Objective question: understand what are the factors contributing the mental health of a person in the tech industry. See if benefits have a high correlation to other variables

Understanding the columns

Timestamp

Age

Gender

Country

state: If you live in the United States, which state or territory do you live in?

self_employed: Are you self-employed?

family_history: Do you have a family history of mental illness?

treatment: Have you sought treatment for a mental health condition?

work_interfere: If you have a mental health condition, do you feel that it interferes with your work?

no_employees: How many employees does your company or organization have?

remote_work: Do you work remotely (outside of an office) at least 50% of the time?

tech_company: Is your employer primarily a tech company/organization?

benefits: Does your employer provide mental health benefits?

care_options: Do you know the options for mental health care your employer provides?

wellness_program: Has your employer ever discussed mental health as part of an employee wellness program?

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

anonymity: Is your anonymity protected if you choose to take advantage of mental health or substance abuse treatment resources?

leave: How easy is it for you to take medical leave for a mental health condition?

mental_health_consequence: Do you think that discussing a mental health issue with your employer would have negative consequences?

phys_health_consequence: Do you think that discussing a physical health issue with your employer would have negative consequences?

coworkers: Would you be willing to discuss a mental health issue with your coworkers?

supervisor: Would you be willing to discuss a mental health issue with your direct supervisor(s)?

mental_health_interview: Would you bring up a mental health issue with a potential employer in an interview?

phys_health_interview: Would you bring up a physical health issue with a potential employer in an interview?

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

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

comments: Any additional notes or comments

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Load the CSV data into a DataFrame
df = pd.read_csv('survey.csv')

# Display the first few rows of the DataFrame
```

```
df.head()
```

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	Often	6-25	...	Somewhat easy	
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	Rarely	More than 1000	...	Don't know	
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Rarely	6-25	...	Somewhat difficult	
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100	...	Somewhat difficult	
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	Never	100-500	...	Don't know	

5 rows × 27 columns

We will be dropping the timestamp column because it's contains date, month, year and time the respondent took this questionnaire, which is irrelevant for us.

```
# Print how many respondants from each country
print(df['Country'].value_counts())
print("\n \n")
```

```
Country
United States             751
United Kingdom            185
Canada                     72
Germany                    45
Ireland                    27
Netherlands                27
Australia                  21
France                     13
India                      10
New Zealand                 8
Poland                      7
Switzerland                 7
Sweden                      7
Italy                       7
South Africa                6
Belgium                     6
Brazil                      6
Israel                      5
Singapore                   4
Bulgaria                    4
Austria                     3
Finland                     3
Mexico                      3
Russia                      3
Denmark                     2
Greece                      2
Colombia                    2
Croatia                     2
Portugal                    2
Moldova                     1
Georgia                     1
Bahamas, The                1
China                       1
Thailand                    1
Czech Republic              1
Norway                      1
Romania                     1
Nigeria                     1
Japan                       1
Hungary                     1
Bosnia and Herzegovina      1
Uruguay                     1
Spain                       1
Zimbabwe                    1
Latvia                      1
Costa Rica                  1
```

```
Slovenia                    1
Philippines                 1
Name: count, dtype: int64
```

We will also be dropping the country and state column because there is not enough respondents from each country to conclude if their country faces bigger mental health issues compared to other countries therefore it becomes irrelevant. Thus, the state column also becomes irrelavent because that column is only relevant to the US. We will also be dropping the comments column because it is an optional question and many decided to forego answering it.

```
# Drop timestamp, country, state, and comments columns
df.drop(columns=['Timestamp', 'Country', 'state', 'comments'], inplace = True)
```

Look for values that don't make sense or are outliers for the age column. I am setting the limit from 18-72.

```
# Get unique values and their counts from the 'age' column
age_counts = df['Age'].value_counts()

# Print the counts
print(age_counts)
```

```
Age
 29             85
 32             82
 26             75
 27             71
 33             70
 28             68
 31             67
 34             65
 30             63
 25             61
 35             55
 23             51
 24             46
 37             43
 38             39
 36             37
 40             33
 39             33
 43             28
 22             21
 41             21
 42             20
 21             16
 45             12
 46             12
 44             11
 19              9
 18              7
 48              6
 50              6
 20              6
 51              5
 49              4
 56              4
 57              3
 54              3
 55              3
 47              2
 60              2
 99999999999     1
 5               1
-1               1
 11              1
 8               1
 61              1
 53              1
-29              1
-1726            1
 65              1
 62              1
 58              1
 329             1
 72              1
```

```
Name: count, dtype: int64
```

```
# Filter out age outliers
df = df[(df['Age'] >= 18) & (df['Age'] <= 100)]

# Print the filtered age counts to verify
filtered_age_counts = df['Age'].value_counts()
print(filtered_age_counts)
```

```
Age
29    85
32    82
26    75
27    71
33    70
28    68
31    67
34    65
30    63
25    61
35    55
23    51
24    46
37    43
38    39
36    37
39    33
40    33
43    28
41    21
22    21
42    20
21    16
45    12
46    12
44    11
19     9
18     7
48     6
50     6
20     6
51     5
49     4
56     4
55     3
57     3
54     3
47     2
60     2
58     1
62     1
65     1
53     1
61     1
72     1
Name: count, dtype: int64
```

Let's look at genders.

```
# Prints all unique gender variables
print(df['Gender'].unique())
```

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

We see that there was most likely no dropbox option for this question because there were many different unique values. I will be cleaning this up to be Male, Female, and other.

```
# Create a copy of the DataFrame
clean_df = df.copy()
```

```
# Replace the gender values for 'Male'
clean_df['Gender'].replace(['Male ', 'male', 'M', 'm', 'Male', 'Cis Male',
                            'Man', 'cis male', 'Mail', 'Male-ish', 'Male (CIS)',
                            'Cis Man', 'msle', 'Malr', 'Mal', 'maile', 'Make'], 'Male', inplace=True)

# Replace the gender values for 'Female'
clean_df['Gender'].replace(['Female ', 'female', 'F', 'f', 'Woman', 'Female',
                            'femail', 'Cis Female', 'cis-female/femme', 'Femake', 'Female (cis)',
                            'woman'], 'Female', inplace=True)

# Replace the gender values for 'Other'
clean_df['Gender'].replace(['Female (trans)', 'queer/she/they', 'non-binary',
                            'fluid', 'queer', 'Androgyne', 'Trans-female', 'male leaning androgynous',
                            'Agender', 'A little about you', 'Nah', 'All',
                            'ostensibly male, unsure what that really means',
                            'Genderqueer', 'Enby', 'p', 'Neuter', 'something kinda male?',
                            'Guy (-ish) ^_^', 'Trans woman'], 'Other', inplace=True)

# Count the occurrences of each category
gender_counts = clean_df['Gender'].value_counts()
print(gender_counts)
```

```
Gender
Male      986
Female    247
Other      18
Name: count, dtype: int64
```

We must now look at null values.

```
# Count null values for each column
clean_df.isnull().sum()
```

```
Age                            0
Gender                         0
self_employed                 18
family_history                 0
treatment                      0
work_interfere               262
no_employees                   0
remote_work                    0
tech_company                   0
benefits                       0
care_options                   0
wellness_program               0
seek_help                      0
anonymity                      0
leave                          0
mental_health_consequence      0
phys_health_consequence        0
coworkers                      0
supervisor                     0
mental_health_interview        0
phys_health_interview          0
mental_vs_physical             0
obs_consequence                0
dtype: int64
```

I will be replacing self_employed null values to "No" because only 1.4% are self employed so it is safe to assume that. I will be replacing work_interfere null values to "Don't know."

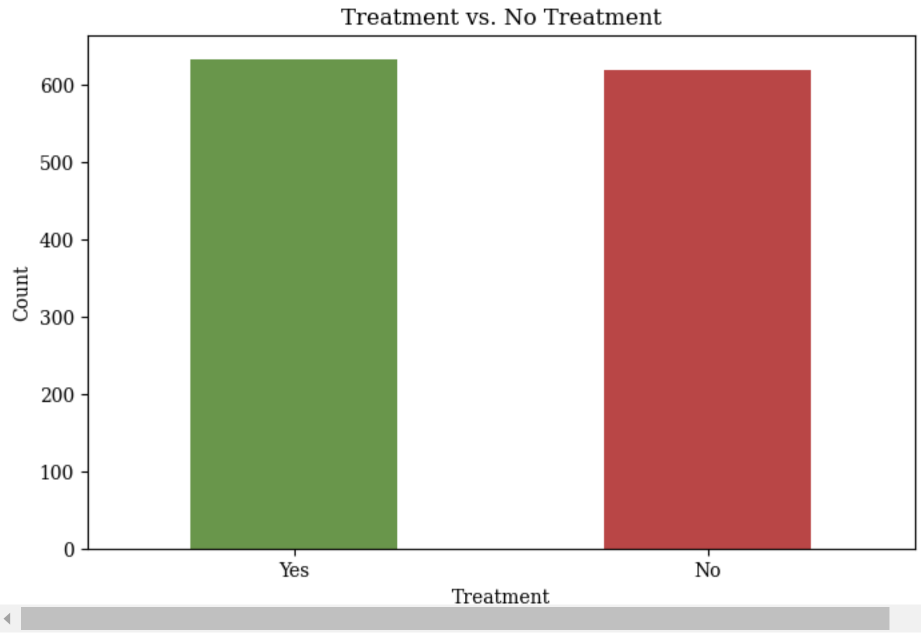
```
# Replace null values in 'self_employed' with 'No'
clean_df['self_employed'].fillna('No', inplace=True)

