Public Health Awareness

Problem Definition:

Public health awareness plays a crucial role in preventing diseases, promoting healthy behaviors, and ensuring timely access to healthcare services. However, there are persistent challenges in effectively disseminating health information and fostering awareness among diverse populations. This data analysis project aims to identify key issues, patterns, and opportunities for enhancing public health awareness.

Abstract:

This data analysis project, powered by IBM Cognos, aims to address the multifaceted challenges surrounding public health awareness in [Specify Region/Population]. Leveraging the advanced analytics capabilities of IBM Cognos, the study will systematically assess existing awareness levels, pinpoint demographic disparities, and scrutinize the efficacy of diverse information dissemination channels. With a particular focus on cultural and linguistic considerations, the project will utilize IBM Cognos to tailor health communication strategies to the unique needs of diverse populations. The analytics platform will facilitate an in-depth exploration of the prevalence and impact of health-related misinformation within the community, offering actionable insights for corrective measures. Additionally, the study will employ IBM Cognos to analyze the intricate relationship between public health awareness and healthcare access/utilization, identifying barriers and proposing evidence-based interventions. The anticipated outcomes encompass targeted recommendations for interventions, culturally sensitive communication strategies, and enhancements in overall healthcare access, all achieved through the sophisticated analytics capabilities of IBM Cognos. This project endeavors to empower stakeholders with actionable insights for fostering a healthier community in [Specify Region/Population].

Requried tools:

1. **IBM Cognos Analytics:** For advanced analytics, reporting, and dashboard creation.
2. **Python (Pandas, NumPy, SciPy):** For statistical analysis and advanced data manipulation.
3. **Jupyter Notebooks:** Interactive data analysis and visualization using Python.
4. **Tableau:** Creating interactive and visually appealing dashboards.
5. **R (ggplot2):** Programming language for statistical graphics and data visualization.
6. **OpenRefine:** For advanced data cleaning and transformation tasks.
7. **MySQL or PostgreSQL:** Relational databases for structured data storage.
8. **Scikit-Learn (Python):** For machine learning tasks, if applicable.
9. **QGIS:** For geospatial data analysis and visualization.
10. **Git:** Version control for tracking changes in your code and collaborating with others.

Problem Definition and Design Thinking:

Problem Definition involves precisely articulating challenges to guide effective solutions. Design Thinking is a human-centric approach that fosters empathy, ideation, prototyping, and testing, ensuring solutions align with user needs. Integrating these processes enhances problem-solving by cultivating a deep understanding of issues and iteratively creating solutions that resonate with end-users.

DataSet link : https://www.kaggle.com/datasets/osmi/mental-health-in-tech-survey

Code :

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly as pio
import matplotlib.colors as mcolors
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from IPython.display import display
import ipywidgets as widgets
from fuzzywuzzy import fuzz
import scipy as misc
from category_encoders.ordinal import OrdinalEncoder
import plotly
plotly.offline.init_notebook_mode (connected = True)
palette = ['#87CEEB','#B0E0E6','#C1F8C1', '#C1F8D8', '#D8BFD8','#8D4C66',
'#FFB6C1','#F0B2B2','#F8D8C1', '#FFFACD', '#D2B48C','#d6b39f', '#c48d86',
'#ad6b75']
palette1 = ['#8D4C66', '#AD6B75','#FFB6C1','#F0B2B2', '#C1F8D8',
'#D8BFD8','#E0B0FF','#FFB6C1','#F0B2B2','#F8D8C1', '#FFFACD',
'#D2B48C','#d6b39f', '#c48d86']
df=pd.read_csv('/kaggle/input/mental-health-in-tech-survey/survey.csv')
df.head(10).style.applymap(lambda x : "background-color: #C1F8D8")
print(f'DataTypes in given dataset: \n{df.dtypes}')
df.describe().style.applymap(lambda x : "background-color: #C1F8D8")
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])
fig.add_trace(go.Pie(labels=df['state'].loc[df['treatment'] ==
'Yes'].value_counts().index.to_list()[:10], values=df['state'].loc[df['treatment'] ==
'Yes'].value_counts()[:10], name="Treatment - Yes", marker=dict(colors=palette)),
              1, 1)
fig.add_trace(go.Pie(labels=df['state'].loc[df['treatment'] ==
'No'].value_counts().index.to_list()[:10], values=df['state'].loc[df['treatment'] ==
```

