DAC_Phase4

October 26, 2023

1 Importing required packages

[17]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.datasets import make_classification
from sklearn.preprocessing import LabelEncoder

train_df = pd.read_csv("C:\\Users\\Student\\Downloads\\survey.csv")

print(train_df.shape)

print(train_df.describe())

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
```

2 Data Cleaning

[18]:
```
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()
train_df.head(5)
```

[18]:

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

  mental_vs_physical obs_consequence
0                Yes              No
1         Don't know              No
2                 No              No
3                 No             Yes
4         Don't know              No

[5 rows x 24 columns]
```

[19]:
```
#Handling Missing Data

defaultInt = 0
defaultString = 'NaN'
defaultFloat = 0.0

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 = []

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)
```

[19]:

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

	mental_vs_physical	obs_consequence
0	Yes	No
1	Don't know	No
2	No	No
3	No	Yes
4	Don't know	No

[5 rows x 24 columns]

[20]:
```
gender = train_df['Gender'].str.lower()

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

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)

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']
```

[21]:
```
train_df['Age'].fillna(train_df['Age'].median(), inplace = True)
# Fill with median() 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

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)
```

[22]:
```
#Getting Unique Values
train_df['self_employed'] = train_df['self_employed'].replace([defaultString],␣
 ↪'No')
print(train_df['self_employed'].unique())
```

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

[23]:
```
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"]
```

3 Preprocessing

[24]:
```
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])

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

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

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']
```

[24]:

	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	\
0	1	0	2	
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]
```

[25]:
```
for feature in train_df:
    print(labelDict['label_' + feature])
```

