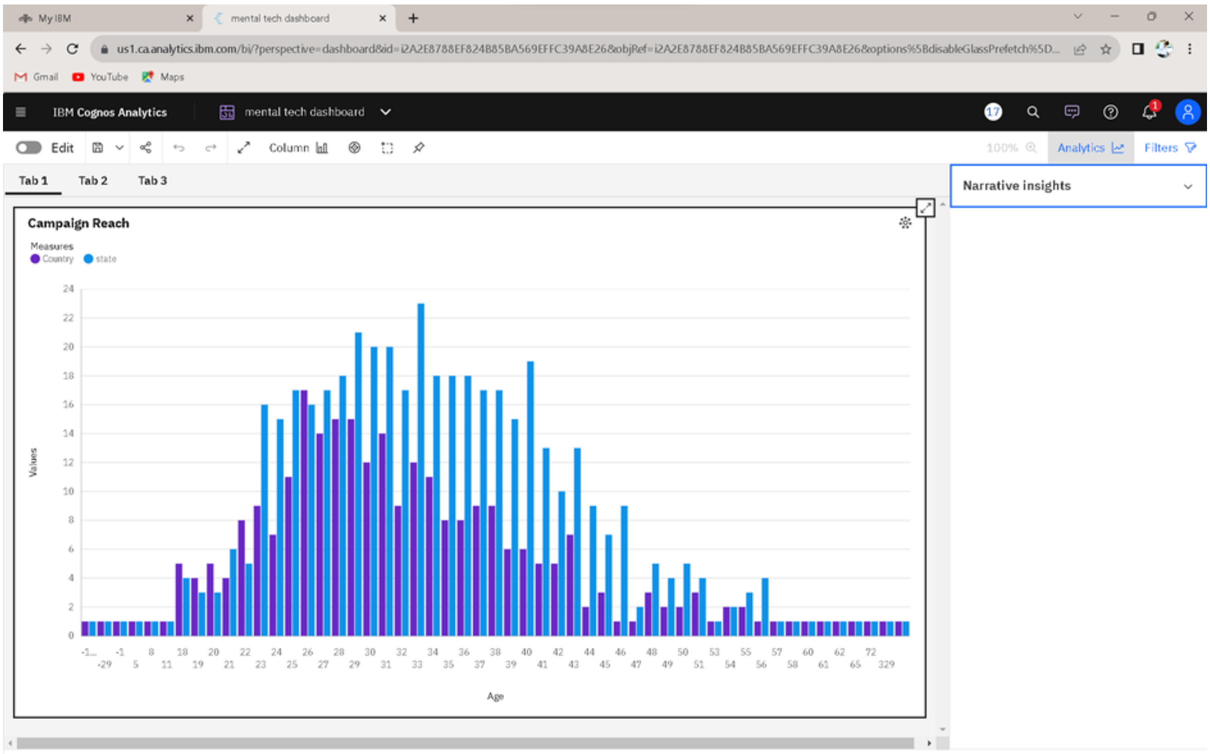

IBM Cognos Analytics
* New dashboard
Edit
Selected sources /
Mental health in tech survey
Search
Gender
Country
state
self_employed
family_history
treatment
work_interfere
no_employees
remote_work
tech_company
benefits
Tab 1
Tab 2
Tab 3
Tab 4
Tab 5
mental_vs_physical
self_employed by no_employees
1-5
100-500
26-100
500-1000
6-25
More than 1000
no_employees
0
0.5
1
1.5
2
2.5
3
self_employed (Count distinct)


Edit
Selected sources /
Mental health in tech survey
Search
Gender
Country
state
self_employed
family_history
treatment
work_interfere
no_employees
remote_work
tech_company
benefits
Tab 1
Tab 2
Tab 3
Tab 4
Tab 5
mental_health_interview by treatment
3.5
3
2.5
2
1.5
1
0.5
0
mental_health_interview (Count distinct)
No
Yes
treatment
treatment by
phys_health_interview
phys_health_interview
No
Maybe
Yes
2
2
2
Search
11:00 PM

