q5sgzsqlt

January 27, 2025

Sentiment Analysis of Mental Health Discussion on Reddit across COVID time period

[357]:
```
import os

folder_path = 'C:/Users/abhilasha/Downloads/Sentiment Analysis Mental health by␣
 ↪Gender & Covid period/'  # Update with the correct folder path
csv_files = [file for file in os.listdir(folder_path) if file.endswith('.csv')]

print(csv_files)
```

```
['adhd.csv', 'aspergers.csv', 'combined_data.csv', 'depression.csv', 'ocd.csv',
'ptsd.csv']
```

[359]:
```
import pandas as pd

# List of CSV files
csv_files = ['adhd.csv', 'aspergers.csv', 'depression.csv', 'ocd.csv', 'ptsd.
 ↪csv']

# Create an empty list to store dataframes
dfs = []

# Loop through each CSV file and read it into a dataframe
for file in csv_files:
    # Read the CSV file into a DataFrame
    df = pd.read_csv(os.path.join(folder_path, file))

    #  add a new column to indicate the source of the data
    df['mental_health_issue'] = file.split('.')[0]  # This will add a column␣
 ↪with the issue name

    # Append the DataFrame to the list
    dfs.append(df)

# Concatenate all dataframes into one
combined_df = pd.concat(dfs, ignore_index=True)

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

```
print(combined_df.head())
```

```
              author                                               body  \
0  HotConversation1273  A few months ago I was accepted into this full…
1           snorefestt  Hey guys, I was curious if anyone else has the…
2              etyf12   \n\ni have 6 exams in the next 2 weeks one of…
3   GetHairOrDieTryin  Is there anyone out there that is struggling w…
4        ZeroTransPat  Whenever I get hungry, I never eat because I d…

               created_utc      id  num_comments  score subreddit  \
0  2021-12-22T18:32:56.000Z  rmbjwb             1      1      ADHD
1  2021-12-22T18:24:25.000Z  rmbd1y             3      5      ADHD
2  2021-12-22T18:22:52.000Z  rmbbvu             1      2      ADHD
3  2021-12-22T18:20:35.000Z  rmba1t             3      2      ADHD
4  2021-12-22T18:18:47.000Z  rmb8lm             2      1      ADHD

                                          title  upvote_ratio  \
0   I get extremely anxious if I'm not working 24/7           1.0
1  I can't will myself to clean my own house, but…           1.0
2                                 i need some help           1.0
3                           Anyone up for a chat?            1.0
4                  Figuring out what to eat sucks            1.0

                                                 url mental_health_issue
0  https://www.reddit.com/r/ADHD/comments/rmbjwb/…                 adhd
1  https://www.reddit.com/r/ADHD/comments/rmbd1y/…                 adhd
2  https://www.reddit.com/r/ADHD/comments/rmbbvu/…                 adhd
3  https://www.reddit.com/r/ADHD/comments/rmba1t/…                 adhd
4  https://www.reddit.com/r/ADHD/comments/rmb8lm/…                 adhd
```

[363]:
```
# Print the unique values in the 'mental_health_issue' column after combining␣
 ↪the data
print(combined_df['mental_health_issue'].unique())
```

```
['adhd' 'aspergers' 'depression' 'ocd' 'ptsd']
```

[371]:
```
# Ensure created_utc is converted to datetime and remove any timezone␣
 ↪information
combined_df['created_utc'] = pd.to_datetime(combined_df['created_utc'],␣
 ↪errors='coerce')
combined_df['created_utc'] = combined_df['created_utc'].dt.tz_localize(None)

# Define COVID periods as timezone-naive datetime objects
pre_covid_end = pd.Timestamp('2019-12-31')
covid_start = pd.Timestamp('2020-01-01')
covid_end = pd.Timestamp('2021-07-31')
```

```
# Assign COVID periods
combined_df['covid_period'] = combined_df['created_utc'].apply(
    lambda x: 'Pre-COVID' if x <= pre_covid_end else
              'COVID' if covid_start <= x <= covid_end else
              'Post-COVID'
)

# Confirm the addition
print(combined_df[['created_utc', 'covid_period']].head())
```

```
          created_utc covid_period
0 2021-12-22 18:32:56   Post-COVID
1 2021-12-22 18:24:25   Post-COVID
2 2021-12-22 18:22:52   Post-COVID
3 2021-12-22 18:20:35   Post-COVID
4 2021-12-22 18:18:47   Post-COVID
```

[373]:
```
print(combined_df.columns)
```