```
[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]
['female', 'male', 'trans']
['No', 'Yes']
['No', 'Yes']
['No', 'Yes']
["Don't know", 'Never', 'Often', 'Rarely', 'Sometimes']
['1-5', '100-500', '26-100', '500-1000', '6-25', 'More than 1000']
['No', 'Yes']
['No', 'Yes']
["Don't know", 'No', 'Yes']
['No', 'Not sure', 'Yes']
["Don't know", 'No', 'Yes']
["Don't know", 'No', 'Yes']
["Don't know", 'No', 'Yes']
["Don't know", 'Somewhat difficult', 'Somewhat easy', 'Very difficult', 'Very
easy']
['Maybe', 'No', 'Yes']
['Maybe', 'No', 'Yes']
['No', 'Some of them', 'Yes']
['No', 'Some of them', 'Yes']
['Maybe', 'No', 'Yes']
['Maybe', 'No', 'Yes']
["Don't know", 'No', 'Yes']
['No', 'Yes']
['0-20', '21-30', '31-65', '66-100']
```

4 Checking Null values

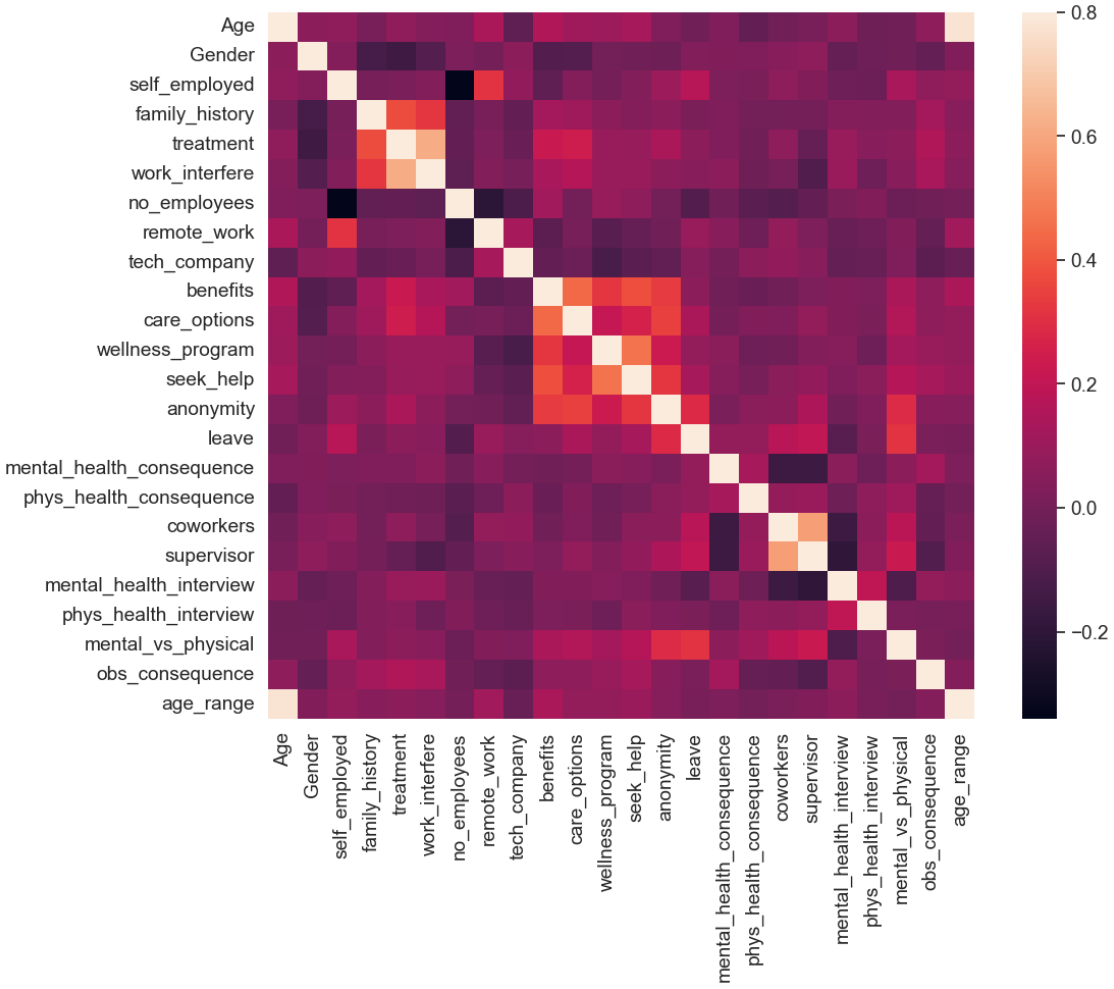
