PUBLIC HEALTH AWARENESS

TEAM MEMBERS:

KEERTHI VARSHINI S - 2021115053

KONETI SASANK - 2021115054

KOWSHIK T - 2021115055

LEKHA S - 2021115056

SANTHOSH - 2021115311

Objectives:

In this phase defines start to building the Project by loading and preprocessing the dataset and perform different analysis and visualization using IBM Cognos.

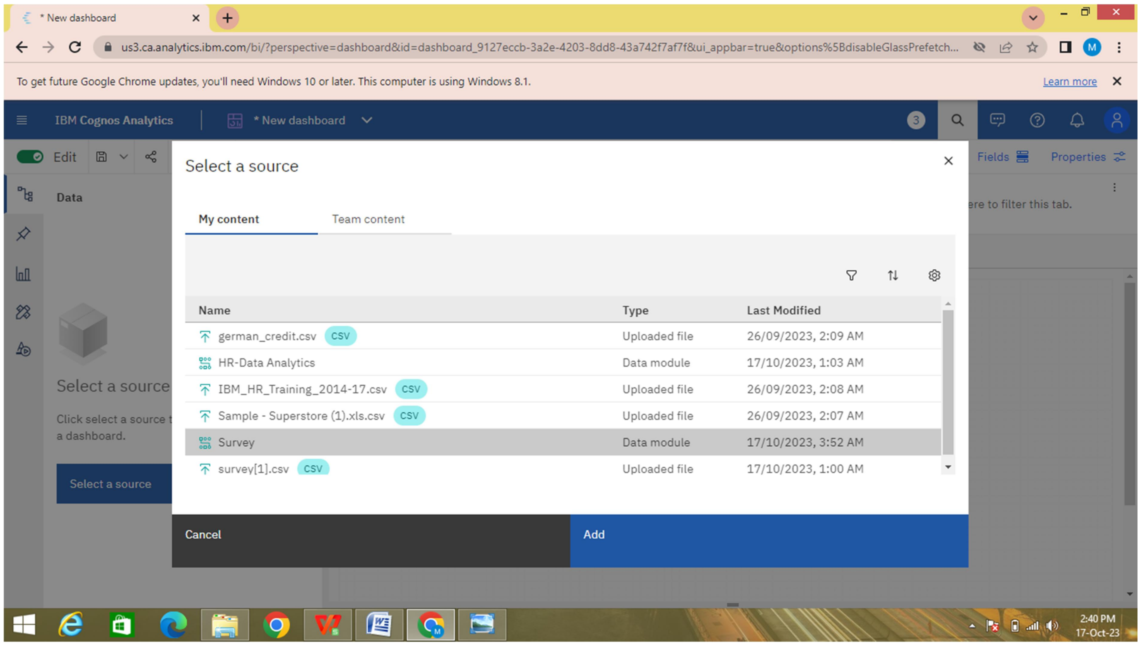
Visualization in IBM Cognos

Steps Involved in data loading on IBM cognos.

Step 1:

1. Login to your IBM cognos
2. Click more menu from the left side
3. Select new tab
4. Click Dashboard tap
5. Select Template for your dashboard
6. Now Dashboard is created and select your data source

