INITIAL MODULE IMPORT

```
# Linking google drive (comment out below if not needed)

#from google.colab import drive
#import os

#drive.mount('/content/drive', force_remount = True)

#notebook_path = r"/content/drive/MyDrive/Colab Notebooks/ML
Assignment/"
#os.chdir(notebook_path)
#!pwd

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

DATA PREPROCESSING - IMPORT, DATA SIZE OVERVIEW

```
data = pd.read_csv('mentalhealth_dataset.csv') #Read dataset CSV
print(len(data))
print("----------")
print(data.dtypes) #View types f
print("----------")
print(data.describe()) #Describes the data in detail(count, means,
standard deviations,...)
print("----------")
```

```
1000
----------
Timestamp                          object
Gender                             object
Age                                 int64
Course                             object
YearOfStudy                        object
CGPA                              float64
Depression                          int64
Anxiety                             int64
PanicAttack                         int64
SpecialistTreatment                 int64
SymptomFrequency_Last7Days          int64
HasMentalHealthSupport              int64
SleepQuality                        int64
StudyStressLevel                    int64
StudyHoursPerWeek                   int64
AcademicEngagement                  int64
```

```
dtype: object
----------
                Age         CGPA   Depression      Anxiety  PanicAttack  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000

mean     21.402000     3.122530     0.483000     0.474000     0.458000

std       2.373611     0.810961     0.499961     0.499573     0.498482

min      18.000000     2.000000     0.000000     0.000000     0.000000

25%      19.000000     2.250000     0.000000     0.000000     0.000000

50%      21.000000     3.250000     0.000000     0.000000     0.000000

75%      24.000000     4.000000     1.000000     1.000000     1.000000

max      25.000000     4.000000     1.000000     1.000000     1.000000


       SpecialistTreatment  SymptomFrequency_Last7Days  \
count          1000.000000                   1000.0000
mean              0.067000                      3.4980
std               0.250147                      2.3081
min               0.000000                      0.0000
25%               0.000000                      1.7500
50%               0.000000                      3.0000
75%               0.000000                      6.0000
max               1.000000                      7.0000

       HasMentalHealthSupport  SleepQuality  StudyStressLevel  \
count             1000.000000   1000.000000       1000.000000
mean                 0.067000      2.983000          3.045000
std                  0.250147      1.417999          1.417386
min                  0.000000      1.000000          1.000000
25%                  0.000000      2.000000          2.000000
50%                  0.000000      3.000000          3.000000
75%                  0.000000      4.000000          4.000000
max                  1.000000      5.000000          5.000000

       StudyHoursPerWeek  AcademicEngagement
count        1000.000000         1000.000000
mean            9.746000            3.055000
std             5.651497            1.422673
min             1.000000            1.000000
25%             5.000000            2.000000
50%             9.000000            3.000000
75%            15.000000            4.000000
```

```
max              19.000000              5.000000
----------
```

DATA PREPROCESSING - HANDLING POTENTIALLY MISSING DATA

```
print(data.isnull().sum())

Timestamp                   0
Gender                      0
Age                         0
Course                      0
YearOfStudy                 0
CGPA                        0
Depression                  0
Anxiety                     0
PanicAttack                 0
SpecialistTreatment         0
SymptomFrequency_Last7Days  0
HasMentalHealthSupport      0
SleepQuality                0
StudyStressLevel            0
StudyHoursPerWeek           0
AcademicEngagement          0
dtype: int64
```

No missing data, filling null values not necessary.

DATA PREPROCESSING - RENAMING COLUMNS

```
df = data

#Properly adding spaces and simplifying names
df.rename(columns={'HasMentalHealthSupport' : 'Mental
Support'},inplace=True)
df.rename(columns={'StudyHoursPerWeek':'StudyHour/Week'},inplace=True)
df.rename(columns={'PanicAttack' : 'Panic Attack'},inplace=True)
df.rename(columns={'SpecialistTreatment' : 'Specialist
Treatment'},inplace=True)
df.rename(columns={'SymptomFrequency_Last7Days' : 'Symptom
Frequency'},inplace=True)
df.rename(columns={'SleepQuality' : 'Sleep Quality'},inplace=True)
df.rename(columns={'StudyStressLevel' : 'Study Stress
Level'},inplace=True)
df.rename(columns={'AcademicEngagement' : 'Academic
Engagement'},inplace=True)

#Replace "year 1" in YearOfStudy column to "Year 1"
df["YearOfStudy"] = df['YearOfStudy'].replace('year 1','Year 1')
df["YearOfStudy"] = df['YearOfStudy'].replace('year 2','Year 2')
```

```
df["YearOfStudy"] = df['YearOfStudy'].replace('year 3','Year 3')
df["YearOfStudy"] = df['YearOfStudy'].replace('year 4','Year 4')
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"Timestamp\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 16,\n
\"samples\": [\n          \"13/7/2020\",\n          \"18/7/2020\",\n
\"18/7/2021\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Male\",\n          \"Female\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2,\n        \"min\": 18,\n
\"max\": 25,\n        \"num_unique_values\": 8,\n        \"samples\":
[\n          18,\n          21\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Course\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 49,\n
\"samples\": [\n          \"Fiqh\",\n          \"Human Sciences \"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"YearOfStudy\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          \"Year
4\",\n          \"Year 1\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"CGPA\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.8109607613543679,\n        \"min\":
2.0,\n        \"max\": 4.0,\n        \"num_unique_values\": 187,\n
\"samples\": [\n          3.82,\n          2.94\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Depression\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Anxiety\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Panic Attack\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Specialist Treatment\",\n      \"properties\": {\n
```

```
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Symptom Frequency\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2,\n        \"min\": 0,\n
\"max\": 7,\n        \"num_unique_values\": 8,\n        \"samples\":
[\n          0,\n          7\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Mental Support\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Sleep Quality\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 5,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          1,\n          3\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Study Stress Level\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 5,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          4,\n          3\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"StudyHour/Week\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 5,\n        \"min\": 1,\n
\"max\": 19,\n        \"num_unique_values\": 19,\n        \"samples\":
[\n          8,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Academic Engagement\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 5,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          5,\n          4\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

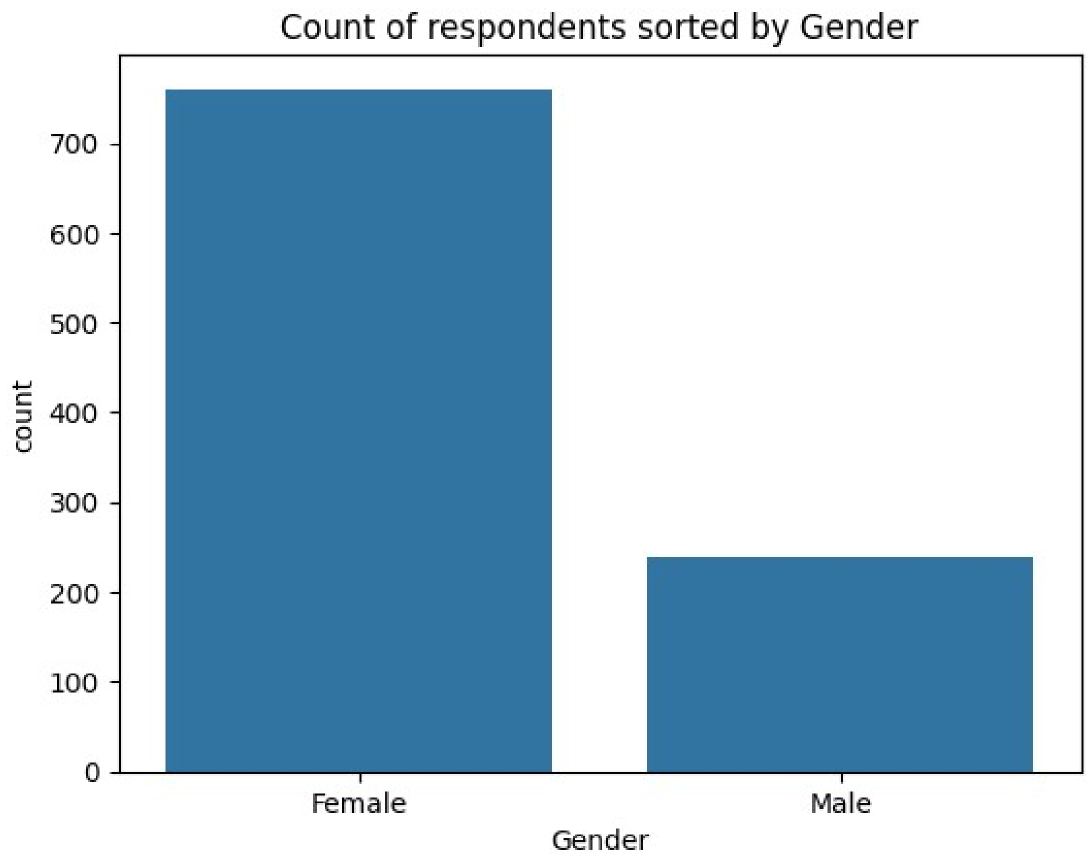

DATA PREPROCESSING COMPLETE

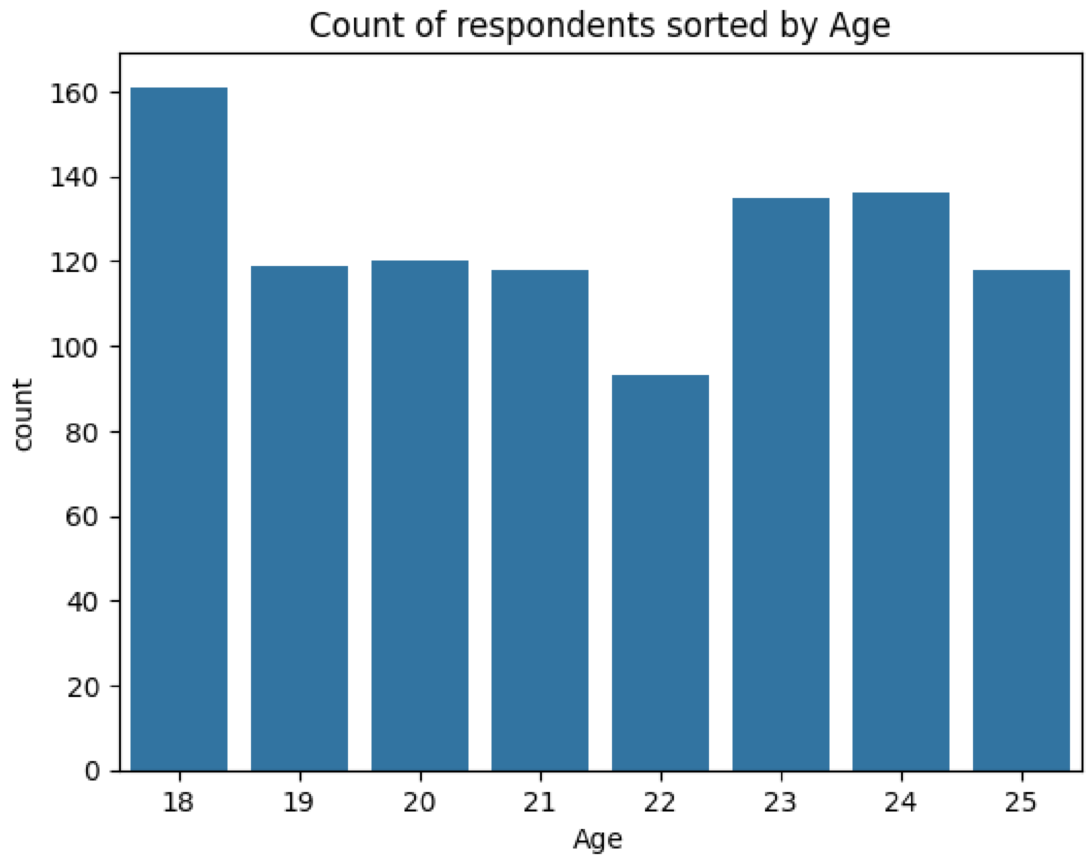
EXPLORATORY DATA ANALYSIS (EDA) PHASE - Noor Hannan Bin Noor Hamsuruddin(1211104293)