# Replace null values in 'work_interfere' with 'Don't know'
clean_df['work_interfere'].fillna("Don't know", inplace=True)

# Verify changes by checking for null values
clean_df.isnull().sum()
```

```
Age                          0
Gender                       0
```

```
self_employed              0
family_history             0
treatment                  0
work_interfere             0
no_employees               0
remote_work                0
tech_company               0
benefits                   0
care_options               0
wellness_program           0
seek_help                  0
anonymity                  0
leave                      0
mental_health_consequence  0
phys_health_consequence    0
coworkers                  0
supervisor                 0
mental_health_interview    0
phys_health_interview      0
mental_vs_physical         0
obs_consequence            0
dtype: int64
```

Plotting Treatment vs No Treatment because that is our target variable

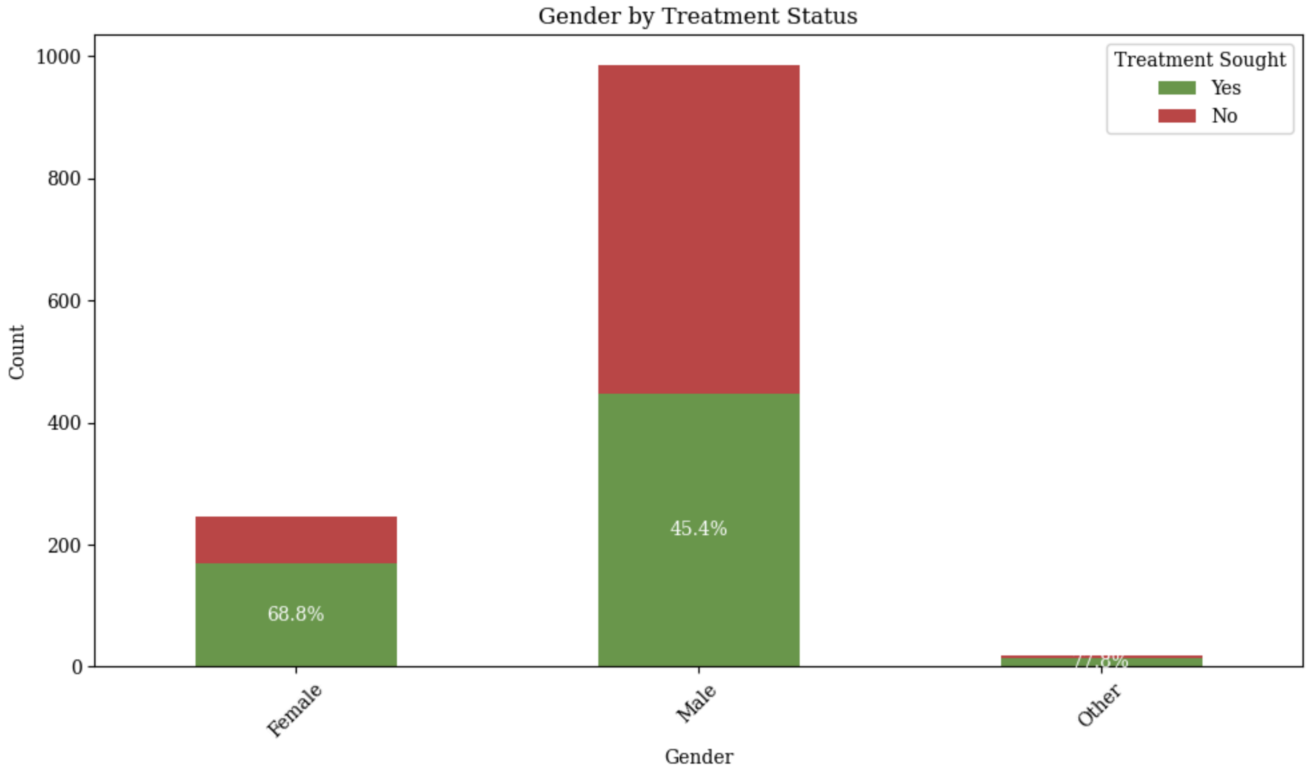
```
# Count the number of responses for treatment
treatment_counts = clean_df['treatment'].value_counts()

# Plotting the bar chart
colors = ['#6a994e', '#bc4749']  # Green for 'Yes', Red for 'No'
treatment_counts.plot(kind='bar', color=colors, figsize=(8, 5))
plt.title('Treatment vs. No Treatment')
plt.xlabel('Treatment')
plt.ylabel('Count')
plt.xticks(rotation=0)  # Rotate labels to horizontal
plt.show()
```

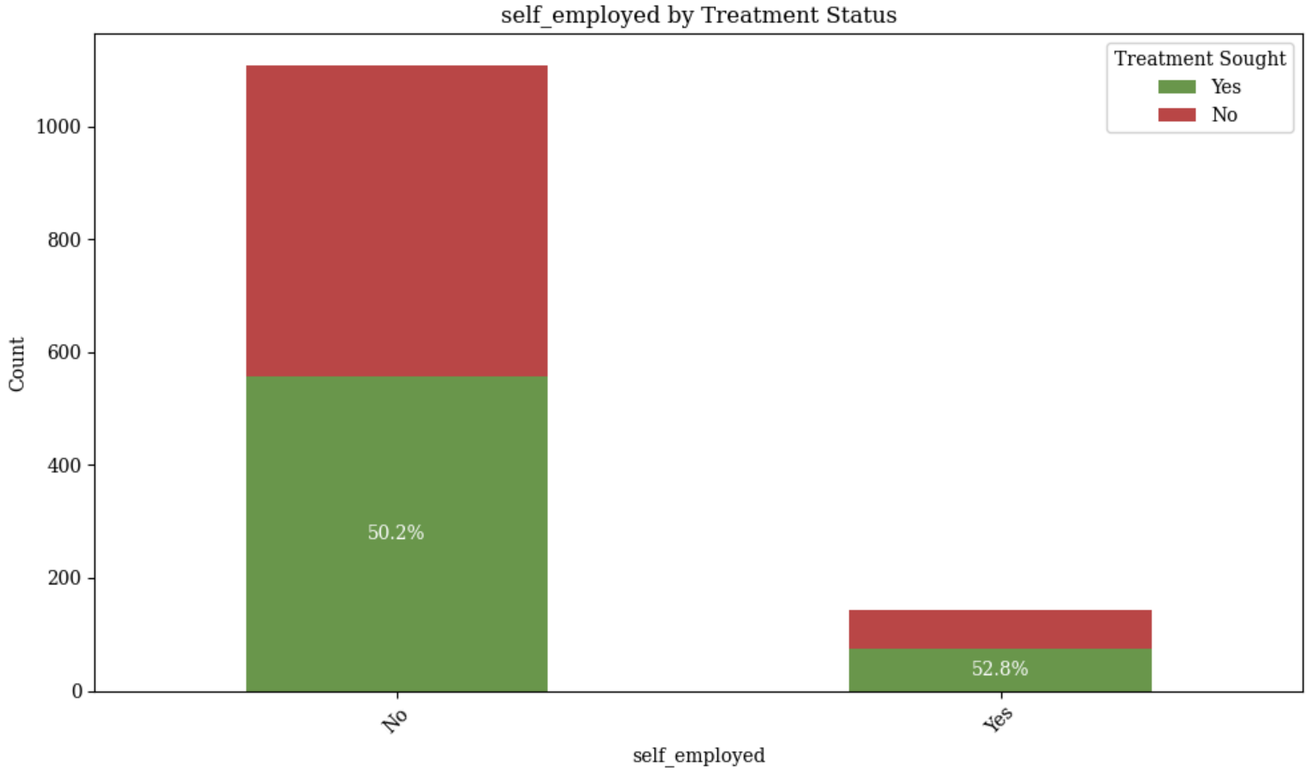


We will now analyze each column and make assumptions based on what we find.