[26]:
```
total = train_df.isnull().sum().sort_values(ascending=False)
percent = (train_df.isnull().sum()/train_df.isnull().count()).
  ↪sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
print(missing_data)
```

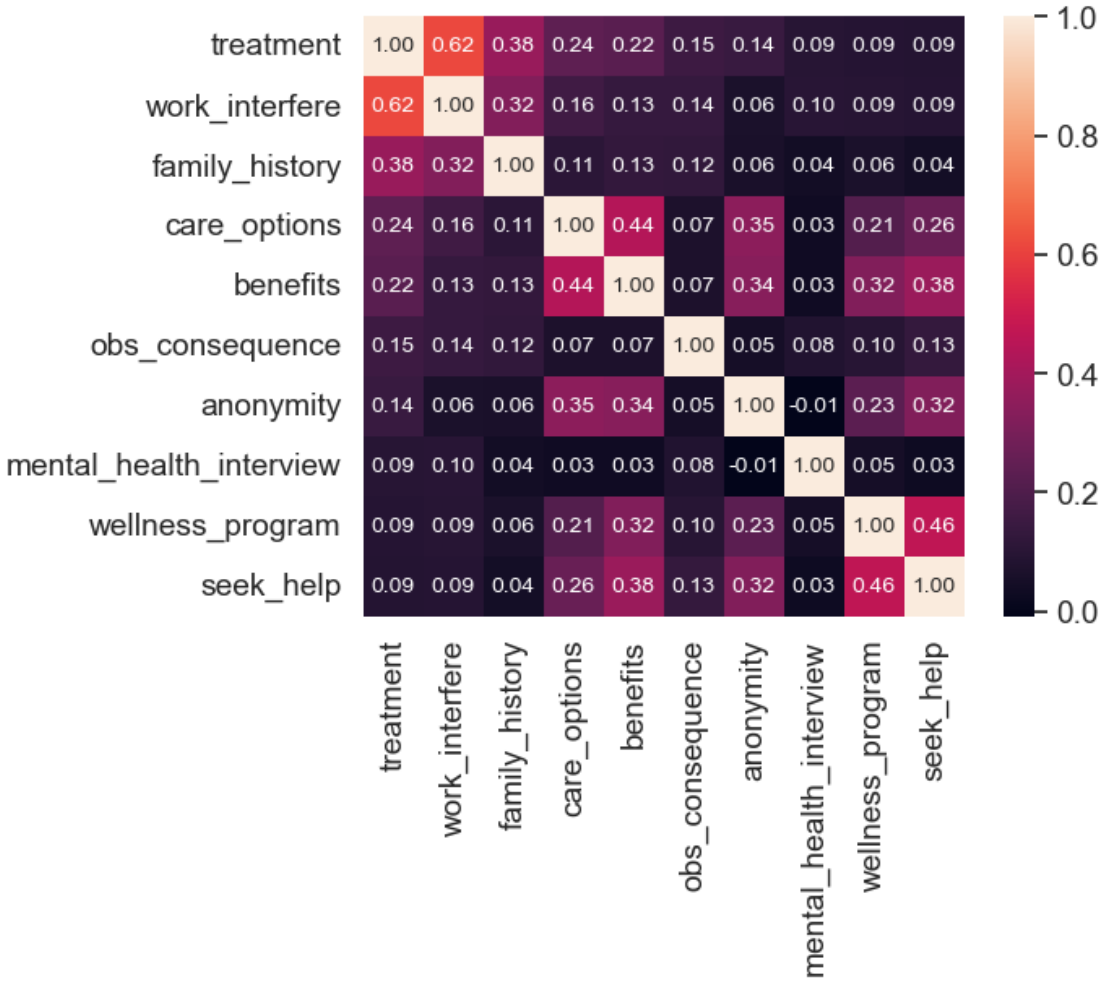
```
                           Total  Percent
Age                            0      0.0
Gender                         0      0.0
obs_consequence                0      0.0
mental_vs_physical             0      0.0
phys_health_interview          0      0.0
mental_health_interview        0      0.0
supervisor                     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
```

5 Visualization of Correlation

[27]:
```
corrmat = train_df.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
plt.show()