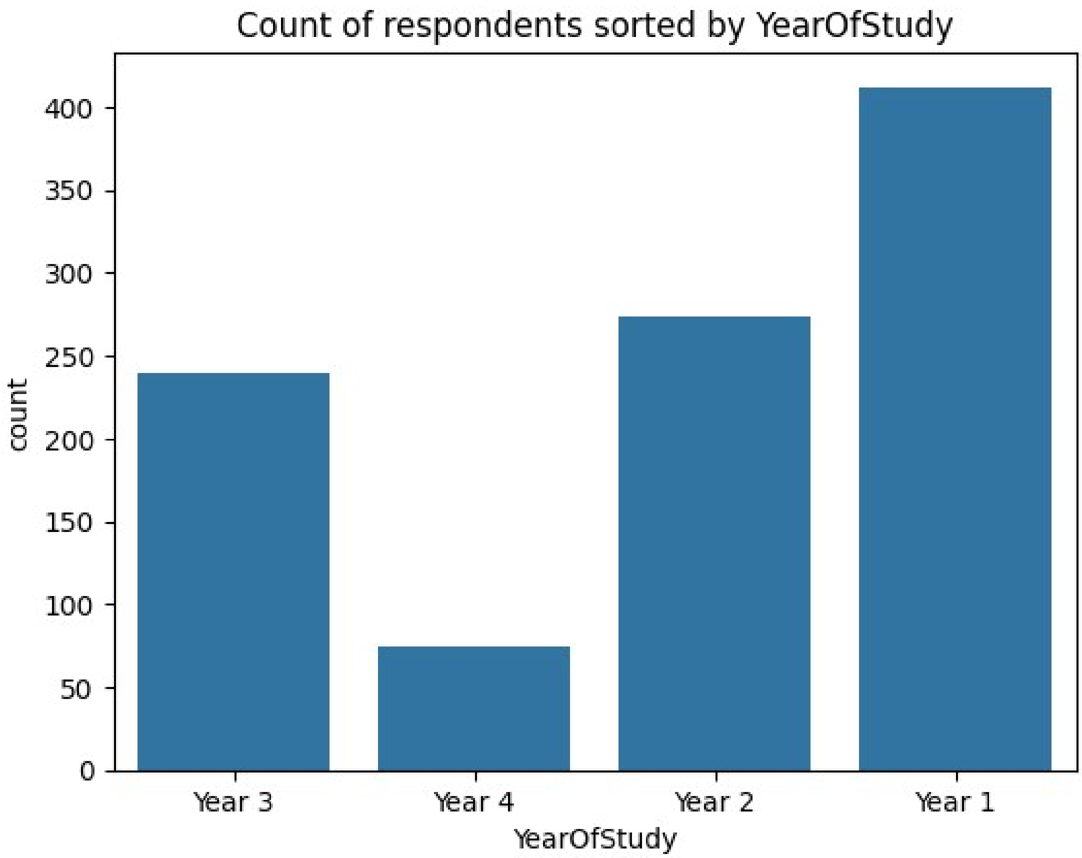
PART 1 - View data in bar plots

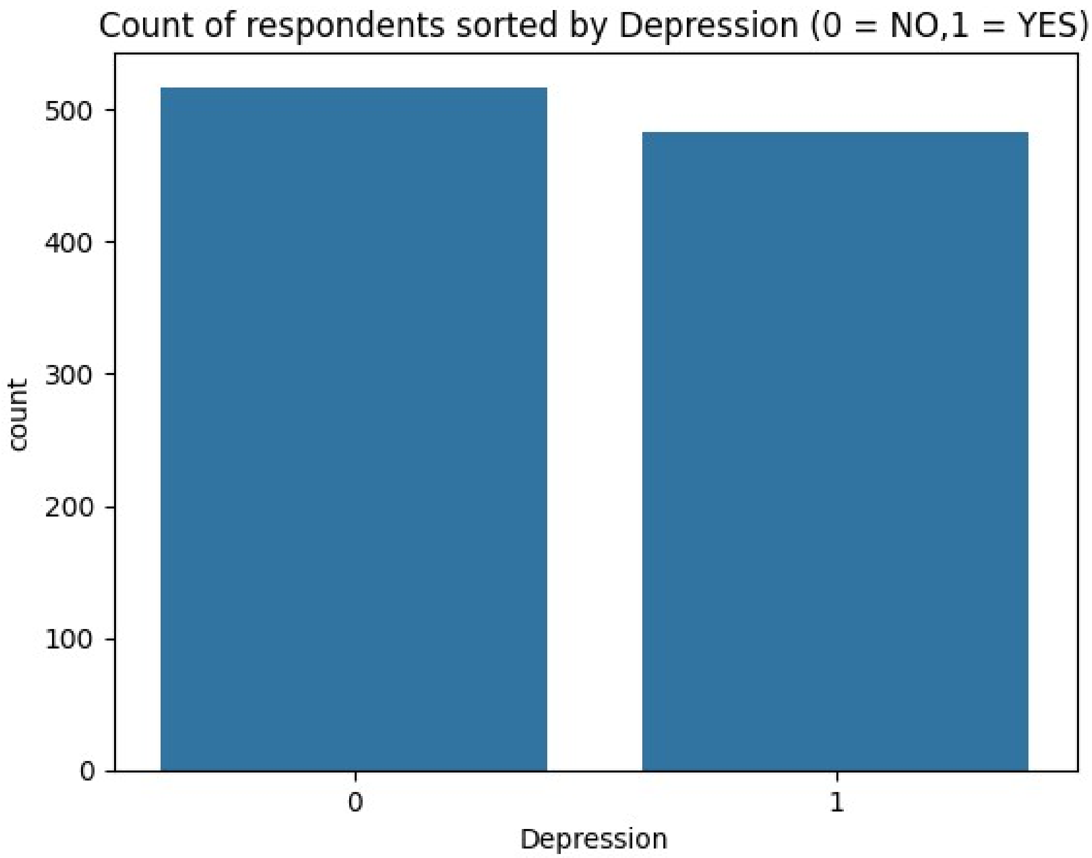
```
#Show count plots of numeric data
columns = df.columns.tolist()
columns_to_remove = ['Timestamp', 'CGPA','Course']
for col in columns_to_remove:
    if col in columns:
        columns.remove(col)
```

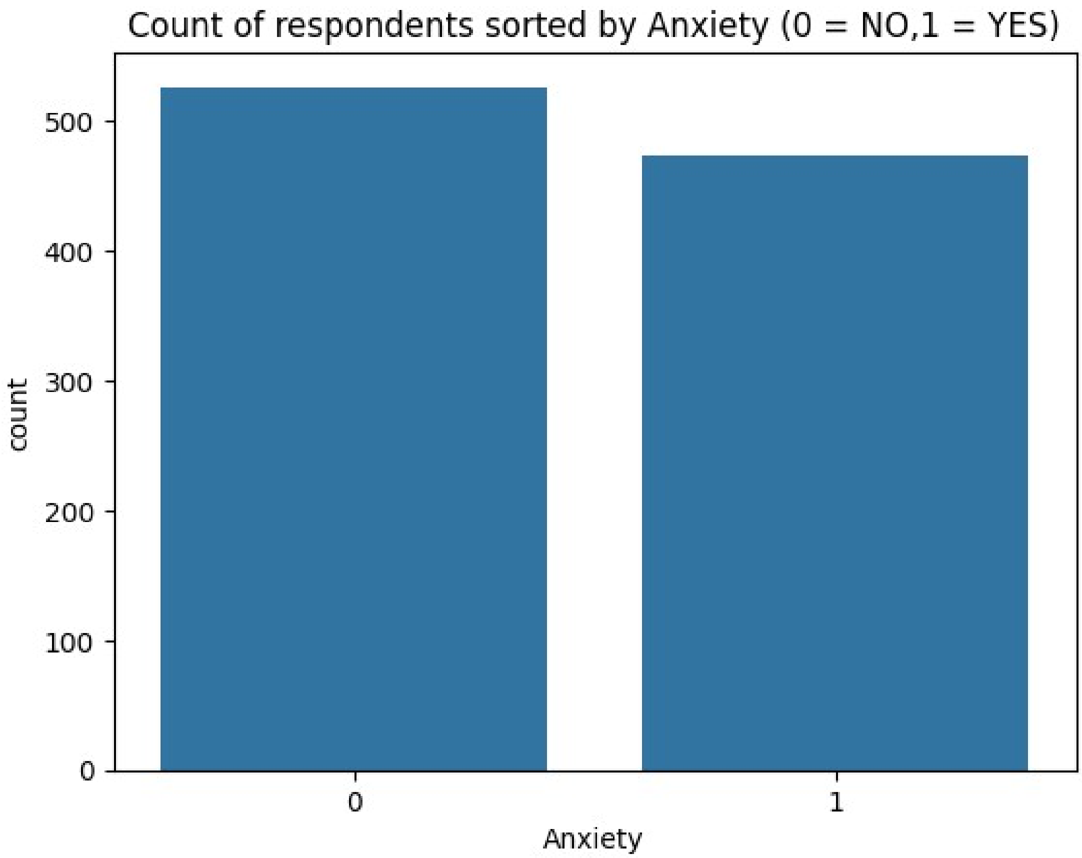
```
for i in columns:
    sns.countplot(x=i,data=df)
    if(df[i].isin([0,1]).all()):
        plt.title(f"Count of respondents sorted by {i} (0 = NO,1 =
YES)")
    else:
        plt.title(f"Count of respondents sorted by {i}")
    plt.show()
```