```
# Change default font to Serif
matplotlib.rcParams['font.family'] = 'serif'

for column in clean_df.columns:
    if clean_df[column].dtype == 'object':  # Filtering for categorical columns
        # Group the data by the categorical column and 'treatment' status
        grouped = clean_df.groupby([column, 'treatment']).size().unstack(fill_value=0)
```

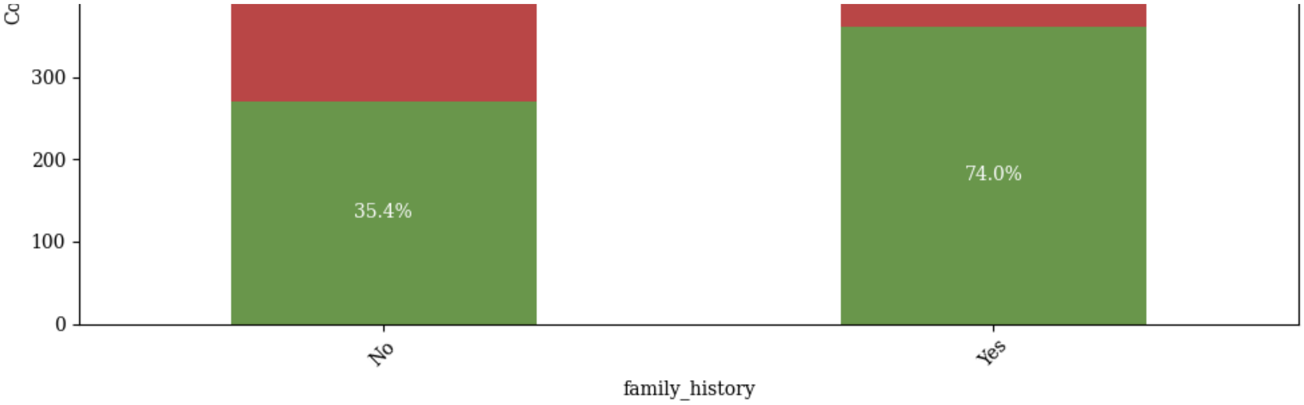
```
# Reverse the order of stacking by sorting the columns if both 'Yes' and 'No' are present
if 'Yes' in grouped.columns and 'No' in grouped.columns:
    grouped = grouped[['Yes', 'No']]
colors = ['#6a994e', '#bc4749']  # 'Yes' in green and 'No' in red

# Create a stacked bar plot with specified colors
ax = grouped.plot(kind='bar', stacked=True, color=colors, figsize=(10, 6))

# Calculate the total counts for each category to find percentages
totals = grouped.sum(axis=1)

# Annotate percentages for 'Yes' treatment sought
for i, (name, row) in enumerate(grouped.iterrows()):
    if 'Yes' in row.index:  # Check if 'Yes' category exists for treatment
        value = row['Yes']
        pct = f"{(value / totals[name] * 100):.1f}%"  # Calculate percentage
        x = i
        y = value / 2  # Position for annotation in the middle of the 'Yes' section
        ax.text(x, y, pct, ha='center', va='center', color='white')  # White text for visibility

# Customize the plot
plt.title(f'{column} by Treatment Status')
plt.xlabel(column)
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Treatment Sought', loc='upper right')
plt.tight_layout()