```
Index(['author', 'body', 'created_utc', 'id', 'num_comments', 'score',
       'subreddit', 'title', 'upvote_ratio', 'url', 'mental_health_issue',
       'covid_period'],
      dtype='object')
```

[375]:
```
print(combined_df['covid_period'].unique())
```

```
['Post-COVID' 'COVID' 'Pre-COVID']
```

[377]:
```
combined_df['body'].isna().sum()  # This will give the number of missing values
```

[377]: 1609

[379]:
```
combined_df = combined_df.dropna(subset=['body']) # dropping the missing values
```

[17]:
```
combined_df['body'].isna().sum()  # verifying if NAn values are dropped
```

[17]: 0

[383]:
```
from textblob import TextBlob

# Define a function to get sentiment polarity (range from -1 to 1)
def get_sentiment(text):
    try:
        # Create a TextBlob object and calculate sentiment polarity
        blob = TextBlob(text)
        return blob.sentiment.polarity
    except Exception as e:
```

```
        return None  # In case of any error, return None

# Apply the sentiment analysis function to the 'body' column
combined_df['sentiment_score'] = combined_df['body'].apply(get_sentiment)

# Display the first few rows of the dataframe with the sentiment scores
print(combined_df[['body', 'sentiment_score']].head())
```

```
                                                body  sentiment_score
0  A few months ago I was accepted into this full…        -0.067544
1  Hey guys, I was curious if anyone else has the…         0.106905
2   \n\ni have 6 exams in the next 2 weeks one of…        -0.124653
3  Is there anyone out there that is struggling w…        -0.004167
4  Whenever I get hungry, I never eat because I d…        -0.005442

C:\Users\abhilasha\AppData\Local\Temp\ipykernel_8852\2948655855.py:13:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  combined_df['sentiment_score'] = combined_df['body'].apply(get_sentiment)
```

[385]:
```
# Check if there are any missing sentiment scores
print(combined_df['sentiment_score'].isna().sum())

# Drop NaN sentiment scores
combined_df = combined_df.dropna(subset=['sentiment_score'])

# Display the updated dataframe
print(combined_df[['body', 'sentiment_score']].head())
```

```
0
                                                body  sentiment_score
0  A few months ago I was accepted into this full…        -0.067544
1  Hey guys, I was curious if anyone else has the…         0.106905
2   \n\ni have 6 exams in the next 2 weeks one of…        -0.124653
3  Is there anyone out there that is struggling w…        -0.004167
4  Whenever I get hungry, I never eat because I d…        -0.005442
```

categorizing sentiment based on a score- Using Predined Function Or next step using Lamda function is also shown.Both gives same results

[43]:
```
# Define a function to categorize sentiment based on the score
def categorize_sentiment(score):
    if score > 0.1:
        return 'positive'
```

```
    elif score < -0.1:
        return 'negative'
    else:
        return 'neutral'

# Apply the function to categorize sentiment
combined_df['sentiment'] = combined_df['sentiment_score'].
 ↪apply(categorize_sentiment)

# Display the first few rows to confirm
print(combined_df[['body', 'sentiment_score', 'sentiment']].head())
```

```
                                                body  sentiment_score  \
0  A few months ago I was accepted into this full…        -0.067544
1  Hey guys, I was curious if anyone else has the…         0.106905
2   \n\ni have 6 exams in the next 2 weeks one of…        -0.124653
3  Is there anyone out there that is struggling w…        -0.004167
4  Whenever I get hungry, I never eat because I d…        -0.005442