```
'No'].value_counts()[:10], name="Treatment - No", marker=dict(colors=palette)),
              1, 2)

fig.update_traces(hole=0.4, hoverinfo="label+percent+name")

fig.update_layout(
    title_text="State and Treatment",
    annotations=[dict(text='Yes', x=0.19, y=0.5, font_size=20, showarrow=False),
                 dict(text='No', x=0.78, y=0.5, font_size=20, showarrow=False)]
)

fig.show()
df = df.drop(columns=['state','comments', 'Timestamp'])
print('\033[1m' + 'Columns in updated Dataframe :' + '\033[0m', len(df.columns))
df['self_employed'] = df['self_employed']\
                    .fillna(pd.Series(np.random.choice(['Yes', 'No'], p=[0.117647,
0.882353], size=len(df))))

df['work_interfere'] = df['work_interfere']\
                    .fillna(pd.Series(np.random.choice(['Sometimes', 'Never', 'Rarely',
'Often']
                                                       , p=[0.467337, 0.214070, 0.173869, 0.144724],
size=len(df))))
print('\033[1m' + 'Total empty values in the Dataset :' + '\033[0m' ,
df.isnull().sum().sum())
df_encoding = df
encoder = OrdinalEncoder()
df_encoding = encoder.fit_transform(df.drop(['Age'], axis=1))
df_encoding['Age'] = df.Age
df_encoding.head(10).style.applymap(lambda x : "background-color: #C1F8D8")
corr = df_encoding.corr(method ='spearman')
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(15, 15))
cmap = sns.diverging_palette(220, 5, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap='Purples', vmax=.05, center=0,square=True,
linewidths=.4, cbar_kws={"shrink": .5})
plt.show()
cmap = mcolors.ListedColormap('#C1F8D8')
gender_values =
df.Gender.value_counts().sort_values(ascending=False).to_frame()
gender_values = gender_values.rename(columns={'Gender': 'count'})
table_gender = gender_values.style.background_gradient(cmap=cmap)
table_gender
def winsorization_outliers(df):
    out=[]
    for i in df:
        q1 = np.percentile(df , 1)
        q3 = np.percentile(df , 99)
        if i > q3 or i < q1:
            out.append(i)
```

```
    print("Outliers:",out)
    return out
outliers = winsorization_outliers(df.Age)
data_age = df.loc[~df.Age.isin(outliers)]
fig = px.violin(data_age, y="Age", x="treatment", color="Gender", box=True,
points="all", color_discrete_sequence=palette)
fig.show()
age = []
for i in df.Age:
    if (i<18) or (i>99):
        age.append(31)
    else:
        age.append(i)
df['Age'] = age

other  = ['A little about you', 'p', 'Nah', 'Enby', 'Trans-female','something kinda
male?','queer/she/they','non-binary','All','fluid', 'Genderqueer','Androgyne',
'Agender','Guy (-ish) ^_^', 'male leaning androgynous','Trans woman','Neuter',
'Female (trans)','queer','ostensibly male, unsure what that really means','trans']
male   = ['male', 'Maleleaning androgynous', 'cis Male','something kinda Male?',
'Male(CIS)','ostensibly Male, unsure what that really means','cis Male','Male','M',
'm', 'Male-ish', 'maile','Cis Male','Mal', 'Male (CIS)','Make','Male ', 'Man', 'msle','cis
male', 'Cis Man','Malr','Mail']
female = ['Female', 'Female(cis)','Trans-Female','cis-Female/femme','Trans
Female','Female(trans)', 'Female(cis)"female','Cis Female', 'F','f','Femake',
'woman','Female ','cis-female/femme','Female (cis)','femail','Woman','female']

df['Gender'].replace(to_replace = other, value = 'other', inplace=True)
df['Gender'].replace(to_replace = male, value = 'M', inplace=True)
df['Gender'].replace(to_replace = female, value = 'F', inplace=True)