# Show the plot
plt.show()
```


Gender by Treatment Status

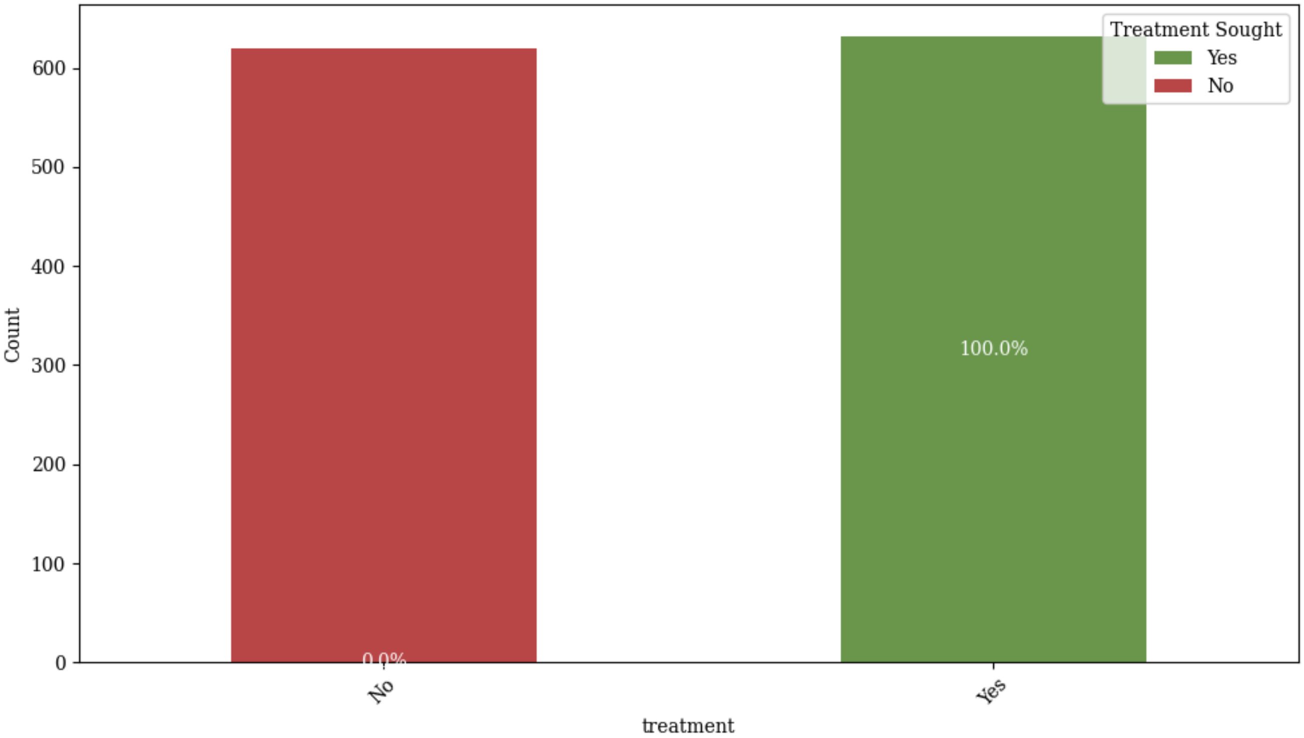

self_employed by Treatment Status

family_history by Treatment Status

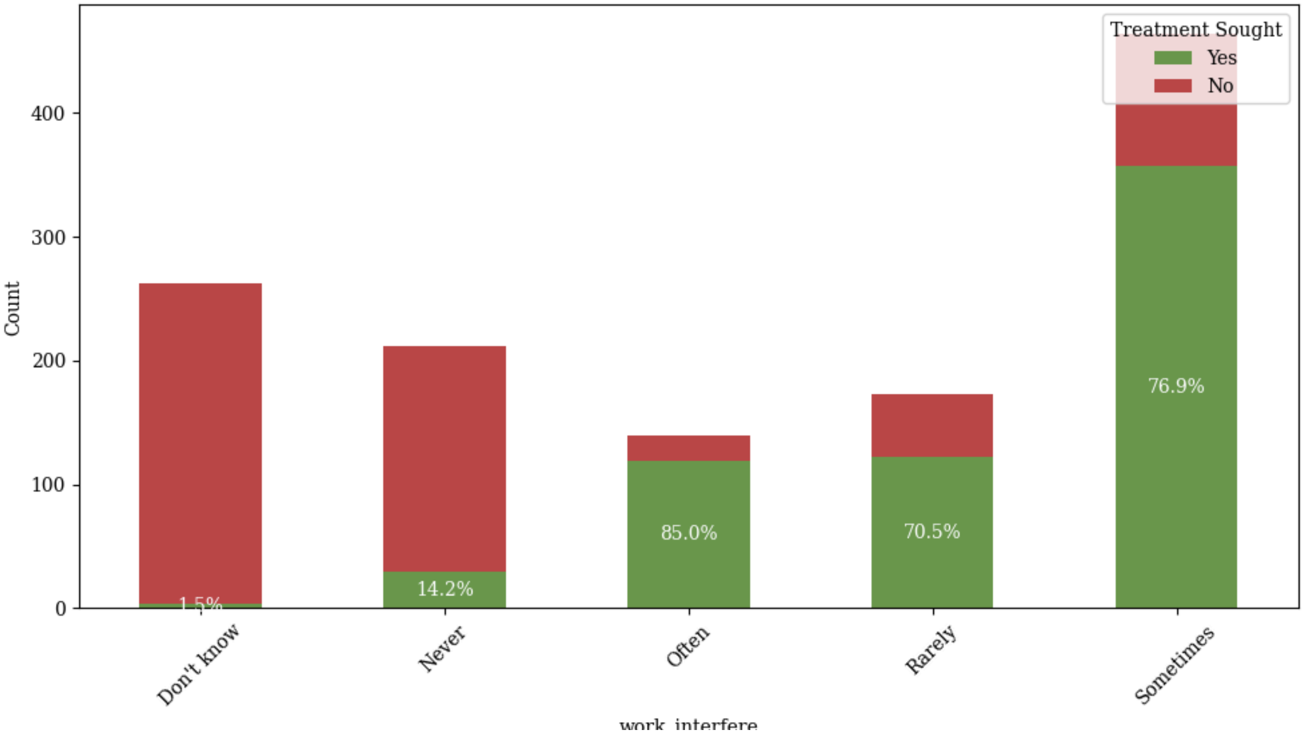
Treatment Sought
🟩 Yes
🟥 No
unt
400
500
600
700
800



