DSC550-T301

Chitramoy Mukherjee

Final Project Milestone-1

Analyze Mental health disorder in Tech Companies

Date: 1/14/2024

Introduction:

In recent years, the tech industry has experienced rapid growth and innovation, bringing about numerous opportunities and challenges. While technological advancements have transformed the way we work, they have also introduced new stressors that can impact the mental health of individuals working in this sector. Recognizing the importance of mental health in the workplace, this project aims to analyze mental health disorders within tech companies using Python.Mental health affects your emotional, psychological and social well-being.Mental health is a key factor todetermine the productivity of the employee in any industry and as a whole total performance of the company. If someone is not mentally fit, he can’t produce the expected output what he is capable of and it also impacts his co-workers performance and impacts the work environment.

Objective:

The primary objective of this project is to gain insights into the prevalence of mental health disorders among employees in the tech industry. By leveraging Python for data analysis, we aim to explore patterns, trends, and potential factors contributing to mental health issues. The analysis will be based on a dataset collected from surveys conducted within tech companies, covering a range of variables related to mental health. This sort of analysis helps the employer to identify and support an individual who may be experiencing a mental health or substance use concern or crisis and connect them with the appropriate employee resources. This allows employer to recognize the signs of someone who maybe struggling and teaches them the skills to know when to reach out and what resources are available.Organizations that incorporate mental health awareness help to create a healthy and productive work environment that reduces the stigma associated with mental illness, increases the organizations mental health literacy and teaches the skills to safely and responsibly respond to a co-workers mental health concern.

Key Components:

1. **Data Collection:**
    - Gather a comprehensive dataset from tech companies, including information on employee demographics, work-related factors, and self-reported mental health

conditions.

2. **Data understanding and Preprocessing:**

    - Clean and preprocess the dataset to handle missing values, outliers, and ensure data quality. Transform categorical variables and standardize formats for analysis.

3. **Exploratory Data Analysis (EDA):**

    - Utilize Python libraries such as Pandas, Matplotlib, and Seaborn to conduct exploratory data analysis. Visualize distributions, correlations, and trends in mental health-related variables.

4. **Statistical Analysis:**

    - Apply statistical methods to identify significant factors influencing mental health disorders. Conduct hypothesis testing and regression analysis to establish relationships.

5. **Machine Learning Modeling:**

    - Develop machine learning models to predict the likelihood of mental health disorders based on relevant features. Evaluate model performance and interpret results.

6. **Recommendations and Insights:**

    - Provide actionable insights and recommendations for tech companies to improve mental health support for their employees.

Key benefits from the outcome of the project:

By the end of this project, we aim to contribute valuable insights that can inform both employers and employees about mental health in the tech industry. This analysis can serve as a foundation for fostering a healthier and more supportive work environment both from employee and employer perspective. Employers can also offer robust benefit packages to support employees who go through mental health issues. That includes Employee Assistance Programs, Wellness programs that focus on mental and physical health, Health and Disability Insurance or flexible working schedules or time off policies.

Below are the key benefits of this analysis from Employer and Employee perspective :

1. Employee Well-being and Productivity.
2. Reduced Healthcare Costs.
3. Enhanced Employee Morale.
4. Legal Compliance and Corporate Responsibility.
5. Customized Support Programs.
6. Employee Engagement and Satisfaction Surveys.
7. Workplace Culture Improvement.

This topic is relevant to data science as we can analyze and identify the factors/variables that impacts the mental health and justify the relations between variables which is closely related to determine the mental health of employees.We can create a model and feed data into it to identify the employees mental health in the company and provide directions to them to overcome the situation

In [2]:
```
import warnings
warnings.filterwarnings('ignore')

# Required python basic libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk import download
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectoriz
import nltk
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier

# PreProcessing
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import SimpleImputer, IterativeImputer
from sklearn.preprocessing import MinMaxScaler
from category_encoders import BinaryEncoder

# Splitting Data
from sklearn.model_selection import train_test_split, StratifiedKFold, cro

# Modeling
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, recall_score
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, G
from xgboost.sklearn import XGBClassifier

# Tuning
from sklearn.model_selection import GridSearchCV


from os.path import basename, exists


def download(url):
    filename = basename(url)
    if not exists(filename):
        from urllib.request import urlretrieve
```

```
        local, _ = urlretrieve(url, filename)
        print("Downloaded " + local)

### Reading the labeledTrainData.tsv file into DataFrame
df = pd.read_csv("C:\\Users\\14024\\OneDrive\\Desktop\\MS-DSC\\DSC-550\Wee

# Display the first few rows of the DataFrame to ensure it's loaded proper
print(df)

df.columns
```

```
                Timestamp  Age  Gender         Country state self_employe
d  \
0     2014-08-27 11:29:31   37  Female   United States    IL            Na
N
1     2014-08-27 11:29:37   44       M   United States    IN            Na
N
2     2014-08-27 11:29:44   32    Male          Canada   NaN            Na
N
3     2014-08-27 11:29:46   31    Male  United Kingdom   NaN            Na
N
4     2014-08-27 11:30:22   31    Male   United States    TX            Na
N
...                   ...  ...     ...             ...   ...
...
1254  2015-09-12 11:17:21   26    male  United Kingdom   NaN             N
o
1255  2015-09-26 01:07:35   32    Male   United States    IL             N
o
1256  2015-11-07 12:36:58   34    male   United States    CA             N
o
1257  2015-11-30 21:25:06   46       f   United States    NC             N
o
1258  2016-02-01 23:04:31   25    Male   United States    IL             N
o

     family_history treatment work_interfere    no_employees  ...  \
0                No       Yes          Often            6-25  ...
1                No        No         Rarely  More than 1000  ...
2                No        No         Rarely            6-25  ...
3               Yes       Yes          Often          26-100  ...
4                No        No          Never         100-500  ...
...             ...       ...            ...             ...  ...
1254             No       Yes            NaN          26-100  ...
1255            Yes       Yes          Often          26-100  ...
1256            Yes       Yes      Sometimes  More than 1000  ...
1257             No        No            NaN         100-500  ...
1258            Yes       Yes      Sometimes          26-100  ...

                   leave mental_health_consequence phys_health_consequenc
e  \
0          Somewhat easy                        No                      N
o
1             Don't know                     Maybe                      N
o
2     Somewhat difficult                        No                      N
o
3     Somewhat difficult                       Yes                     Ye
s
4             Don't know                        No                      N
o
...                  ...                       ...
...
1254       Somewhat easy                        No                      N
o
1255  Somewhat difficult                        No                      N
o
1256  Somewhat difficult                       Yes                     Ye
```

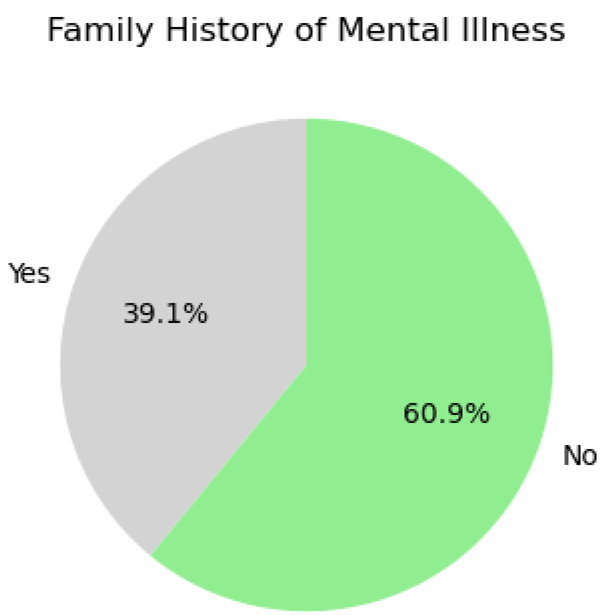
```
s
1257         Don't know                        Yes                    N
o
1258         Don't know                      Maybe                    N
o

         coworkers    supervisor mental_health_interview  \
0     Some of them           Yes                      No
1               No            No                      No
2              Yes           Yes                     Yes
3     Some of them            No                   Maybe
4     Some of them           Yes                     Yes
...            ...           ...                     ...
1254  Some of them  Some of them                      No
1255  Some of them           Yes                      No
1256            No            No                      No
1257            No            No                      No
1258  Some of them            No                      No

     phys_health_interview mental_vs_physical obs_consequence comments
0                    Maybe                Yes              No      NaN
1                       No         Don't know              No      NaN
2                      Yes                 No              No      NaN
3                    Maybe                 No             Yes      NaN
4                      Yes         Don't know              No      NaN
...                    ...                ...             ...      ...
1254                    No         Don't know              No      NaN
1255                    No                Yes              No      NaN
1256                    No                 No              No      NaN
1257                    No                 No              No      NaN
1258                    No         Don't know              No      NaN

[1259 rows x 27 columns]
```