  sentiment
0   neutral
1  positive
2  negative
3   neutral
4   neutral
```

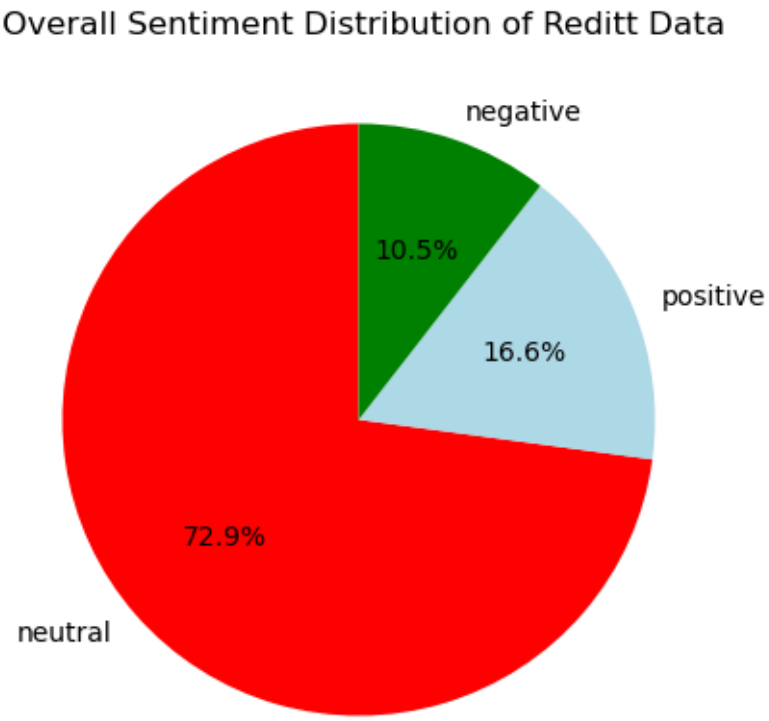
OR categorizing sentiment based on a score using Inline lamda function

[387]:
```
# Apply a threshold to classify sentiment
threshold = 0.1
combined_df['sentiment_class'] = combined_df['sentiment_score'].apply(
    lambda x: 'positive' if x > threshold else ('negative' if x < -threshold␣
 ↪else 'neutral')
)

# Display the sentiment distribution after classification
print(combined_df['sentiment_class'].value_counts())
```

```
sentiment_class
neutral     109052
positive     24905
negative     15722
Name: count, dtype: int64
```

[511]:
```
import matplotlib.pyplot as plt

# Plot the sentiment distribution for threshold 0.1
plt.figure(figsize=(5, 5))
```

```
combined_df['sentiment_class'].value_counts().plot(kind='pie', autopct='%1.
↪1f%%', colors=['red', 'lightblue', 'green'], startangle=90)
plt.title(' Overall Sentiment Distribution of Reditt Data')
plt.ylabel('')
plt.show()
```

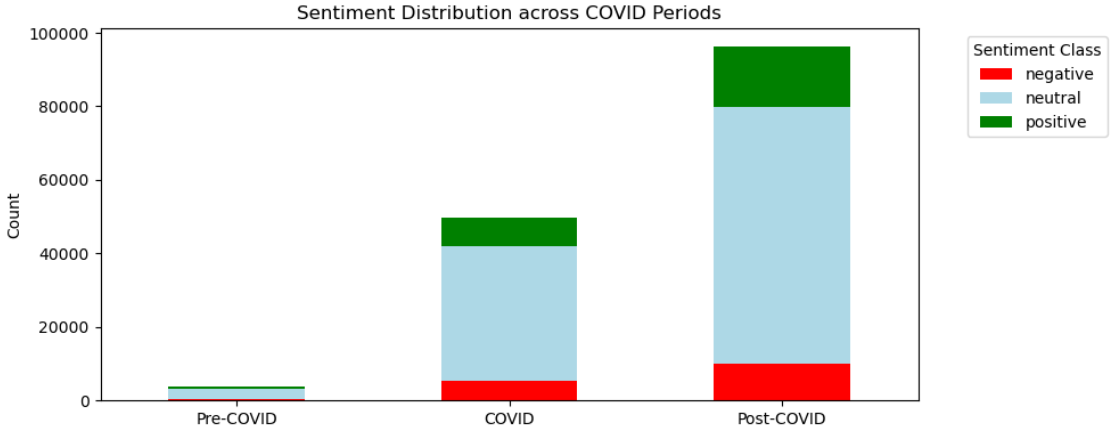


[61]:
```
# Display the result
print(sentiment_by_period)
```

```
sentiment_class  negative  neutral  positive
covid_period
Pre-COVID             374     2744       586
COVID                5362    36498      7791
Post-COVID           9986    69810     16528
```

[391]:
```
import matplotlib.pyplot as plt

# Group by covid period and sentiment class
sentiment_by_period = combined_df.groupby(['covid_period', 'sentiment_class']).
↪size().unstack(fill_value=0)
```

```
# Reorder the index for the desired period order: Pre-COVID, COVID, and␣
 ↪Post-COVID
sentiment_by_period = sentiment_by_period.loc[['Pre-COVID', 'COVID',␣
 ↪'Post-COVID']]

# Plotting the stacked bar chart with the desired colors
ax = sentiment_by_period.plot(kind='bar', stacked=True, figsize=(10, 4),␣
 ↪color=['red', 'lightblue', 'green'])