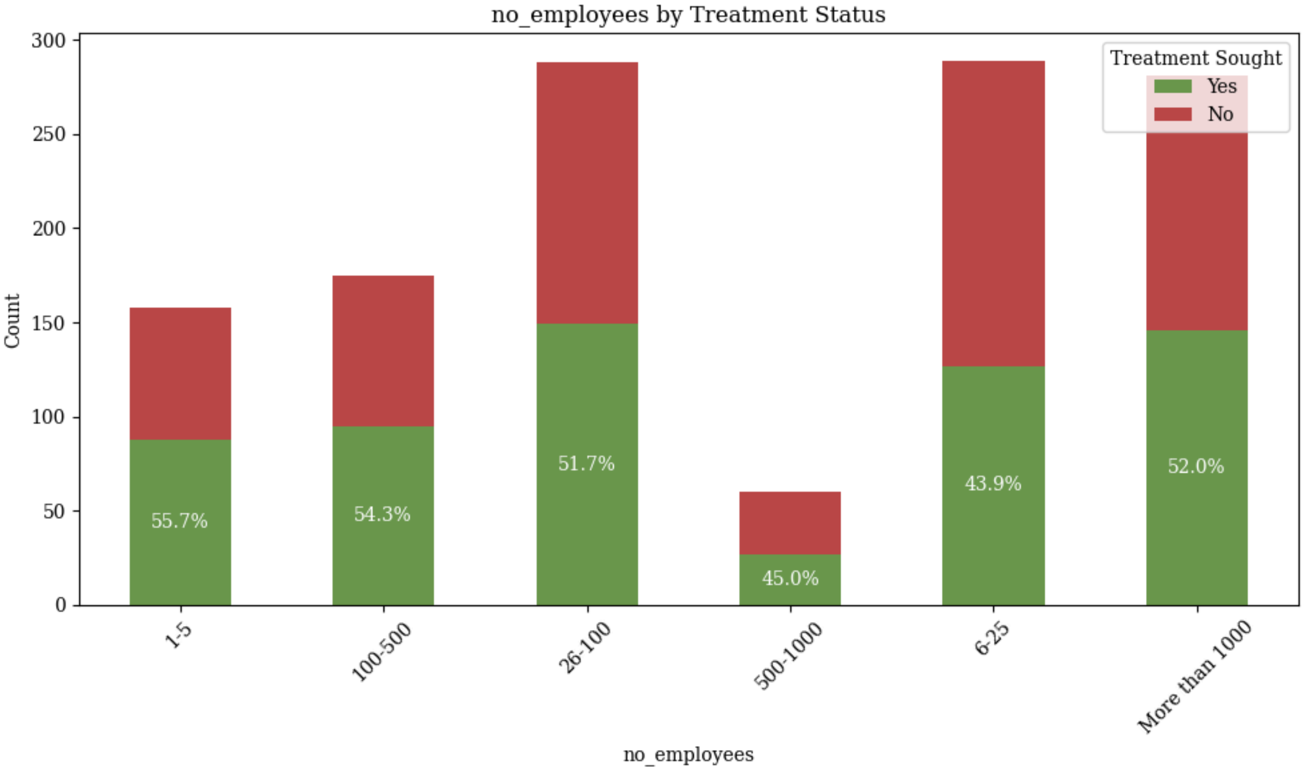
treatment by Treatment Status

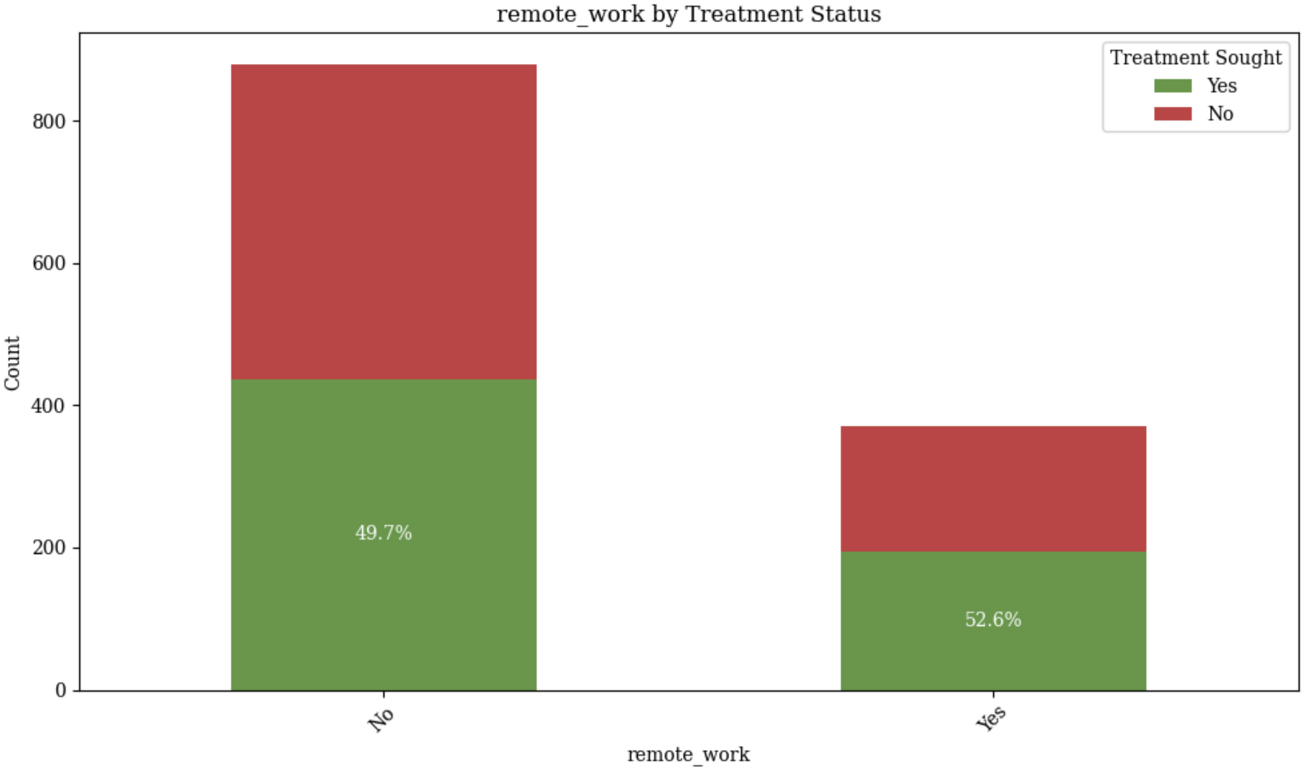


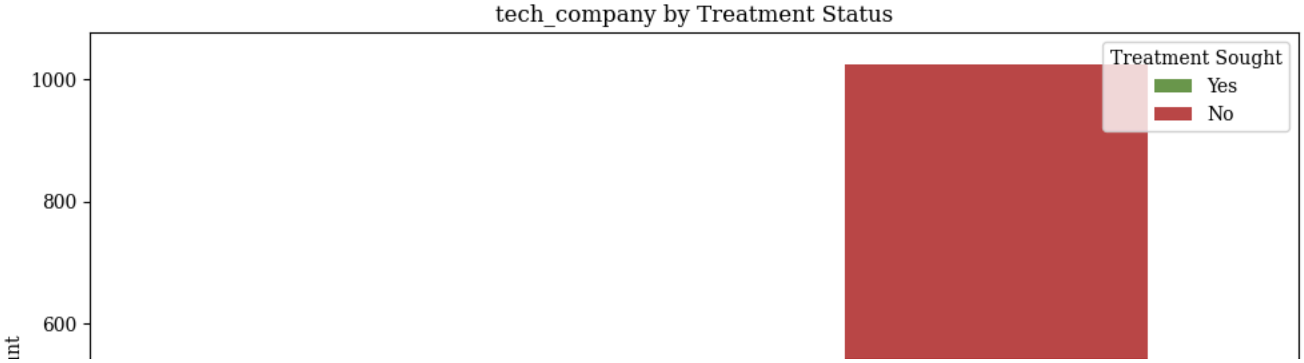
work_interfere by Treatment Status

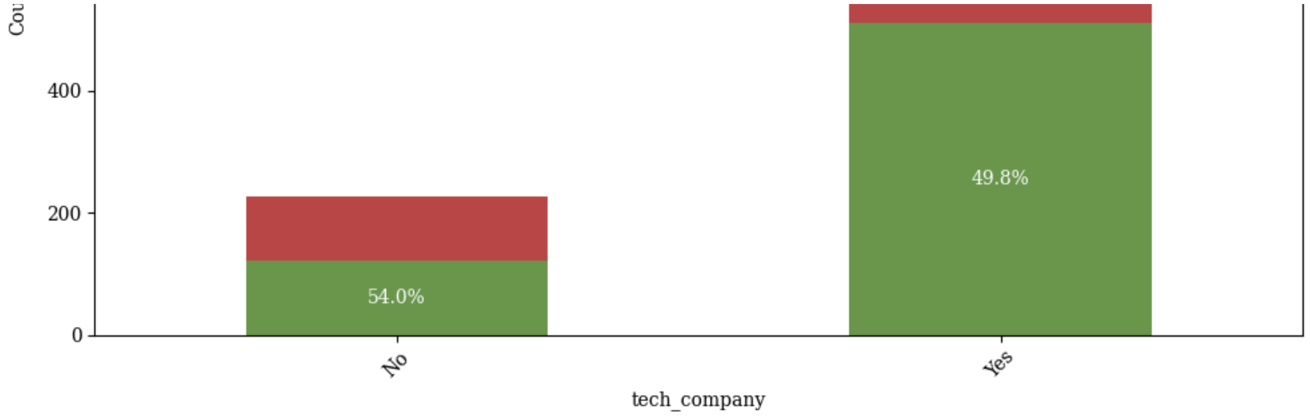


work_interfere

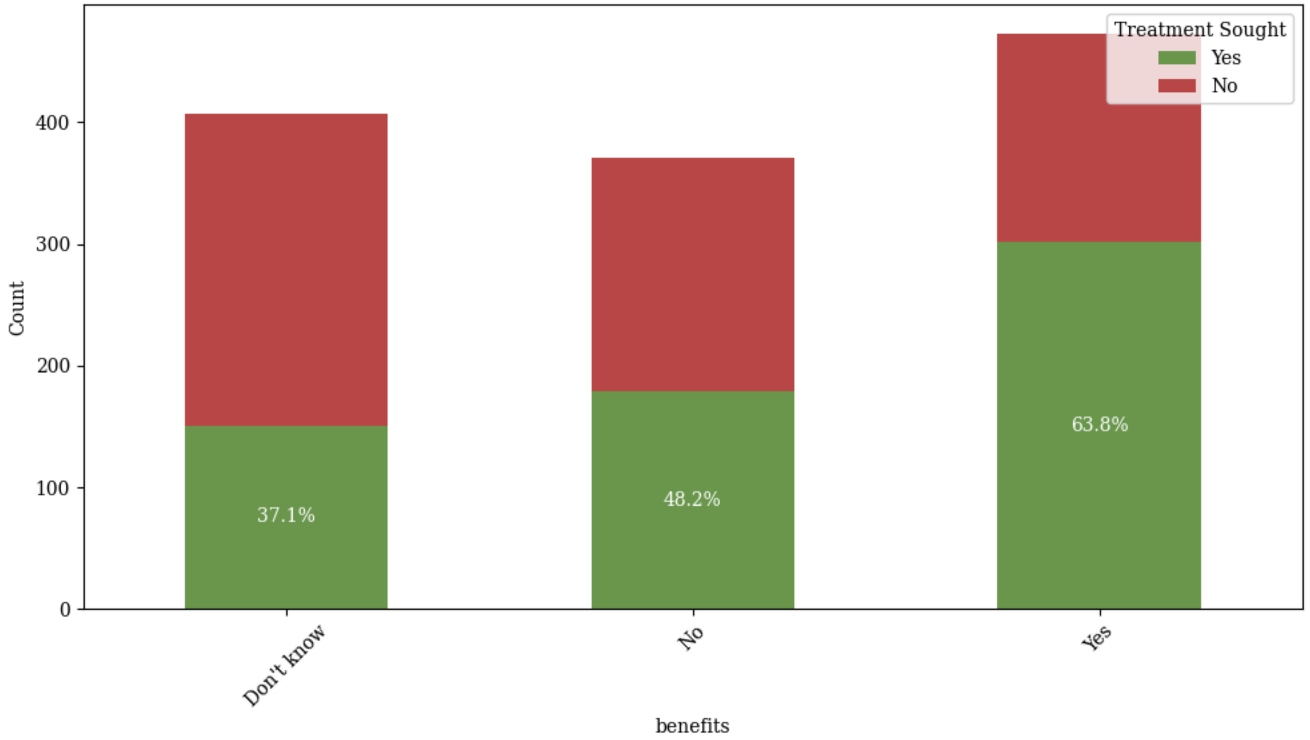