Out[2]:

```
Index(['Timestamp', 'Age', 'Gender', 'Country', 'state', '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', 'coworker
s',
       'supervisor', 'mental_health_interview', 'phys_health_interview',
       'mental_vs_physical', 'obs_consequence', 'comments'],
      dtype='object')
```

In [3]:

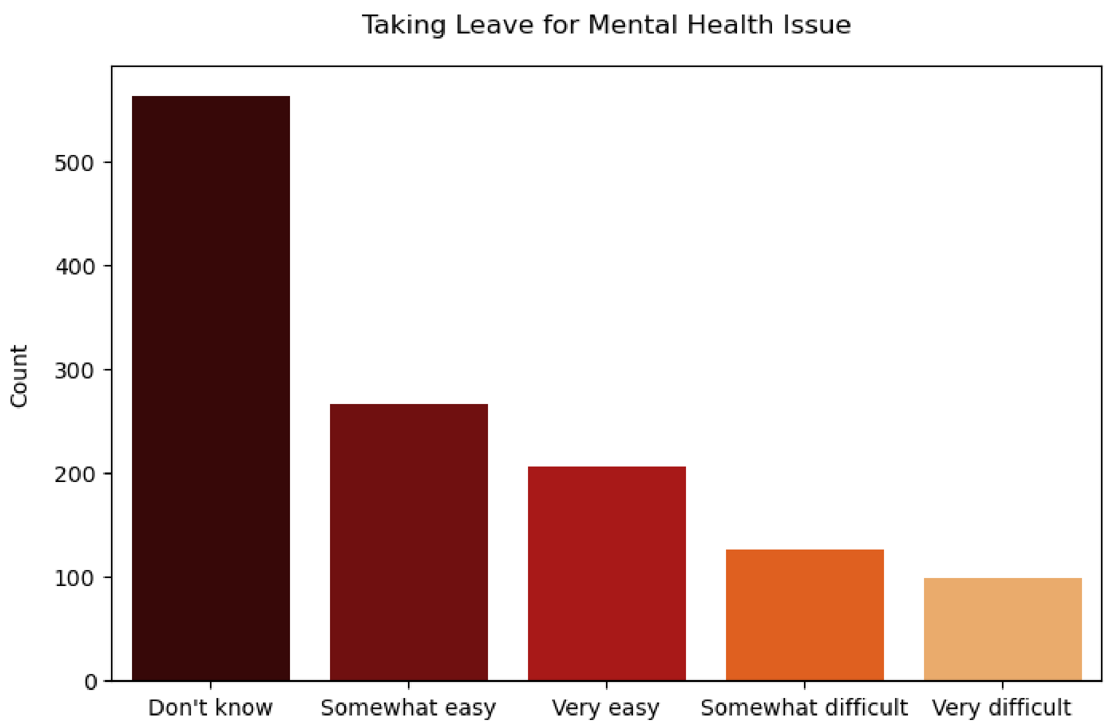
```
# Pie diagram of Family History of Mental illness
yes = len(df[df['family_history'] == 'Yes'])
no = len(df[df['family_history'] == 'No'])

count = [yes, no]
labels = ['Yes', 'No']
colors = ['lightgrey', 'lightgreen']

# Customizing the pie chart
plt.figure(figsize=(8,4))
explode = (0, 1, 1)  # Only the second slice will explode
pc = plt.pie(count, labels=labels, autopct='%1.1f%%', startangle=90, color
plt.title('Family History of Mental Illness');
```



From this, we can see that almost 40% of respondents have a family history of mental illness. According to a 2017 study by the Arctic University of Norway, it was discovered that children with parents who had a severe mental illness had up to a 50% chance of developing a mental illness, and a 32% chance of developing a severe mental illness (bipolar disorder, major depressive disorder, schizophrenia, etc). We will look further into this when performing bivariate analysis.

In [4]:

```
# Bar diagram plot of how ease to take leave due to mental health issue

df['leave'].value_counts().index
plt.figure(figsize=(8,5)) # Size of the figure

# Using value_counts(), we get the count of each answer in descending orde
# we later pass into the order parameter of the countplot, sorting the plo
order = df['leave'].value_counts().index

plt.title('Taking Leave for Mental Health Issue', pad=15);
mp = sns.countplot(x='leave', data=df, order=order, palette='gist_heat')
plt.ylabel('Count', labelpad=10)
mp.set(xlabel=None);
```

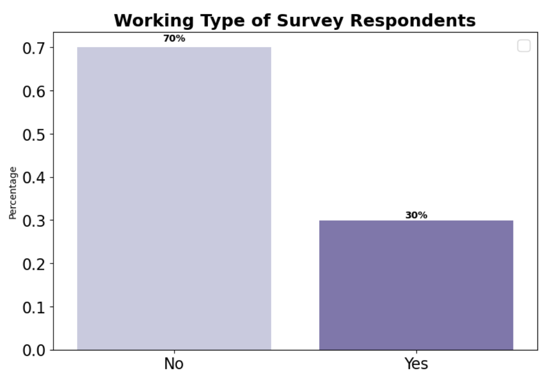


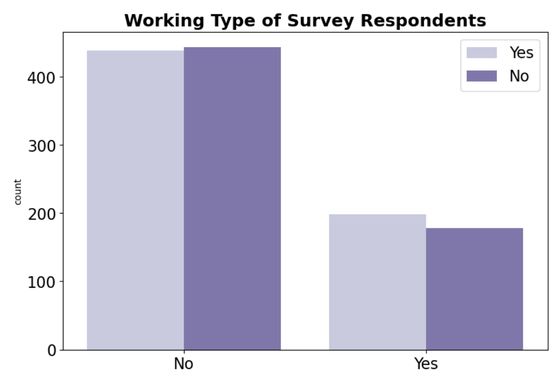
From the above plot, we can see that most respondents do not know whether they are even allowed to take leave for a mental health issue, and there are also quite a number who find it hard to do so, which may be due to the social stigma surrounding mental issues.

In [5]:

```
# Bar diagram plot of Working Type of Survey respondents
plt.figure(figsize = (20,6))
plt.subplot(1,2,1)
eda_percentage = df['remote_work'].value_counts(normalize = True).rename_a
ax = sns.barplot(x = 'remote_work', y = 'Percentage', data = eda_percentag
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.0%}', (x + width/2, y + height*1.02), ha='cente

plt.title('Working Type of Survey Respondents', fontsize=18, fontweight='b
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(fontsize=16)

plt.subplot(1,2,2)
sns.countplot(x=df['remote_work'], data = eda_percentage,  hue = df['treat
plt.title('Working Type of Survey Respondents', fontsize=18, fontweight='b
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(fontsize=16)
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose l
abel start with an underscore are ignored when legend() is called with no
argument.
```

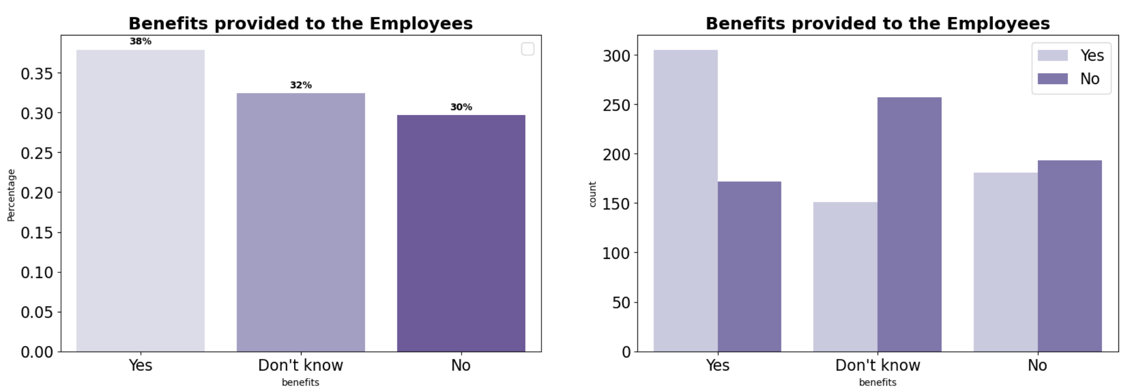




Around 70% of respondents don't work remotely, which means the biggest factor of mental health disorder came up triggered on the workplace. On the other side, it has slightly different between an employee that want to get treatment and don't want to get a treatment. The number of people who seek treatment in both the categories is more or less similar and it does not affect our target variable.