7. Select the data source



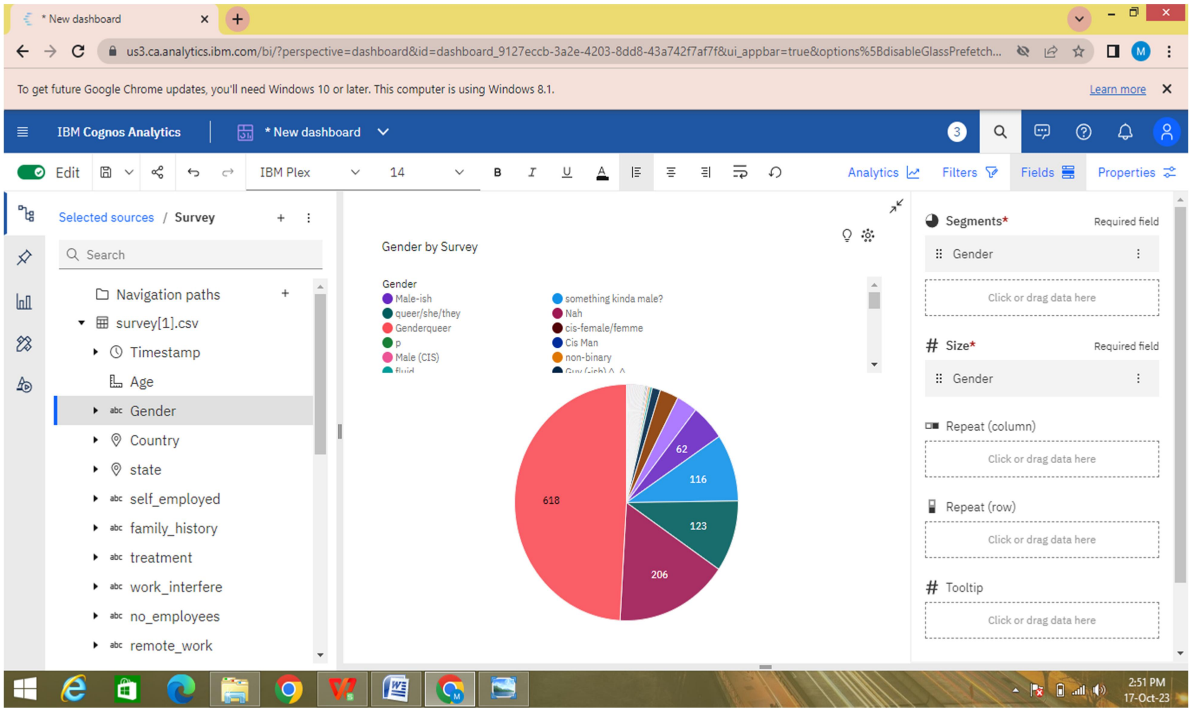
Visualization

After creating the dashboard, the next step is to visualize the data

In IBM Cognos

1. Goes to the Corresponding Dashboard

2. select the visualizations tab in the left side of title bar



In the above screen shot displays the Pie chart in Gender by survey.