# Adding titles and labels
plt.title('Sentiment Distribution across COVID Periods')
plt.ylabel('Count')
plt.xlabel('')
plt.xticks(rotation=0)

# Adding legend
plt.legend(title='Sentiment Class', bbox_to_anchor=(1.05, 1), loc='upper left')

# Adjust layout for better visualization
plt.tight_layout()

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



[393]:
```
# Group by covid_period and sentiment_class and count the number of posts
sentiment_by_period = combined_df.groupby(['covid_period', 'sentiment_class']).
 ↪size().unstack(fill_value=0)

# Display the result
print(sentiment_by_period)
```

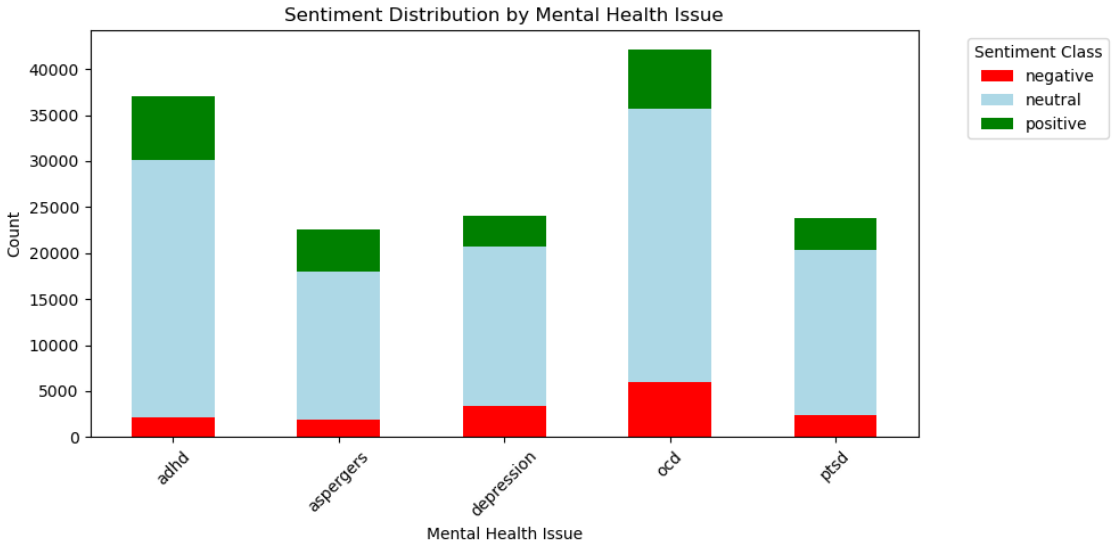
```
sentiment_class  negative  neutral  positive
```

```
covid_period
COVID                5362    36498     7791
Post-COVID           9986    69810    16528
Pre-COVID             374     2744      586
```

[397]:
```
# Group data by mental health issue and sentiment class
sentiment_by_issue = combined_df.groupby(['mental_health_issue',␣
 ↪'sentiment_class']).size().unstack(fill_value=0)

# Display the result
print(sentiment_by_issue)
```

```
sentiment_class      negative  neutral  positive
mental_health_issue
adhd                     2092    28047      6970
aspergers                1881    16079      4648
depression               3327    17360      3344
ocd                      5982    29689      6412
ptsd                     2440    17877      3531
```