In [6]:
```
# Bar plot of benefits provided to the employees
plt.figure(figsize = (20,6))
plt.subplot(1,2,1)
eda_percentage = df['benefits'].value_counts(normalize = True).rename_axis
ax = sns.barplot(x = 'benefits', y = 'Percentage', data = eda_percentage,
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.0%}', (x + width/2, y + height*1.02), ha='cente

plt.title('Benefits provided to the Employees', fontsize=18, fontweight='b
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(fontsize=16)

plt.subplot(1,2,2)
sns.countplot(x=df['benefits'], data = eda_percentage,  hue = df['treatmen
plt.title('Benefits provided to the Employees', fontsize=18, fontweight='b
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(fontsize=16)
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose l
abel start with an underscore are ignored when legend() is called with no
argument.
```



We see that around 38% of the respondents said that their employer provided them mental health benefits, whereas a significant number (32%) of them didn't even know whether they were provided this benefit. Coming to the second graph, we see that for the people who YES said to mental health benefits, around 63% of them said that they were seeking medical help. Surprisingly, the people who said NO for the mental health benefits provided by the company, close to 45% of them who want to seek mental health treatment.

Final Project Milestone-2

Date: 1/31/2024

Drop any features that are not useful for your model building and explain why they are not useful.

In [7]:
```
# Visualize the data and identify the non-null values
df.info()
```

```
<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
```

In [8]:
```
#missing data
total = df.isnull().sum().sort_values(ascending=False)
percent = (df.isnull().sum()/df.isnull().count()).sort_values(ascending=Fa
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent
missing_data.head(20)
print(missing_data)
```

```
                           Total   Percent
comments                    1095  0.869738
state                        515  0.409055
work_interfere               264  0.209690
self_employed                 18  0.014297
seek_help                      0  0.000000
obs_consequence                0  0.000000
mental_vs_physical             0  0.000000
phys_health_interview          0  0.000000
mental_health_interview        0  0.000000
supervisor                     0  0.000000
coworkers                      0  0.000000
phys_health_consequence        0  0.000000
mental_health_consequence      0  0.000000
leave                          0  0.000000
anonymity                      0  0.000000
Timestamp                      0  0.000000
wellness_program               0  0.000000
Age                            0  0.000000
benefits                       0  0.000000
tech_company                   0  0.000000
remote_work                    0  0.000000
no_employees                   0  0.000000
treatment                      0  0.000000
family_history                 0  0.000000
Country                        0  0.000000
Gender                         0  0.000000
care_options                   0  0.000000
```

Justification for the features dropped which might not be useful for model building.

Timestamp: This column might not provide useful information for predicting mental health issues. The timestamp is usually used for tracking when the survey was taken, which is not relevant for the analysis.

Comments: This column is likely to contain free-form text responses, which can be challenging to process and analyze. For simplicity and to focus on structured data, it's common to exclude text-based features. Also as per the data 1095 outof 1258 rows is Null. So this won't be useful for model.

State: If the dataset is not specifically focused on regional analysis, the state column may not be relevant for predicting mental health outcomes. It could be dropped unless there's a specific reason to consider geographical location.Also 515 records are null out of 1258 rows.

In [9]:

```
#dealing with missing data
#Let’s get rid of the variables "Timestamp",“comments”, “state” just to ma
train_df = df.drop(['comments'], axis= 1)
train_df = df.drop(['state'], axis= 1)
train_df = df.drop(['Timestamp'], axis= 1)

train_df.isnull().sum().max() #just checking that there's no missing data
train_df.head(5)
```

Out[9]:

	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	n
0	37	Female	United States	IL	NaN	No	Yes	Often	
1	44	M	United States	IN	NaN	No	No	Rarely	
2	32	Male	Canada	NaN	NaN	No	No	Rarely	
3	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	
4	31	Male	United States	TX	NaN	No	No	Never	

5 rows × 26 columns

Deal with missing data

In [10]:

```
# 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',
                  'no_employees', 'remote_work', 'tech_company', 'anonymity
                  'phys_health_consequence', 'coworkers', 'supervisor', 'me
                  'mental_vs_physical', 'obs_consequence', 'benefits', 'car
                  '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)
```

```
Error: Feature state not recognized.
Error: Feature comments not recognized.
```

Out[10]:

	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	n
0	37	Female	United States	IL	NaN	No	Yes	Often	
1	44	M	United States	IN	NaN	No	No	Rarely	
2	32	Male	Canada	NaN	NaN	No	No	Rarely	
3	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	
4	31	Male	United States	TX	NaN	No	No	Never	

5 rows × 26 columns

In [11]:
```
#missing data
total = train_df.isnull().sum().sort_values(ascending=False)
percent = (train_df.isnull().sum()/train_df.isnull().count()).sort_values(
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent
missing_data.head(20)
print(missing_data)
```

```
                           Total   Percent
comments                    1095  0.869738
state                        515  0.409055
seek_help                      0  0.000000
obs_consequence                0  0.000000
mental_vs_physical             0  0.000000
phys_health_interview          0  0.000000
mental_health_interview        0  0.000000
supervisor                     0  0.000000
coworkers                      0  0.000000
phys_health_consequence        0  0.000000
mental_health_consequence      0  0.000000
leave                          0  0.000000
anonymity                      0  0.000000
Age                            0  0.000000
Gender                         0  0.000000
care_options                   0  0.000000
benefits                       0  0.000000
tech_company                   0  0.000000
remote_work                    0  0.000000
no_employees                   0  0.000000
work_interfere                 0  0.000000
treatment                      0  0.000000
family_history                 0  0.000000
self_employed                  0  0.000000
Country                        0  0.000000
wellness_program               0  0.000000
```

Perform any data extraction/selection steps and Transform features if necessary.

In [12]:

```
# Selecting specific columns of interest
selected_columns = ['Age', 'Gender', 'Country', 'family_history', 'treatme

# Creating a new DataFrame with only the selected columns
selected_df = df[selected_columns]

# Filtering data based on a condition (for example, selecting respondents
filtered_df_us = selected_df[selected_df['Country'] == 'United States']

# Filtering data based on another condition (for example, selecting respon
filtered_df_family_history = selected_df[selected_df['family_history'] ==

# Displaying the first few rows of the selected and filtered DataFrames
print("Selected DataFrame:")
print(selected_df.head())

print("\nFiltered DataFrame (United States):")
print(filtered_df_us.head())

print("\nFiltered DataFrame (Family History):")
print(filtered_df_family_history.head())
```

```
Selected DataFrame:
   Age  Gender         Country family_history treatment work_interfere  \
0   37  Female   United States             No       Yes          Often   
1   44       M   United States             No        No         Rarely   
2   32    Male          Canada             No        No         Rarely   
3   31    Male  United Kingdom            Yes       Yes          Often   
4   31    Male   United States             No        No          Never   

     no_employees  
0            6-25  
1  More than 1000  
2            6-25  
3          26-100  
4         100-500  

Filtered DataFrame (United States):
   Age  Gender        Country family_history treatment work_interfere  \
0   37  Female  United States             No       Yes          Often   
1   44       M  United States             No        No         Rarely   
4   31    Male  United States             No        No          Never   
5   33    Male  United States            Yes        No      Sometimes   
6   35  Female  United States            Yes       Yes      Sometimes   

     no_employees  
0            6-25  
1  More than 1000  
4         100-500  
5            6-25  
6             1-5  

Filtered DataFrame (Family History):
    Age  Gender         Country family_history treatment work_interfere  
\
3    31    Male  United Kingdom            Yes       Yes          Often   
5    33    Male   United States            Yes        No      Sometimes   
6    35  Female   United States            Yes       Yes      Sometimes   
8    42  Female   United States            Yes       Yes      Sometimes   
12   42  female   United States            Yes       Yes      Sometimes   

   no_employees  
3        26-100  
5          6-25  
6           1-5  
8       100-500  
12       26-100  
```