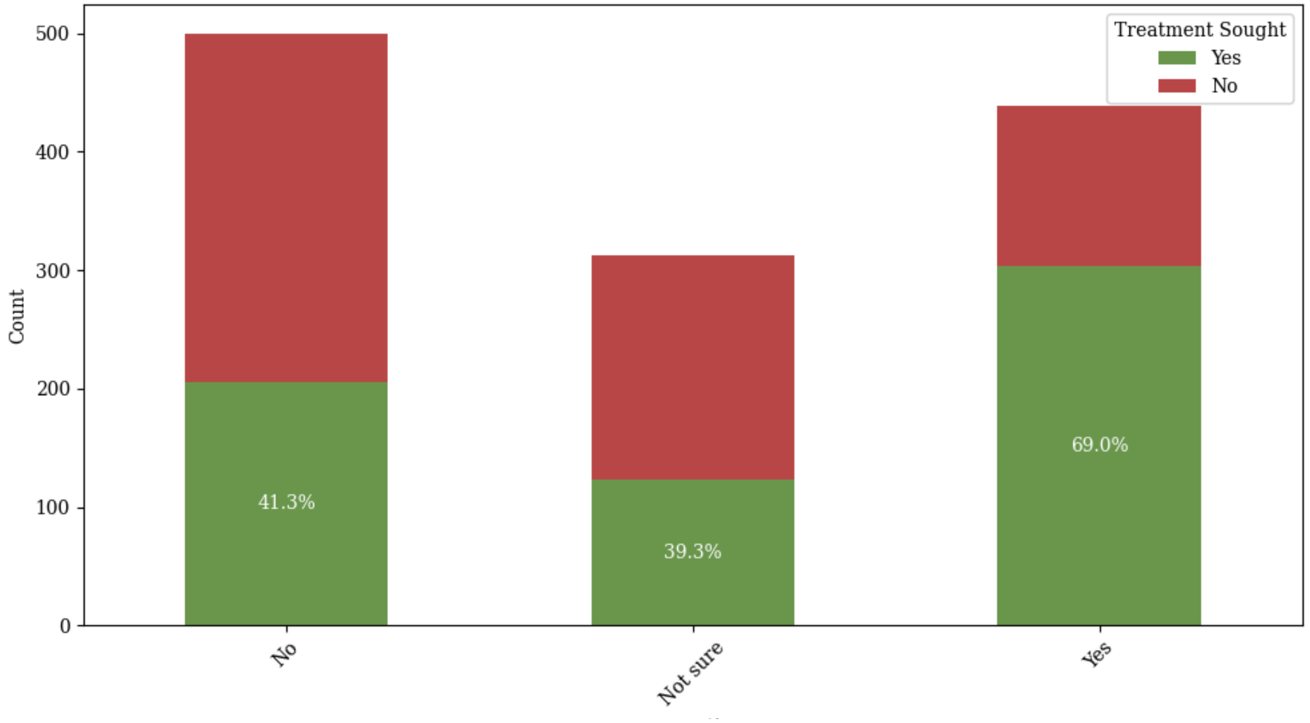




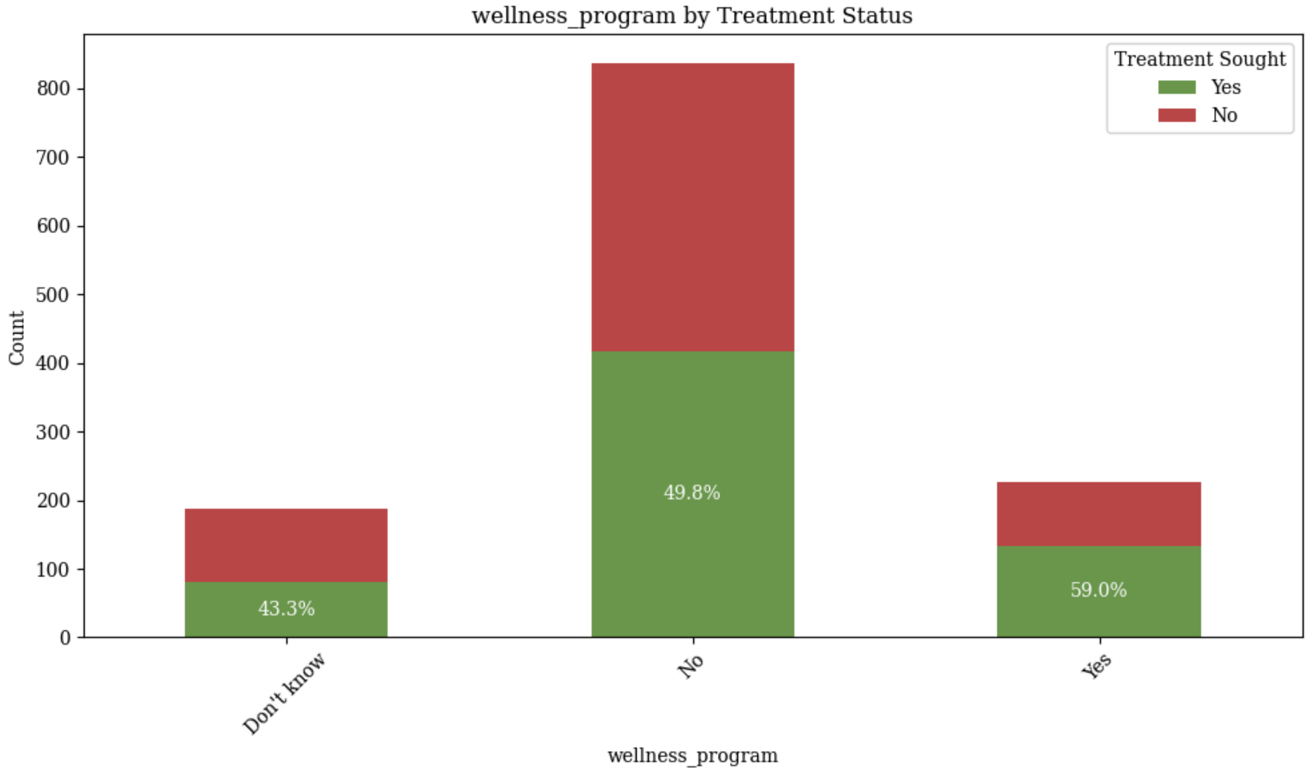
benefits by Treatment Status

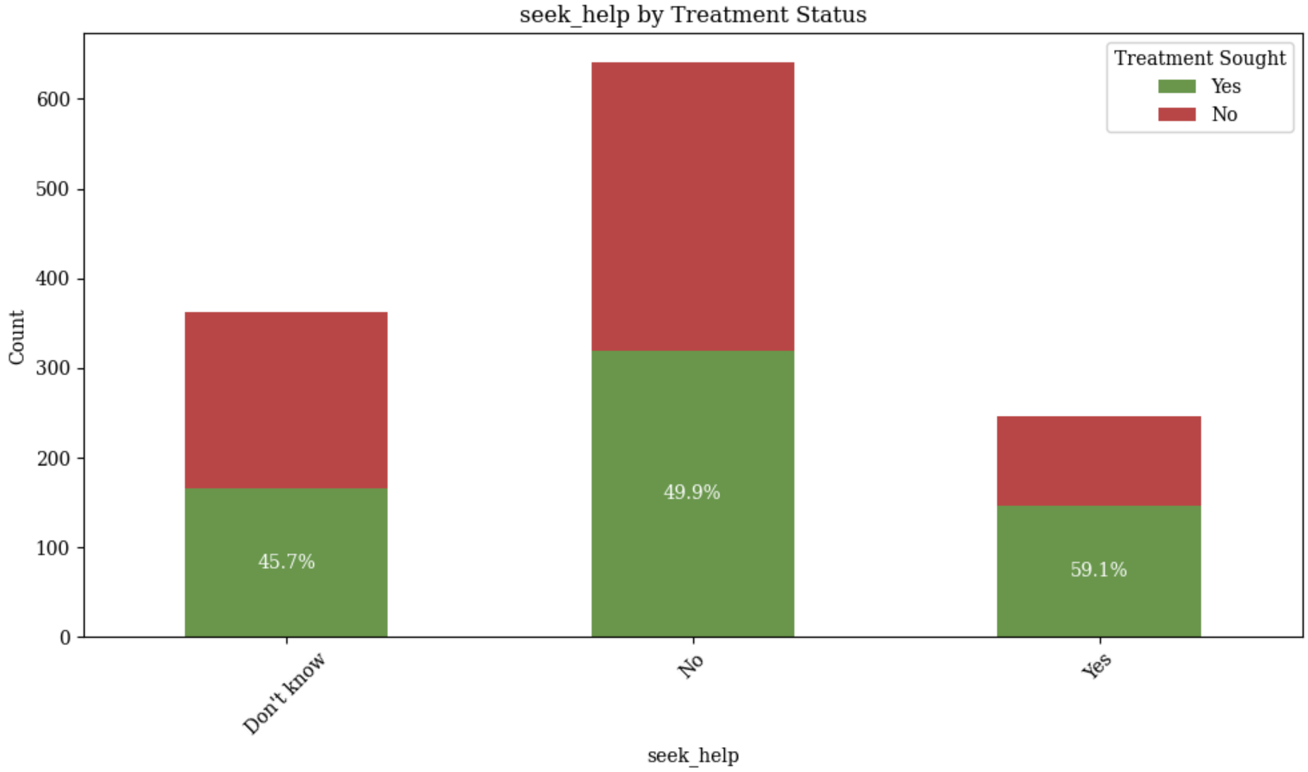


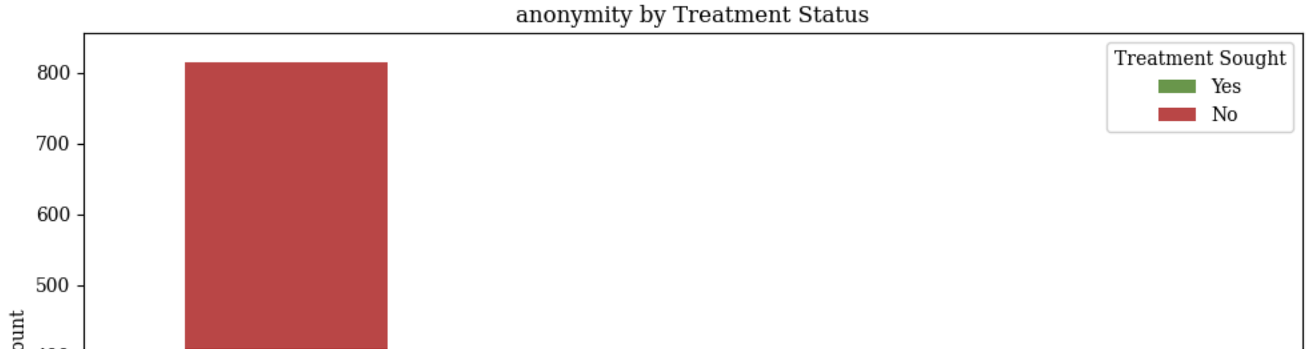
care_options by Treatment Status

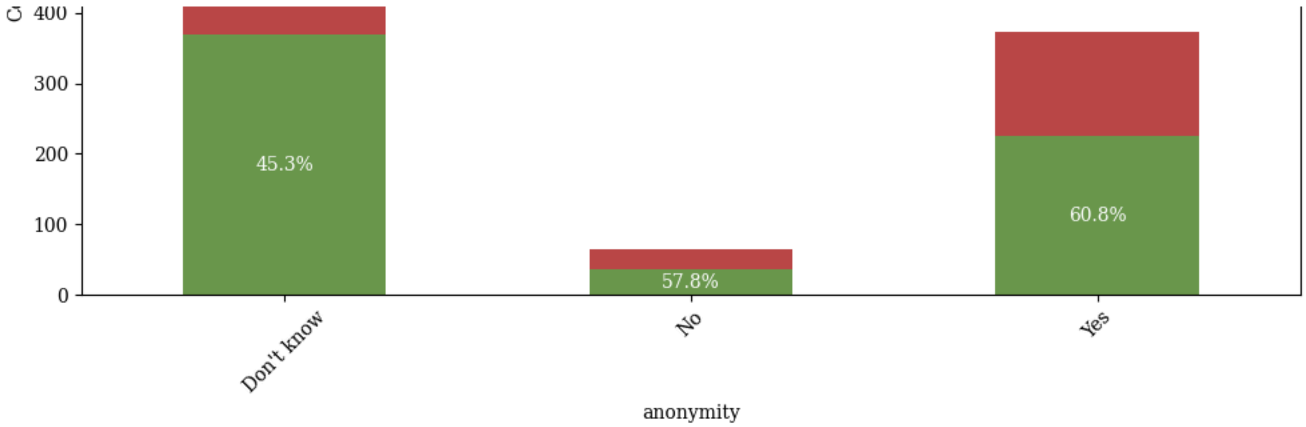


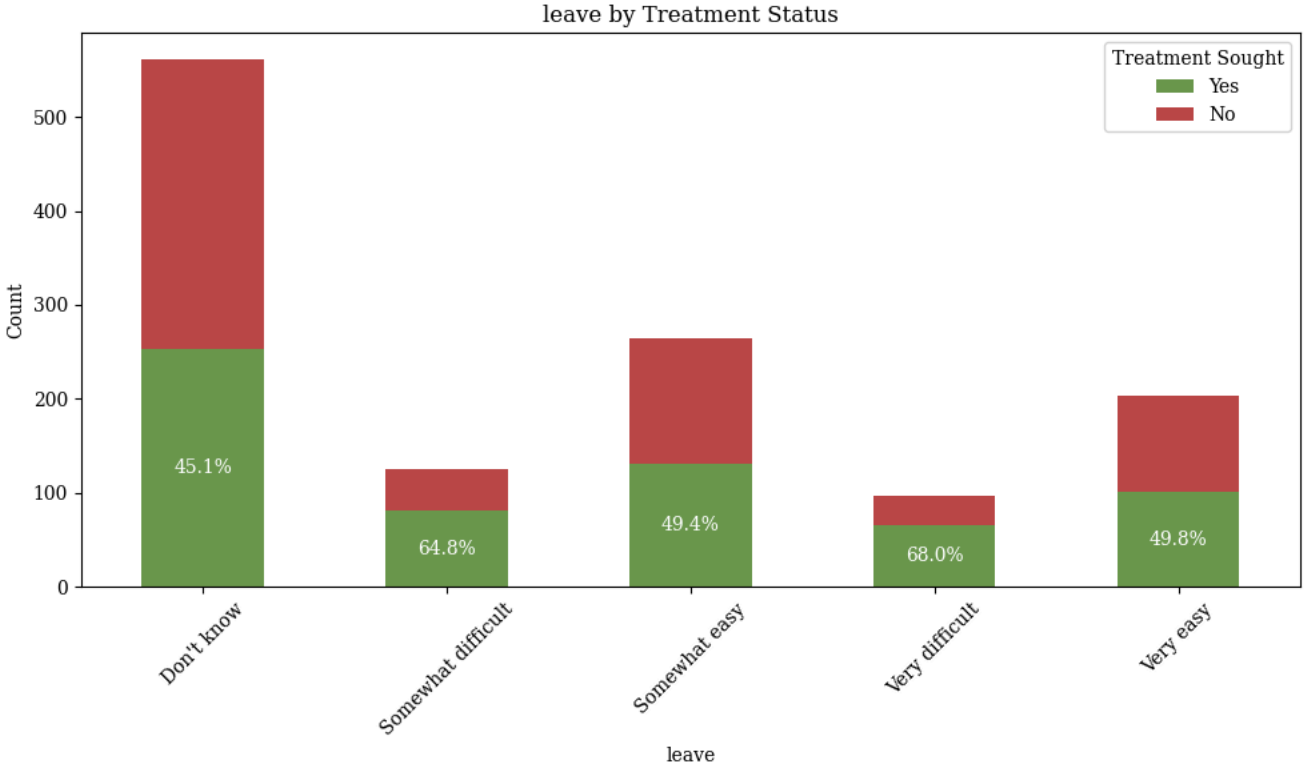