After performing these activities a comprehensive document will be created to demonstrate the ability to Communicate and share finding.

```python
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt
import seaborn as sns

from scipy import stats

from scipy.stats import randint


# prep

from sklearn.model_selection import train_test_split
from sklearn import preprocessing

from sklearn.datasets import make_classification

from sklearn.preprocessing import binarize, LabelEncoder, MinMaxScaler

# models

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier,

ExtraTreesClassifier


# Validation libraries

from sklearn import metrics

from sklearn.metrics import accuracy_score, mean_squared_error,
precision_recall_curve

from sklearn.model_selection import cross_val_score


#Neural Network

from sklearn.neural_network import MLPClassifier


#Bagging

from sklearn.ensemble import BaggingClassifier, AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier

#Naive bayes

from sklearn.naive_bayes import GaussianNB


#Stacking

from mlxtend.classifier import StackingClassifier
```

```python
# Any results you write to the current directory are saved as output.
#reading in CSV's from a file path
train_df = pd.read_csv("C:\\Users\Manikandan\Downloads\survey.csv")

#Pandas: whats the data row count?
print(train_df.shape)

#Pandas: whats the distribution of the data?
print(train_df.describe())
```

```
#Pandas: What types of data do i have?

print(train_df.info())

(1259, 27)

                   Age
count  1.259000e+03
mean  7.942815e+07

std     2.818299e+09
min    -1.726000e+03
25%     2.700000e+01
50%     3.100000e+01
75%     3.600000e+01
max     1.000000e+11
<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
None

pip install mlxtend
```

```
Defaulting to user installation because normal site-packages is not
writeable

Collecting mlxtend

  Obtaining dependency information for mlxtend from
https://files.pythonhosted.org/packages/73/da/d5d77a9a7a135c948dbf8d3b
873655b105a152d69e590150c83d23c3d070/mlxtend-0.23.0-py3-none-
any.whl.metadata

  Downloading mlxtend-0.23.0-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: scipy>=1.2.1 in c:\programdata\
anaconda3\lib\site-packages (from mlxtend) (1.11.1)

Requirement already satisfied: numpy>=1.16.2 in c:\programdata\
anaconda3\lib\site-packages (from mlxtend) (1.24.3)

Requirement already satisfied: pandas>=0.24.2 in c:\programdata\
anaconda3\lib\site-packages (from mlxtend) (2.0.3)

Requirement already satisfied: scikit-learn>=1.0.2 in c:\programdata\
anaconda3\lib\site-packages (from mlxtend) (1.3.0)

Requirement already satisfied: matplotlib>=3.0.0 in c:\programdata\
anaconda3\lib\site-packages (from mlxtend) (3.7.2)

Requirement already satisfied: joblib>=0.13.2 in c:\users\harsh\
appdata\roaming\python\python311\site-packages (from mlxtend) (1.1.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\
programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0-

>mlxtend) (3.0.9)

Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\
anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend)
(2023.3.post1)

Requirement already satisfied: tzdata>=2022.1 in c:\programdata\
anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2023.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\
anaconda3\lib\site-packages (from scikit-learn>=1.0.2->mlxtend)
(2.2.0)

Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\
lib\site-packages (from python-dateutil>=2.7->matplotlib>=3.0.0-
```

```
#dealing with missing data

#Let's get rid of the variables "Timestamp","comments", "state" just
to make our lives easier.

train_df = train_df.drop(['comments'], axis= 1)
train_df = train_df.drop(['state'], axis= 1)
train_df = train_df.drop(['Timestamp'], axis= 1)

train_df.isnull().sum().max() #just checking that there's no missing
data missing...

train_df.head(5)
```

```
   Age  Gender         Country self_employed family_history treatment
\
0   37  Female   United States           NaN             No       Yes

1   44       M   United States           NaN             No        No

2   32    Male          Canada           NaN             No        No

3   31    Male  United Kingdom           NaN            Yes       Yes

4   31    Male   United States           NaN             No        No


  work_interfere    no_employees remote_work tech_company ...
anonymity \

0          Often            6-25          No          Yes  ...
Yes
1         Rarely  More than 1000          No           No  ...  Don't
know
2         Rarely            6-25          No          Yes  ...  Don't
know
3          Often          26-100          No          Yes  ...