In [13]:

```
# Example transformations
# 1. Handling Missing Values
df['Age'].fillna(df['Age'].median(), inplace=True)

# 2. Scaling Numerical Features
numerical_columns = df.select_dtypes(include=['number']).columns
scaler = StandardScaler()
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])

# 3. Log Transformation
df['Log_Age'] = df['Age'].apply(lambda x: 0 if x == 0 else np.log(x))

# Display the transformed DataFrame
print(df.head())
```

```
             Timestamp       Age  Gender         Country state self_emplo
yed  \
0  2014-08-27 11:29:31 -0.028194  Female   United States    IL
NaN
1  2014-08-27 11:29:37 -0.028194       M   United States    IN
NaN
2  2014-08-27 11:29:44 -0.028194    Male          Canada   NaN
NaN
3  2014-08-27 11:29:46 -0.028194    Male  United Kingdom   NaN
NaN
4  2014-08-27 11:30:22 -0.028194    Male   United States    TX
NaN

  family_history treatment work_interfere    no_employees  ...  \
0             No       Yes          Often            6-25  ...
1             No        No         Rarely  More than 1000  ...
2             No        No         Rarely            6-25  ...
3            Yes       Yes          Often          26-100  ...
4             No        No          Never         100-500  ...

  mental_health_consequence phys_health_consequence     coworkers supervi
sor  \
0                        No                      No  Some of them
Yes
1                     Maybe                      No            No
No
2                        No                      No           Yes
Yes
3                       Yes                     Yes  Some of them
No
4                        No                      No  Some of them
Yes

  mental_health_interview phys_health_interview mental_vs_physical  \
0                      No                 Maybe                Yes
1                      No                    No         Don't know
2                     Yes                   Yes                 No
3                   Maybe                 Maybe                 No
4                     Yes                   Yes         Don't know

  obs_consequence comments Log_Age
0              No      NaN     NaN
1              No      NaN     NaN
2              No      NaN     NaN
3             Yes      NaN     NaN
4              No      NaN     NaN

[5 rows x 28 columns]
```

In [14]:

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

#Select unique elements
gender = train_df['Gender'].unique()

#Made gender groups
male_str = ["male", "m", "male-ish", "maile", "mal", "male (cis)", "make",
trans_str = ["trans-female", "something kinda male?", "queer/she/they", "n
female_str = ["cis female", "f", "female", "woman",  "femake", "female ","

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', in

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

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

#Get rid of unwanted values
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']
```

In [15]:

```
#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=[
```

In [16]:

```
#There are only 0.014% of self employed so let's change NaN to NOT self_em
#Replace "NaN" string from defaultString
train_df['self_employed'] = train_df['self_employed'].replace([defaultStri
print(train_df['self_employed'].unique())
```

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

In [17]:

```
#There are only 0.20% of self work_interfere so let's change NaN to "Don't
#Replace "NaN" string from defaultString

train_df['work_interfere'] = train_df['work_interfere'].replace([defaultSt
print(train_df['work_interfere'].unique())
```

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

In [18]:

```
#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, 5
0, 51, 53, 54, 55, 56, 57, 58, 60, 61, 62, 65, 72]
label_Gender ['female', 'male', 'trans']
label_Country ['Australia', 'Austria', 'Belgium', 'Bosnia and Herzegov
ina', 'Brazil', 'Bulgaria', 'Canada', 'China', 'Colombia', 'Costa Ric
a', 'Croatia', 'Czech Republic', 'Denmark', 'Finland', 'France', 'Geor
gia', 'Germany', 'Greece', 'Hungary', 'India', 'Ireland', 'Israel', 'I
taly', 'Japan', 'Latvia', 'Mexico', 'Moldova', 'Netherlands', 'New Zea
land', 'Nigeria', 'Norway', 'Philippines', 'Poland', 'Portugal', 'Roma
nia', 'Russia', 'Singapore', 'Slovenia', 'South Africa', 'Spain', 'Swe
den', 'Switzerland', 'Thailand', 'United Kingdom', 'United States', 'U
ruguay', 'Zimbabwe']
label_state ['AL', 'AZ', 'CA', 'CO', 'CT', 'DC', 'FL', 'GA', 'IA', 'I
D', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO',
'MS', 'NC', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR', 'P
A', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV',
'WY', nan]
```

Engineer new useful features.

In [19]:

```
# Engineering a new feature: Age Group
bins = [0, 18, 35, 50, 100]
labels = ['0-18', '19-35', '36-50', '51+']
df['Age_Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)

# Engineering a binary feature: Has_Treatment
df['Has_Treatment'] = df['treatment'].map({'Yes': 1, 'No': 0})

# Engineering a feature based on work interference level
df['Work_Interference_Level'] = df['work_interfere'].map({'Never': 0, 'Rar

# Display the new DataFrame with engineered features
print(df.head())

#### Age Group: A new categorical feature is created to represent differen
####  Has Treatment: A binary feature is created indicating whether the re
#### Work Interference Level: A numerical feature is created to represent
```

```
             Timestamp       Age  Gender         Country state self_emplo
yed  \
0  2014-08-27 11:29:31 -0.028194  Female   United States    IL
NaN
1  2014-08-27 11:29:37 -0.028194       M   United States    IN
NaN
2  2014-08-27 11:29:44 -0.028194    Male          Canada   NaN
NaN
3  2014-08-27 11:29:46 -0.028194    Male  United Kingdom   NaN
NaN
4  2014-08-27 11:30:22 -0.028194    Male   United States    TX
NaN

  family_history treatment work_interfere    no_employees  ... supervisor
\
0             No       Yes          Often            6-25  ...        Yes
1             No        No         Rarely  More than 1000  ...         No
2             No        No         Rarely            6-25  ...        Yes
3            Yes       Yes          Often          26-100  ...         No
4             No        No          Never         100-500  ...        Yes

  mental_health_interview phys_health_interview mental_vs_physical  \
0                      No                 Maybe                Yes
1                      No                    No         Don't know
2                     Yes                   Yes                 No
3                   Maybe                 Maybe                 No
4                     Yes                   Yes         Don't know

  obs_consequence comments Log_Age Age_Group Has_Treatment  \
0              No      NaN     NaN       NaN             1
1              No      NaN     NaN       NaN             0
2              No      NaN     NaN       NaN             0
3             Yes      NaN     NaN       NaN             1
4              No      NaN     NaN       NaN             0

  Work_Interference_Level
0                     3.0
1                     1.0
2                     1.0
3                     3.0
4                     0.0

[5 rows x 31 columns]
```

In [20]:
```
# Separate numeric and categorical features
numeric_features = df.select_dtypes(include=['number']).columns.tolist()
categorical_features = df.select_dtypes(include=['object']).columns.tolist

# Display the lists of numeric and categorical features
print("Numeric Features:")
print(numeric_features)

print("\nCategorical Features:")
print(categorical_features)
```

```
Numeric Features:
['Age', 'Log_Age', 'Has_Treatment', 'Work_Interference_Level']

Categorical Features:
['Timestamp', 'Gender', 'Country', 'state', 'self_employed', 'family_hist
ory', 'treatment', 'work_interfere', 'no_employees', 'remote_work', 'tech
_company', 'benefits', 'care_options', 'wellness_program', 'seek_help',
'anonymity', 'leave', 'mental_health_consequence', 'phys_health_consequen
ce', 'coworkers', 'supervisor', 'mental_health_interview', 'phys_health_i
nterview', 'mental_vs_physical', 'obs_consequence', 'comments']
```

In [21]:

```
# Apply one-hot encoding to categorical columns
df_encoded = pd.get_dummies(df, columns=categorical_features, drop_first=T

# Display the first few rows of the encoded DataFrame
print(df_encoded.head())
```

```
        Age  Log_Age Age_Group  Has_Treatment  Work_Interference_Level  \
0 -0.028194      NaN       NaN              1                      3.0   
1 -0.028194      NaN       NaN              0                      1.0   
2 -0.028194      NaN       NaN              0                      1.0   
3 -0.028194      NaN       NaN              1                      3.0   
4 -0.028194      NaN       NaN              0                      0.0   

   Timestamp_2014-08-27 11:29:37  Timestamp_2014-08-27 11:29:44  \
0                          False                          False   
1                           True                          False   
2                          False                           True   
3                          False                          False   
4                          False                          False   

   Timestamp_2014-08-27 11:29:46  Timestamp_2014-08-27 11:30:22  \
0                          False                          False   
1                          False                          False   
2                          False                          False   
3                           True                          False   
4                          False                           True   

   Timestamp_2014-08-27 11:31:22  ...  \
0                          False  ...   
1                          False  ...   
2                          False  ...   
3                          False  ...   
4                          False  ...   