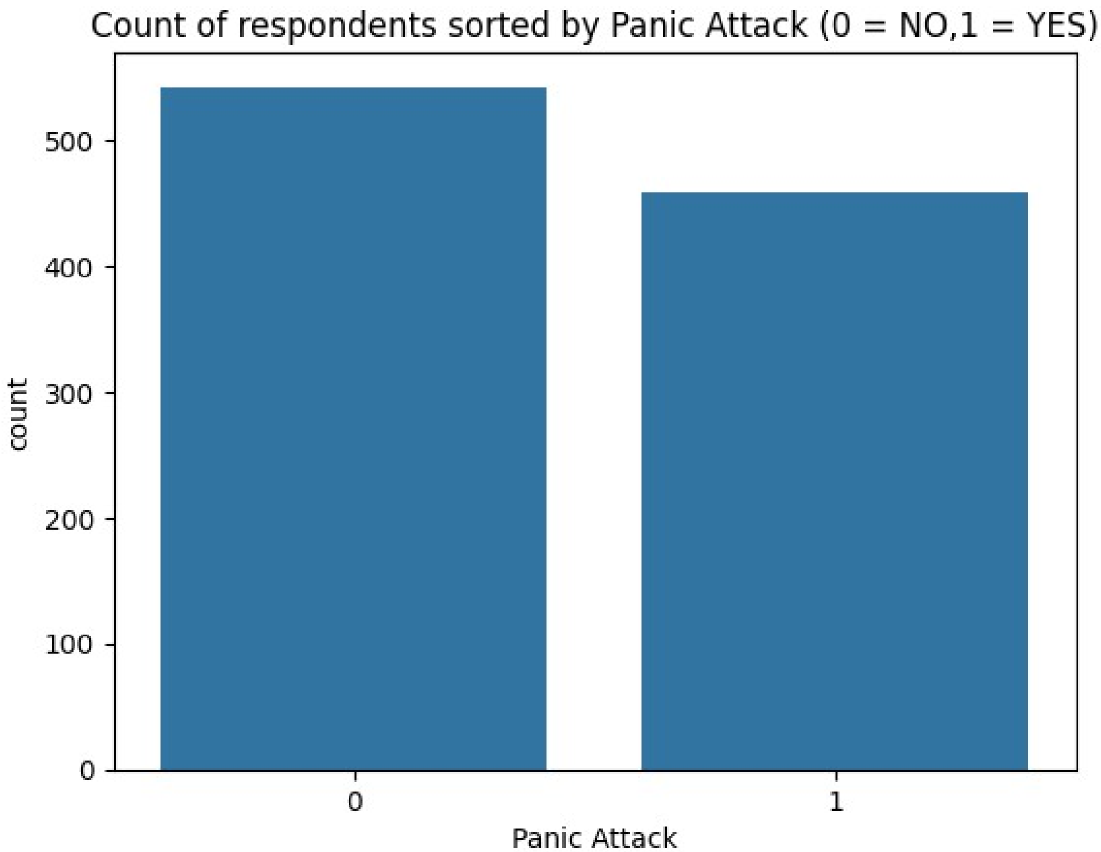


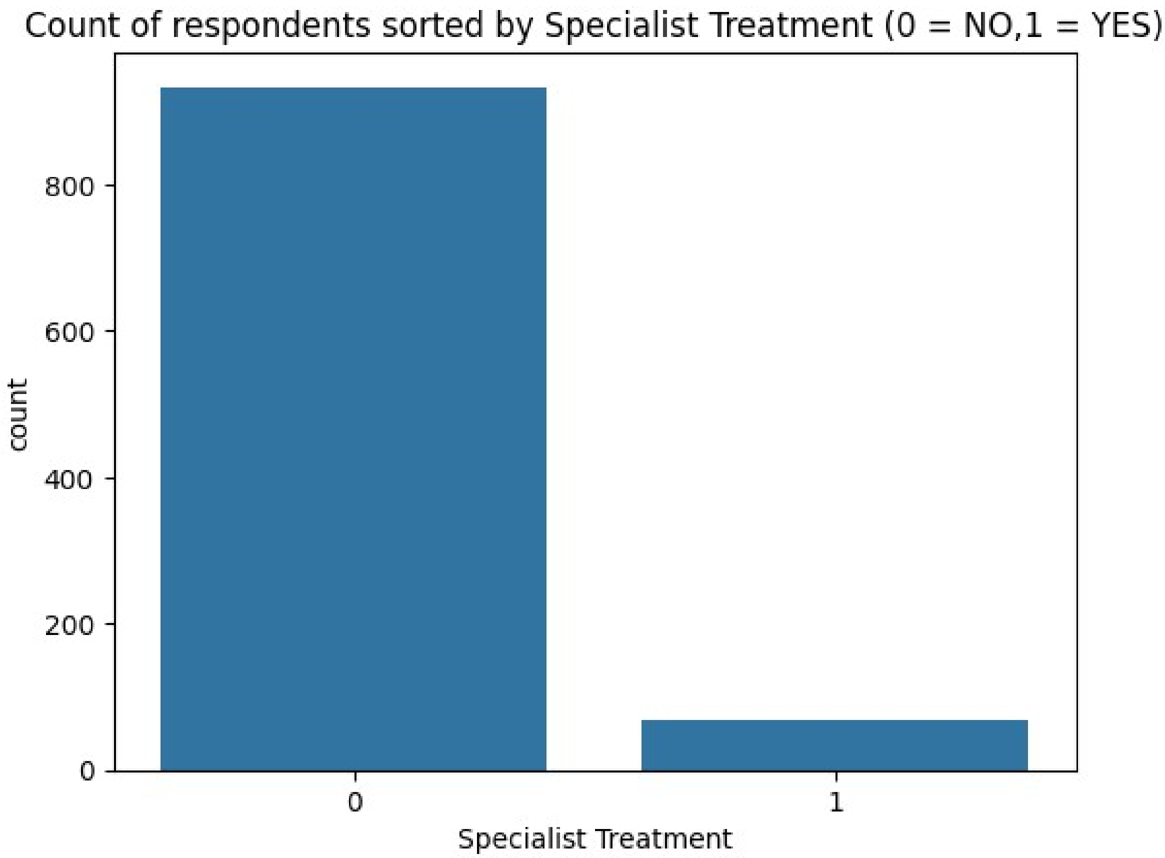

Count of respondents sorted by Age
count
160
140
120
100
80
60
40
20
0
18
19
20
21
22
23
24
25
Age

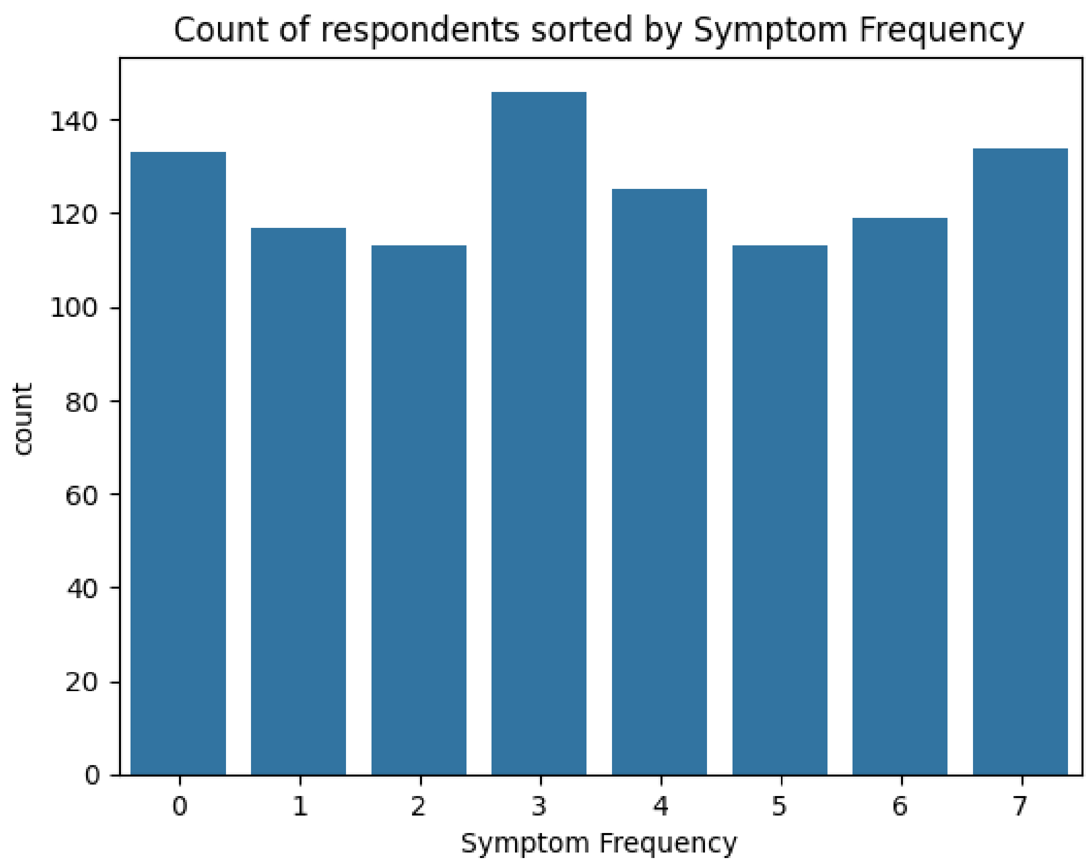

Count of respondents sorted by YearOfStudy
400
350
300
250
200
150
100
50
0
count
Year 3
Year 4
Year 2
Year 1
YearOfStudy

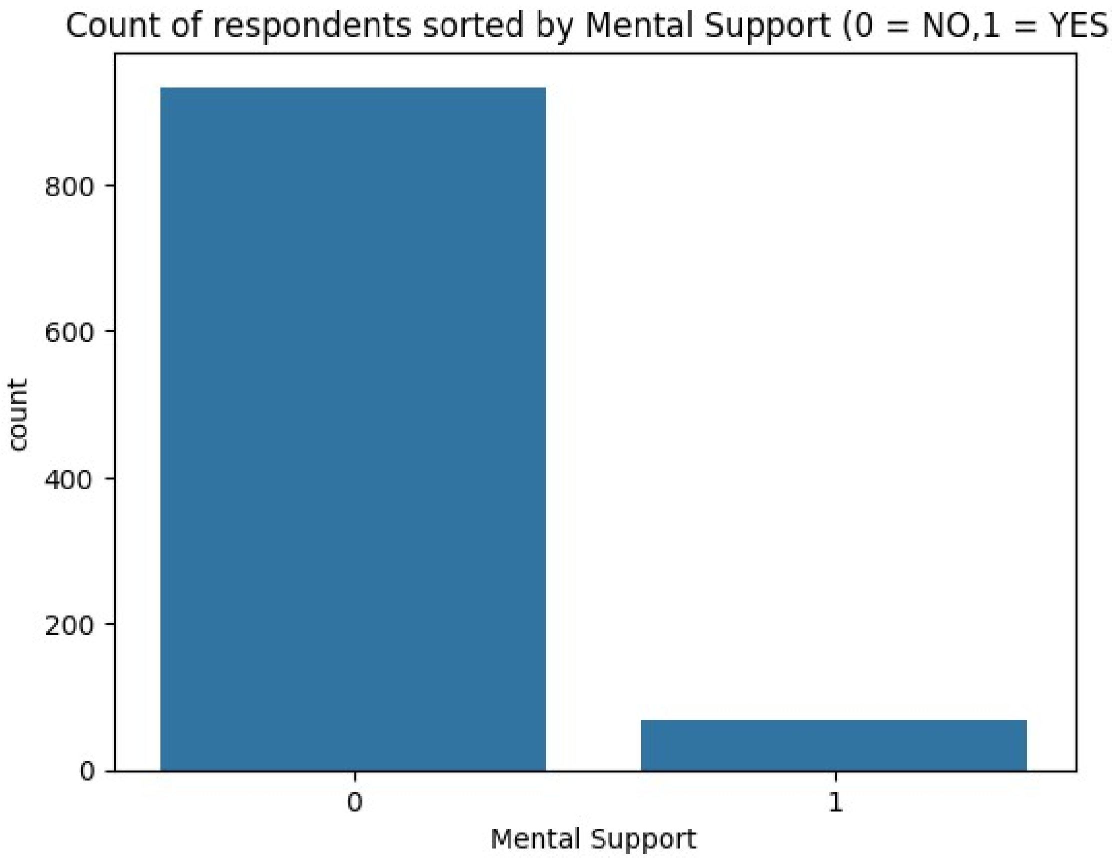

Count of respondents sorted by Depression (0 = NO,1 = YES)
count
Depression
0
100
200
300
400
500
0
1

