```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.cluster import KMeans



# Load the dataset
file_path = '/content/teenagers_mental_health_dataset.csv'
data = pd.read_csv(file_path)


# Display basic information about the dataset
print("Dataset Info:")
print(data.info())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   ID                      50 non-null     int64 
 1   Age                     50 non-null     int64 
 2   Department              50 non-null     object
 3   Gender                  50 non-null     object
 4   Stress_Level            50 non-null     object
 5   Stress_Sources          50 non-null     object
 6   Sleep_Hours             50 non-null     int64 
 7   Academic_Performance    50 non-null     object
 8   Relaxation_Activities   50 non-null     object
 9   Physical_Activity       50 non-null     object
 10  Social_Media_Usage      50 non-null     object
 11  Social_Media_Impact     50 non-null     object
 12  Financial stress        49 non-null     object
 13  Preferred_App_Features  50 non-null     object
dtypes: int64(3), object(11)
memory usage: 5.6+ KB
None
```

```
# Display the first few rows of the dataset
print("\nFirst few rows:")
print(data.head())
```

```
First few rows:
   ID  Age Department  Gender Stress_Level     Stress_Sources  Sleep_Hours  \
0   1   20       FMSH  Female    Very High  Academic Pressure            6   
1   2   24       FMSH    Male     Moderate            No Data            5   
2   3   19        FOE    Male         High  Academic Pressure            8   
3   4   21       FMSM  Female     Very Low            No Data            5   
4   5   20        FOE    Male     Moderate            No Data            6   

  Academic_Performance Relaxation_Activities Physical_Activity  \
0                    A               Sports          Moderate   
1                    D               Gaming               Low   
2                    D    Listening to Music               Low   
3                    A               Gaming               Low   
4                    A               Gaming               Low   

  Social_Media_Usage Social_Media_Impact Financial stress  \
0          Heavy Use          Positively         Moderate   
1          Heavy Use          Negatively              Low   
2       Moderate Use          Positively         Moderate   
3       Moderate Use          Negatively         Moderate   
4          Heavy Use          Positively             High   

         Preferred_App_Features  
0                         Music  
1                         Books  
2  Books, Games, Music, Chatbot  
3                       Chatbot  
4  Games, Music, Chatbot, Books  
```

```
# Check for missing values
print("\nMissing values:")
print(data.isnull().sum())
```

```
Missing values:
ID                        0
Age                       0
Department                0
Gender                    0
Stress_Level              0
Stress_Sources            0
Sleep_Hours               0
Academic_Performance      0
Relaxation_Activities     0
Physical_Activity         0
Social_Media_Usage        0
Social_Media_Impact       0
Financial stress          1
Preferred_App_Features    0
dtype: int64
```

```
# Summary statistics
print("\nSummary statistics:")
print(data.describe(include='all'))
```

```
Summary statistics:
                ID        Age Department  Gender Stress_Level  \
count   50.00000  50.000000         50      50           50
unique       NaN        NaN          4       2            5
top          NaN        NaN        FOE  Female         High
freq         NaN        NaN         20      25           17
mean    25.50000  21.260000        NaN     NaN          NaN
std     14.57738   1.468166        NaN     NaN          NaN
min      1.00000  19.000000        NaN     NaN          NaN
25%     13.25000  20.000000        NaN     NaN          NaN
50%     25.50000  21.000000        NaN     NaN          NaN
75%     37.75000  22.000000        NaN     NaN          NaN
max     50.00000  24.000000        NaN     NaN          NaN

           Stress_Sources  Sleep_Hours Academic_Performance  \
count                  50    50.000000                   50
unique                  2          NaN                    4
top     Academic Pressure          NaN                    C
freq                   29          NaN                   17
mean                  NaN     5.780000                  NaN
std                   NaN     1.502243                  NaN
min                   NaN     4.000000                  NaN
25%                   NaN     4.000000                  NaN
50%                   NaN     6.000000                  NaN
75%                   NaN     7.000000                  NaN
max                   NaN     8.000000                  NaN

       Relaxation_Activities Physical_Activity Social_Media_Usage  \
count                     50                50                 50
unique                     5                 3                  3
top                   Sports          Moderate          Heavy Use
freq                      15                24                 28
mean                     NaN               NaN                NaN
std                      NaN               NaN                NaN
min                      NaN               NaN                NaN
25%                      NaN               NaN                NaN
50%                      NaN               NaN                NaN
75%                      NaN               NaN                NaN
max                      NaN               NaN                NaN

       Social_Media_Impact Financial stress Preferred_App_Features
count                   50               49                     50
unique                   2                3                     29
top             Positively         Moderate                Chatbot
freq                    33               25                      4
mean                   NaN              NaN                    NaN
std                    NaN              NaN                    NaN
min                    NaN              NaN                    NaN
25%                    NaN              NaN                    NaN
50%                    NaN              NaN                    NaN
75%                    NaN              NaN                    NaN
max                    NaN              NaN                    NaN
```

```
# Correlation analysis
if 'EducationalPerformance' in data.columns:
    print("\nCorrelation with Educational Performance:")
    print(data.corr()['EducationalPerformance'].sort_values(ascending=False))
```

```
# Visualizations
# Distribution of educational performance
plt.figure(figsize=(8, 6))
sns.histplot(data['Academic_Performance'], kde=True, bins=20, color='blue')
plt.title('Distribution of Educational Performance')
plt.xlabel('Educational Performance')
plt.ylabel('Frequency')
plt.show()
```

```
# Heatmap of correlations
plt.figure(figsize=(10, 8))
# Select only numeric features for correlation calculation
numeric_data = data.select_dtypes(include=np.number)
correlation_matrix = numeric_data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