```

```
No
4        Never        100-500        Yes        Yes  ...  Don't
know


                leave mental_health_consequence
phys_health_consequence \
0        Somewhat easy                    No

No          Don't know                  Maybe
1
No   Somewhat difficult                    No
2
     Somewhat difficult                   Yes
No
3           Don't know                     No


      coworkers supervisor mental_health_interview
phys_health_interview \
0 Some of them          Yes                  No
Maybe
1            No          No                  No

No          Yes         Yes                 Yes
2
3   Some of them         No               Maybe
Maybe
4   Some of themYes     Yes                 Yes


  mental_vs_physical obs_consequence
0                Yes              No
          Don't know              No
1                                 No
                  No             Yes
2                 No              No
[5 rows x 24 columns]
```

```
# Assign default values for each data type

defaultInt = 0
defaultString = 'NaN'
defaultFloat = 0.0

# Create lists by data tpe

intFeatures = ['Age']

stringFeatures = ['Gender', 'Country', 'self_employed',
'family_history', 'treatment', 'work_interfere',

                 'no_employees', 'remote_work', 'tech_company',
'anonymity', 'leave', 'mental_health_consequence',
```

```
                 'phys_health_consequence', 'coworkers', 'supervisor',
'mental_health_interview', 'phys_health_interview',

                 'mental_vs_physical', 'obs_consequence', 'benefits',
'care_options', 'wellness_program',

                 'seek_help']
floatFeatures = []

# Clean the NaN's

for feature in train_df:

    if feature in intFeatures:

        train_df[feature] = train_df[feature].fillna(defaultInt)
    elif feature in stringFeatures:

        train_df[feature] = train_df[feature].fillna(defaultString)
    elif feature in floatFeatures:

        train_df[feature] = train_df[feature].fillna(defaultFloat)
    else:

        print('Error: Feature %s not recognized.' % feature)
train_df.head(5)
```

```
   Age  Gender         Country self_employed family_history treatment
\
0   37  Female   United States           NaN             No       Yes

1   44       M   United States           NaN             No        No

2   32    Male          Canada           NaN             No        No

3   31    Male  United Kingdom           NaN            Yes       Yes

4   31    Male   United States           NaN             No        No


  work_interfere    no_employees remote_work tech_company ...
anonymity \

0          Often            6-25          No          Yes  ...
Yes
1         Rarely  More than 1000          No           No  ...  Don't
know
2         Rarely            6-25          No          Yes  ...  Don't
know
3          Often          26-100          No          Yes  ...
No
4          Never         100-500         Yes          Yes  ...  Don't
know
```

```
No
2
  Somewhat difficult                        No
No
3

Yes Somewhat difficult                      Yes
4

No
           Don't know                       No



      coworkers supervisor mental_health_interview
phys_health_interview  \
0  Some of them        Yes                      No
Maybe
1            No         No                      No
No
2           Yes        Yes                     Yes
Yes
3  Some of them         No                   Maybe
Maybe
4  Some of them        Yes                     Yes
Yes
```

```
0                 Yes              No
          Don't know               No
1                                  No
                   No             Yes
2                  No              No
```

```
[5 rows x 24 columns]

#clean 'Gender'

#Slower case all columm's elements
gender = train_df['Gender'].str.lower()
#print(gender)

#Select unique elements

gender = train_df['Gender'].unique()
```

```python
#Made gender groups

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"]
```

```
for (row, col) in train_df.iterrows():


    if str.lower(col.Gender) in male_str:
        train_df['Gender'].replace(to_replace=col.Gender,

value='male', inplace=True)


    if str.lower(col.Gender) in female_str:
        train_df['Gender'].replace(to_replace=col.Gender,

value='female', inplace=True)


    if str.lower(col.Gender) in trans_str:
        train_df['Gender'].replace(to_replace=col.Gender,

value='trans', inplace=True)


#Get rid of bullshit

stk_list = ['A little about you', 'p']

train_df = train_df[~train_df['Gender'].isin(stk_list)]

print(train_df['Gender'].unique())
```

```
['female' 'male' 'trans']
```

```
#complete missing age with mean