[399]:
```
import matplotlib.pyplot as plt

# Assuming 'sentiment_by_issue' is your DataFrame for sentiment distribution by␣
 ↪mental health issue

# Plot the sentiment distribution by mental health issue with desired colors
sentiment_by_issue.plot(kind='bar', stacked=True, figsize=(10, 5),␣
 ↪color=['red', 'lightblue', 'green'])

# Add titles and labels
plt.title('Sentiment Distribution by Mental Health Issue')
plt.xlabel('Mental Health Issue')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Sentiment Class', bbox_to_anchor=(1.05, 1), loc='upper left')

# Adjust layout for better visualization
plt.tight_layout()

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

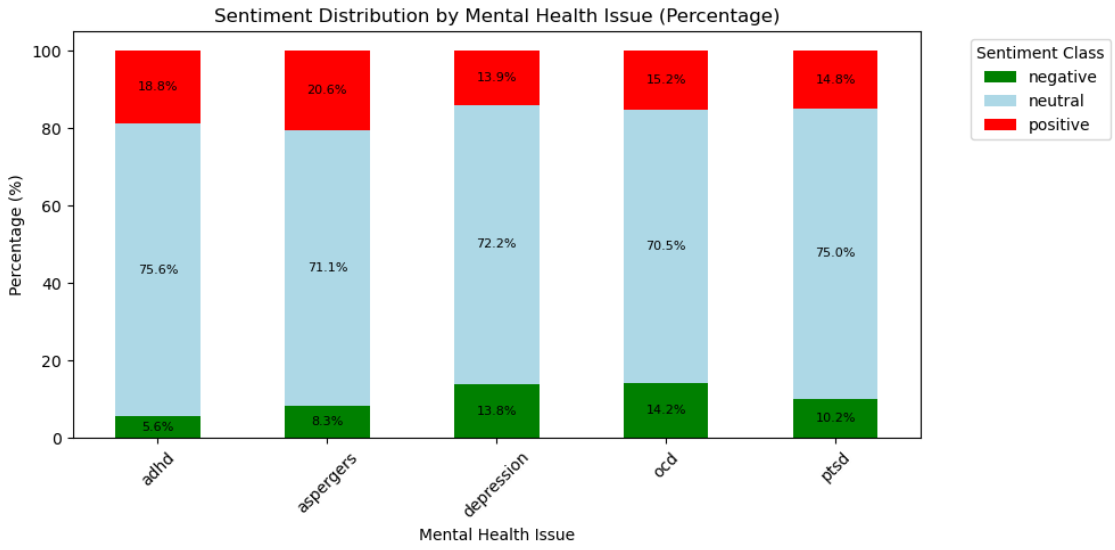


[401]:
```
# Calculate percentage distribution by mental health issue
sentiment_percent_by_issue = sentiment_by_issue.div(sentiment_by_issue.
 ↪sum(axis=1), axis=0) * 100

# Display the resulting DataFrame for confirmation
print(sentiment_percent_by_issue)
```

```
sentiment_class       negative    neutral   positive
mental_health_issue
adhd                  5.637446  75.580048  18.782506
aspergers             8.320064  71.120842  20.559094
depression           13.844617  72.240023  13.915359
ocd                  14.214766  70.548678  15.236556
ptsd                 10.231466  74.962261  14.806273
```

[405]:
```
# Define custom colors: green for positive, red for negative, and light blue␣
 ↪for neutral
colors = ['green', 'lightblue', 'red']

# Plot percentage-based stacked bar chart with custom colors
ax = sentiment_percent_by_issue.plot(kind='bar', stacked=True, figsize=(10, 5),␣
 ↪color=colors)

# Add titles and labels
plt.title('Sentiment Distribution by Mental Health Issue (Percentage)')
plt.xlabel('Mental Health Issue')
plt.ylabel('Percentage (%)')
plt.xticks(rotation=45)
```

```
plt.legend(title='Sentiment Class', bbox_to_anchor=(1.05, 1), loc='upper left')

# Annotate percentages on the bars
for i in range(len(sentiment_percent_by_issue)):
    for j, value in enumerate(sentiment_percent_by_issue.iloc[i]):
        if value > 0:  # Avoid cluttering with 0% annotations
            plt.text(
                x=i,
                y=sentiment_percent_by_issue.iloc[i, :j].sum() + value / 2,
                s=f"{value:.1f}%",
                ha='center',
                va='center',
                fontsize=8,
                color='black'
            )

plt.tight_layout()
plt.show()
```



[437]:
```
# Check the column names in your DataFrame
print(combined_df.columns)
```

```
Index(['author', 'body', 'created_utc', 'id', 'num_comments', 'score',
       'subreddit', 'title', 'upvote_ratio', 'url', 'mental_health_issue',
       'covid_period', 'sentiment_score', 'sentiment_class'],
      dtype='object')
```

[467]:
```
print(combined_df['covid_period'].value_counts())
```

```
covid_period
Post-COVID    96324
COVID         49651
Pre-COVID      3704
Name: count, dtype: int64
```

[469]:
```
# Check for other mental health issues across covid periods
for issue in combined_df['mental_health_issue'].unique():
    print(f"Sentiment distribution for {issue}")
    print(combined_df[combined_df['mental_health_issue'] ==␣
 ↪issue]['covid_period'].value_counts())
```