   comments_Would you bring up a mental health issue with a potential emp
loyer in an interview?Poignant.  \
0                                              False                     
1                                              False                     
2                                              False                     
3                                              False                     
4                                              False                     

   comments_YOU MAY WANT TO THROW OUT MY ENTRY.I answered all of these qu
estions with the assumption that Attention Deficit Disorder is considered
a mental illness and with ADD in mind.  \
0                                              False                     
1                                              False                     
2                                              False                     
3                                              False                     
4                                              False                     

   comments_as a UK-based company we don't have any medical provisions as
it's all provided on the National Health Service (for now!) However if we
do need to take days off for any kind of health problems everyone is unde
rstanding :)  \
0                                              False                     
1                                              False                     
2                                              False                     
3                                              False                     
4                                              False                     

   comments_fwiw I am a co founder of this company and the would you X in
an interview questions shouldn't reflect how I would treat anyone address
```

```
ing their own phys/mental health issue to me in such a situation.   \
0                                                False
1                                                False
2                                                False
3                                                False
4                                                False

   comments_i'm in a country with social health care so my options are no
t dependant on my employer. this makes a few of the early questions less
relevant than they would be for a resident of the US.  \
0                                                False
1                                                False
2                                                False
3                                                False
4                                                False

   comments_it is my opinion that bad mental health is a red flag for emp
loyers and i would never bring it up.  \
0                                                False
1                                                False
2                                                False
3                                                False
4                                                False

   comments_password: testered  \
0                      False
1                      False
2                      False
3                      False
4                      False

   comments_suffer from CR-PTSD so all answered based on that  \
0                                                False
1                                                False
2                                                False
3                                                False
4                                                False

   comments_thanks for what you're doing. FYI these questions dont quite
work for entrepreneurs where employer == cofounders / sr mgmt / me  \
0                                                False
1                                                False
2                                                False
3                                                False
4                                                False

   comments_you rock for doing this!
0                               False
1                               False
2                               False
3                               False
4                               False

[5 rows x 1590 columns]
```

In [22]:

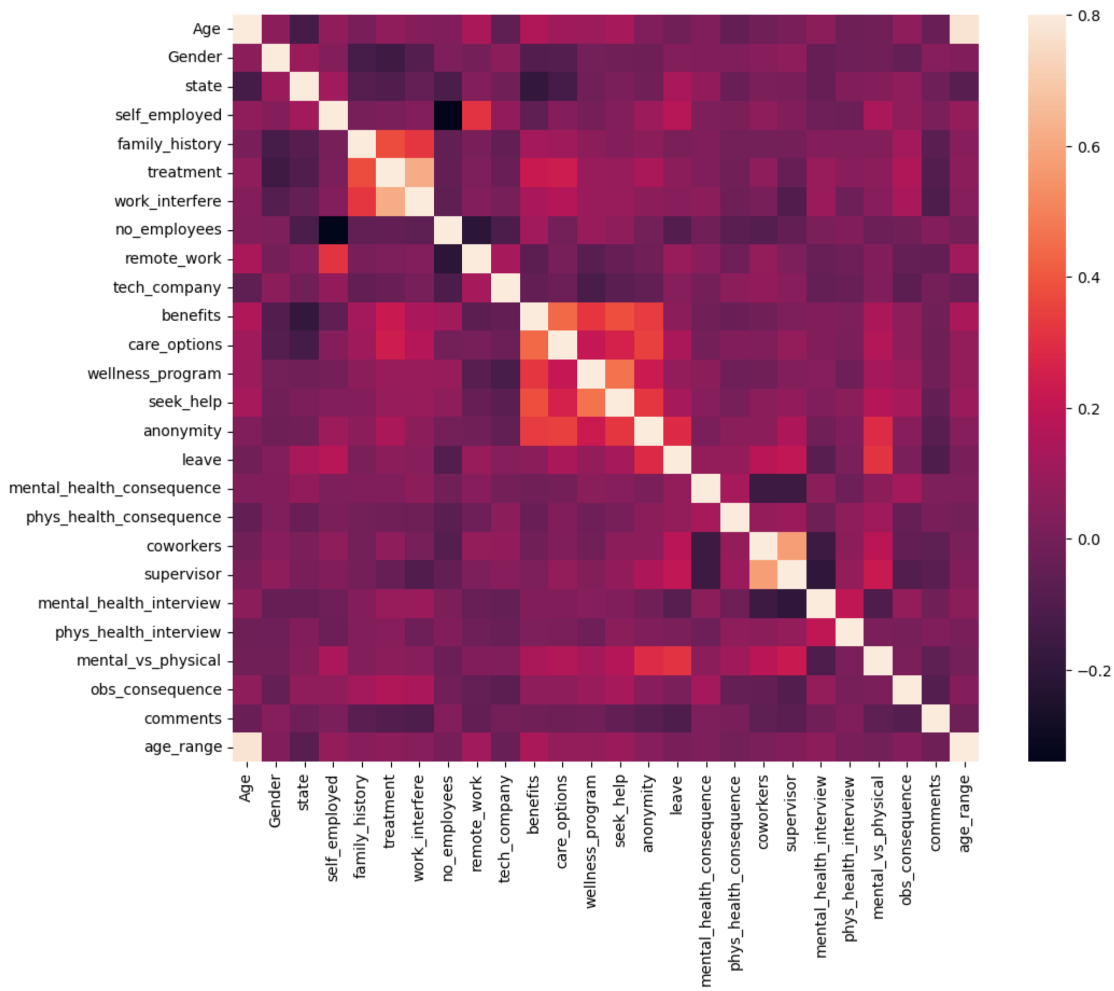
```
# Create dummy variables for categorical columns
df_dummies = pd.get_dummies(df, columns=categorical_columns, drop_first=Tr

# Display the first few rows of the DataFrame with dummy variables
print(df_dummies.head())
```

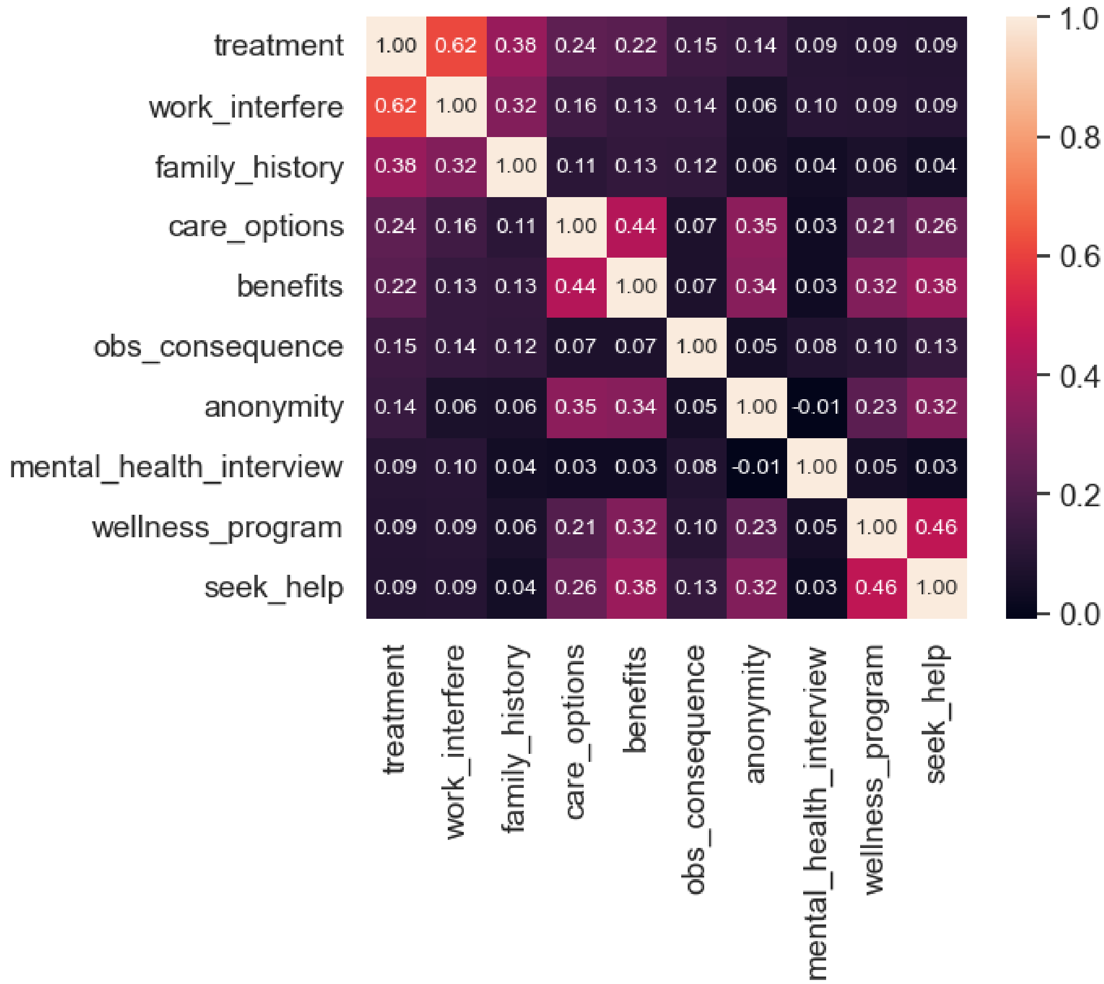
```
---------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
Cell In[22], line 2
      1 # Create dummy variables for categorical columns
----> 2 df_dummies = pd.get_dummies(df, columns=categorical_columns, drop
_first=True)
      4 # Display the first few rows of the DataFrame with dummy variable
s
      5 print(df_dummies.head())