train_df['Age'].fillna(train_df['Age'].median(), inplace = True)


# Fill with media() values < 18 and > 120

s = pd.Series(train_df['Age'])
s[s<18] = train_df['Age'].median()
train_df['Age'] = s

s = pd.Series(train_df['Age'])
s[s>120] = train_df['Age'].median()
train_df['Age'] = s

#Ranges of Age

train_df['age_range'] = pd.cut(train_df['Age'], [0,20,30,65,100],
labels=["0-20", "21-30", "31-65", "66-100"], include_lowest=True)

#There are only 0.014% of self employed so let's change NaN to NOT
self_employed
```

```python
#Replace "NaN" string from defaultString
train_df['self_employed'] =
train_df['self_employed'].replace([defaultString], 'No')
print(train_df['self_employed'].unique())
```

```
['No' 'Yes']
```

```python
#There are only 0.20% of self work_interfere so let's change NaN to
"Don't know

#Replace "NaN" string from defaultString


train_df['work_interfere']= train_df['work_interfere'].replace
([defaultString], 'Don\'t know' )
print(train_df['work_interfere'].unique())
```

```
['Often' 'Rarely' 'Never' 'Sometimes' "Don't know"]
```

```python
#Encoding data

labelDict = {}

for feature in train_df:

    le = preprocessing.LabelEncoder()
    le.fit(train_df[feature])
    le_name_mapping = dict(zip(le.classes_,

le.transform(le.classes_)))

    train_df[feature] = le.transform(train_df[feature])

    # Get labels

labelKey = 'label_' + feature labelValue = [*le_name_mapping]
labelDict[labelKey] =labelValue


for key, value in labelDict.items():
    print(key, value)
```

```
#Get rid of 'Country'

train_df = train_df.drop(['Country'], axis= 1)
train_df.head()

label_Age [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,

33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,

50, 51, 53, 54, 55, 56, 57, 58, 60, 61, 62, 65, 72]

label_Gender ['female', 'male', 'trans']

label_Country ['Australia', 'Austria', 'Belgium', 'Bosnia and
Herzegovina', 'Brazil', 'Bulgaria', 'Canada', 'China', 'Colombia',
'Costa Rica', 'Croatia', 'Czech Republic', 'Denmark', 'Finland',
'France', 'Georgia', 'Germany', 'Greece', 'Hungary', 'India',

'Ireland', 'Israel', 'Italy', 'Japan', 'Latvia', 'Mexico', 'Moldova',
'Netherlands', 'New Zealand', 'Nigeria', 'Norway', 'Philippines',
'Poland', 'Portugal', 'Romania', 'Russia', 'Singapore', 'Slovenia',
'South Africa', 'Spain', 'Sweden', 'Switzerland', 'Thailand', 'United
Kingdom', 'United States', 'Uruguay', 'Zimbabwe']

label_self_employed ['No', 'Yes']
label_family_history ['No', 'Yes']
label_treatment ['No', 'Yes']

label_work_interfere ["Don't know", 'Never', 'Often', 'Rarely',
'Sometimes']

label_no_employees ['1-5', '100-500', '26-100', '500-1000', '6-25',

'More than 1000']
label_remote_work ['No', 'Yes']
label_tech_company ['No', 'Yes']

label_benefits ["Don't know", 'No', 'Yes']
label_care_options ['No', 'Not sure', 'Yes']
label_wellness_program ["Don't know", 'No', 'Yes']
```

```
label_seek_help ["Don't know", 'No', 'Yes']
label_anonymity ["Don't know", 'No', 'Yes']

label_leave ["Don't know", 'Somewhat difficult', 'Somewhat easy',
'Very difficult', 'Very easy']

label_mental_health_consequence ['Maybe', 'No', 'Yes']
label_phys_health_consequence ['Maybe', 'No', 'Yes']
label_coworkers ['No', 'Some of them', 'Yes']
label_supervisor ['No', 'Some of them', 'Yes']
label_mental_health_interview ['Maybe', 'No', 'Yes']
label_phys_health_interview ['Maybe', 'No', 'Yes']
label_mental_vs_physical ["Don't know", 'No', 'Yes']
label_obs_consequence ['No', 'Yes']

label_age_range ['0-20', '21-30', '31-65', '66-100']


   Age  Gender  self_employed  family_history  treatment
work_interfere  \
0   19       0              0               0          1
2
1   26       1              0               0          0
3
2   14       1              0               0          0
3
3   13       1              0               1          1
2
4   13       1              0               0          0
1


   no_employees  remote_work  tech_company  benefits  ...  leave  \
0             4            0             1         2  ...      2
1             5            0             0         0  ...      0
2             4            0             1         1  ...      1
3             2            0             1         1  ...      1
4             1            1             1         2  ...      0


   mental_health_consequence phys_health_consequence coworkers
supervisor \

0                          1                       1         1
2
1                          0                       1         0
0
2                          1                       1         2
2
3                          2                       2         1
0
4                          1                       1         1
2
```