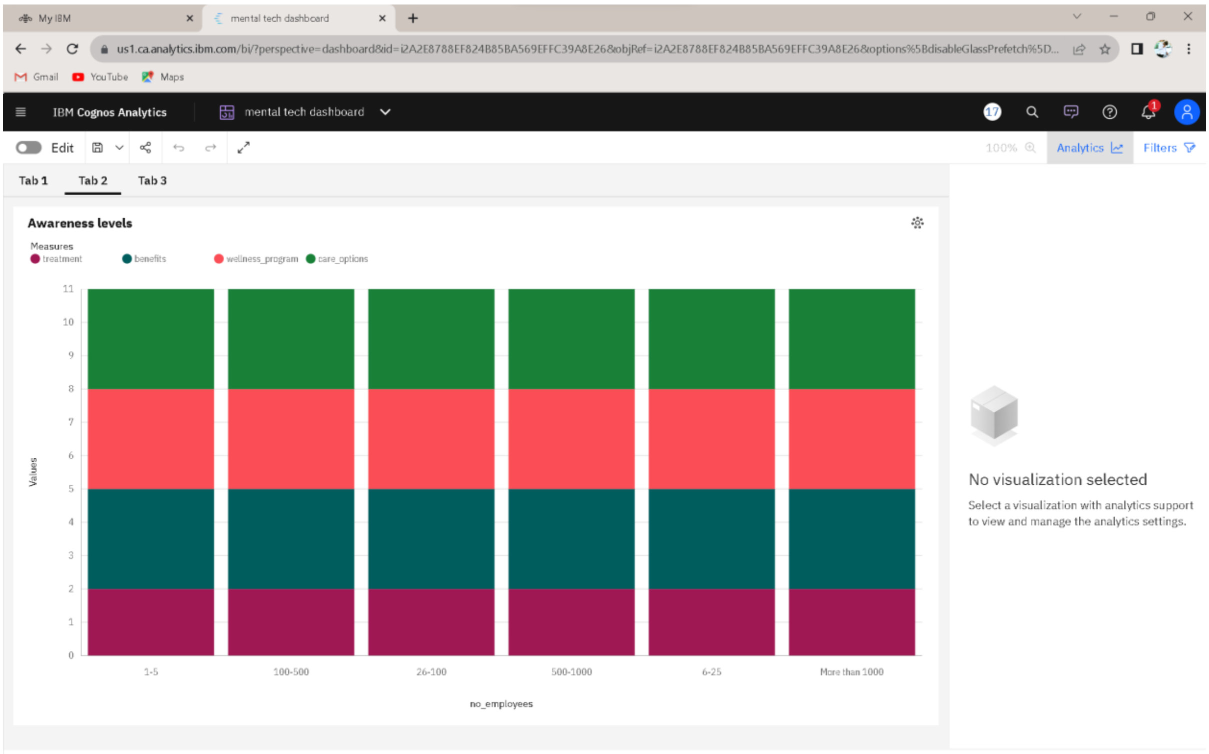


6.Dashboard and Report using ibm cognos tool

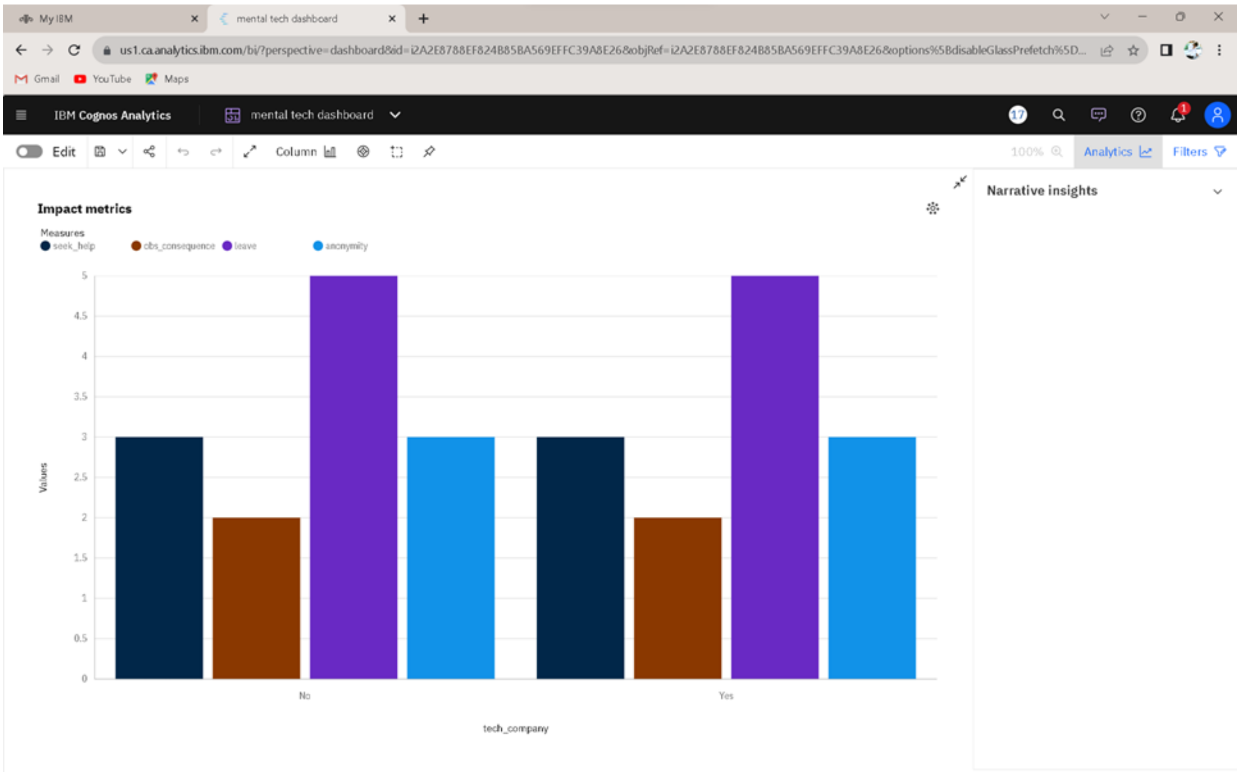
Campaign Analysis



Awareness Level



Impact metrics



7.User experience:

- Data analytics with tools like IBM Cognos can be a game-changer for website owners focused on public health awareness.
- It provides valuable insights into how people use their websites.
- These insights can help in several ways: First, by understanding what pages users visit the most and where they leave the site, website owners can make those pages better and fix the weak spots.
- Second, they can figure out what kinds of information are most interesting to their audience and create more of that.
- Third, they can test different website designs and content to see what works best. Plus, analytics can show if the website is slow or has errors that need fixing.
- It's like having a map to guide them in making their website more useful and engaging for people interested in public health.

8.Conclusion

In conclusion, using data analytics tools like IBM Cognos can be a powerful ally for website owners aiming to enhance the user experience in public health awareness. By understanding how people navigate the website, what content they prefer, and where improvements are needed, owners can tailor their site to be more user-friendly and engaging. It's like having a compass to guide them toward providing vital health information effectively. By continuously analyzing and acting on data-driven insights, they can ensure that their website becomes a valuable resource for promoting public health and well-being, ultimately serving the needs of their audience more efficiently and effectively.