k = 10
cols = corrmat.nlargest(k, 'treatment')['treatment'].index
cm = np.corrcoef(train_df[cols].values.T)
sns.set(font_scale=1.25)
```

```
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f',␣
 ↪annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values)
plt.show()
```





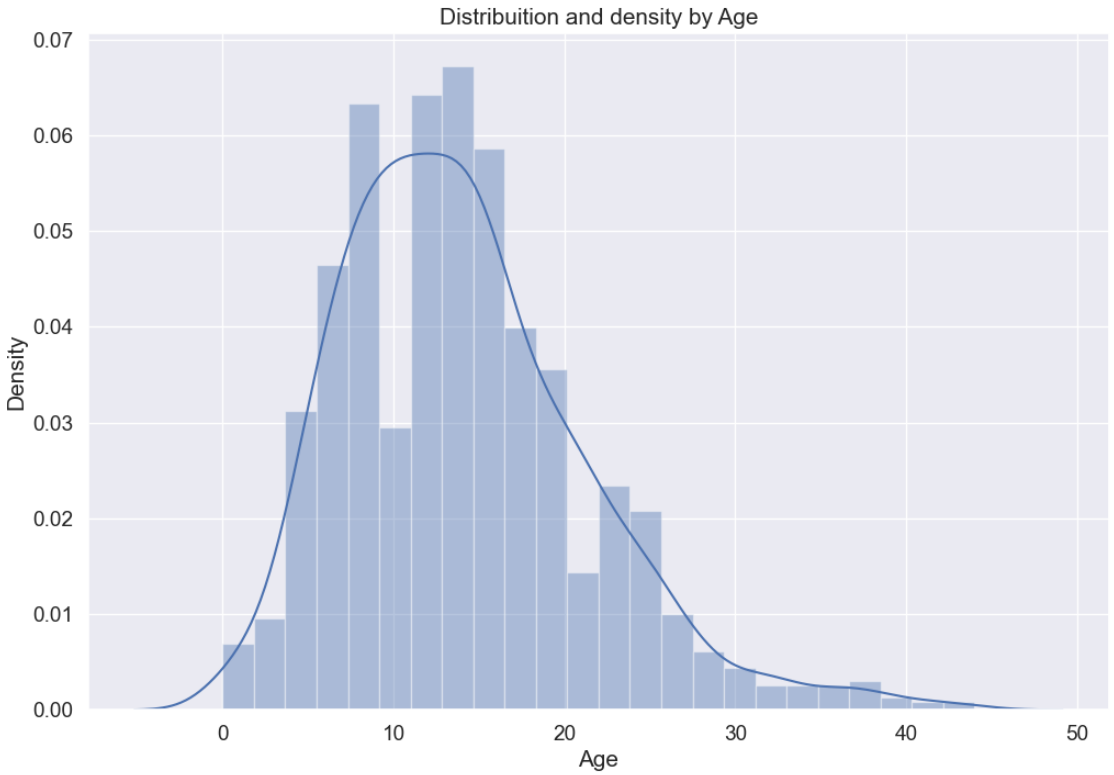
6 Visualization

[28]:
```
plt.figure(figsize=(12,8))
sns.distplot(train_df["Age"], bins=24)
plt.title("Distribuition and density by Age")
plt.xlabel("Age")
```