mental_health_interview phys_health_interview mental_vs_physical

```
1                        1                       1                   0

2                        2                       2                   1

3                        0                       0                   1

4                        2                       2                   0
```

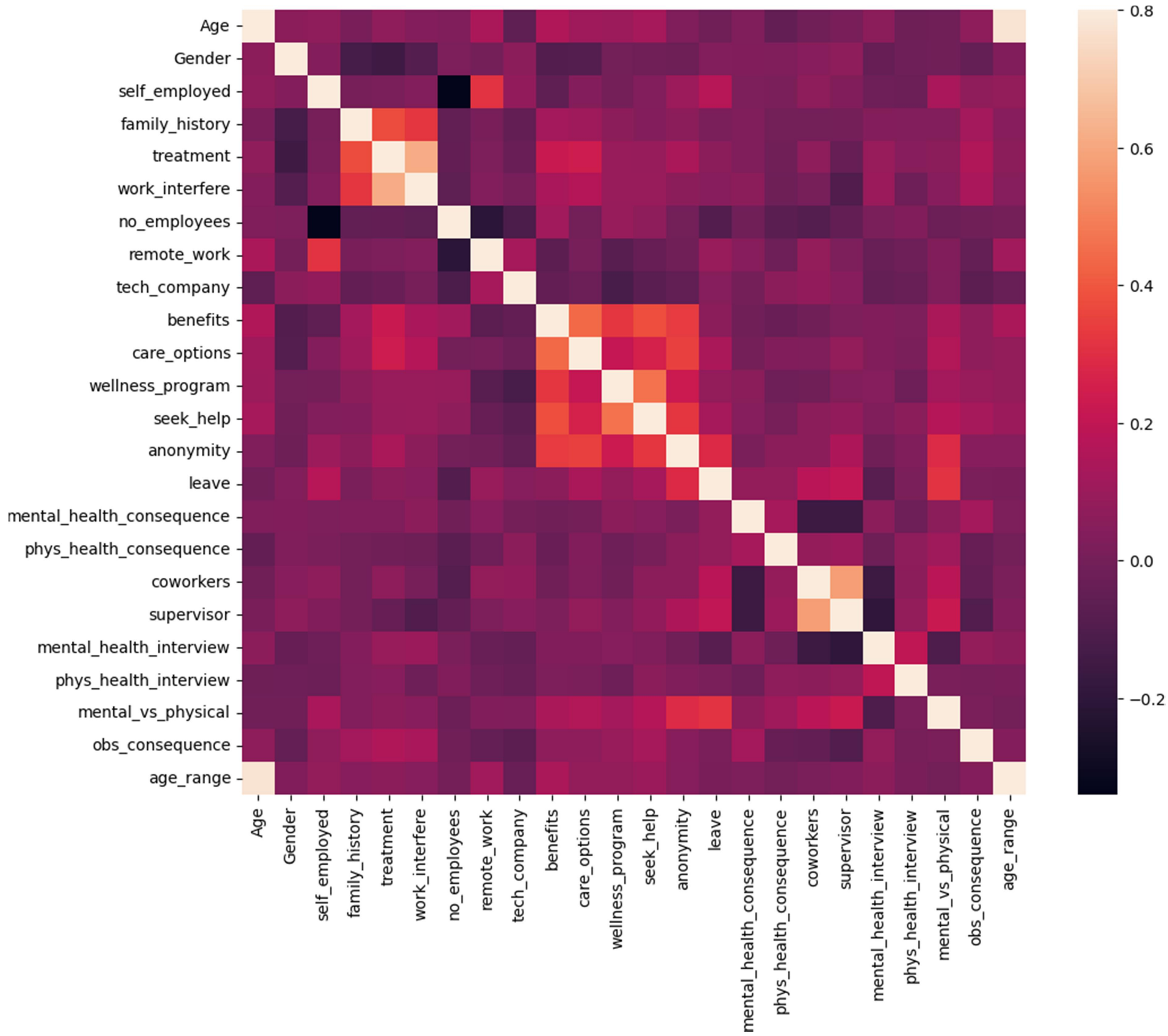
```
   obs_consequence  age_range
0                0          2
1                0          2
2                0          2
3                1          2
4                0          2

[5 rows x 24 columns]
```