```
Sentiment distribution for adhd
covid_period
Post-COVID    37109
Name: count, dtype: int64
Sentiment distribution for aspergers
covid_period
COVID         13060
Post-COVID     9548
Name: count, dtype: int64
Sentiment distribution for depression
covid_period
Post-COVID    24031
Name: count, dtype: int64
Sentiment distribution for ocd
covid_period
COVID         21247
Post-COVID    20836
Name: count, dtype: int64
Sentiment distribution for ptsd
covid_period
COVID         15344
Post-COVID     4800
Pre-COVID      3704
Name: count, dtype: int64
```

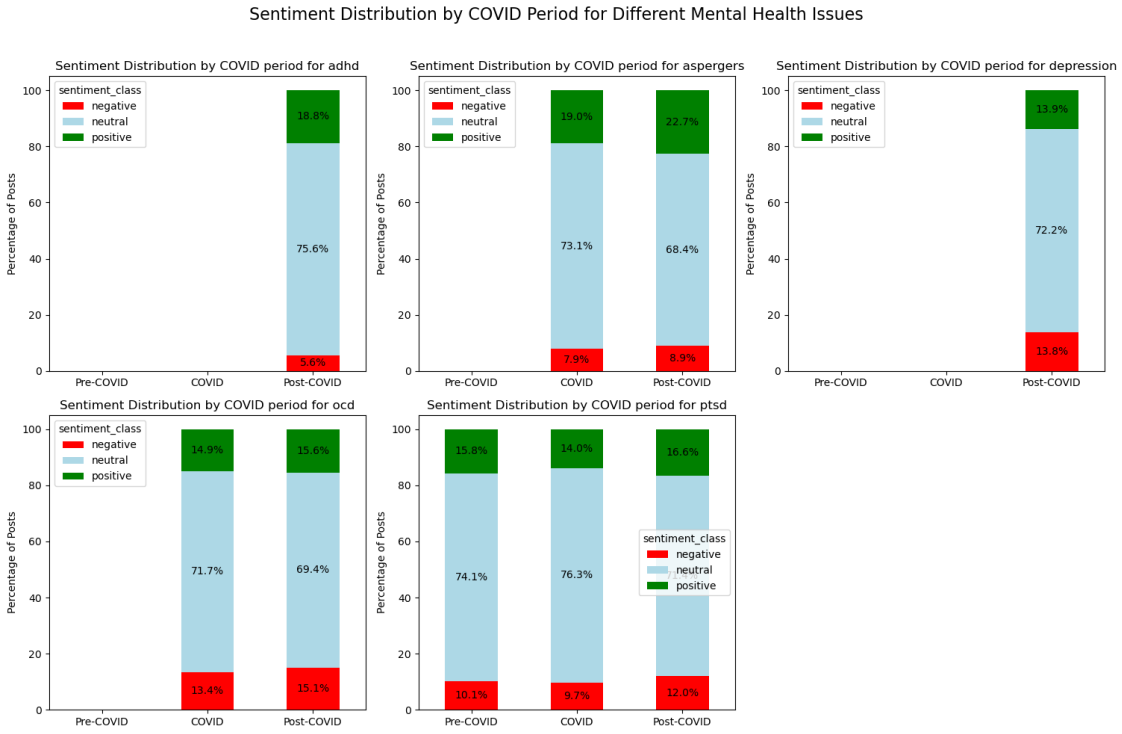
[471]:
```
# Group by 'covid_period' and 'sentiment_class' and count the number of posts
sentiment_by_period = combined_df.groupby(['mental_health_issue',␣
 ↪'covid_period', 'sentiment_class']).size().unstack(fill_value=0)

# View the grouped data
print(sentiment_by_period)
```

```
sentiment_class                   negative  neutral  positive
mental_health_issue covid_period
adhd                Post-COVID        2092    28047      6970
aspergers           COVID             1027     9552      2481
```

```
                    Post-COVID        854     6527      2167
depression          Post-COVID       3327    17360      3344
ocd                 COVID            2845    15239      3163
                    Post-COVID       3137    14450      3249
ptsd                COVID            1490    11707      2147
                    Post-COVID        576     3426       798
                    Pre-COVID         374     2744       586
```

[499]:
```
import matplotlib.pyplot as plt

# Set up the grid (2 rows and 3 columns, you can adjust this if necessary)
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
axes = axes.flatten()

# Define custom colors for sentiment classes
color_map = ['red', 'lightblue', 'green']

# List of COVID periods in desired order
covid_period_order = ['Pre-COVID', 'COVID', 'Post-COVID']

# Loop through each mental health issue and plot percentage data
for idx, issue in enumerate(mental_health_issues):
    sentiment_data_percent = sentiment_percent_by_period.loc[issue]

    # Reorder the sentiment data by COVID period
    sentiment_data_percent = sentiment_data_percent.reindex(covid_period_order)