```
C:\Users\Student\AppData\Local\Temp\ipykernel_7084\3111768160.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

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

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

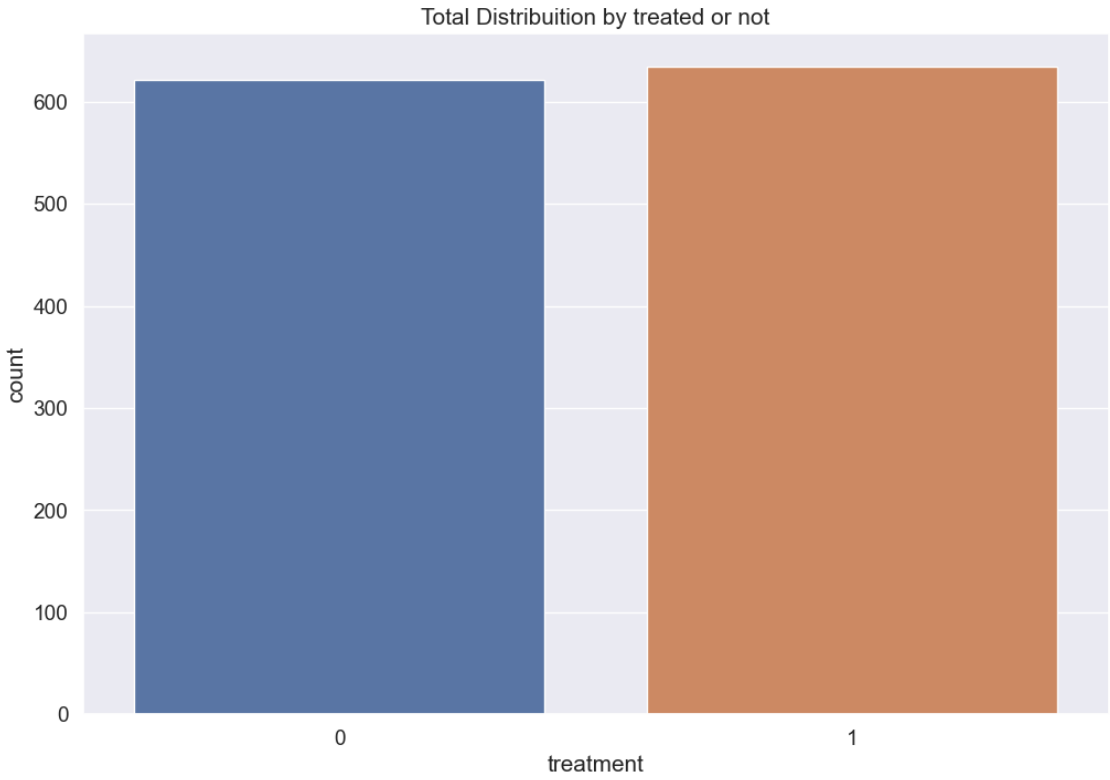
```
sns.distplot(train_df["Age"], bins=24)
```

[28]: Text(0.5, 0, 'Age')