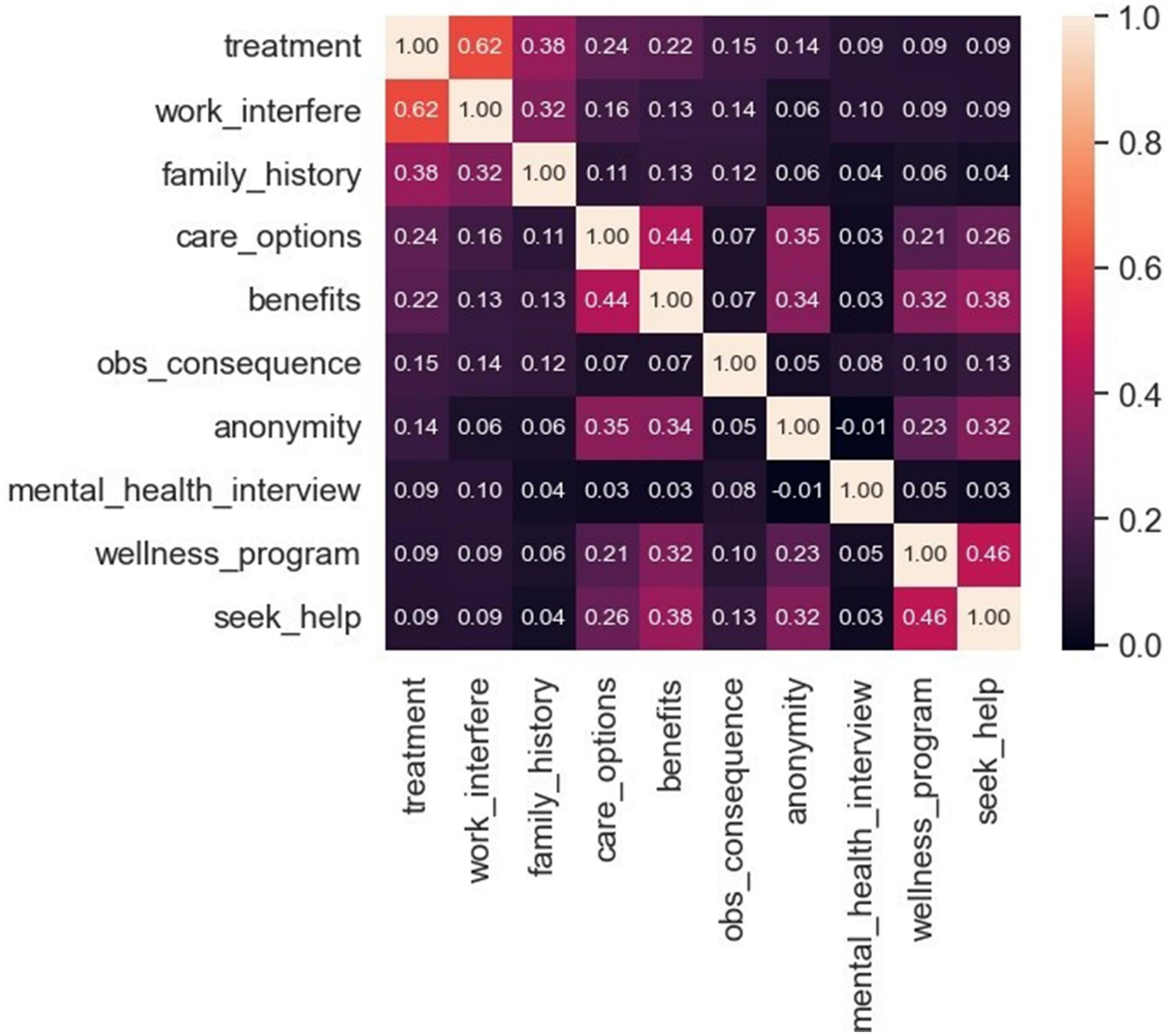
```
#missing data

total = train_df.isnull().sum().sort_values(ascending=False)
percent =

(train_df.isnull().sum()/train_df.isnull().count()).sort_values(ascend
ing=False)

missing_data = pd.concat([total, percent], axis=1, keys=['Total',
'Percent'])

missing_data.head(20)
print(missing_data)
```

```
                           Total  Percent
Age                            0      0.0
Gender                         0      0.0
obs_consequence                0      0.0
mental_vs_physical             0      0.0
phys_health_interview          0      0.0
mental_health_interview        0      0.0
supervisor                     0      0.0
coworkers                      0      0.0
phys_health_consequence        0      0.0
mental_health_consequence      0      0.0
leave                          0      0.0
anonymity                      0      0.0
seek_help                      0      0.0
wellness_program               0      0.0
care_options                   0      0.0
benefits                       0      0.0
tech_company                   0      0.0
remote_work                    0      0.0
no_employees                   0      0.0
work_interfere                 0      0.0
```

```
treatment                          0      0.0
family_history                     0      0.0
self_employed                      0      0.0
age_range                          0      0.0
```

```
#correlation matrix

corrmat = train_df.corr()

f, ax = plt.subplots(figsize=(12, 9)) sns.heatmap(corrmat, vmax=.8,
square=True);plt.show()

#treatment correlation matrix

k = 10 #number of variables for heatmap

cols = corrmat.nlargest(k, 'treatment')['treatment'].indexcm =
np.corrcoef(train_df[cols].values.T) sns.set(font_scale=1.25)
```


Age
Gender
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
age_range
0.8
0.6
0.4
0.2
0.0
−0.2




```
# Distribiution and density by Age plt.figure(figsize=(12,8))
sns.distplot(train_df["Age"], bins=24) plt.title("Distribiution and density by Age")
plt.xlabel("Age")
```

C:\Users\harsh\AppData\Local\Temp\ipykernel_21212\2516591640.py:3:UserWarning:

`distplot` is a deprecated function and will be removed in seabornv0.14.0.

Please adapt your code to use either `displot` (a figure-levelfunction with similar flexibility) or `histplot` (an axes-level function forhistograms).

```
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

    sns.distplot(train_df["Age"], bins=24)Text(0.5, 0,

'Age')
```