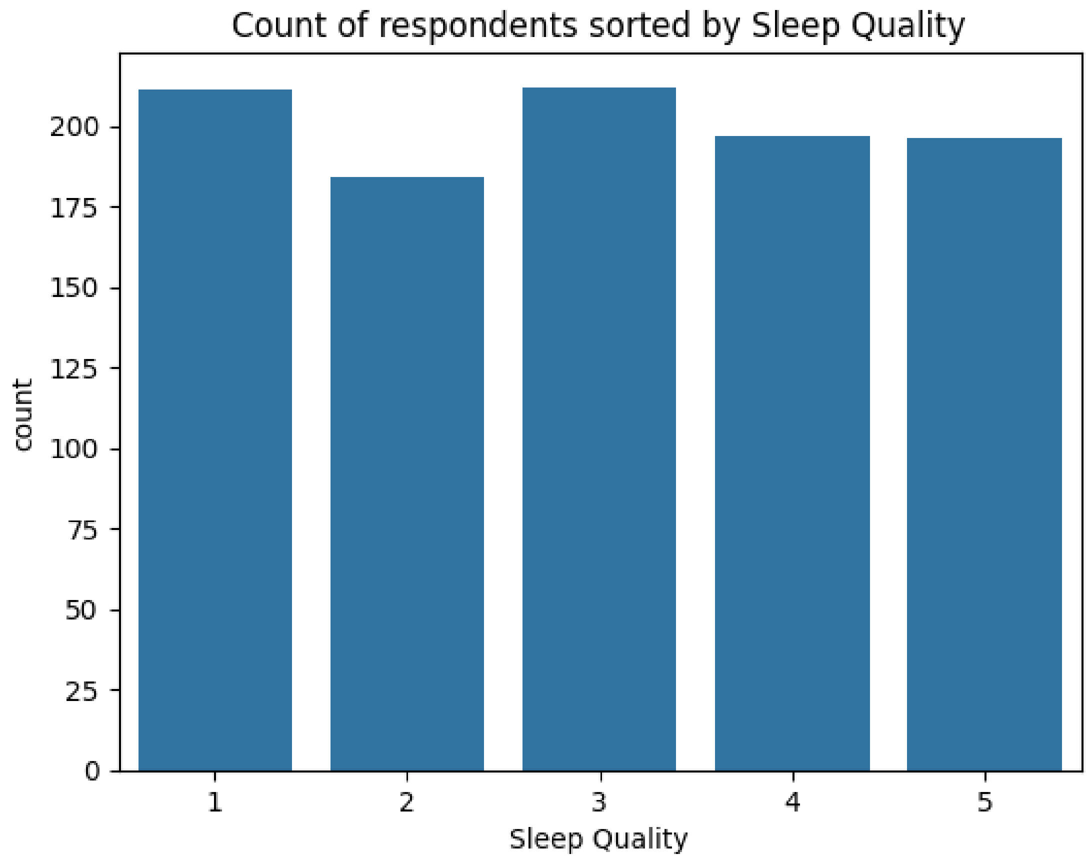

Count of respondents sorted by Anxiety (0 = NO,1 = YES)
count
500
400
300
200
100
0
0
1
Anxiety


Count of respondents sorted by Panic Attack (0 = NO,1 = YES)
count
500
400
300
200
100
0
0
1
Panic Attack


Count of respondents sorted by Specialist Treatment (0 = NO,1 = YES)
800
600
400
200
0
count
0
1
Specialist Treatment

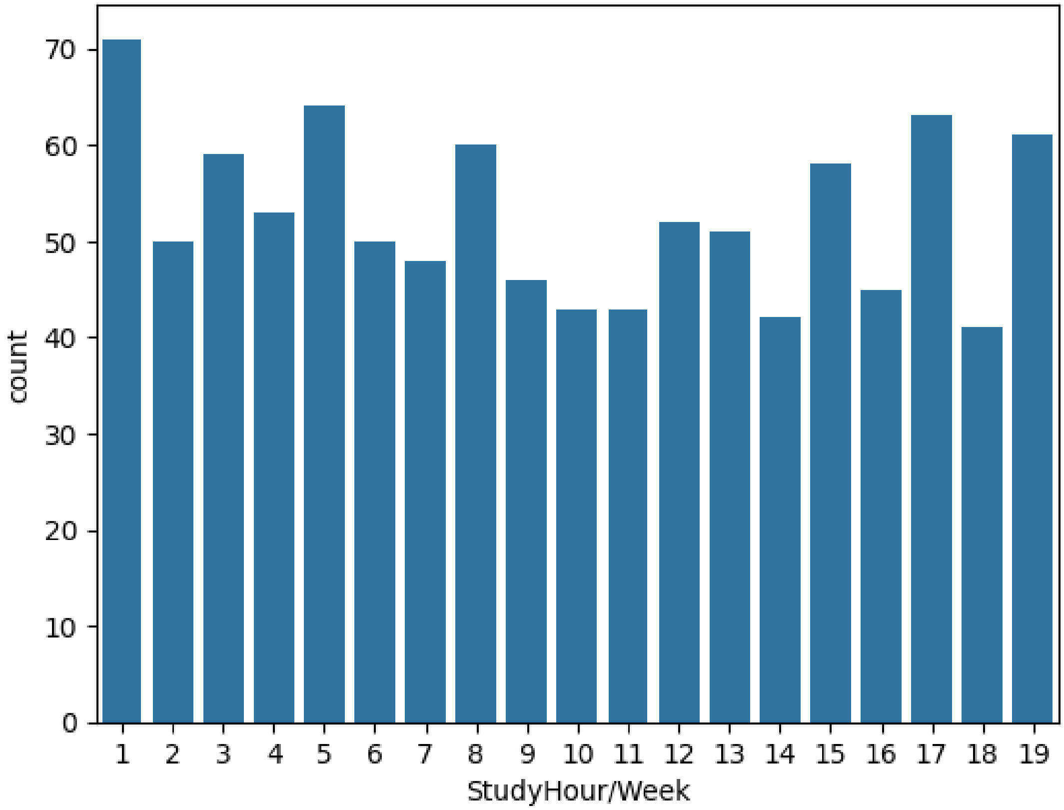

Count of respondents sorted by Symptom Frequency
count
0
20
40
60
80
100
120
140
0
1
2
3
4
5
6
7
Symptom Frequency

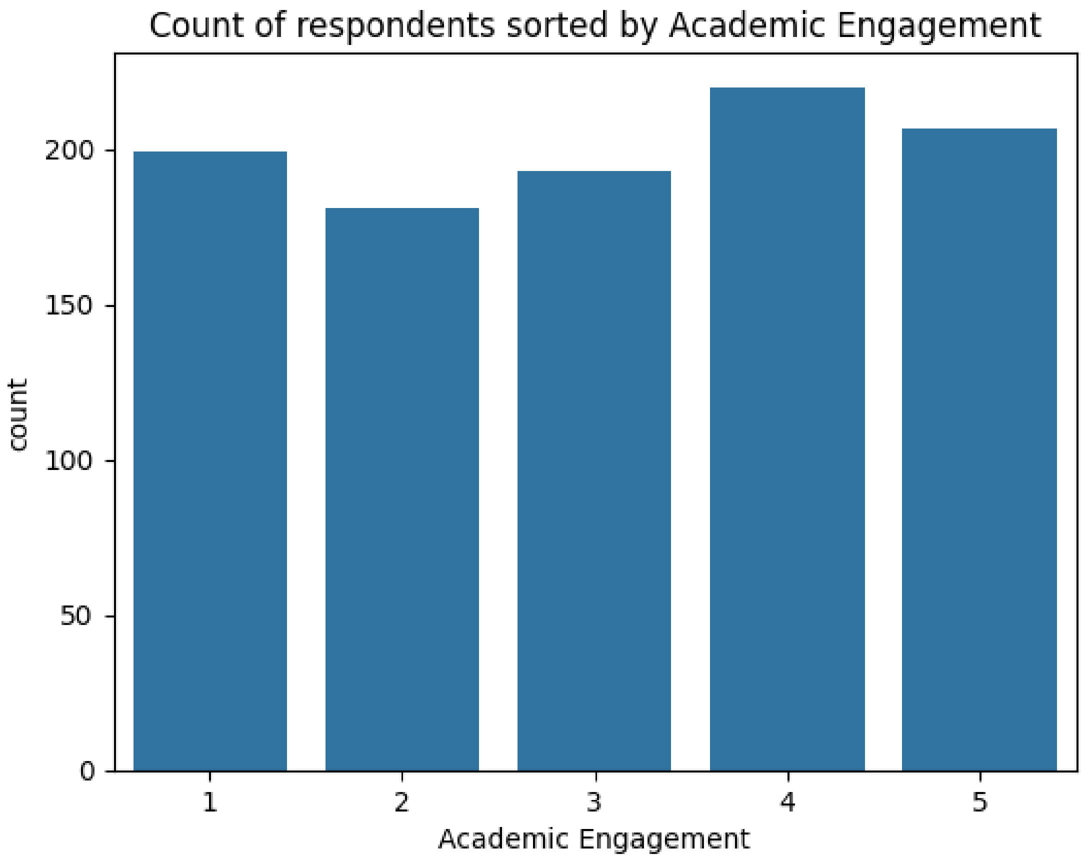

Count of respondents sorted by Mental Support (0 = NO,1 = YES)
count
800
600
400
200
0
0
1
Mental Support


Count of respondents sorted by Sleep Quality
count
0
25
50
75
100
125
150
175
200
1
2
3
4
5
Sleep Quality


Count of respondents sorted by Study Stress Level
count
0
50
100
150
200
1
2
3
4
5
Study Stress Level