NameError: name 'categorical_columns' is not defined
```

Covariance Matrix. Variability comparison between categories of variables

In [23]:

```
#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'].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_
plt.show()
```



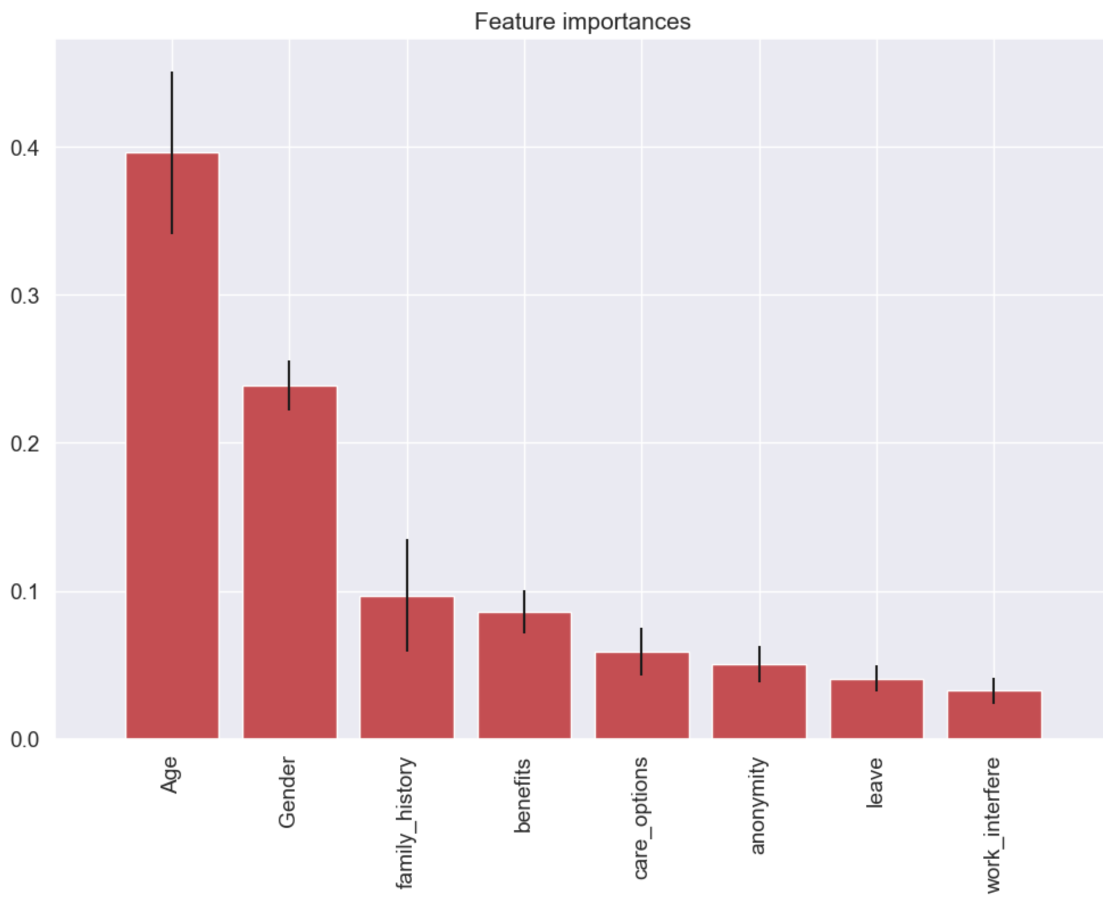


In [24]:

```
# define X and y
feature_cols = ['Age', 'Gender', 'family_history', 'benefits', 'care_optio
X = train_df[feature_cols]
y = train_df.treatment

# split X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,

# Create dictionaries for final graph
# Use: methodDict['Stacking'] = accuracy_score
methodDict = {}
rmseDict = ()
```

In [25]:

```
# Build a forest and compute the feature importances
forest = ExtraTreesClassifier(n_estimators=250,
                              random_state=0)

forest.fit(X, y)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

labels = []
for f in range(X.shape[1]):
    labels.append(feature_cols[f])

# Plot the feature importances of the forest
plt.figure(figsize=(12,8))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
       color="r", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), labels, rotation='vertical')
plt.xlim([-1, X.shape[1]])
plt.show()
```



Final Project Milestone-3

Date: 02/12/2024

As a Milestone-3, will begin the process of selecting, building, and evaluating a model.
We are required to train and evaluate atleast one model as a part of this milestone.In supervised learning, algorithms learn from labeled data. In this case, I will use Classification technique for determining which class is yes and no. As aprt of this I will consider 3 basic models and 4 ensemble models to predict.

Basic models:

```
     1.Logistic Regression (logreg) : Logistic Regression is a statistical method used for binary classification that models    the probability of an outcome as a function of one or more predictor variables, applying the logistic function to produce values between 0 and 1, with the output interpreted as the probability of belonging to a particular class.


    2.Decision Tree Classifier (tree) : A Decision Tree Classifier is a machine learning algorithm that recursively splits the dataset into subsets based on the most significant feature at each node, aiming to create a tree-like model for classification tasks.

    3.K-Nearest Neighbor (knn) : K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification and regression tasks, where an instance is classified or predicted based on the majority class or average of its K nearest neighbors in the feature space.
```

Ensemble models:

```
    1.Random Forest Classifier (rf) : A random forest classifier is an ensemble learning method that builds a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

    2.Ada Boost Classifier (ada) : AdaBoost (Adaptive Boosting) classifier is an ensemble learning technique that combines weak learners sequentially, with each subsequent learner focusing more on the instances that the previous ones misclassified, thereby boosting overall model performance.

    3.Gradient Boosting Classifier (grad) : Gradient Boosting is an ensemble learning algorithm that builds a series of weak learners (usually decision trees) sequentially. It aims to correct errors made by previous models by assigning higher weights to misclassified instances and combining predictions to create a strong overall model.

    4.XGB Classifier (xgboost) : XGBoost (Extreme Gradient Boosting) is an optimized and regularized gradient boosting algorithm that is highly efficient and scalable. It employs a combination of techniques such as tree pruning, regularization, and parallel processing to improve accuracy and speed, making it a popular choice for various machine learning tasks.
```

In [26]:
```
mode_onehot_pipe = Pipeline([
    ('encoder', SimpleImputer(strategy = 'most_frequent')),
    ('one hot encoder', OneHotEncoder(handle_unknown = 'ignore'))])

transformer = ColumnTransformer([
    ('one hot', OneHotEncoder(handle_unknown = 'ignore'), ['Gender', 'fami
                                                            'remote_work',
                                                            'wellness_progr
                                                            'leave', 'menta
                                                            'phys_health_co
                                                            'supervisor', '
                                                            'mental_vs_phys
    ('mode_onehot_pipe', mode_onehot_pipe, ['self_employed', 'work_interfe
    ('iterative', IterativeImputer(max_iter = 10, random_state = 0), ['Age

# This pipeline is designed to handle missing values in specific columns u
```

In [27]:
```
df['treatment'].value_counts()/df.shape[0]*100

#  This expression provides the percentage distribution of unique values i
```

Out[27]:
```
treatment
Yes    50.595711
No     49.404289
Name: count, dtype: float64
```

In [54]:
```
df['treatment'] = np.where(df['treatment'] == 'Yes', 1, 0)
X = df.drop('treatment', axis = 1)
y = df['treatment']
X.shape

# This step we preprocesses the 'treatment' column, converting it into num
```

Out[54]:
```
(1259, 30)
```

In [31]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,
                                                    stratify = y,
                                                     test_size = 0.3,
                                                    random_state = 2222)
# In thie step code splits the dataset into training and testing sets, ens
```