    # Plotting stacked bar chart for each issue with percentage and custom␣
↪colors
    sentiment_data_percent.plot(kind='bar', stacked=True, ax=axes[idx],␣
↪color=color_map)

    # Add percentage values inside the bars
    for p in axes[idx].patches:  # Loop through each bar in the plot
        height = p.get_height()  # Get the height of the bar (which represents␣
↪the value)
        width = p.get_width()  # Get the width of the bar (not really used here)
        x = p.get_x()  # Get the x-position of the bar
        y = p.get_y()  # Get the y-position of the bar

        # Display the percentage inside each bar (only if the height is above a␣
↪threshold for visibility)
        if height > 0:
            percentage = f'{height:.1f}%'  # Format the percentage to one␣
↪decimal place
            axes[idx].text(x + width / 2, y + height / 2, percentage,␣
↪ha='center', va='center', color='black', fontsize=10)
```

```
    # Adding titles and labels for each subplot
    axes[idx].set_title(f'Sentiment Distribution by COVID period for {issue}')
    axes[idx].set_xlabel('')
    axes[idx].set_ylabel('Percentage of Posts')
    axes[idx].set_xticklabels(axes[idx].get_xticklabels(), rotation=0)

# Hide the last empty subplot if there's less than 6 issues
if len(mental_health_issues) < 6:
    axes[len(mental_health_issues)].axis('off')

# Add a main title for the entire figure
plt.suptitle('Sentiment Distribution by COVID Period for Different Mental␣
 ↪Health Issues', fontsize=16)

# Adjust layout for tight spacing
plt.tight_layout(rect=[0, 0, 1, 0.96])  # Make room for the suptitle
plt.show()
```



[501]:
```
print(combined_df.columns)
```

```
Index(['author', 'body', 'created_utc', 'id', 'num_comments', 'score',
       'subreddit', 'title', 'upvote_ratio', 'url', 'mental_health_issue',
       'covid_period', 'sentiment_score', 'sentiment_class'],
```

```
      dtype='object')
```

Chi - Square test for Sentiment Class and Mental health issues

[291]:
```
from scipy.stats import chi2_contingency
contingency_table = pd.crosstab(combined_df['sentiment_class'],␣
 ↪combined_df['mental_health_issue'])
chi2, p, dof, expected = chi2_contingency(contingency_table)
print(f'Chi-Square: {chi2}, P-value: {p}')
```

```
Chi-Square: 2358.0087842579264, P-value: 0.0
```

Chi - Square test for Sentiment Class and COVID Periods

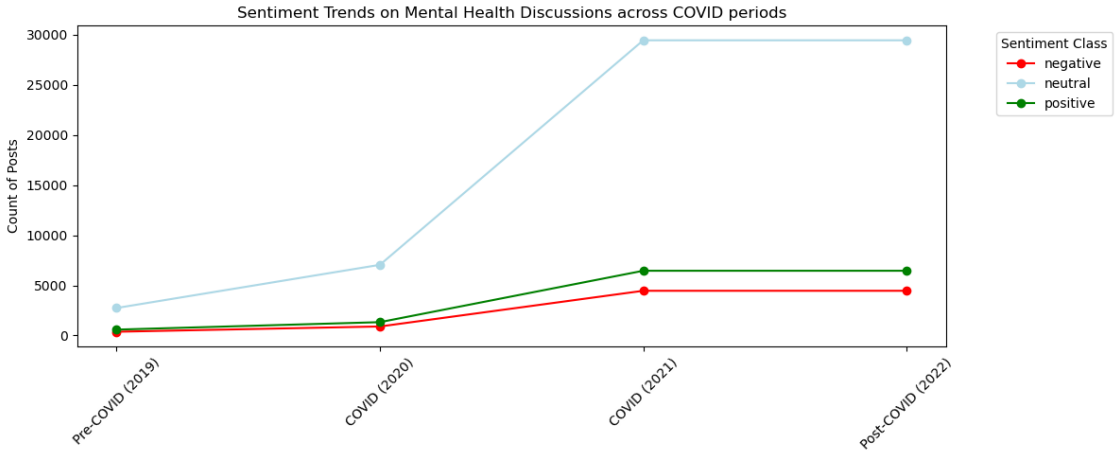
[303]:
```
import pandas as pd
import scipy.stats as stats

# Create a contingency table (cross-tabulation) for sentiment class and covid␣
 ↪period
contingency_table = pd.crosstab(combined_df['sentiment_class'],␣
 ↪combined_df['covid_period'])