Count of respondents sorted by StudyHour/Week



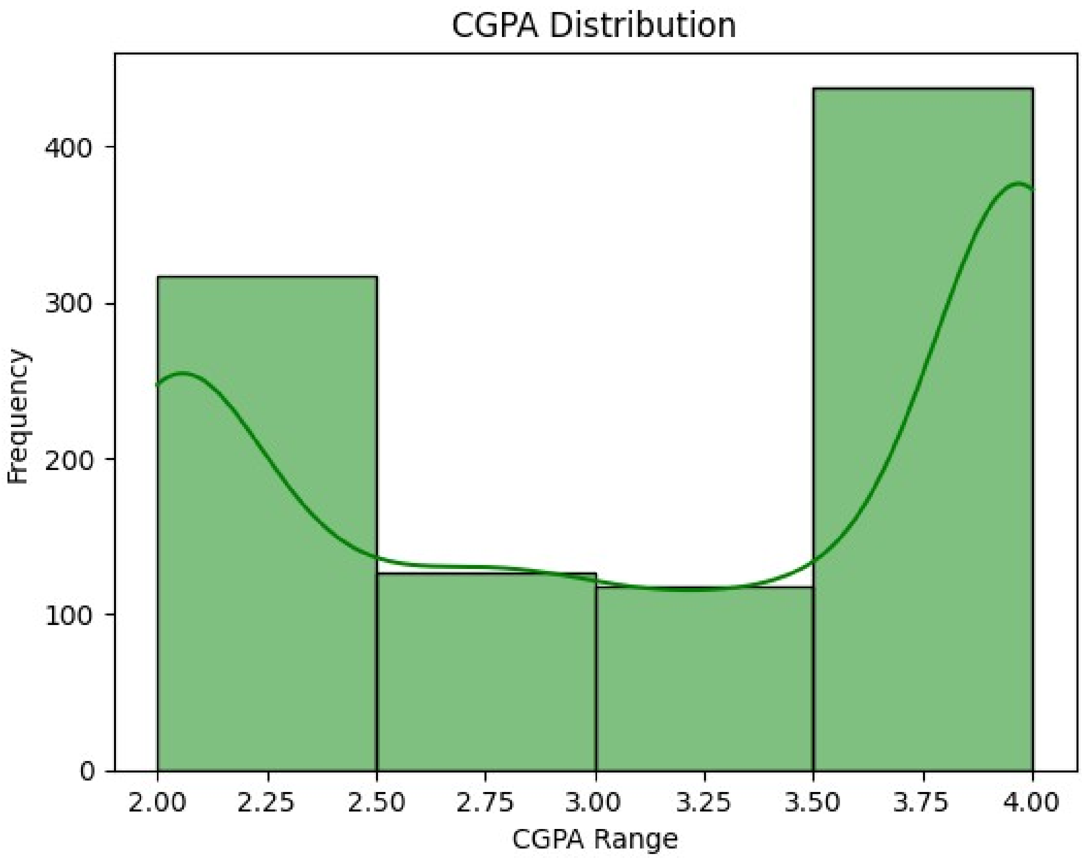



```
#Count number of students courses, since using a bar graph is
unsuitable
course_count = df['Course'].value_counts()
print (course_count)
```

```
Course
Engineering                 180
BCS                         177
BIT                         101
KOE                          39
Biomedical science           33
Engine                       19
Laws                         19
psychology                   17
BENL                         16
CTS                          15
Business Administration      14
Koe                          14
engin                        14
Human Sciences               13
Nursing                      13
Law                          13
Communication                13
```

```
Marine science             12
Psychology                 12
Kirkhs                     12
Malcom                     12
Pendidikan Islam           12
Accounting                 11
DIPLOMA TESL               11
Usuluddin                  11
Fiqh                       11
KIRKHS                     10
Irkhs                      10
Pendidikan islam           10
ENM                         9
Human Resources             9
Mathemathics                9
Fiqh fatwa                  9
TAASL                       9
Radiography                 9
Islamic education           9
Econs                       8
Kop                         8
Benl                        8
Biotechnology               8
koe                         8
Diploma Nursing             8
IT                          8
KENMS                       7
Pendidikan Islam            7
Banking Studies             6
MHSC                        6
ALA                         6
Islamic Education           5
Name: count, dtype: int64
```

```python
#Divide CGPA's into bins
sns.histplot(df["CGPA"], bins=4, kde=True,color='green')
plt.title("CGPA Distribution")
plt.xlabel("CGPA Range")
plt.ylabel("Frequency")
plt.show()
```



CHECKING POTENTIAL RELATED COLUMNS

A. Stress Levels against Course

```
#Check average stress levels for each course(Usage of weighted average
to account for major differences in rows)
total_stress = df.groupby('Course')['Study Stress Level'].sum()
total_counts = df.groupby('Course')['Study Stress Level'].count()

weighted_avg = total_stress / total_counts
weighted_avg = weighted_avg.sort_values(ascending=True)


for index, value in weighted_avg.items():
    print(f"{index} : {value:.2f}")
```

```
Islamic Education : 2.00
BENL : 2.12
Banking Studies : 2.17
Communication  : 2.46
Irkhs : 2.50
Fiqh fatwa  : 2.67
CTS : 2.67
ALA : 2.67
```

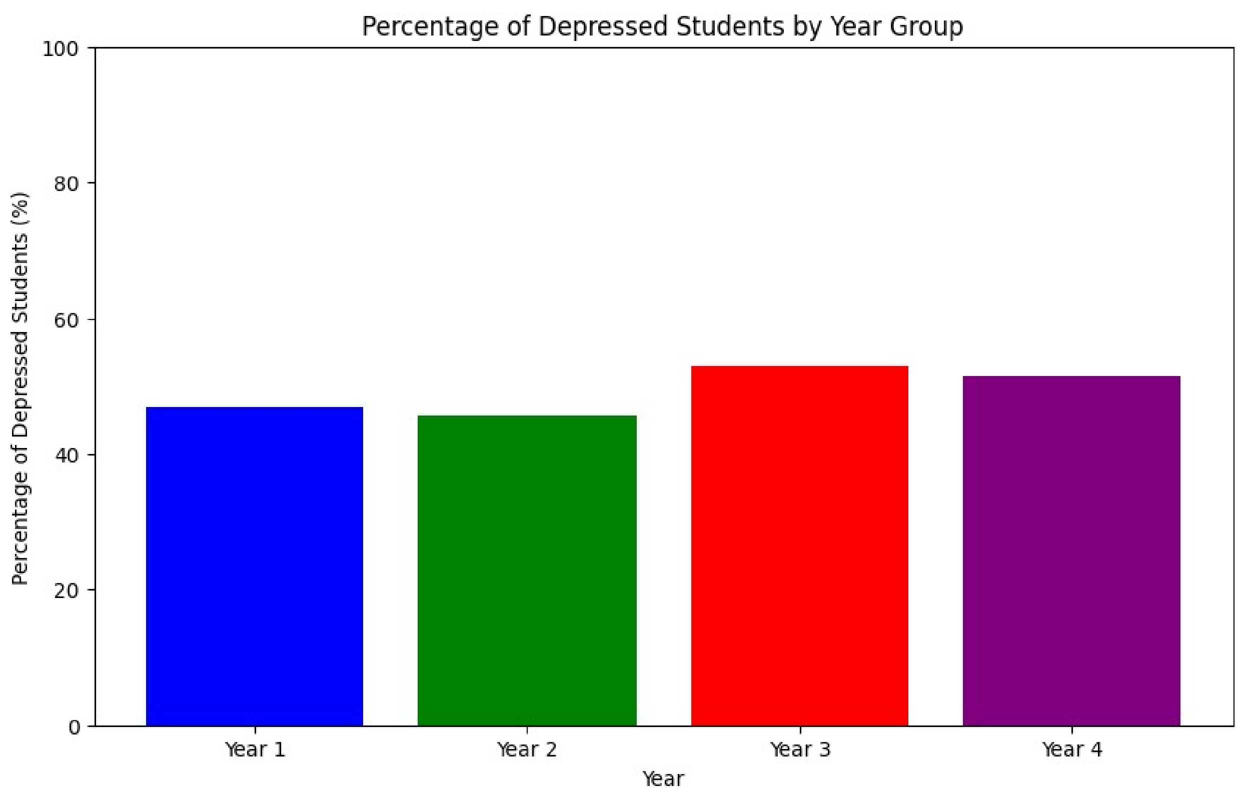
```
Law : 2.69
KENMS : 2.71
Koe : 2.79
engin : 2.86
koe : 2.88
Pendidikan islam : 2.90
KIRKHS : 2.90
Usuluddin  : 2.91
Biomedical science : 2.91
Accounting  : 2.91
Kirkhs : 2.92
Engineering : 2.92
Mathemathics : 3.00
MHSC : 3.00
KOE : 3.03
BCS : 3.05
Psychology : 3.08
Engine : 3.11
Human Resources : 3.11
Radiography : 3.11
TAASL : 3.11
IT : 3.12
Kop : 3.12
BIT : 3.15
Laws : 3.21
Pendidikan Islam : 3.29
Marine science : 3.33
psychology : 3.35
Fiqh : 3.36
Econs : 3.38
Human Sciences  : 3.46
Nursing  : 3.46
Pendidikan Islam  : 3.50
Business Administration : 3.50
DIPLOMA TESL : 3.55
Islamic education : 3.56
ENM : 3.56
Biotechnology : 3.62
Malcom : 3.83
Diploma Nursing : 3.88
Benl : 3.88
```

B. Depression Levels for each Year

```
#Percentage of depressed respondents per year
depression_percentage = df.groupby('YearOfStudy')['Depression'].mean()
*100

depression_percentage = depression_percentage.reset_index()
```

```python
# Print the percentage of depressed students for each year
for index, row in depression_percentage.iterrows():
    print(f"{row['YearOfStudy']} : {row['Depression']:.2f}% claim to
be depressed")

plt.figure(figsize=(10, 6))
plt.bar(depression_percentage['YearOfStudy'],
depression_percentage['Depression'], color=['blue', 'green', 'red',
'purple'])
plt.xlabel('Year')
plt.ylabel('Percentage of Depressed Students (%)')
plt.title('Percentage of Depressed Students by Year Group')
plt.ylim(0, 100)  # Set the y-axis limit to 0-100%
plt.show()
```