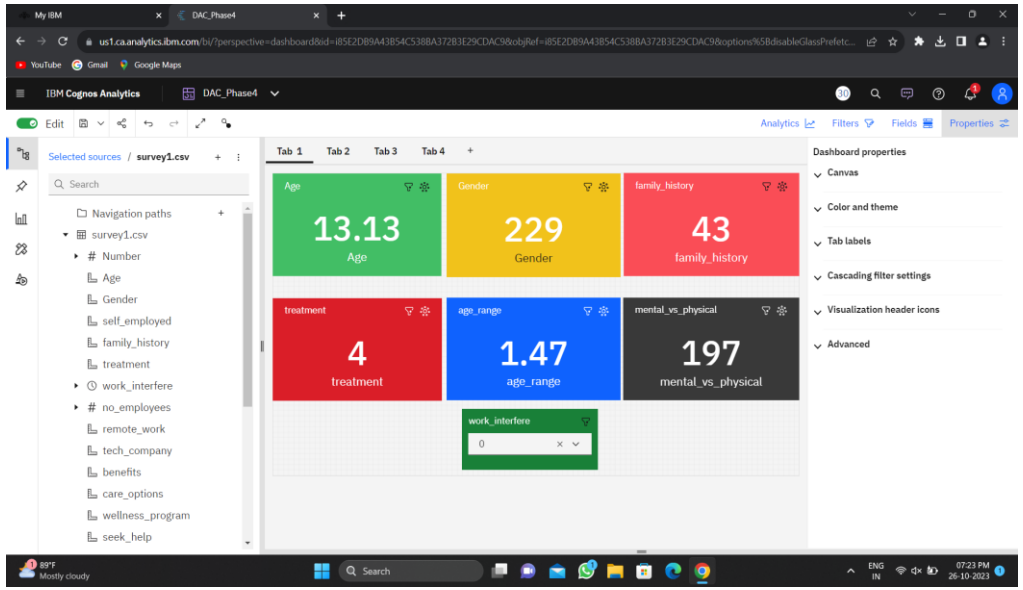


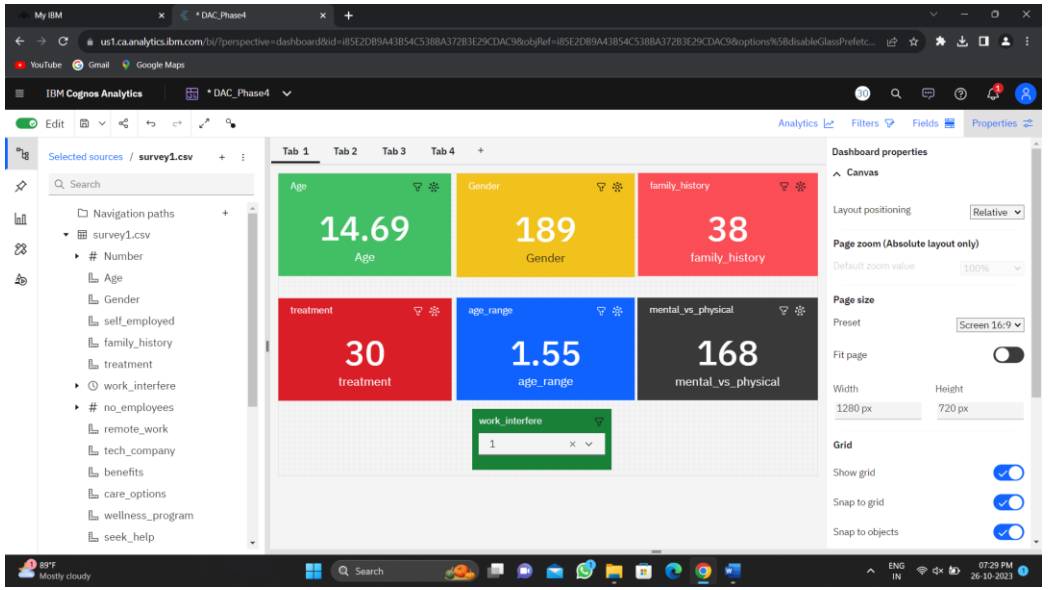
[29]:
```
plt.figure(figsize=(12,8))
labels = labelDict['label_Gender']
g = sns.countplot(x="treatment", data=train_df)

plt.title('Total Distribuition by treated or not')
```