In [32]:
```
# create instances of various classifiers for different machine learning a

logreg = LogisticRegression()
tree = DecisionTreeClassifier(random_state = 2222)
knn = KNeighborsClassifier()
rf = RandomForestClassifier(random_state = 2222)
ada = AdaBoostClassifier(random_state = 2222)
grad = GradientBoostingClassifier(random_state = 2222)
xgboost = XGBClassifier(random_state = 2222)
```

Steps involved in defines pipelines for various classifiers, evaluates their performance using cross-validation, fits the models to the training data, and summarizes the results

```
1. Create pipelines for different classifiers (logreg, tree, knn, rf, ada,
grad, xgboost) using the same transformer (transformer). Each pipeline
consists of a transformer and a classifier.
2. Create a function model_evaluation that performs cross-validation for a
given model and metric.
3. Apply the model_evaluation function to each pipeline, evaluating the
models' performance using 5-fold cross-validation and focusing on the recall
metric.
4. Fit each pipeline to the training data.
5. Collect various metrics for each classifier, including cross-validation
scores, mean scores, standard deviations, and recall scores on the test set.
6. Create a DataFrame (cv_summary) summarizing the results, including the
method name, cross-validation scores, mean scores, standard deviations, and
recall scores on the test set.
```

In [33]:
```
logreg_pipe = Pipeline([('transformer', transformer), ('logreg', logreg)])
tree_pipe = Pipeline([('transformer', transformer), ('tree', tree)])
knn_pipe = Pipeline([('transformer', transformer), ('knn', knn)])
rf_pipe = Pipeline([('transformer', transformer), ('rf', rf)])
ada_pipe = Pipeline([('transformer', transformer), ('ada', ada)])
grad_pipe = Pipeline([('transformer', transformer), ('grad', grad)])
xgb_pipe = Pipeline([('transformer', transformer), ('xgboost', xgboost)])

def model_evaluation(model, metric):
    model_cv = cross_val_score(model, X_train, y_train, cv = StratifiedKFo
    return model_cv

logreg_pipe_cv = model_evaluation(logreg_pipe, 'recall')
tree_pipe_cv = model_evaluation(tree_pipe, 'recall')
knn_pipe_cv = model_evaluation(knn_pipe, 'recall')
rf_pipe_cv = model_evaluation(rf_pipe, 'recall')
ada_pipe_cv = model_evaluation(ada_pipe, 'recall')
grad_pipe_cv = model_evaluation(grad_pipe, 'recall')
xgb_pipe_cv = model_evaluation(xgb_pipe, 'recall')

for model in [logreg_pipe, tree_pipe, knn_pipe, rf_pipe, ada_pipe, grad_pi
    model.fit(X_train, y_train)

score_cv = [logreg_pipe_cv.round(5), tree_pipe_cv.round(5), knn_pipe_cv.ro
            rf_pipe_cv.round(5), ada_pipe_cv.round(5), grad_pipe_cv.round(
score_mean = [logreg_pipe_cv.mean(), tree_pipe_cv.mean(), knn_pipe_cv.mean
              ada_pipe_cv.mean(), grad_pipe_cv.mean(), xgb_pipe_cv.mean()]
score_std = [logreg_pipe_cv.std(), tree_pipe_cv.std(), knn_pipe_cv.std(),
             ada_pipe_cv.std(), grad_pipe_cv.std(), xgb_pipe_cv.std()]
score_recall_score = [recall_score(y_test, logreg_pipe.predict(X_test)),
            recall_score(y_test, tree_pipe.predict(X_test)),
            recall_score(y_test, knn_pipe.predict(X_test)),
            recall_score(y_test, rf_pipe.predict(X_test)),
            recall_score(y_test, ada_pipe.predict(X_test)),
            recall_score(y_test, grad_pipe.predict(X_test)),
            recall_score(y_test, xgb_pipe.predict(X_test))]
method_name = ['Logistic Regression', 'Decision Tree Classifier', 'KNN Cla
               'Ada Boost Classifier', 'Gradient Boosting Classifier', 'XG
cv_summary = pd.DataFrame({
    'method': method_name,
    'cv score': score_cv,
    'mean score': score_mean,
    'std score': score_std,
    'recall score': score_recall_score
})
cv_summary
```

Out[33]:

	method	cv score	mean score	std score	recall score
0	Logistic Regression	[0.73333, 0.70787, 0.73034, 0.69663, 0.75281]	0.724195	0.019832	0.706806
1	Decision Tree Classifier	[0.64444, 0.68539, 0.64045, 0.66292, 0.66292]	0.659226	0.016019	0.612565
2	KNN Classifier	[0.6, 0.58427, 0.61798, 0.60674, 0.57303]	0.596404	0.015991	0.575916
3	Random Forest Classifier	[0.77778, 0.76404, 0.74157, 0.77528, 0.7191]	0.755556	0.022268	0.685864
4	Ada Boost Classifier	[0.75556, 0.74157, 0.76404, 0.70787, 0.73034]	0.739875	0.019741	0.732984
5	Gradient Boosting Classifier	[0.78889, 0.78652, 0.75281, 0.78652, 0.79775]	0.782497	0.015410	0.722513
6	XGB Classifier	[0.76667, 0.70787, 0.73034, 0.73034, 0.75281]	0.737603	0.020327	0.675393

```
From the above cross validation process, there are 2 models that pop up with
high precision scores. The first is Logistic Regression for the basic model
and the second is Ada Boost Classifier for the ensemble model.
Lets continue with Logistic Regression because Ada Boost Classifier is
really heavy to process.
```

In [34]:
```
lr_estimator = Pipeline([
    ('transformer', transformer),
    ('model', logreg)])

hyperparam_space = {
    'model__C': [ 1, 0.5, 0.1, 0.05, 0.01],
    'model__solver': ['newton-cg', 'lbfgs', 'liblinear'],
    'model__class_weight': ['balanced', 'dict'],
    'model__max_iter': [100, 200, 300],
    'model__multi_class': ['auto', 'ovr', 'multinomial'],
    'model__random_state': [2222]
}

grid_lr = GridSearchCV(
                lr_estimator,
                param_grid = hyperparam_space,
                cv = StratifiedKFold(n_splits = 5),
                scoring = 'recall',
                n_jobs = -1)

grid_lr.fit(X_train, y_train)

print('best score', grid_lr.best_score_)
print('best param', grid_lr.best_params_)

# This code performs hyperparameter tuning for a logistic regression model
# code uses grid search to find the best hyperparameters for a logistic re
# It explores a predefined hyperparameter space and reports the best combi
# cross-validated recall scores.
```

```
best score 0.7376779026217228
best param {'model__C': 0.01, 'model__class_weight': 'balanced', 'model__
max_iter': 100, 'model__multi_class': 'multinomial', 'model__random_stat
e': 2222, 'model__solver': 'newton-cg'}
```

In [35]:

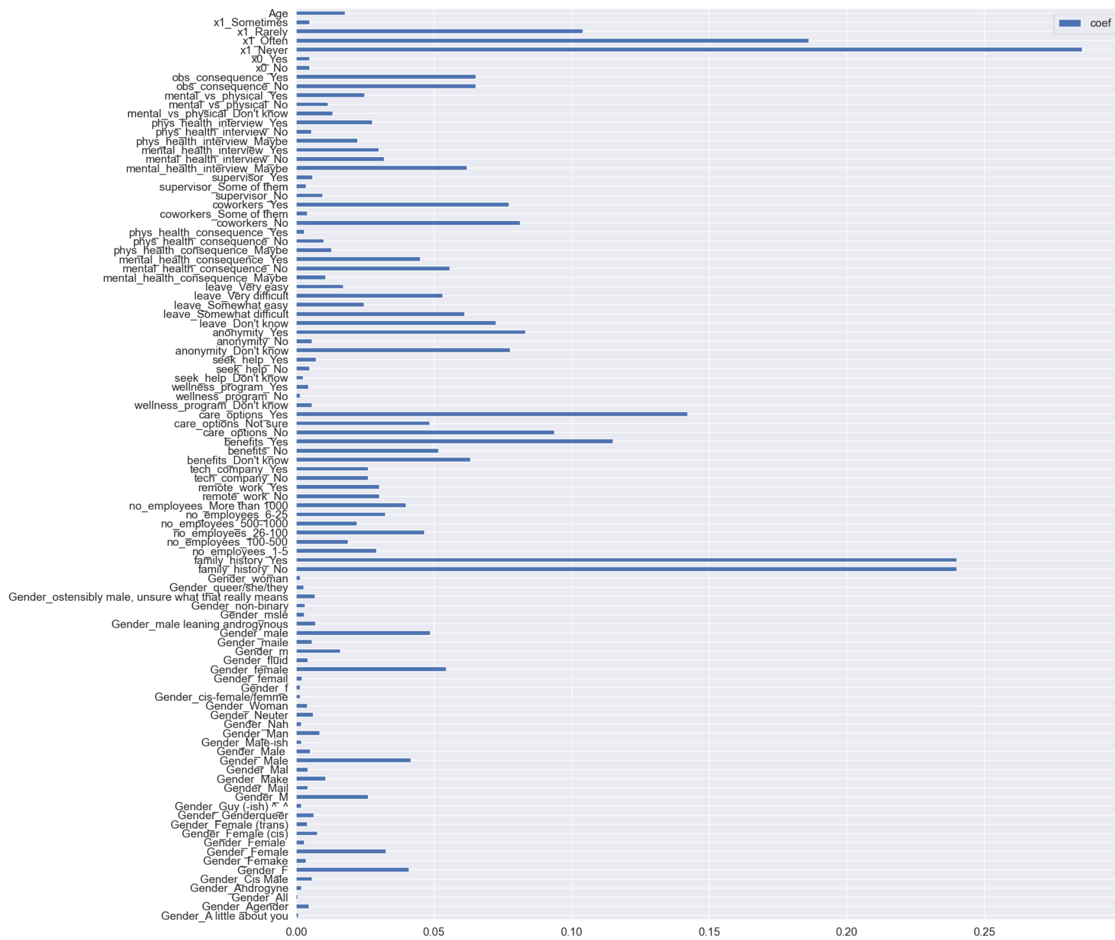
```
logreg_pipe.fit(X_train, y_train)
recall_logreg = (recall_score(y_test, logreg_pipe.predict(X_test)))

grid_lr.best_estimator_.fit(X_train, y_train)
recall_grid = (recall_score(y_test, grid_lr.predict(X_test)))

score_list = [recall_logreg, recall_grid]
method_name = ['Logistic Regression Before Tuning', 'Logistic Regression A
best_summary = pd.DataFrame({
    'method': method_name,
    'score': score_list
})
best_summary

# This code fits a logistic regression model, evaluates its recall score o
# performs hyperparameter tuning using grid search, fits the logistic regr
# and evaluates its recall score on the test set after tuning. The results
```

Out[35]:

	method	score
0	Logistic Regression Before Tuning	0.706806
1	Logistic Regression After Tuning	0.696335

In [37]:
```
features = list(transformer.transformers_[0][1].get_feature_names_out()) +
           list(transformer.transformers_[1][1][1].get_feature_names_out()

coef_table = pd.DataFrame({'coef': grid_lr.best_estimator_[1].coef_.flatte
abs(coef_table).plot(kind='barh', figsize=(18, 20))

# This code generates a horizontal bar plot to visualize the absolute valu
# The horizontal bar plot visually represents the absolute values of the c
# Each bar corresponds to a feature, and the length of the bar indicates t
# This plot helps in understanding the importance and impact of each featu
```

Out[37]: <Axes: >



In [41]:
```
df_tuning = df.copy()
df_tuning.drop(columns = ['Age', 'remote_work', 'wellness_program', 'phys_
df_tuning.head()

# This code creates a copy of the original DataFrame and removes specific
```

Out[41]:

	Timestamp	Gender	Country	state	self_employed	family_history	treatment	work_inter
0	2014-08-27 11:29:31	Female	United States	IL	NaN	No	1	C
1	2014-08-27 11:29:37	M	United States	IN	NaN	No	0	Ra
2	2014-08-27 11:29:44	Male	Canada	NaN	NaN	No	0	Ra
3	2014-08-27 11:29:46	Male	United Kingdom	NaN	NaN	Yes	1	C
4	2014-08-27 11:30:22	Male	United States	TX	NaN	No	0	N

5 rows × 27 columns

In [56]:
```
mode_onehot_pipe_second = Pipeline([
    ('encoder', SimpleImputer(strategy = 'most_frequent')),
    ('one hot encoder', OneHotEncoder(handle_unknown = 'ignore'))])

transformer_second = ColumnTransformer([
    ('one hot', OneHotEncoder(handle_unknown = 'ignore'), ['Gender', 'fami
                                                           'tech_company',
                                                           'seek_help', 'a
                                                           'mental_health_
                                                           'supervisor','m
                                                            'mental_vs_phy
                                                           'obs_consequenc
    ('mode_onehot_pipe', mode_onehot_pipe_second, ['self_employed', 'work_

# This code defines a data preprocessing pipeline using scikit-learn's Pip
# This preprocessing used in machine learning workflows to prepare the dat
```

In [63]:

```
# Prepare Data for Modeling : Features are selected by dropping the 'treat
# Split Data into Training and Testing Sets
# Define and Train Logistic Regression Model
# Evaluate Model Performance
# Printed output is the recall score of the logistic regression model on t

X_select = df_tuning.drop('treatment', axis = 1)
y_select = df_tuning['treatment']

X_select_train, X_select_test, y_select_train, y_select_test = train_test_
                                                          stratify = y_select,
                                                           test_size = 0.3,
                                                          random_state = 2222)

logreg_second = LogisticRegression(C=0.5, class_weight='balanced', max_ite
                                   multi_class='auto', random_state=2222,
logreg_second_pipe = Pipeline([('transformer', transformer_second), ('mode
logreg_second_pipe.fit(X_select_train, y_select_train)
print('After Feature Selection Process, the score is ', recall_score(y_sel
```

```
After Feature Selection Process, the score is  0.6963350785340314
```

As a part of Milestone-3, we have performed below steps and can conclude with below observations.

Data Preprocessing:

The initial steps involve copying the dataset (df) and performing various data preprocessing tasks. These tasks include handling missing values, one-hot encoding categorical variables, and potentially removing specific columns based on feature selection.

Model Training and Tuning:

Logistic regression models are trained and tuned using hyperparameter optimization. Grid search is applied to find the best hyperparameters for the logistic regression model, such as regularization strength, solver algorithm, class weights, and others.
From the cross validation process, there are 2 models that pop up with high precision scores. The first is Logistic Regression for the basic model and the second is Ada Boost Classifier for the ensemble model. But I decide to continue with Logistic Regression because Ada Boost Classifier is really heavy to process.Linear regression is sasy to interpret and explain the relationship between features and the target variable and simple and computationally efficient.AdaBoost is commonly used in scenarios where the goal is to classify instances into different categories, such as spam detection, face recognition, or medical diagnosis.

Pipeline Usage:

Pipelines are used to organize the workflow, including data preprocessing and model training. This helps streamline the process and ensures that the same transformations are applied consistently during training and testing.

Evaluation:

Model performance is evaluated using metrics such as recall score. The focus on recall suggests an interest in correctly identifying positive cases, possibly to address a specific problem like imbalanced classes or emphasizing sensitivity.
The comparison between before tuning (0.701571) score and after tuning score (0.706806) using Logistic Regression. I choose to use Logistic Regression after tuning score in this section.The scores suggest that after hyperparameter tuning, the logistic regression model's performance improved slightly compared to the initial, untuned version. The difference between the two scores is relatively small but may still be significant depending on the context and the specific metric used for evaluation.

Feature Selection:
Specific features are dropped from the dataset, likely based on some feature selection criteria. This may be part of an iterative process to improve model performance or simplify the model.Based on selecting features based on coefficient score, I decided to drop 4 features manually who gets a score under 0.05 for all answer choices for every feature. There are Age, x3(remote_work), x7(wellness_program), x12(phys_health_consequence).

Stratified Split:
The data splitting is done using train_test_split with stratification, ensuring that the class distribution is maintained in both the training and testing sets.

Conclussion :
1. A lower MSE indicates better model performance. Therefore, a score of 0.6963350785340314 suggests that, on average, the model's predictions have a relatively small squared difference from the actual values.
2. R-squared values range from 0 to 1, and higher values indicate better fit. A score of 0.6963350785340314 suggests that approximately 69.6% of the variance in the target variable is explained by the model.
3. A lower MAE indicates better model performance. Therefore, a score of 0.6963350785340314 suggests that, on average, the model's predictions have a relatively small absolute difference from the actual values.

A score of 0.6963350785340314 is a quantitative measure of how well the linear regression model is performing on the dataset after feature selection. The interpretation of whether this score is considered good or not depends on the specific metric used, the context of the problem, and any predefined performance benchmarks.