```
# Bar Chart: Count of Mental Health Categories
plt.figure(figsize=(8, 6))
sns.countplot(x='Relaxation_Activities', data=data, palette='viridis')
plt.title('Count of Relaxation_Activities')
plt.xlabel('Mental Health Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

```
# Pie Chart: Distribution of Stress Levels
stress_counts = data['Stress_Level'].value_counts()
plt.figure(figsize=(8, 8))
wedges, texts, autotexts = plt.pie(
    stress_counts,
    labels=stress_counts.index,
    autopct='%1.1f%%',
    startangle=140,
    colors=sns.color_palette('pastel')
)
plt.legend(wedges, stress_counts.index, title="Stress Levels", loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))
plt.title('Distribution of Stress Levels')
plt.show()
```

```
# Line Chart: Age vs Average Sleep Hours
age_sleep = data.groupby('Age')['Sleep_Hours'].mean().reset_index()
plt.figure(figsize=(10, 6))
sns.lineplot(data=age_sleep, x='Age', y='Sleep_Hours', marker='o', color='blue')
plt.title('Average Sleep Hours by Age')
plt.xlabel('Age')
plt.ylabel('Average Sleep Hours')
plt.grid(True)
plt.show()
```

```
# Select relevant features for clustering
features = ['Stress_Level', 'Sleep_Hours', 'Physical_Activity', 'Social_Media_Usage']

# Encode categorical features
label_encoder = LabelEncoder()
for col in ['Stress_Level', 'Physical_Activity', 'Social_Media_Usage']:
    data[col] = label_encoder.fit_transform(data[col])

# Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data[features])

# Determine the optimal number of clusters using the Elbow Method
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data_scaled)
    inertia.append(kmeans.inertia_)

# Plot the Elbow Curve
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal Clusters')
plt.show()
```

```
# Apply K-Means clustering with the optimal number of clusters (e.g., k=3)
kmeans = KMeans(n_clusters=3, random_state=42)
data['Cluster'] = kmeans.fit_predict(data_scaled)

# Analyze the clusters
cluster_analysis = data.groupby('Cluster')[features].mean()

# Visualize the clusters using a scatter plot (e.g., Stress_Level vs Sleep_Hours
sns.scatterplot(
    x=data['Sleep_Hours'],
    y=data['Stress_Level'],
    hue=data['Cluster'],
    palette='viridis',
    style=data['Cluster'],
    s=100
)
plt.title('Clusters of Stress Levels')
plt.xlabel('Sleep Hours')
plt.ylabel('Stress Level')
plt.legend(title='Cluster')
plt.show()
```

```
# Display the cluster analysis
print("Cluster Analysis:")
print(cluster_analysis)
```

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

# Select relevant columns for Association Rule Mining
columns = ['Preferred_App_Features', 'Relaxation_Activities']
data = data[columns].fillna('')

# Combine preferences into transactional format
data['Transactions'] = data['Preferred_App_Features'] + ',' + data['Relaxation_Activities']
transactions = data['Transactions'].str.split(',').tolist()

# Create a one-hot encoded DataFrame for the transactions
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_data = te.fit(transactions).transform(transactions)
one_hot_data = pd.DataFrame(te_data, columns=te.columns_)

# Step 1: Find frequent itemsets
frequent_itemsets = apriori(one_hot_data, min_support=0.1, use_colnames=True)
print("Frequent Itemsets:")
print(frequent_itemsets)

# Step 2: Generate association rules

rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.0, support_only=False, num_itemsets=2)

# Filter rules for higher relevance (e.g., confidence > 0.7)
filtered_rules = rules[(rules['confidence'] > 0.7) & (rules['lift'] > 1.2)]
print("\nAssociation Rules:")
print(filtered_rules)

# Step 3: Analyze and visualize results
import matplotlib.pyplot as plt
import seaborn as sns

# Plot support vs confidence
plt.figure(figsize=(8, 6))
sns.scatterplot(x='support', y='confidence', size='lift', hue='lift', data=filtered_rules, palette='viridis', sizes=(40, 400))
plt.title('Support vs Confidence')
```