print('\033[1m' + 'Unique values in updated Gender column :' + '\033[0m',
df.Gender.unique())
print('\033[1m' + 'Range of column Age (Before) :' + '\033[0m', (df.Age.min(),
df.Age.max()))
print('\033[1m' + 'Range of column Age :' + '\033[0m', (df.Age.min(), df.Age.max()))
prop = pd.DataFrame(df.Gender.value_counts(normalize=True))
prop.style.applymap(lambda x : "background-color: #C1F8D8")
a= sns.histplot(df,x='Age',hue='seek_help', palette=palette)
df_ = df.drop(['Age', 'Country'], axis=1)
buttons = []
i = 0
vis = [False] * 24

for col in df_.columns:
    vis[i] = True
    buttons.append({'label' : col,
            'method' : 'update',
            'args'   : [{'visible' : vis},
            {'title'  : col}] })
    i+=1
```

```
    vis = [False] * 24

fig = go.Figure()
for col in df_.columns:
    fig.add_trace(go.Pie(
            values = df_[col].value_counts(),
            labels = df_[col].value_counts().index,
            title = dict(text = 'Distribution of {}'.format(col),
                         font = dict(size=18, family = 'Times New Roman'),
                         ),
            hole = 0.4,
            hoverinfo='label+percent',))
fig.update_traces(hoverinfo='label+percent',
                  textinfo='label+percent',
                  textfont_size=12,
                  opacity = 0.8,
                  showlegend = False,
                  marker = dict(colors = sns.color_palette(palette1).as_hex(),
                           line=dict(color='#000000', width=1)))

fig.update_layout(margin=dict(t=0, b=0, l=0, r=0),
   updatemenus = [dict(
                      type = 'dropdown',
                      x = 1.15,
                      y = 0.85,
                      showactive = True,
                      active = 0,
                      buttons = buttons)],
                annotations=[
                           dict(text = "<b>Choose<br>Column<b> : ",
                           showarrow=False,
                           x = 1.06, y = 0.92, yref = "paper", align = "left")])
for i in range(1,22):
    fig.data[i].visible = False
fig.show()
male   = df[df.Gender == 'M'].drop(['Gender', 'Age', 'Country'], axis=1)
female = df[df.Gender == 'F'].drop(['Gender', 'Age', 'Country'], axis=1)
other  = df[df.Gender == 'other'].drop(['Gender', 'Age', 'Country'], axis=1)
  font = dict(size=18, family = 'Times New Roman'),
                         ),
            hole = 0.4,
            hoverinfo='label+percent',),1,1)
for col in female.columns:
    fig.add_trace(go.Pie(
            values = female[col].value_counts(),
            labels = female[col].value_counts().index,
            title = dict(text = 'Female distribution<br>of {}'.format(col),
                         font = dict(size=18, family = 'Times New Roman'),
                         ),
            hole = 0.4,
            hoverinfo='label+percent',),1,2)
```

```
 fig.update_traces(hoverinfo='label+percent',
                  textinfo='label+percent',
                  textfont_size=14,
                  opacity = 0.7,
                  showlegend = False,
                  marker = dict(colors = sns.color_palette(palette1).as_hex(),
                           line=dict(color='#000000', width=1)))
 fig.update_traces(row=1, col=2, hoverinfo='label+percent',
                  textinfo='label+percent',
   textfont_size=12,
                  opacity = 0.7,
                  showlegend = False,
                  marker = dict(colors = sns.color_palette(palette).as_hex(),
                           line=dict(color='#000000', width=1)))
 fig.update_layout(margin=dict(t=0, b=0, l=0, r=0),
                  font_family   = 'Times New Roman',
                  updatemenus = [dict(
                      type = 'dropdown',
                      x = 0.62,
                      y = 0.91,
                      showactive = True,
                      active = 0,
                      buttons = buttons)],
                annotations=[
                           dict(text = "<b>Choose<br>Column<b> : ",
                               font = dict(size = 14),
                           showarrow=False,
                           x = 0.5, y = 1, yref = "paper", align = "left")])
 for i in range(1,42):
     fig.data[i].visible = False
 fig.data[21].visible = True
fig.show()
```