care_options

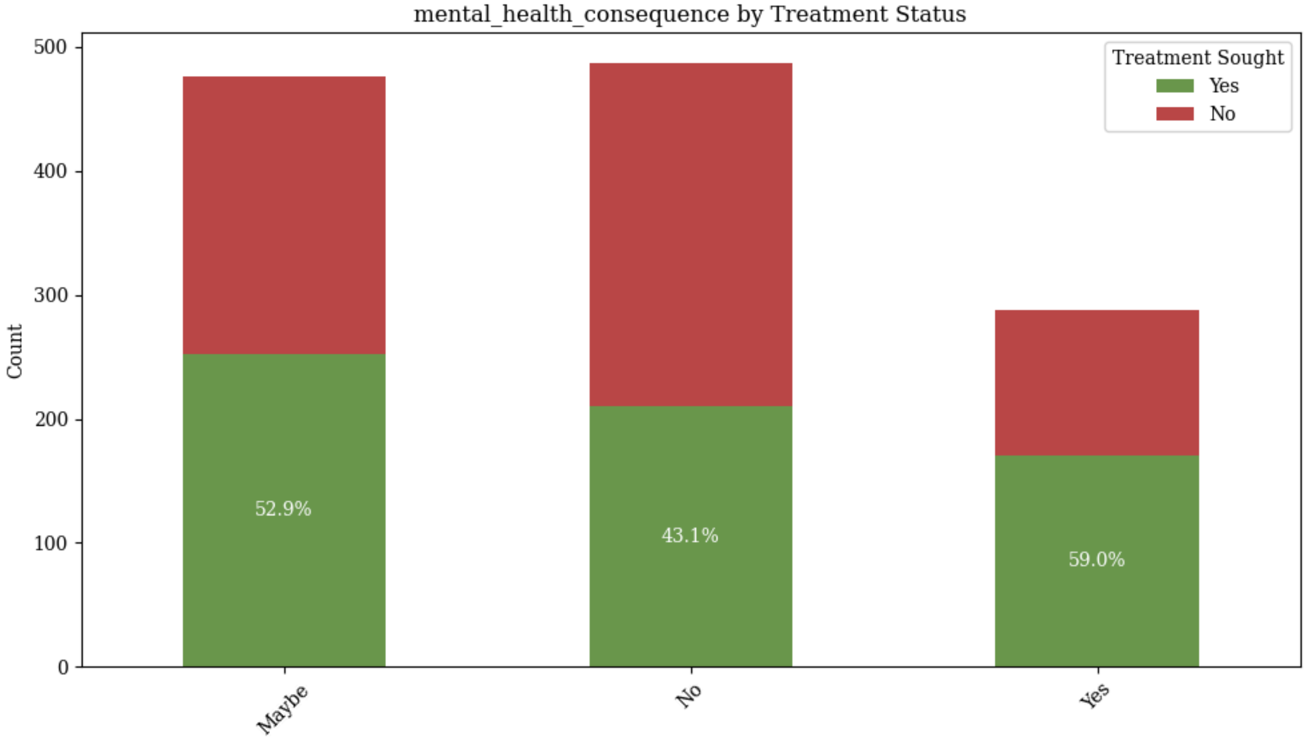




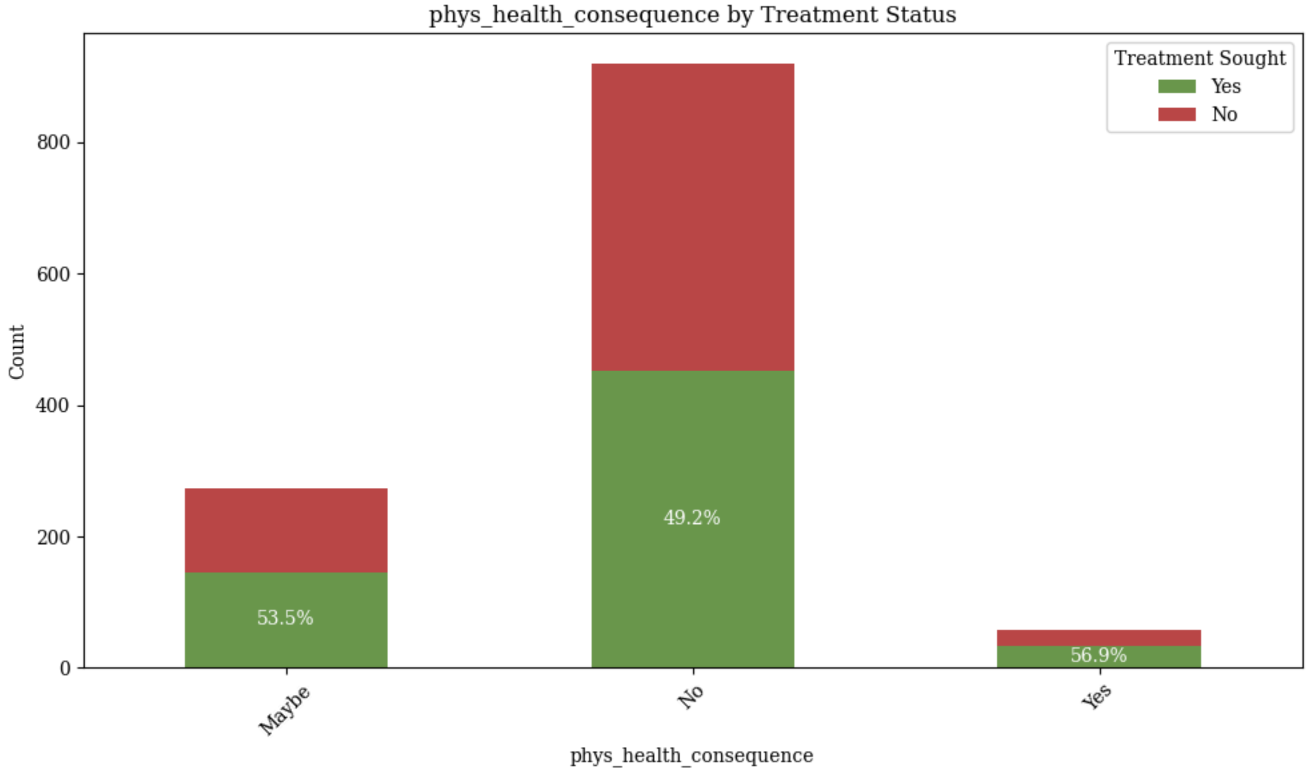


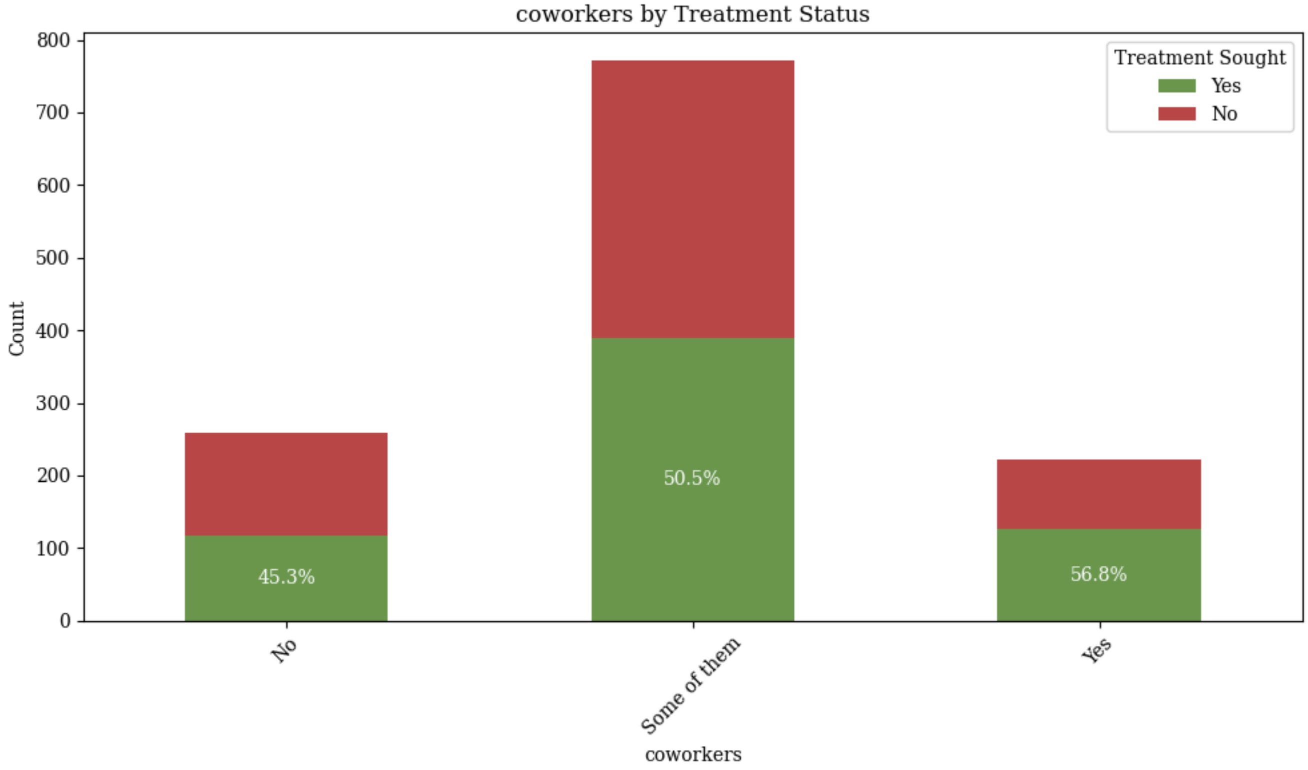


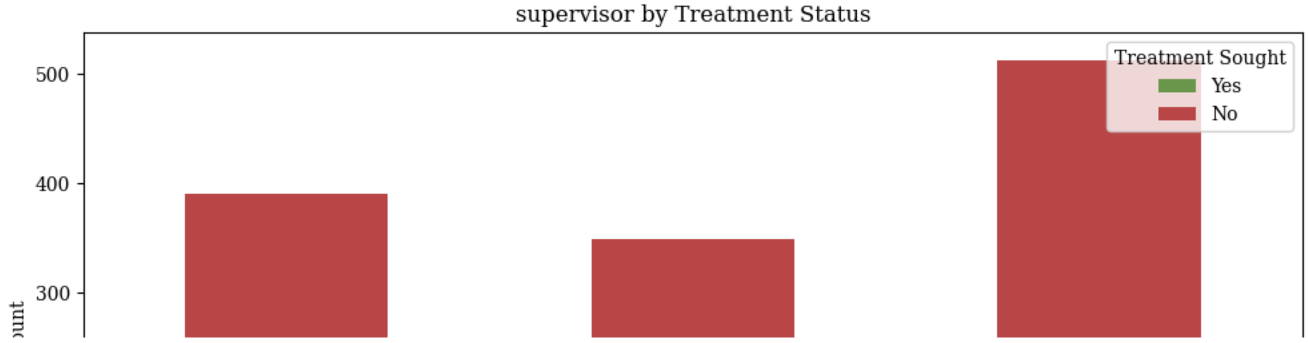

leave by Treatment Status


mental_health_consequence by Treatment Status

mental_health_consequence


phys_health_consequence by Treatment Status
Treatment Sought
🟩 Yes
🟥 No
Count
Maybe
No
Yes
phys_health_consequence


coworkers by Treatment Status
Treatment Sought
🟩 Yes
🟥 No
Count
No
Some of them
Yes
coworkers


supervisor by Treatment Status
Treatment Sought
🟩 Yes
🟥 No
ount