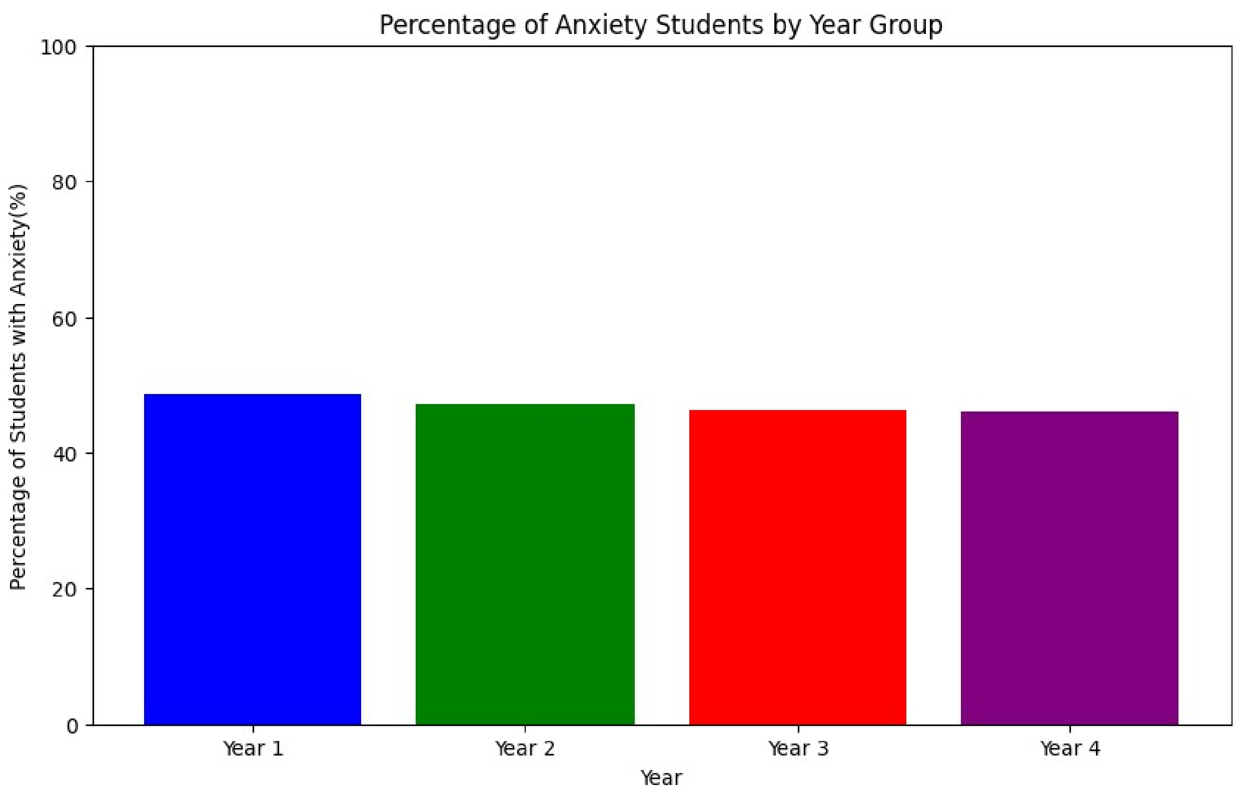
```
Year 1 : 46.84% claim to be depressed
Year 2 : 45.62% claim to be depressed
Year 3 : 52.92% claim to be depressed
Year 4 : 51.35% claim to be depressed
```



C. People in each year who have anxiety

```python
#Percentage of respondents per year who get anxiety
depression_percentage = df.groupby('YearOfStudy')['Anxiety'].mean()
```

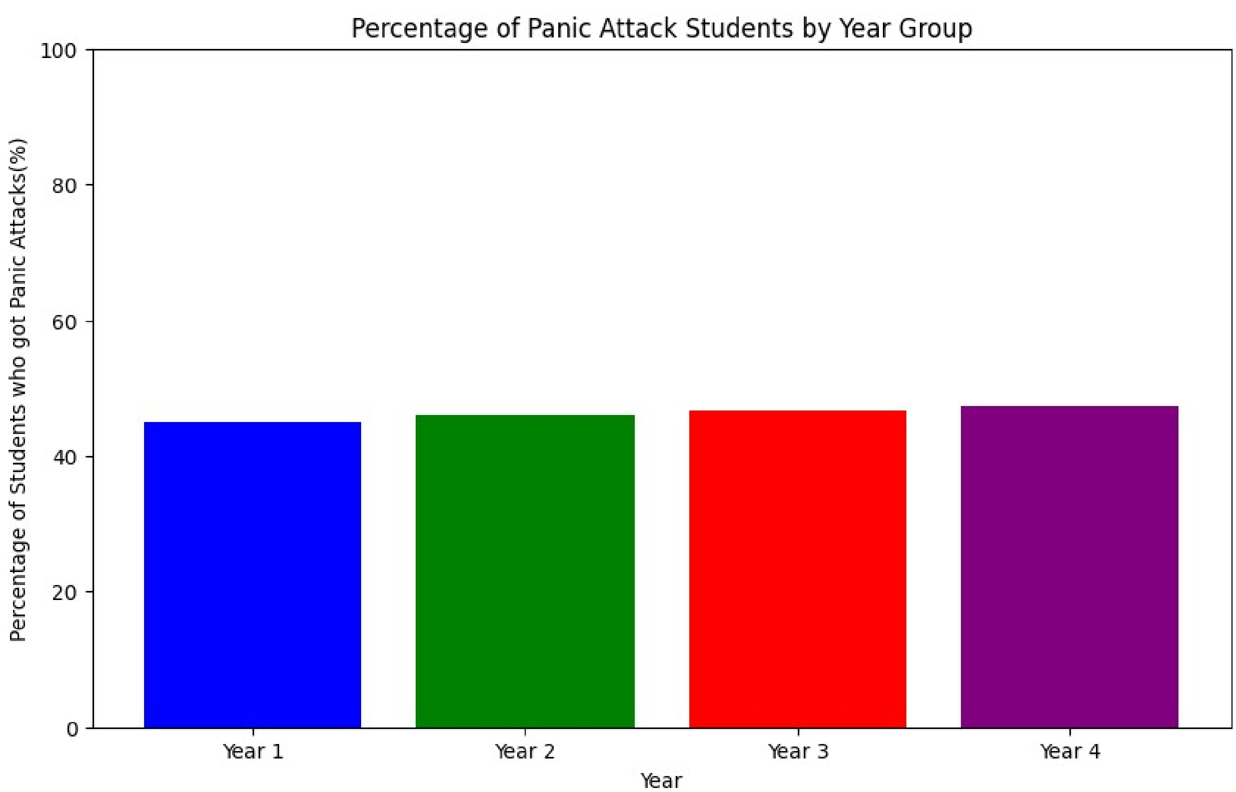
```
*100

depression_percentage = depression_percentage.reset_index()

# Print the percentage of depressed students for each year
for index, row in depression_percentage.iterrows():
    print(f"{row['YearOfStudy']} : {row['Anxiety']:.2f}% claim to be 
have anxiety")

plt.figure(figsize=(10, 6))
plt.bar(depression_percentage['YearOfStudy'], 
depression_percentage['Anxiety'], color=['blue', 'green', 'red', 
'purple'])
plt.xlabel('Year')
plt.ylabel('Percentage of Students with Anxiety(%)')
plt.title('Percentage of Anxiety Students by Year Group')
plt.ylim(0, 100)  # Set the y-axis limit to 0-100%
plt.show()

Year 1 : 48.54% claim to be have anxiety
Year 2 : 47.08% claim to be have anxiety
Year 3 : 46.25% claim to be have anxiety
Year 4 : 45.95% claim to be have anxiety
```



D. Students who have had panic attacks

```python
#Percentage of respondents per year who got panic attacks
depression_percentage = df.groupby('YearOfStudy')['Panic
Attack'].mean() *100

depression_percentage = depression_percentage.reset_index()

# Print the percentage of depressed students for each year
for index, row in depression_percentage.iterrows():
    print(f"{row['YearOfStudy']} : {row['Panic Attack']:.2f}% claim to
be have gotten panic attacks")

plt.figure(figsize=(10, 6))
plt.bar(depression_percentage['YearOfStudy'],
depression_percentage['Panic Attack'], color=['blue', 'green', 'red',
'purple'])
plt.xlabel('Year')
plt.ylabel('Percentage of Students who got Panic Attacks(%)')
plt.title('Percentage of Panic Attack Students by Year Group')
plt.ylim(0, 100)  # Set the y-axis limit to 0-100%
plt.show()
```

```
Year 1 : 44.90% claim to be have gotten panic attacks
Year 2 : 45.99% claim to be have gotten panic attacks
Year 3 : 46.67% claim to be have gotten panic attacks
Year 4 : 47.30% claim to be have gotten panic attacks
```



D. Correlations between columns

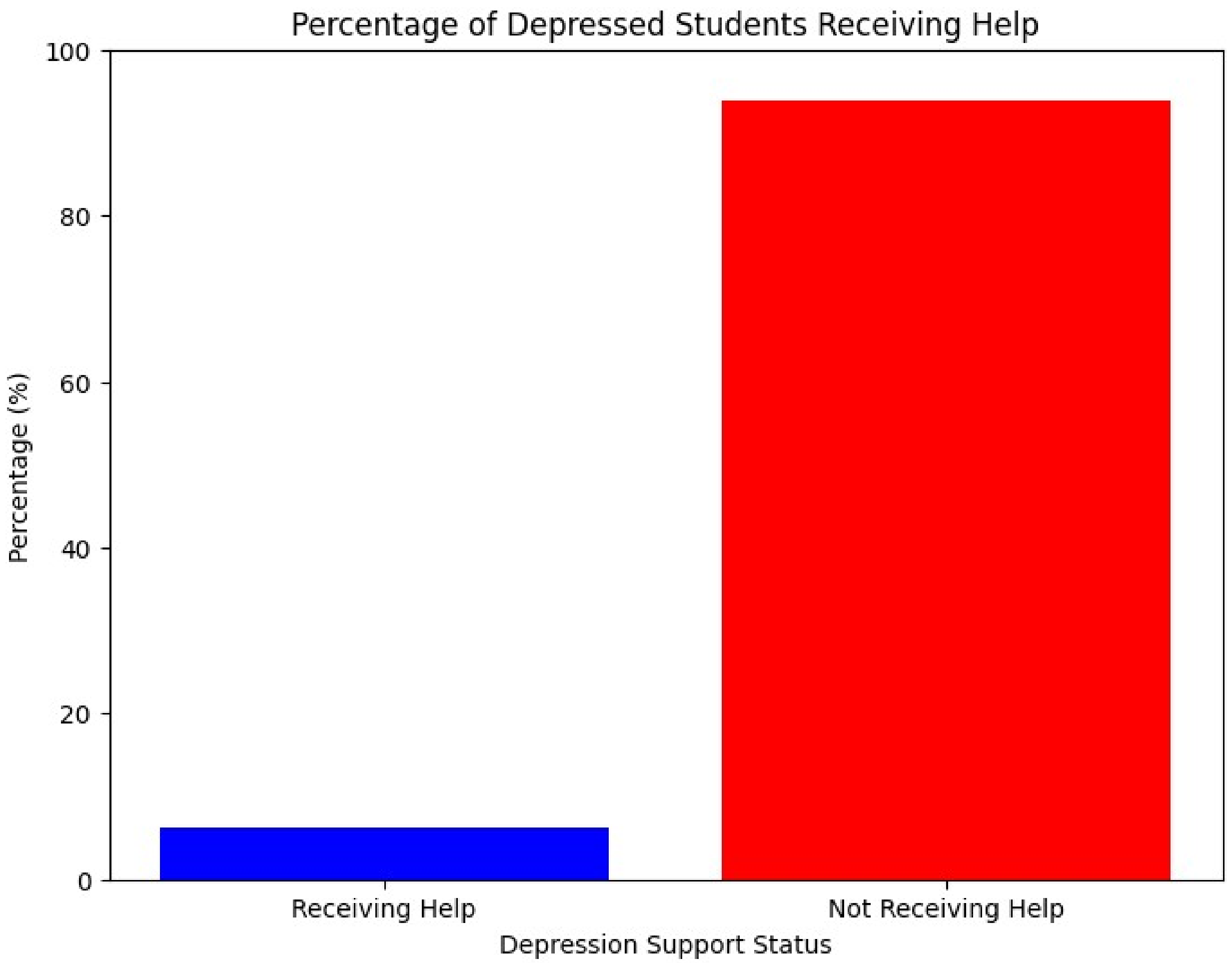
```
df[['Depression', 'Anxiety', 'Panic Attack']].corr()
```


```
{"summary":"{\n  \"name\": \"df[['Depression', 'Anxiety', 'Panic
Attack']]\",\n  \"rows\": 3,\n  \"fields\": [\n    {\n
\"column\": \"Depression\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.5654336108558554,\n
\"min\": -0.004876042334225962,\n        \"max\": 1.0,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          1.0,\n
0.0483252822980871,\n          -0.004876042334225962\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Anxiety\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.5554792648685141,\n        \"min\": 0.027767571812297498,\n
\"max\": 1.0,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          0.0483252822980871,\n          1.0,\n
0.027767571812297498\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Panic Attack\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.5709753877389113,\n        \"min\": -
0.004876042334225962,\n        \"max\": 1.0,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          -
0.004876042334225962,\n          0.027767571812297498,\n          1.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


E. Are depressed students getting the treatment or support they need?