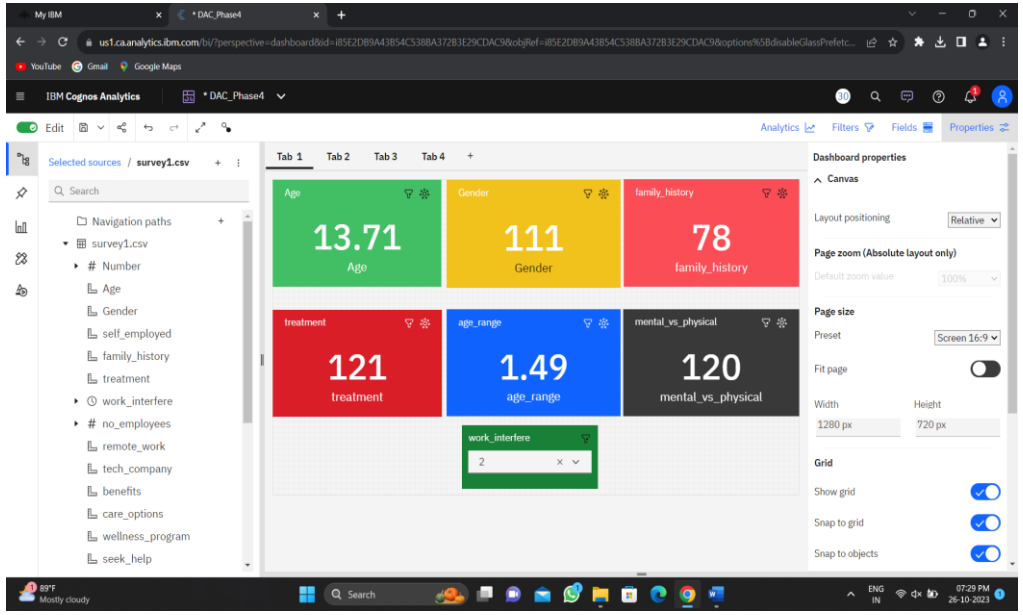
[29]: Text(0.5, 1.0, 'Total Distribuition by treated or not')

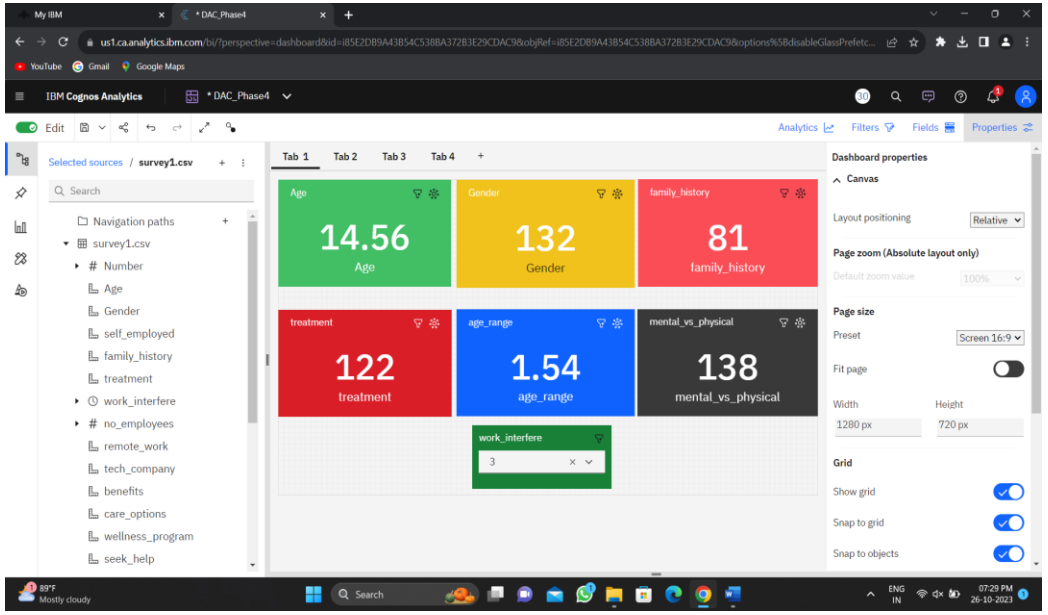

Total Distribuition by treated or not
600
500
400
300
200
100
0
count
0
1
treatment