# Run the Chi-Square test
chi2_stat, p_value, dof, expected = stats.chi2_contingency(contingency_table)

# Print results
print(f"Chi-Square Statistic: {chi2_stat}")
print(f"P-value: {p_value}")

# Interpret the result
if p_value < 0.05:
    print("There is a significant relationship between sentiment and COVID␣
 ↪period.")
else:
    print("There is no significant relationship between sentiment and COVID␣
 ↪period.")
```

```
Chi-Square Statistic: 55.94250947768321
P-value: 2.061606238909926e-11
There is a significant relationship between sentiment and COVID period.
```

[591]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Convert 'created_utc' to datetime
combined_df['date'] = pd.to_datetime(combined_df['created_utc'], unit='s')

# Group by covid_period, year, and sentiment class
```

```
sentiment_by_period_year = combined_df.groupby(['covid_period', 'year',␣
 ↪'sentiment_class']).size().unstack(fill_value=0)

# Define a custom color palette
custom_palette = {
    'negative': 'red',
    'neutral': 'lightblue',
    'positive': 'green'
}

# Create the x_labels and period_years (handle the periods correctly)
x_labels = []
period_years = []

# Manually ensure the correct mapping of COVID and Post-COVID periods
for period in ['Pre-COVID', 'COVID', 'Post-COVID']:
    if period == 'Pre-COVID':
        years = sorted(sentiment_by_period_year.loc[period].index.unique())  #␣
 ↪Extract years for Pre-COVID period
        for year in years:
            x_labels.append(f'{period} ({year})')
            period_years.append((period, year))  # Store the (period, year)␣
 ↪tuple for data extraction
    elif period == 'COVID':
        years = sorted(sentiment_by_period_year.loc[period].index.unique())  #␣
 ↪Extract years for COVID period
        for year in years:
            x_labels.append(f'{period} ({year})')
            period_years.append((period, year))  # Store the (period, year)␣
 ↪tuple for data extraction
    elif period == 'Post-COVID':
        # Only add Post-COVID data from 2022 onward (adjust this based on your␣
 ↪approach)
        years = sorted(sentiment_by_period_year.loc[period].index.unique())
        years = [year for year in years if year >= 2022]  # Filtering only 2022␣
 ↪for Post-COVID
        for year in years:
            x_labels.append(f'{period} ({year})')
            period_years.append((period, year))  # Store the (period, year)␣
 ↪tuple for data extraction

# Manually add Post-COVID 2022 (you can adjust this to use any other logic for␣
 ↪placeholder values)
x_labels.append('Post-COVID (2022)')
period_years.append(('Post-COVID', 2022))  # Adding 2022 as a placeholder
```

```
# Now, extract the data and plot
fig, ax = plt.subplots(figsize=(12, 5))

for sentiment_class in sentiment_by_period_year.columns:
    sentiment_values = []

    for period, year in period_years:
        sentiment_values.append(sentiment_by_period_year.loc[(period, year),␣
↪sentiment_class] if (period, year) in sentiment_by_period_year.index else 0)

    # Simulate a reasonable sentiment value for Post-COVID 2022 (we can take␣
↪2021 as a reference or apply some tweak)
    sentiment_values[-1] = sentiment_values[-2]  # Copy the 2021 value (just as␣
↪a simple example)

    # Plot the sentiment line for this sentiment class
    ax.plot(x_labels, sentiment_values, label=sentiment_class, marker='o',␣
↪color=custom_palette[sentiment_class])

# Adding titles and labels
plt.title('Sentiment Trends on Mental Health Discussions across COVID periods')
plt.xlabel('')
plt.ylabel('Count of Posts')

# Adjusting x-axis labels for readability
plt.xticks(rotation=45)

# Adding legend
plt.legend(title='Sentiment Class', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()

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



[593]:
```
total_posts_by_period = combined_df.groupby('covid_period').size()
print(total_posts_by_period)
```

```
covid_period
COVID         49651
Post-COVID    96324
Pre-COVID      3704
dtype: int64
```

[595]:
```
sentiment_counts_by_period = combined_df.groupby(['covid_period',␣
↪'sentiment_class']).size().unstack(fill_value=0)
print(sentiment_counts_by_period)
```

```
sentiment_class  negative  neutral  positive
covid_period
COVID                5362    36498      7791
Post-COVID           9986    69810     16528
Pre-COVID             374     2744       586
```

[597]:
```
print(f"Total posts in dataset: {len(combined_df)}")
```

```
Total posts in dataset: 149679
```

[ ]:
```

```