```
depressed = df[df['Depression'] == 1].copy()

depressed.loc[:, 'Receiving_Help'] = (depressed['Specialist
Treatment'] == 1) | (depressed['Mental Support'] == 1)
getting_help = depressed['Receiving_Help'].sum()

total_depressed = len(depressed)

getting_help_percent = getting_help / total_depressed * 100
print(f'{getting_help_percent:.2f} % of students are getting treatment
or mental support for their depression, leaving {100 -
getting_help_percent:.2f} % untreated or assisted.')

labels = ['Receiving Help', 'Not Receiving Help']
percentages = [getting_help_percent, 100 - getting_help_percent]

plt.figure(figsize=(8, 6))
plt.bar(labels, percentages, color=['blue', 'red'])
plt.xlabel('Depression Support Status')
plt.ylabel('Percentage (%)')
plt.title('Percentage of Depressed Students Receiving Help')
```

```
plt.ylim(0, 100)  # Set the y-axis limit to 0-100%
plt.show()

6.21 % of students are getting treatment or mental support for their
depression, leaving 93.79 % untreated or assisted.
```



F. Does leaving depression untreated cause a drop in academic engagement?

```
untreated = (df['Depression'] == 1) & ~((df['Specialist Treatment'] ==
1) | (df['Mental Support'] == 1))

untreated_depressed = df[untreated]
mean_untreated_depressed = untreated_depressed['Academic
Engagement'].mean()

engagement_normal = df[~(df['Depression']== 1) &((df['Specialist
Treatment'] == 1) | (df['Mental Support'] == 1))]
mean_academic_engagement = engagement_normal['Academic
Engagement'].mean()
```

```
print(f'Average Academic Engagement (Depressed):
{mean_untreated_depressed:.2f}')
print(f'Average Academic Engagement (Non-Depressed):
{mean_academic_engagement:.2f}')
```

```
Average Academic Engagement (Depressed): 2.91
Average Academic Engagement (Non-Depressed): 3.22
```

PCA Process - Khoo Jen-Au (1211102910)

Further Pre-processing the data for PCA

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA

# Encode the 'Gender' column
label_encoder = LabelEncoder()
df['Gender_Encoded'] = label_encoder.fit_transform(df['Gender'])

features = ['StudyHour/Week', 'Panic Attack', 'Specialist Treatment',
            'Symptom Frequency', 'Sleep Quality', 'Study Stress
Level',
            'Academic Engagement']

# Standardising the numerical features
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[features])

# Confirm Pre-processing results
df[['Gender', 'Gender_Encoded']].head(15), pd.DataFrame(df_scaled,
columns=features)
```

```
(    Gender  Gender_Encoded
 0   Female               0
 1   Female               0
 2   Female               0
 3   Female               0
 4   Female               0
 5   Female               0
 6   Female               0
 7   Female               0
 8   Female               0
 9   Female               0
 10  Female               0
 11  Female               0
 12  Female               0
 13    Male               1
 14    Male               1,
      StudyHour/Week  Panic Attack  Specialist Treatment  Symptom
```

```
Frequency  \
 0         -0.309099     -0.919249                  -0.267976
0.651077
 1          0.576065     -0.919249                  -0.267976           -
1.516291
 2          0.576065      1.087845                  -0.267976           -
0.215870
 3          1.638261     -0.919249                  -0.267976           -
0.215870
 4         -1.194263     -0.919249                  -0.267976           -
1.516291
 ..              ...           ...                        ...
...
 995       -0.486132     -0.919249                  -0.267976
1.084551
 996        0.399032     -0.919249                  -0.267976           -
0.215870
 997       -1.194263      1.087845                  -0.267976           -
1.082817
 998        0.576065      1.087845                  -0.267976
0.651077
 999       -0.309099     -0.919249                  -0.267976           -
1.516291

      Sleep Quality  Study Stress Level  Academic Engagement
 0         0.717567            1.379989            -0.741933
 1         0.717567            0.674113             1.367829
 2        -1.399149           -0.737641            -1.445187
 3         1.423139           -1.443518            -0.741933
 4        -0.693577            0.674113            -0.741933
 ..             ...                 ...                  ...
 995      -1.399149           -0.031764            -1.445187
 996       0.717567           -1.443518            -0.741933
 997       1.423139            0.674113            -1.445187
 998       1.423139            0.674113            -0.038679
 999      -0.693577            1.379989            -1.445187

 [1000 rows x 7 columns])
```

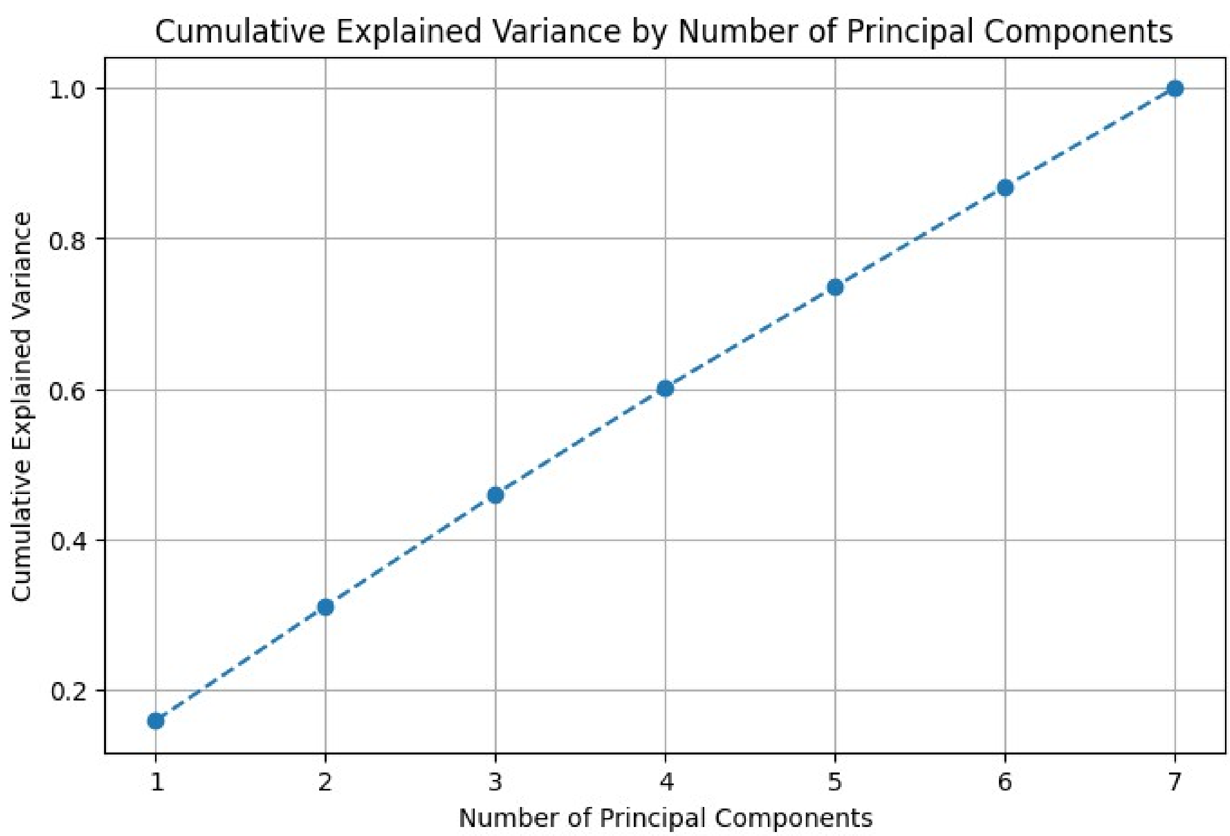
Encoded `Gender` column:

- Female -> 0
- Male -> 1

All numerical features have been scaled to have a mean of 0 and a standard deviation of 1, ensuring they contribute equally to PCA.

```
# Perform PCA
pca = PCA()
```

```
principal_components = pca.fit_transform(df_scaled)

# Explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_variance_ratio = np.cumsum(explained_variance_ratio)

# Plot explained variance
plt.figure(figsize=(8, 5))
plt.plot(range(1, len(explained_variance_ratio) + 1),
cumulative_variance_ratio, marker='o', linestyle='--')
plt.title('Cumulative Explained Variance by Number of Principal
Components')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid(True)
plt.show()

# Display the explained variance ratio for each component
explained_variance_ratio
```




```
array([0.15909991, 0.15207731, 0.14873945, 0.14187378, 0.13393441,
       0.13289771, 0.13137744])
```

The array indicates how much variance each principal component (PC) explains. For example:

- PC1 explains ~15.91% of the variance.
- PC2 explains ~15.21%.
- PC3 explains ~14.87%, and so on.

The components have relatively balanced contributions to variance, with no single dominant component.

Since the dataset is already well-structured, as well as the contributions from features are balanced and beneficial for analysis or model, we may not need to perform PCA.

```
n_components = np.argmax(cumulative_variance_ratio >= 0.90) + 1
print(f"Number of components to retain 90% variance: {n_components}")

Number of components to retain 90% variance: 7
```

Since for most applications, retaining enough components to explain 90% of the variance is considered sufficient. But as seen in the code block above, since we need 7 components to retain that 90% variance, which is all the original components, PCA is not required.

Building a model - Wan Muhammad Atif bin Taram Satiraksa (1211103154)

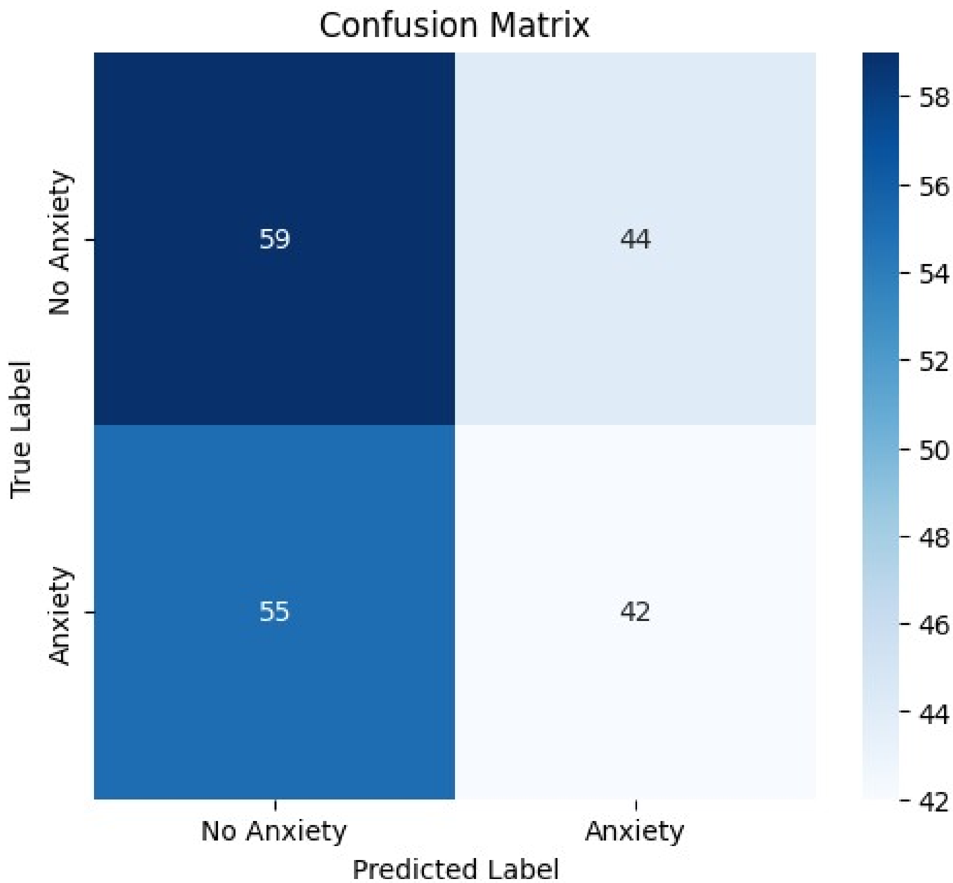
In this section, we will be building a Random Forest model to predict the possibility of students having anxiety based on a list of features.

```
print(df.columns)
# Check unique values in the 'Mental Support' column
unique_values = df['Mental Support'].unique()

print("Unique values in 'Mental Support':", unique_values)


Index(['Timestamp', 'Gender', 'Age', 'Course', 'YearOfStudy', 'CGPA',
       'Depression', 'Anxiety', 'Panic Attack', 'Specialist
Treatment',
       'Symptom Frequency', 'Mental Support', 'Sleep Quality',
       'Study Stress Level', 'StudyHour/Week', 'Academic Engagement',
       'Gender_Encoded'],
      dtype='object')
Unique values in 'Mental Support': [0 1]

from imblearn.over_sampling import ADASYN
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, StratifiedKFold,
GridSearchCV
```

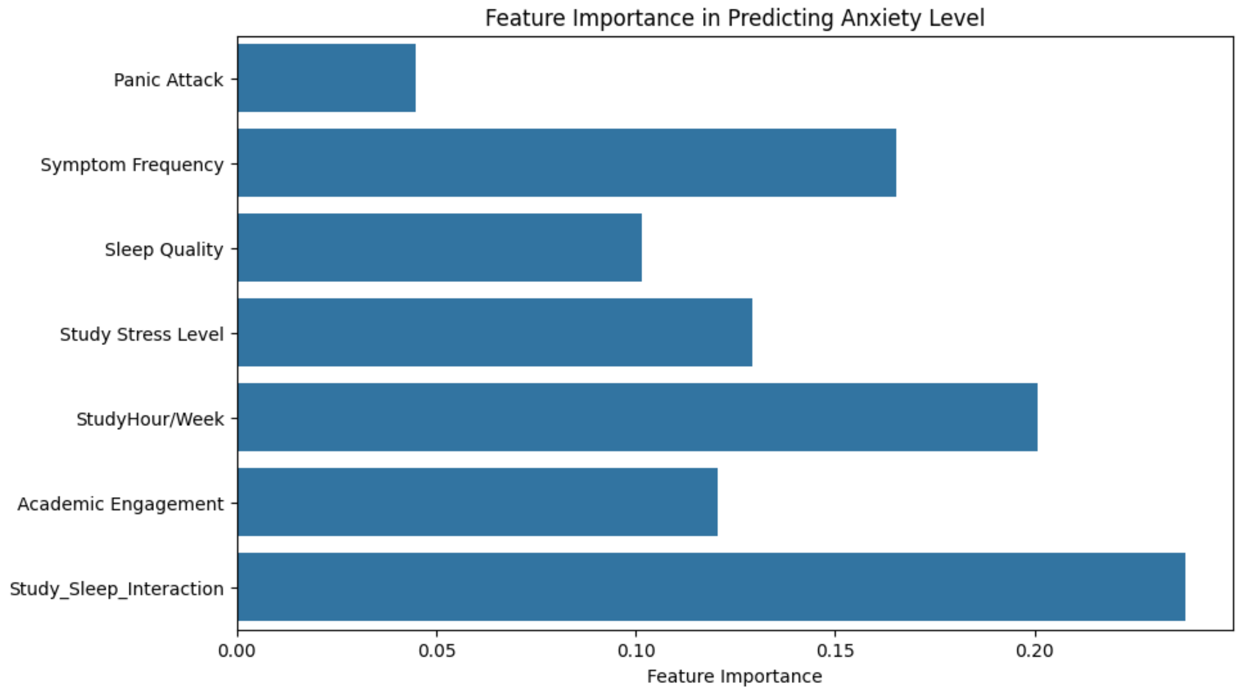
```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier

# Selecting features and target variable
features = ['Panic Attack', 'Symptom Frequency', 'Sleep Quality',
            'Study Stress Level', 'StudyHour/Week', 'Academic
Engagement']

# Create the combined feature for Study Hour/Week and Sleep Quality
df['Study_Sleep_Interaction'] = df['StudyHour/Week'] * df['Sleep
Quality']

# Update the features list
features.append('Study_Sleep_Interaction')

# Extract features and target variable
X = df[features]
y = df['Anxiety']  # Target variable: Anxiety Level

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Random Forest with class weights
rf = RandomForestClassifier(random_state=42, class_weight='balanced')

# Hyperparameter tuning with GridSearchCV
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3,
verbose=2, n_jobs=-1)
grid_search.fit(X_train_scaled, y_train)

# Get best parameters and model
best_rf = grid_search.best_estimator_

# Predict and evaluate with best model
y_pred = best_rf.predict(X_test_scaled)
print("Classification Report:\n", classification_report(y_test,
```

```
y_pred))

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix using Seaborn
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['No
Anxiety', 'Anxiety'], yticklabels=['No Anxiety', 'Anxiety'])

# Labels and title
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')

# Show the plot
plt.show()

# Feature importance
feature_importance = best_rf.feature_importances_
for i, feature in enumerate(features):
    print(f"{feature}: {feature_importance[i]:.4f}")

# Plot feature importance
plt.figure(figsize=(10, 6))
sns.barplot(x=feature_importance, y=features)
plt.xlabel('Feature Importance')
plt.title('Feature Importance in Predicting Anxiety Level')
plt.show()
```

```
Fitting 3 folds for each of 216 candidates, totalling 648 fits
Classification Report:
              precision    recall  f1-score   support

           0       0.52      0.57      0.54       103
           1       0.49      0.43      0.46        97

    accuracy                           0.51       200
   macro avg       0.50      0.50      0.50       200
weighted avg       0.50      0.51      0.50       200
```




```
Panic Attack: 0.0449
Symptom Frequency: 0.1653
Sleep Quality: 0.1016
Study Stress Level: 0.1292
StudyHour/Week: 0.2007
Academic Engagement: 0.1206
Study_Sleep_Interaction: 0.2377
```



EVALUATION & ASSESMENT OF THE QUALITY OF MODELS AND PREDICTION PHASE - Adam Daniel bin Saiful Azly (1211104293)

In this section, we will be evaluating the model of the machine learning implementation in the project and suggest some improvement to it.

(1) Evaluation & Model Assesment

1- Model Performance Analysis

- The model used is Random Forest Classifier with hyperparameter tuning through GridSearchCV.
- Applied balanced class weigths to state potential class imbalance.
- The model is evaluated using:-

1. Classification Report (Precision, Recall, F1-score).
2. Confusion Matrix.
3. Feature Importance Analysis.

2- Observation of Model Results

- If the recall for anxiety class is low: The model (may) misses students who are truely anxious (high false negatives) that is concerning for mental health intervention.

- **Feature Importance scores**: To detect the key predictors that is study stress level, sleep quality, and panic attack.
- **Confusion matrix**: To helps on determine "how well the model distinguishes between anxious and non-anxious atudents.

(2) Explanation and justification of the usefulness and importance of the prediction

- **Early Detection of Mental Health Issues**: The model helps to identify students who are at the risk of of anxiety before the symtomp becomes critical which allows to do earliy remedy.
- **Data-Driven Decision Making**: Institutions and organizations can use the predictions to manage resources effectively to ensure that mental health can help on reaching the students who are needed the most.
- **Reduces the Academic Impact**: Since mental health issues negatively affect academic engagement, targeted interventions can improve students' performance and retention.
- **Designation of personalized Support Strategies**: Academic institutions can create their support programs to fulfill student's needs by unsderstanding the contributed factors towards anxiety.
- **Improving an Overall Well-being**: A proactive method towards mental health to enhance student's benefits and their overall quality of life.

(3) Recommendations for Model Improvement

1- Address the Class Imbalance

- Additional methods that can be implemented in extent to the class weighting including:-

1. Oversampling like SMOTE and ADASYN to generate sanples from underepresented cases.
2. Undersampling to reduce the majority clas, balancing the dataset.

2- Feature Engineering Enhancements

- Since the project introduced an interation feature that is "(studyHour/Week * Sleep Quality)", there can be an additional transformation that we can explore, which are:-

1. Polynomial Features (to capture nonlinear relationships).
2. Log Transformations (to normalize skewed features).

3. Domain-Specific Features: (to combine study stress and panic attack frequency and create a "Crisis Score").

3- Alternative models for better accuracy

1. XGBoost or LightGBM (Often outperform Random Forest in structured data tasks).
2. Stacking Models (Combination of multiple orders which are Random Forest with XGBoost to enhance predictive performance).

4- Explainability with SHAP (SHapley Additive Explanations) Values

1. Offres a more intreptable view of how each feature can affect an individual prediction.
2. Aid on identifying high-risk students with clear justifications.

(4) Recommended actionable insights for Solving the Problem.

1- Institutional interventions based on the insights.

1. High correlation between sleep quality and anxiety (Academic insitutions can apply wellness programs to promote better sleep habits).
2. Study stress is the main factor and predictor (Academic insitutions can itroduce academic stress managemnet programs and counseling services).
3. Panic attack often impact anxiety (Early screening and proactive intervention strategies must be in place).

2- Continuous Model Monitoring and Refinement.

1. Get new data periodically (to retrain and fine-tune the model).
2. Condition: "If any of the features become more predictive over time, future iterations must be prioritized."

3- Leveraging Predictions for Real-WOrld Solutions.

1. Implement model prwedictions to flag at-risk students early.
2. Create a dashboard for counselors (to visualize anxiety risk levels and do timely action).
3. Make sure thatstudent's privacy is intcact while utilizing predictive insights responsibly.