1.Data Counts based on work_interfere:

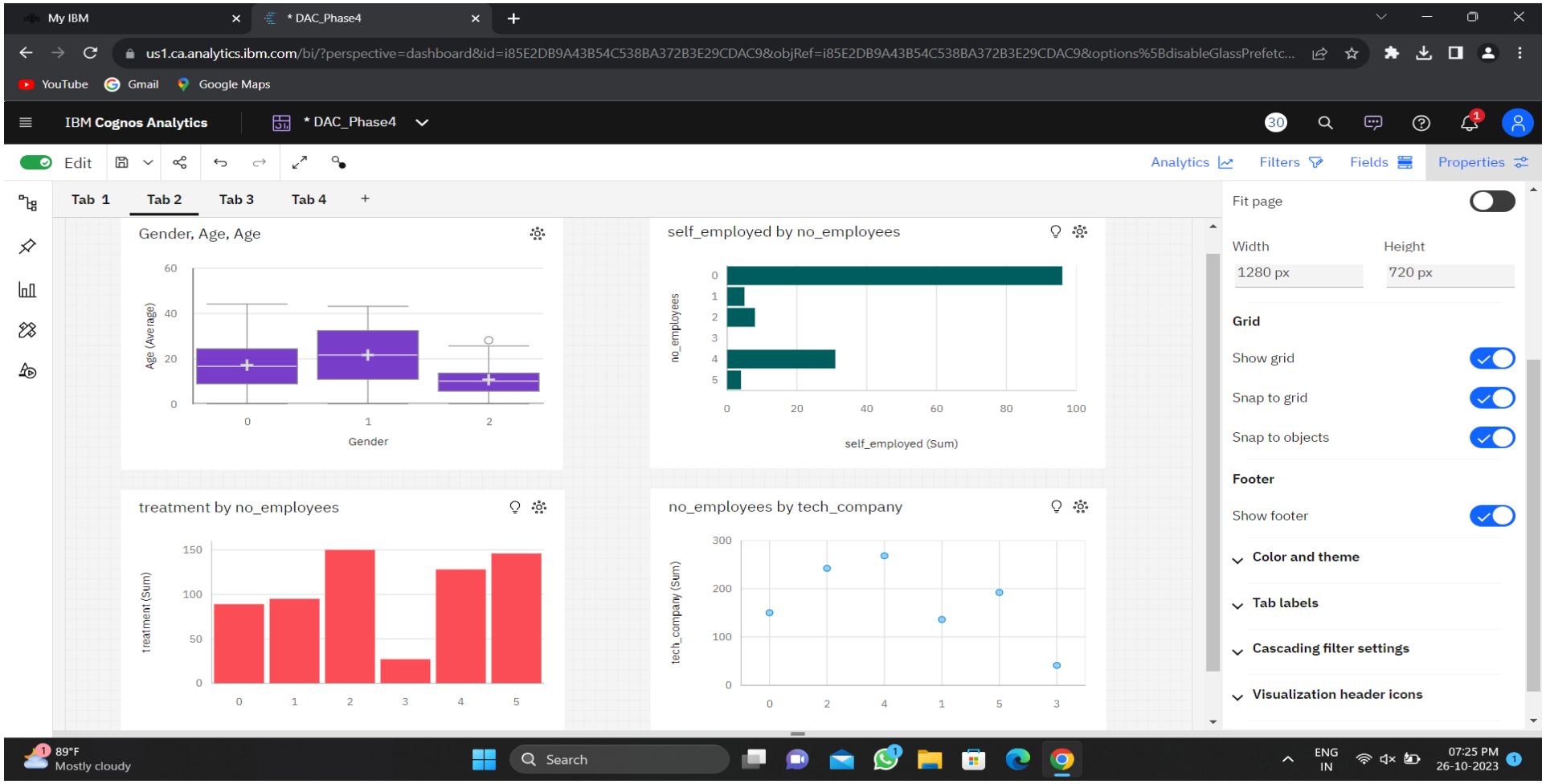




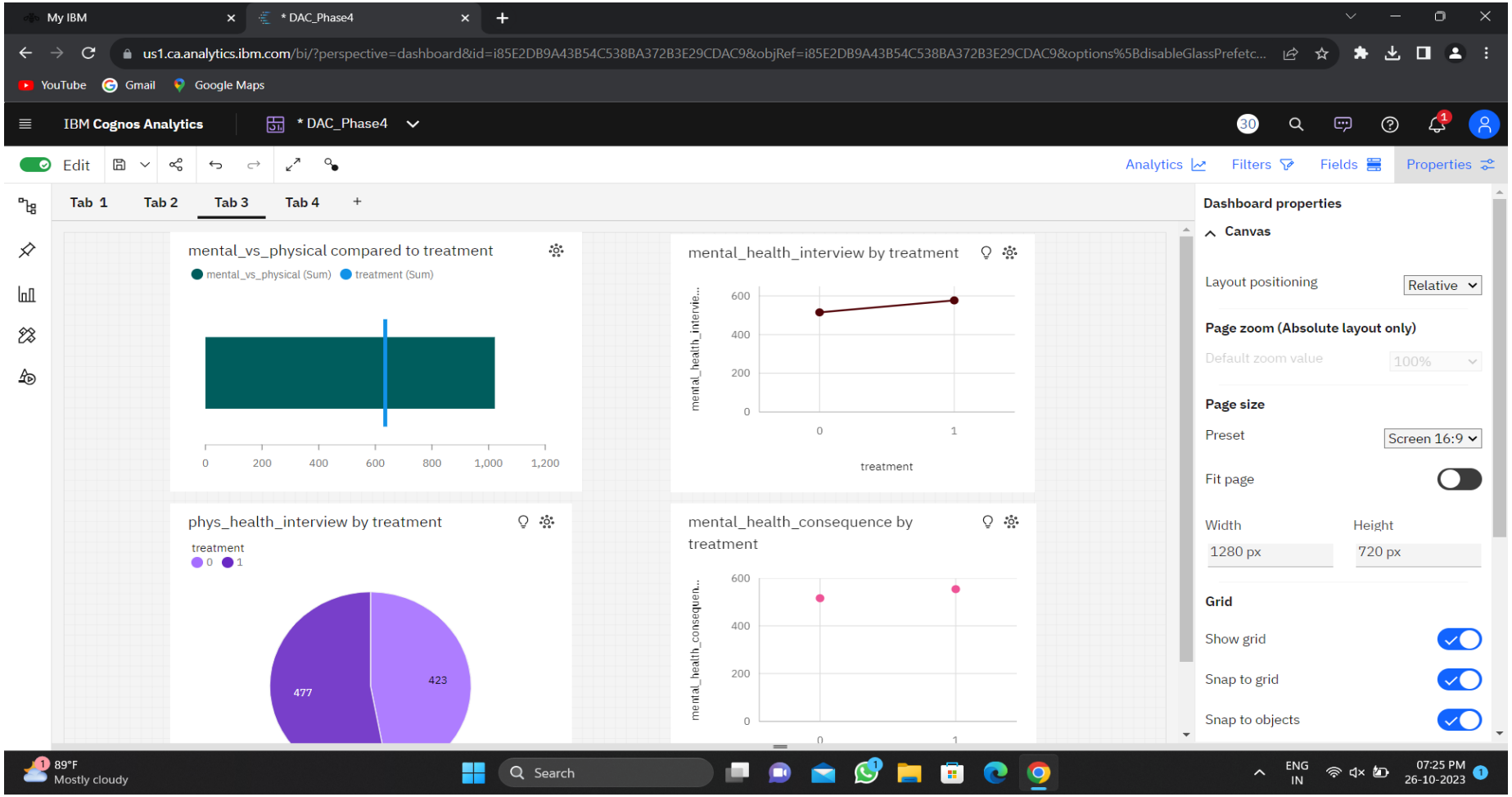




2.Visualization of outliers:



3.Visualisation of Attributes with Treatment:



4.Visualisation of Correlation: