In [4]:
```
import pandas as pd
import numpy as np
import seaborn as sb
import pylab as py
import matplotlib.pyplot as plt
```

In [5]:
```
df = pd.read_csv(r'C:\Users\Motomoto\Downloads\mental_health_social_media_dataset.c
```

In [6]:
```
df.shape #גודל הנתונים
```

Out[6]: (5000, 15)

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 15 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   person_name                  5000 non-null   object 
 1   age                          5000 non-null   int64  
 2   date                         5000 non-null   object 
 3   gender                       5000 non-null   object 
 4   platform                     5000 non-null   object 
 5   daily_screen_time_min        5000 non-null   int64  
 6   social_media_time_min        5000 non-null   int64  
 7   negative_interactions_count  5000 non-null   int64  
 8   positive_interactions_count  5000 non-null   int64  
 9   sleep_hours                  5000 non-null   float64
 10  physical_activity_min        5000 non-null   int64  
 11  anxiety_level                5000 non-null   int64  
 12  stress_level                 5000 non-null   int64  
 13  mood_level                   5000 non-null   int64  
 14  mental_state                 5000 non-null   object 
dtypes: float64(1), int64(9), object(5)
memory usage: 586.1+ KB
```

In [8]:
```
df.head() #הצגת השורות הראשונות בטבלה
```

Out[8]:

| | person_name | age | date | gender | platform | daily_screen_time_min | social_media_ti |
|---|---|---|---|---|---|---|---|
| **0** | Reyansh Ghosh | 35 | 1/1/2024 | Male | Instagram | 320 | |
| **1** | Neha Patel | 24 | 1/12/2024 | Female | Instagram | 453 | |
| **2** | Ananya Naidu | 26 | 1/6/2024 | Male | Snapchat | 357 | |
| **3** | Neha Das | 66 | 1/17/2024 | Female | Snapchat | 190 | |
| **4** | Reyansh Banerjee | 31 | 1/28/2024 | Male | Snapchat | 383 | |

In [9]:
```
df.describe()
```

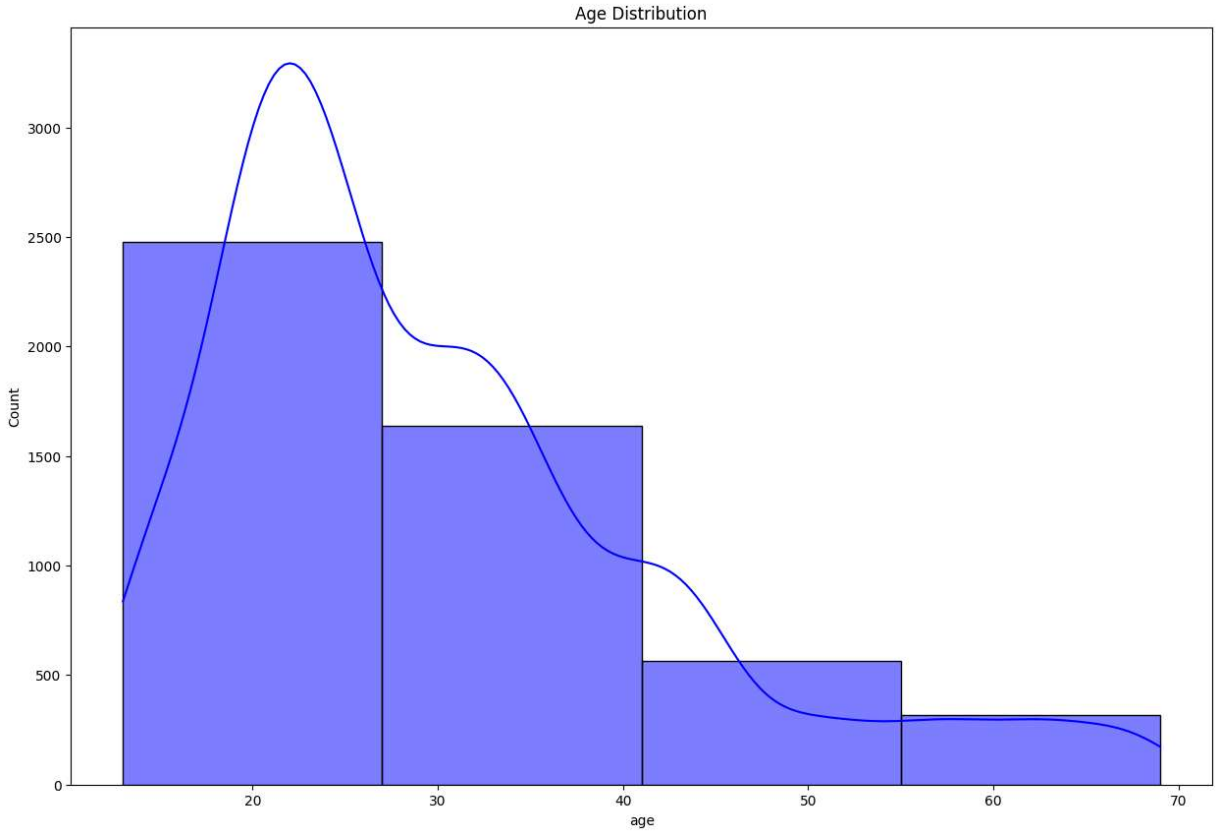
Out[9]:

| | age | daily_screen_time_min | social_media_time_min | negative_interactions_cou |
|---|---|---|---|---|
| **count** | 5000.000000 | 5000.000000 | 5000.000000 | 5000.0000 |
| **mean** | 29.947800 | 373.058200 | 175.331600 | 0.8642 |
| **std** | 12.279936 | 106.003916 | 71.209329 | 0.5551 |
| **min** | 13.000000 | 140.000000 | 35.000000 | 0.0000 |
| **25%** | 21.000000 | 310.000000 | 118.000000 | 1.0000 |
| **50%** | 27.000000 | 388.000000 | 170.000000 | 1.0000 |
| **75%** | 35.250000 | 461.000000 | 231.000000 | 1.0000 |
| **max** | 69.000000 | 520.000000 | 338.000000 | 2.0000 |

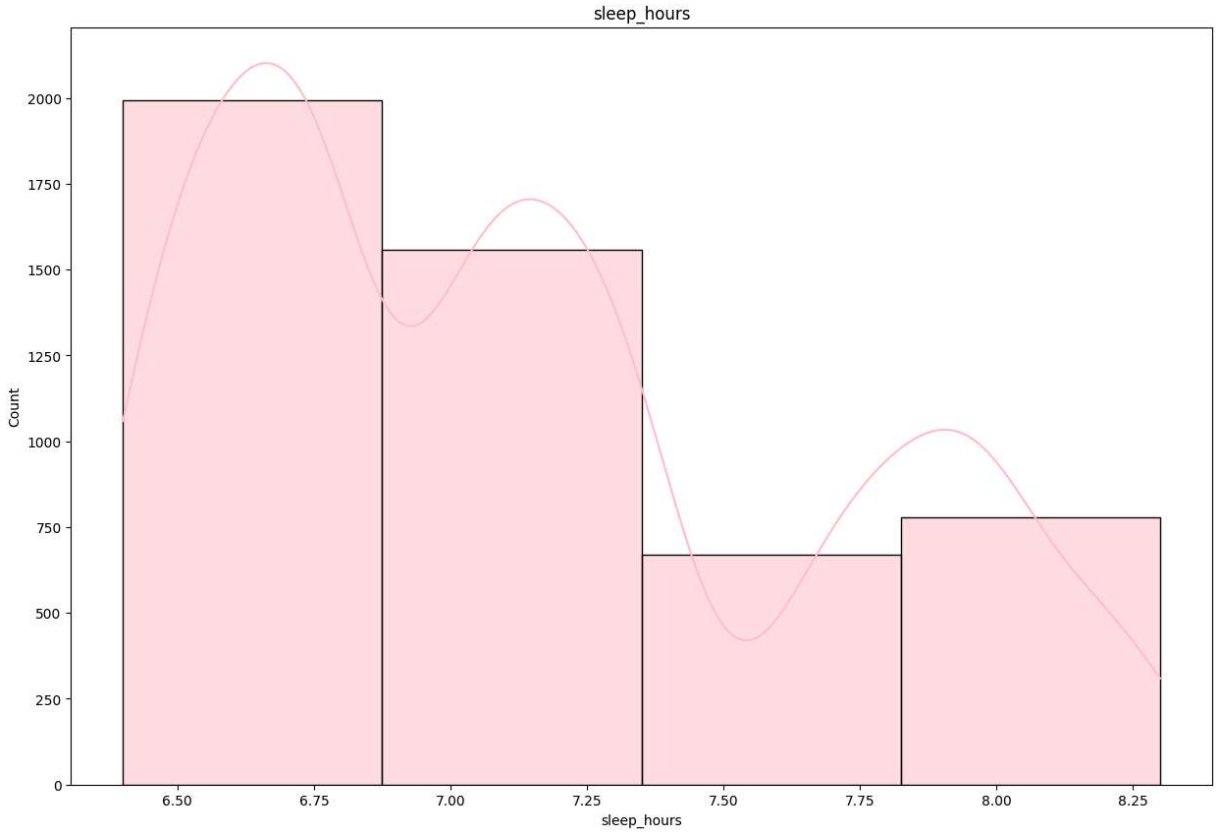
In [10]:
```
df.isnull().sum()
```

Out[10]:
```
person_name                   0
age                           0
date                          0
gender                        0
platform                      0
daily_screen_time_min         0
social_media_time_min         0
negative_interactions_count   0
positive_interactions_count   0
sleep_hours                   0
physical_activity_min         0
anxiety_level                 0
stress_level                  0
mood_level                    0
mental_state                  0
dtype: int64
```

In [11]:
```
#התפלגות טווחי הגילאים בקובץ מחקר זה
plt.figure(figsize=(15,10))
sb.histplot(data=df, x='age', kde=True, bins=4,color='blue')
plt.title("Age Distribution")
plt.show()
```




In [78]:
```
plt.figure(figsize=(15,10))
sns.histplot(data=df, x='sleep_hours', kde=True, bins=4,color='pink')
plt.title("sleep_hours")
plt.show()
#המשתמשים ביותר זמן ברשתות חברתיות ישנים פחות
```



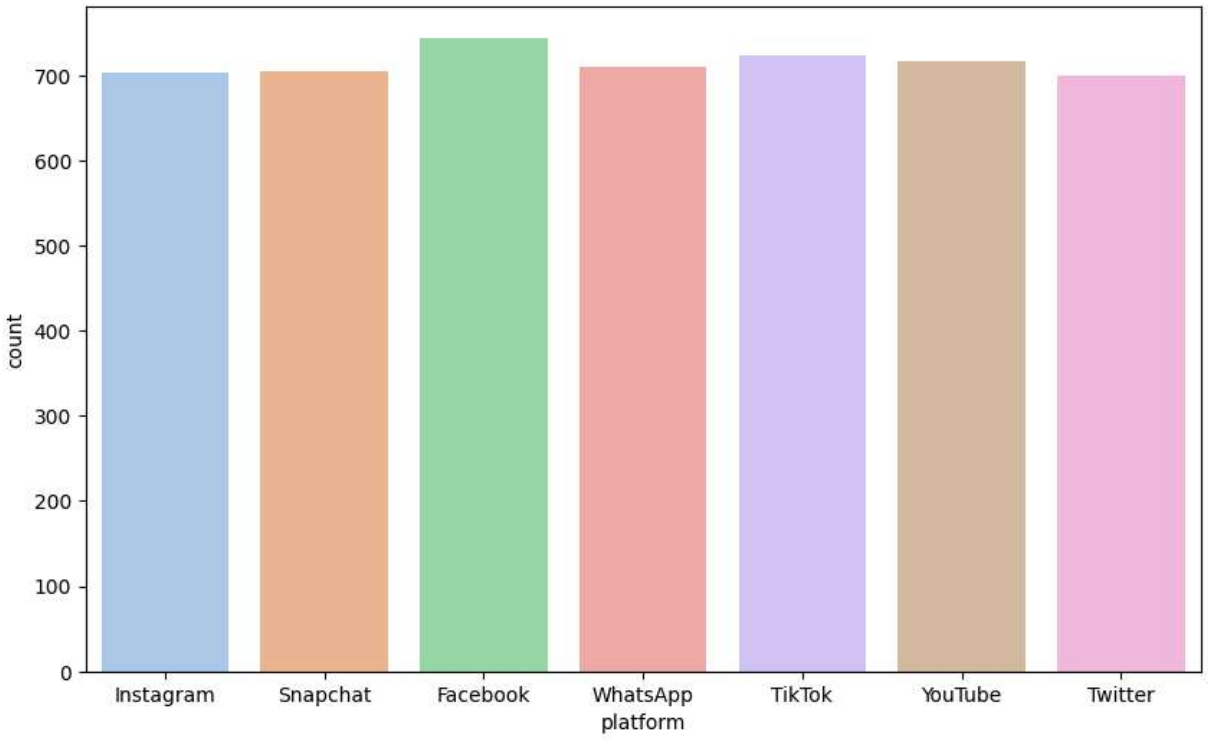

In [38]:
```
df['platform'].value_counts()
```

Out[38]:
```
platform
Facebook     744
TikTok       723
YouTube      716
WhatsApp     710
Snapchat     705
Instagram    703
Twitter      699
Name: count, dtype: int64
```

In [44]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='platform',palette='pastel')
```

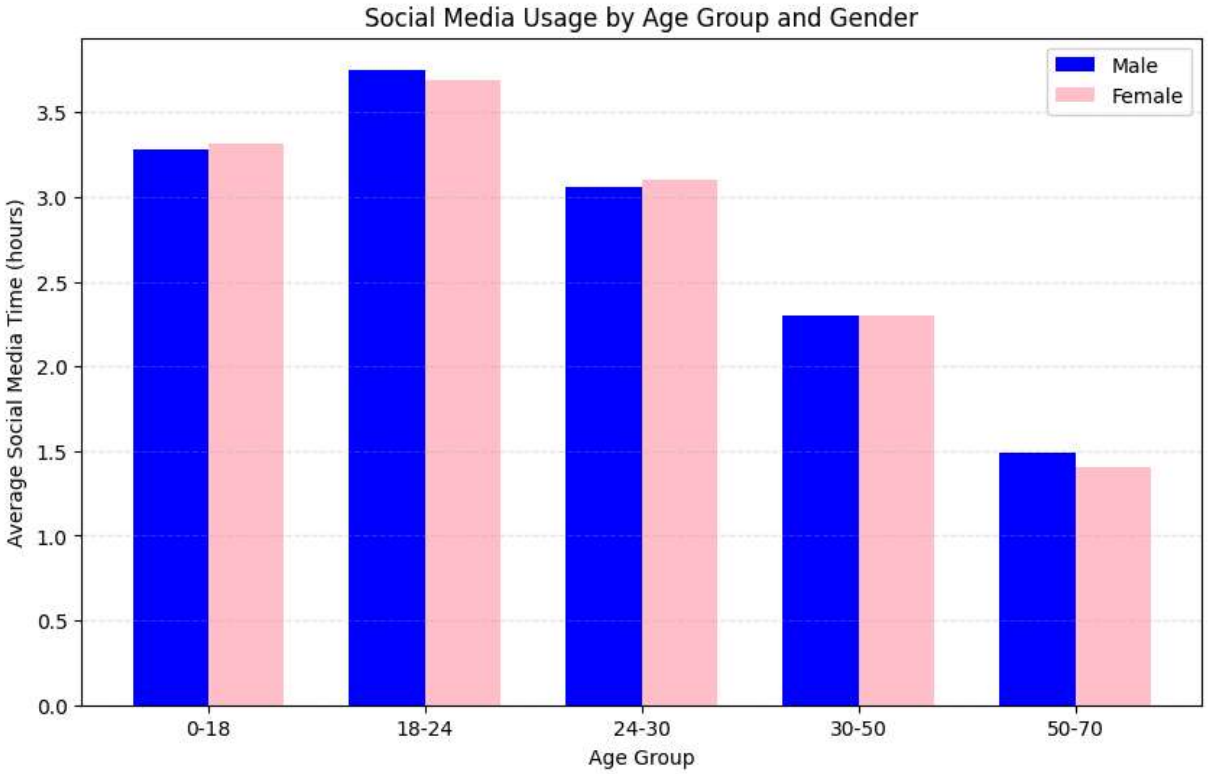
```
C:\Users\Motomoto\AppData\Local\Temp\ipykernel_10112\3266657434.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x='platform',palette='pastel')
```

Out[44]:
```
<Axes: xlabel='platform', ylabel='count'>
```

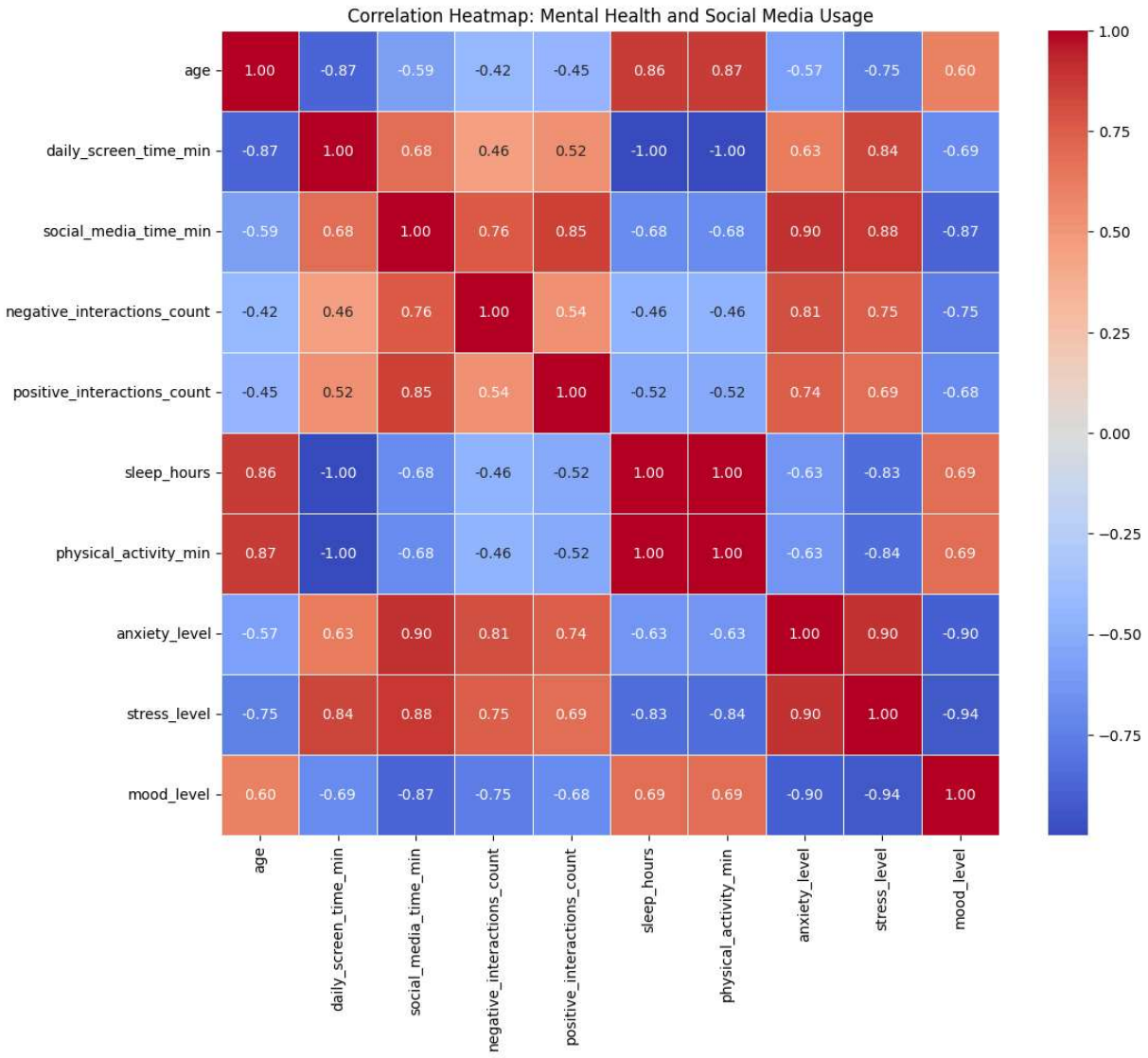



In [14]:
```
df['social_media_time_hours'] = df['social_media_time_min'] / 60
bins = [0, 18, 24, 30, 50, 70]
labels = ['0-18', '18-24', '24-30', '30-50', '50-70']
df['age_group'] = pd.cut(df['age'], bins=bins, labels=labels)

#  חישוב ממוצע זמן ברשתות חברתיות לפי מגדר וקבוצת גיל
usage_stats = df.groupby(['age_group', 'gender'])['social_media_time_hours'].mean()
usage_pivot = usage_stats.pivot(index='age_group', columns='gender', values='social

# בניית הגרף
plt.figure(figsize=(10,6))
x = range(len(usage_pivot))
bar_width = 0.35

# ציור עמודות לגברים (Male)
plt.bar([i - bar_width/2 for i in x], usage_pivot['Male'],
        width=bar_width, color='blue', label='Male')

# ציור עמודות לנשים (Female)
plt.bar([i + bar_width/2 for i in x], usage_pivot['Female'],
        width=bar_width, color='pink', label='Female')

# עיצוב הגרף
plt.xticks(x, usage_pivot.index)
plt.ylabel("Average Social Media Time (hours)")
plt.xlabel("Age Group")
plt.title("Social Media Usage by Age Group and Gender")
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.3)

plt.show()
```

```
C:\Users\Motomoto\AppData\Local\Temp\ipykernel_3108\11942437.py:7: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
```



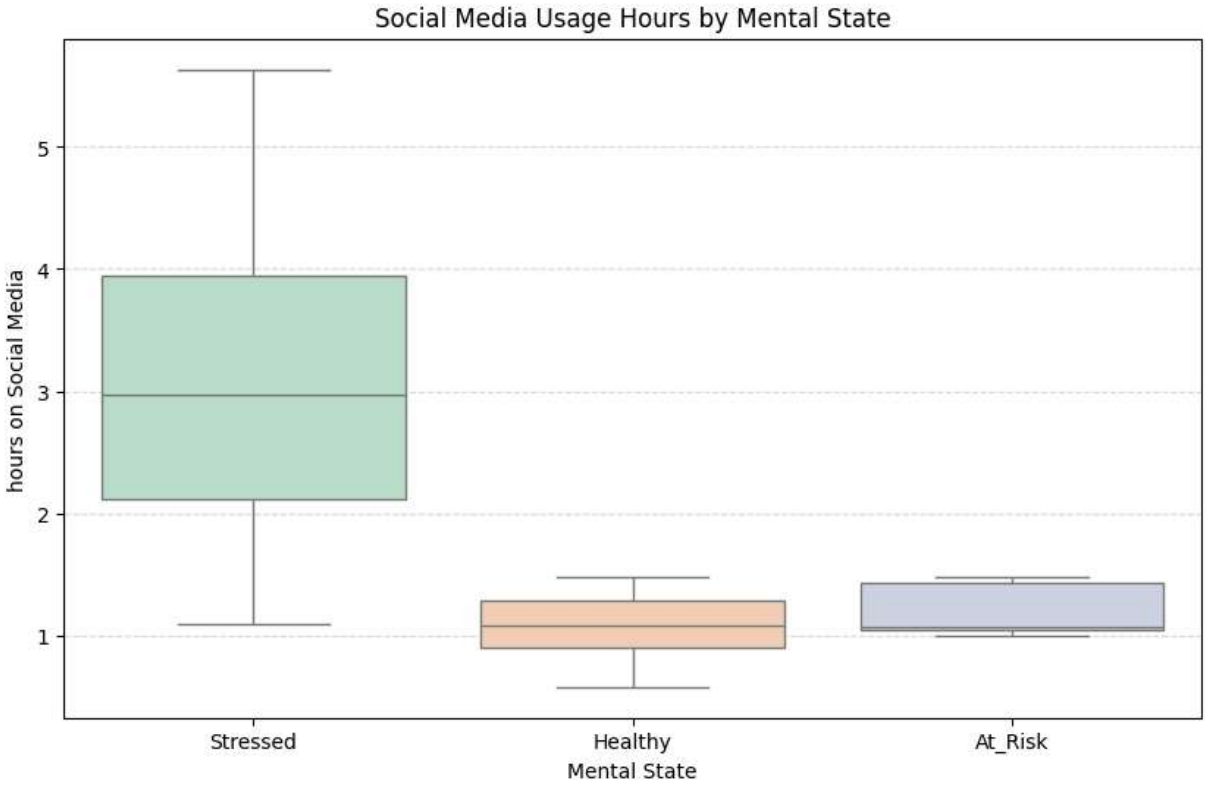

In [5]:
```
numerical_cols = ['age', 'daily_screen_time_min', 'social_media_time_min',
                  'negative_interactions_count', 'positive_interactions_count',
                  'sleep_hours', 'physical_activity_min', 'anxiety_level',
                  'stress_level', 'mood_level']
corr_matrix = df[numerical_cols].corr()

# יצירת גרף מפת חום
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap: Mental Health and Social Media Usage')

#הצגת הגרף והסבר על הקשרים בין הנתונים:
#קרוב ל-1+ (צבע חם/אדום): קשר חיובי חזק. כשמשתנה אחד עולה, השני עולה איתו.
#קרוב ל-1- (צבע קר/כחול): קשר שלילי חזק. כשמשתנה אחד עולה, השני יורד
# סביב 0 אין קשר לינאר בין המספרים
plt.show()
```




In [ ]:
```
#קשר הפוך חזק ללחץ: שעות השינה הן הגורם בעל המתאם השלילי החזק ביותר עם סטרס
#שיפור מצב הרוח: השינה היא הגורם המרכזי שמעלה את רמת מצב הרוח
#אינטראקציות חברתיות ברשת-מקושרות בעלייה בחרדה
#זמן שימוש: ככל שזמן השימוש ברשתות החברתיות עולה, ישנה נטייה לעלייה ברמת הסטרס והחרדה.
#קשר חיובי חזק ככל שרמת הסטרס עולה ככה רמת החרדה
```

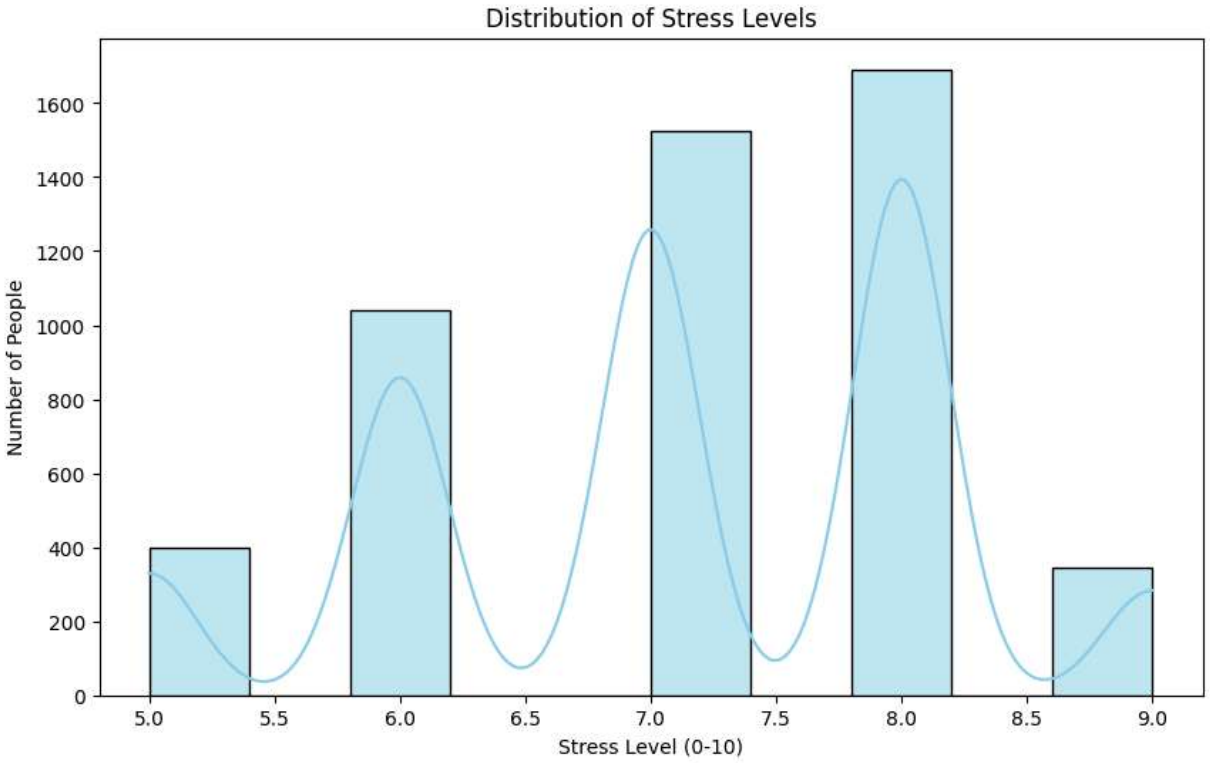
In [45]:
```
import seaborn as sns
import matplotlib.pyplot as plt
df['social_media_time_hours'] = df['social_media_time_min'] / 60
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='mental_state', y='social_media_time_hours', palette='Pastel

plt.title('Social Media Usage Hours by Mental State')
plt.xlabel('Mental State')
plt.ylabel('hours on Social Media')
plt.grid(axis='y', linestyle='--', alpha=0.4)
plt.show()
```

```
C:\Users\Motomoto\AppData\Local\Temp\ipykernel_10112\3612369841.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=df, x='mental_state', y='social_media_time_hours', palette='Paste
l2')
```

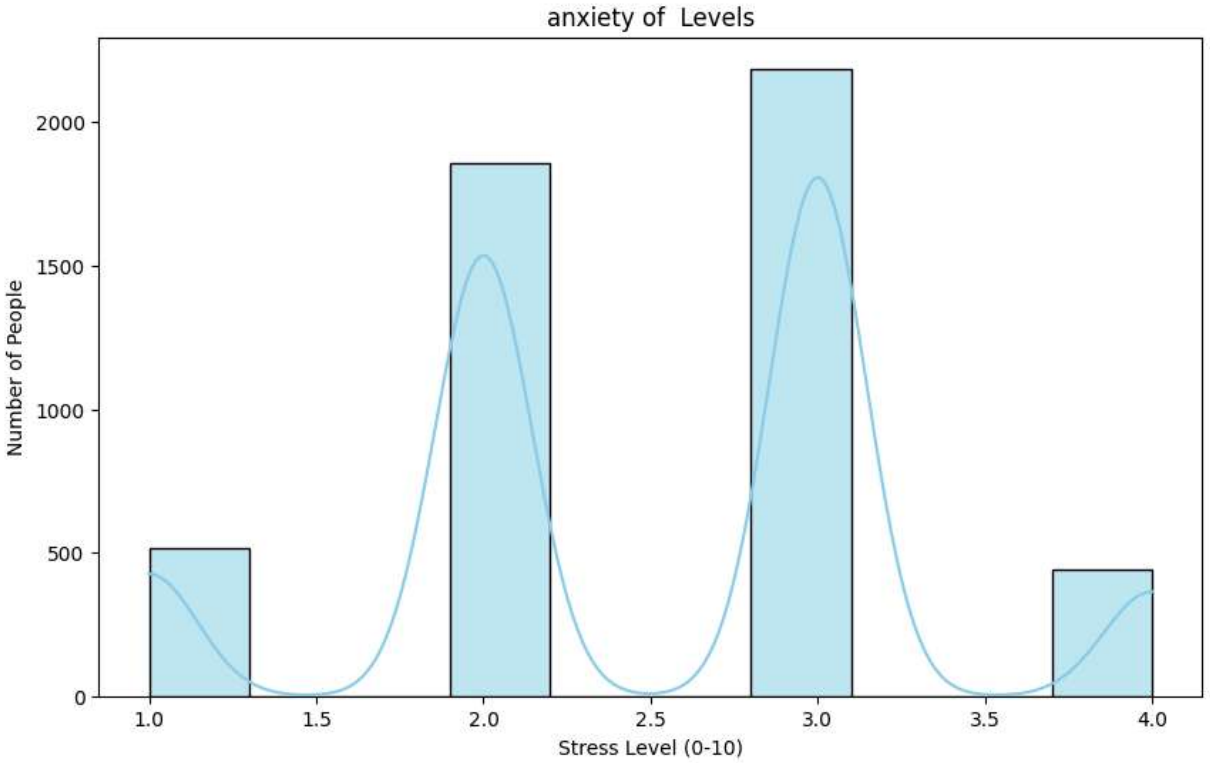



In [ ]:
```
#מסקנה: אנשים לחוצים משתמשים יותר ברשתות החברתיות מאנשים בריאים נפשית
#החציון של זמן השימוש בקבוצת Stressed גבוה משמעותית מזה של קבוצת Healthy.
#אנשים לחוצים קיימים הבדלים גדולים בהרגלי השימוש, ואילו אצל אנשים בריאים השימוש דומה יותר בין רוב המשתתפים
#ניתן לראות טווח צר בקבוצת Stressed הרחב בקבוצת Healthy (IQR) לעומת טווח הרבעונים
```

In [11]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df['stress_level'], bins=10, kde=True, color='skyblue')

plt.title('Distribution of Stress Levels')
plt.xlabel('Stress Level (0-10)')
plt.ylabel('Number of People')
plt.show()
```

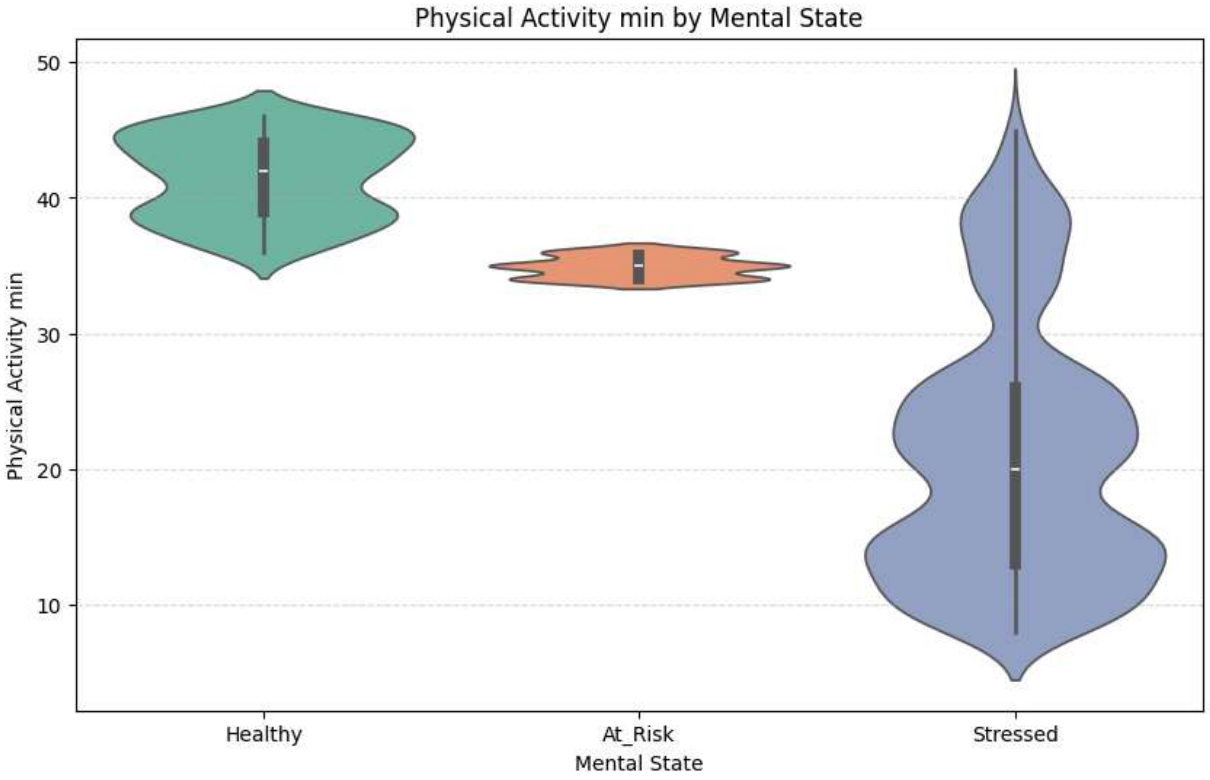



In [14]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df['anxiety_level'], bins=10, kde=True, color='skyblue')

plt.title('anxiety of  Levels')
plt.xlabel('Stress Level (0-10)')
plt.ylabel('Number of People')
plt.show()
```



In [79]:
```
plt.figure(figsize=(10, 6))

order = ['Healthy', 'At_Risk', 'Stressed']

sns.violinplot(data=df, x='mental_state', y='physical_activity_min',
               palette='Set2', order=order)

plt.title('Physical Activity min by Mental State')
plt.xlabel('Mental State')
plt.ylabel('Physical Activity min')
plt.grid(axis='y', linestyle='--', alpha=0.4)
plt.show()
```

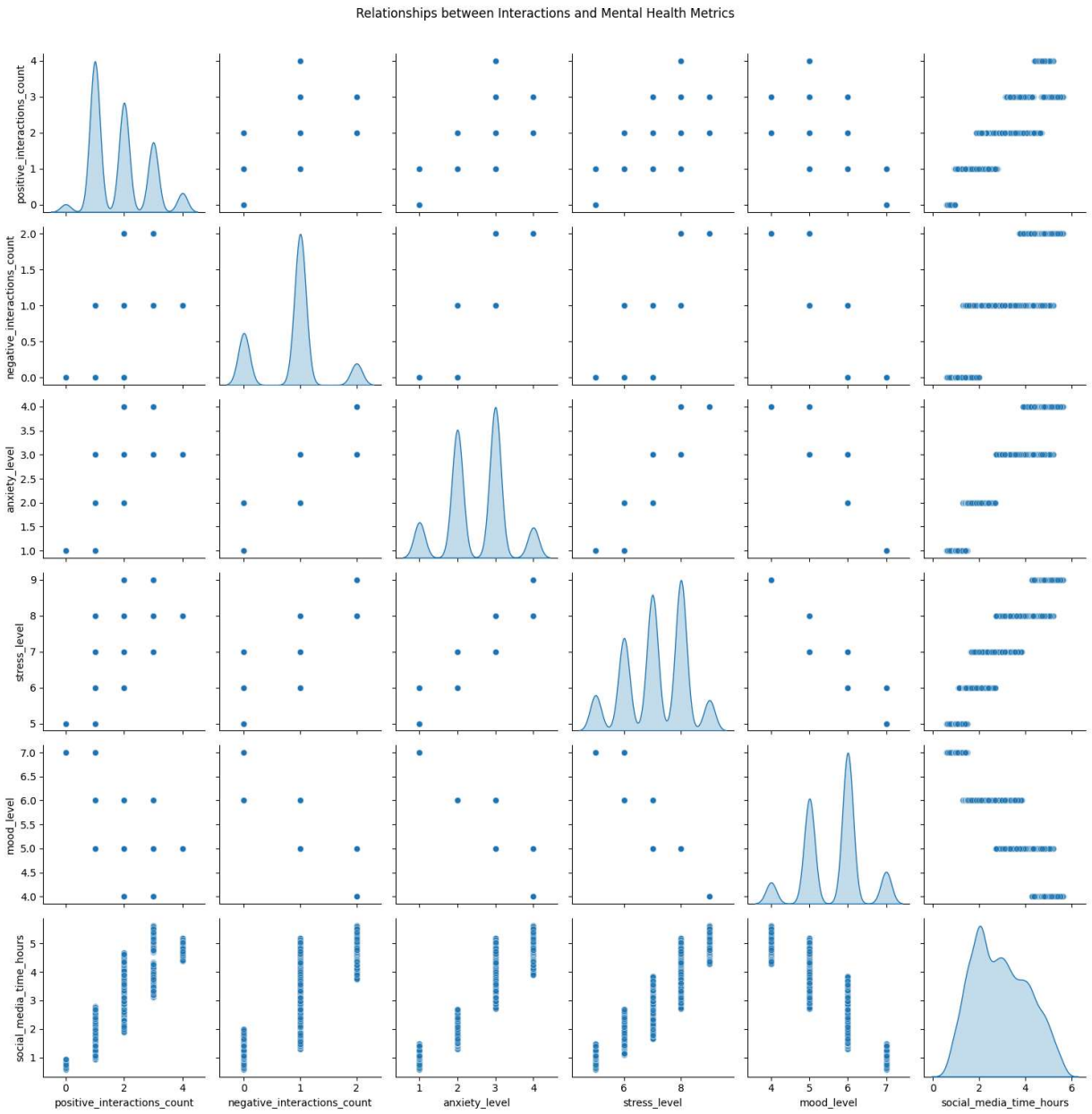
```
C:\Users\Motomoto\AppData\Local\Temp\ipykernel_10112\3634138041.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.violinplot(data=df, x='mental_state', y='physical_activity_min',
```




In [ ]:
```
#גופנית על מצבים נפשים תוצאה---> ככל שאתה מתאמן יותר דקות/שעות ביום אתה יותר בריא פחות חווה מתיחות נפשית
```

In [61]:
```
df['social_media_time_hours'] = df['social_media_time_min'] / 60

columns = ['positive_interactions_count', 'negative_interactions_count',
           'anxiety_level', 'stress_level', 'mood_level', 'social_media_time_hours'

# יצירת מטריקס של גרפי פיזור
sns.pairplot(df[columns], diag_kind='kde', plot_kws={'alpha': 0.6})
plt.suptitle('Relationships between Interactions and Mental Health Metrics', y=1.01
```

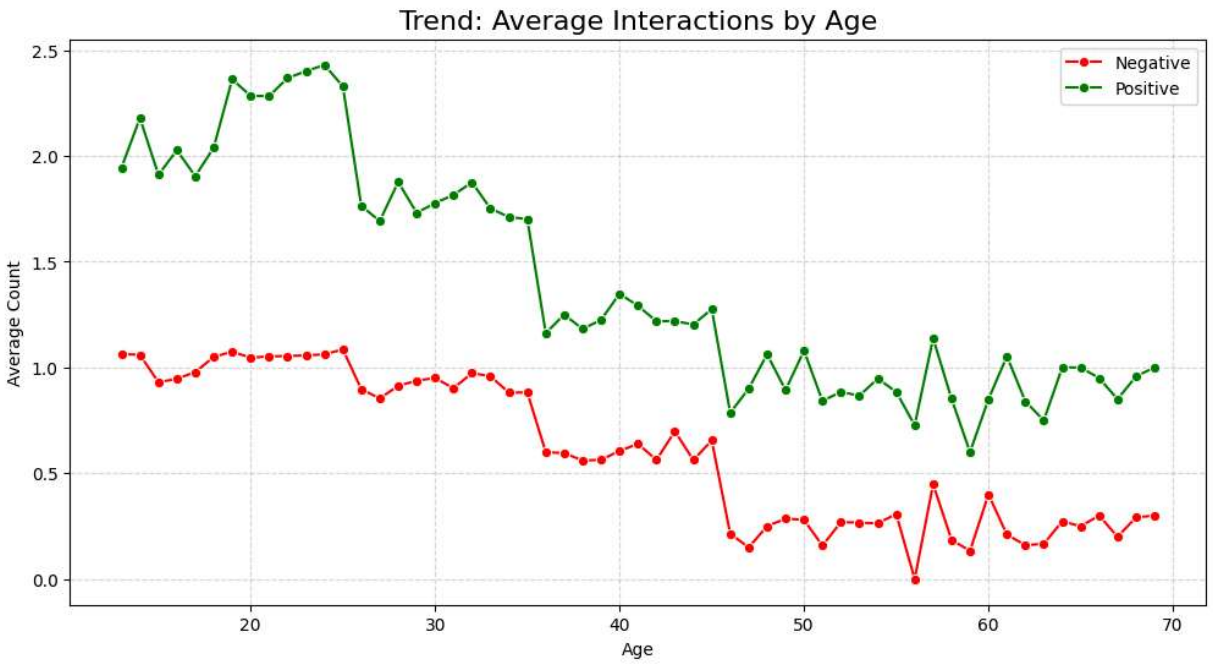
```
plt.tight_layout()
plt.show()
```




In [62]:
```
#הקשר בין הגיל  לאינטקרציות חיוביות או שליליות
```

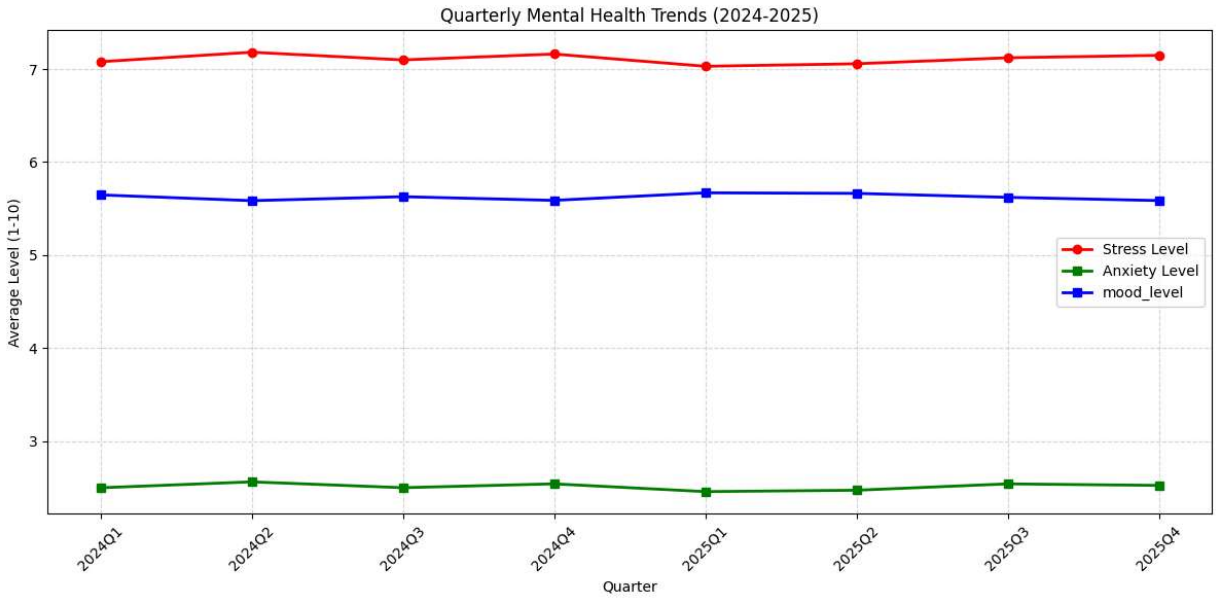
In [66]:
```
#פילטור להקשר בין אטרקציות חיוביות או שליליות
age_stats = df.groupby('age')[['negative_interactions_count', 'positive_interaction
#הגרף
plt.figure(figsize=(12, 6))
sns.lineplot(data=age_stats, x='age', y='negative_interactions_count', label='Negat
sns.lineplot(data=age_stats, x='age', y='positive_interactions_count', label='Posit

plt.title('Trend: Average Interactions by Age', fontsize=16)
plt.xlabel('Age')
plt.ylabel('Average Count')
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
plt.show()
```

```
#ניתן לראות שככל שיותר צעירים יש אינטרקציות יותר חיובות מהפלטפורמות החברתיות כלומר יותר מרוצים
#יותר מועט כלומר הטווח משתנה אין לה ממש זמן ושימוש במדיה החברתית כלומר אין הפעה ניכרת על רמות מנטל סטייט
```




In [ ]:
```
#הקשר בין הזמן עלייה ברמות המנטליות
```

In [74]:
```
df['date'] = pd.to_datetime(df['date'])

#יצירת רבעונים כל שנה
df['quarter'] = df['date'].dt.to_period('Q').astype(str)

# חישוב ממוצעים רבעוניים
quarterly_stats = df.groupby('quarter')[['stress_level', 'anxiety_level', 'mood_lev

# יצירת הגרף
plt.figure(figsize=(12, 6))
plt.plot(quarterly_stats.index, quarterly_stats['stress_level'], marker='o', label=
plt.plot(quarterly_stats.index, quarterly_stats['anxiety_level'], marker='s', label
plt.plot(quarterly_stats.index, quarterly_stats['mood_level'], marker='s', label='m

plt.title('Quarterly Mental Health Trends (2024-2025)')
plt.ylabel('Average Level (1-10)')
plt.xlabel('Quarter')
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




In [ ]:
```
#קבוע-תמידי נשאר גבוהה בכל שמני הרבעונים ברמה7 משמעשהרישתות החברתיות יוצרות לחץ מתמך שלא חולף עם הזמן#
#ות המצבים ניתן לראות ש/לאורך שנים המשתשמשים ברשתות נשארים באותם השפעה של מצבים נפשים בשימוש ברשתות#
#קשר ישיר ברגע שסטרטס עולה גם החרדה עולה והשינוי במצב רוח עולה#
```

```
# =====Imports =====
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, f1_score, classification_report, confusion_matrix
```

```
# =====  Load dataset =====
df = pd.read_csv("mental_health_social_media_dataset.csv")

# Quick look at the data
print(df.shape)
df.head()
```

```
(5000, 15)
```

| | person_name | age | date | gender | platform | daily_screen_time_min | social_media_time_min | negative_interactions_count | positive_interactio |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Reyansh Ghosh | 35 | 1/1/2024 | Male | Instagram | 320 | 160 | 1 | |
| **1** | Neha Patel | 24 | 1/12/2024 | Female | Instagram | 453 | 226 | 1 | |
| **2** | Ananya Naidu | 26 | 1/6/2024 | Male | Snapchat | 357 | 196 | 1 | |
| **3** | Neha Das | 66 | 1/17/2024 | Female | Snapchat | 190 | 105 | 0 | |
| **4** | Reyansh Banerjee | 31 | 1/28/2024 | Male | Snapchat | 383 | 211 | 1 | |

We use mental_state as Y' because the teacher requested this target. This is a multi-class classification problem (Healthy / At_Risk / Stressed). Value counts help us see class imbalance before training models

Most samples belong to the "Stressed" class

```
# ===== Basic checks =====
print(df.columns.tolist())
print(df["mental_state"].value_counts())
```

```
['person_name', 'age', 'date', 'gender', 'platform', 'daily_screen_time_min', 'social_media_time_min', 'negative_interactions_count', 'positive
mental_state
Stressed    4601
Healthy      341
At_Risk       58
Name: count, dtype: int64
```

```
# =====  Drop non-informative columns =====
# Remove identifiers and unused columns
df = df.drop(columns=["person_name", "date"])
```

Models work with numeric labels, so we encode mental_state into integers. // LabelEncoder maps each class name to a number (e.g., Healthy -> 1)

dict(zip(...)) prints the mapping so we can interpret results later

```
# =====  Encode target variable =====
# Convert target labels to numeric values
le_target = LabelEncoder()
df["mental_state_encoded"] = le_target.fit_transform(df["mental_state"])

# Check label mapping
dict(zip(le_target.classes_, le_target.transform(le_target.classes_)))
```

```
{'At_Risk': np.int64(0), 'Healthy': np.int64(1), 'Stressed': np.int64(2)}
```

gender and platform are categorical, so we convert them to dummy variables

One-hot encoding avoids giving an artificial numeric order to categories

This makes features usable for all three models

```
# =====   Encode categorical features =====
# Encode categorical input features
df_encoded = pd.get_dummies(df, columns=["gender", "platform"], drop_first=True)

print(df_encoded.shape)
```

```
(5000, 20)
```

↑ Categorical features encoded using one-hot encoding

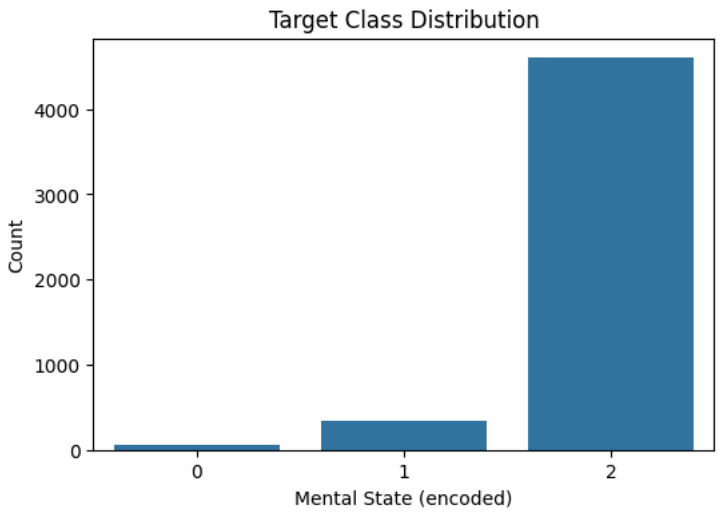
X contains input features used for prediction ----//--- Y is the target label (encoded mental_state) --- / --- We drop target columns from X to prevent data leakage

```
# =====  Define features and target =====
X = df_encoded.drop(columns=["mental_state", "mental_state_encoded"])
y = df_encoded["mental_state_encoded"]

print(X.shape, y.shape)
```

```
(5000, 18) (5000,)
```

↑ Feature matrix and target vector prepared for modeling

```
# ===== Check target distribution =====
# Visualize class imbalance
plt.figure(figsize=(6,4))
sns.countplot(x=y)
plt.title("Target Class Distribution")
plt.xlabel("Mental State (encoded)")
plt.ylabel("Count")
plt.show()
```



Split data into train and test sets to evaluate on unseen samples --/ -- stratify=y keeps the same class proportions in both sets

```
# =====  Train-test split =====
# Split data while preserving class distribution
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print(X_train.shape, X_test.shape)
```

```
(4000, 18) (1000, 18)
```

↑# Train (4000) and test (1000)

```
# ==========================================
# ===== First Model Random Forest model =====
# ==========================================

from sklearn.ensemble import RandomForestClassifier

# Initialize Random Forest classifier
rf_model = RandomForestClassifier(
    n_estimators=200,
    random_state=42,
    class_weight="balanced"
)
```

```
# ===== Train and predict =====
# Train the model
rf_model.fit(X_train, y_train)

# Predict on test set
rf_pred = rf_model.predict(X_test)
```

Accuracy is the overall percent of correct predictions

Macro F1 averages F1 across classes, treating each class equally --/--- mental_state shows strong class imbalance (Stressed >> Healthy, At_Risk) ---/--- Macro F1 gives equal importance to minority classes

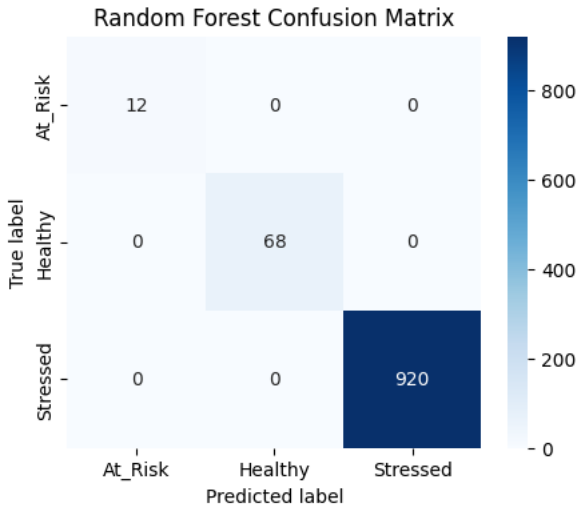
```
# =====  Evaluation metrics =====
rf_acc = accuracy_score(y_test, rf_pred)
rf_f1 = f1_score(y_test, rf_pred, average="macro")  #Macro F1 forces the model to perform well on minority classes.

print("Random Forest Accuracy:", round(rf_acc, 4))
print("Random Forest F1-macro:", round(rf_f1, 4))
```

```
Random Forest Accuracy: 1.0
Random Forest F1-macro: 1.0
```

↑ # Perfect performance likely due to strong relationship between features and target

```
# =====  Confusion matrix =====
cm = confusion_matrix(y_test, rf_pred)

plt.figure(figsize=(5,4))
sns.heatmap(
    cm,
    annot=True,
    fmt="d",
    cmap="Blues",
    xticklabels=le_target.classes_,
    yticklabels=le_target.classes_
)
```

```
plt.title("Random Forest Confusion Matrix")
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.show()
```

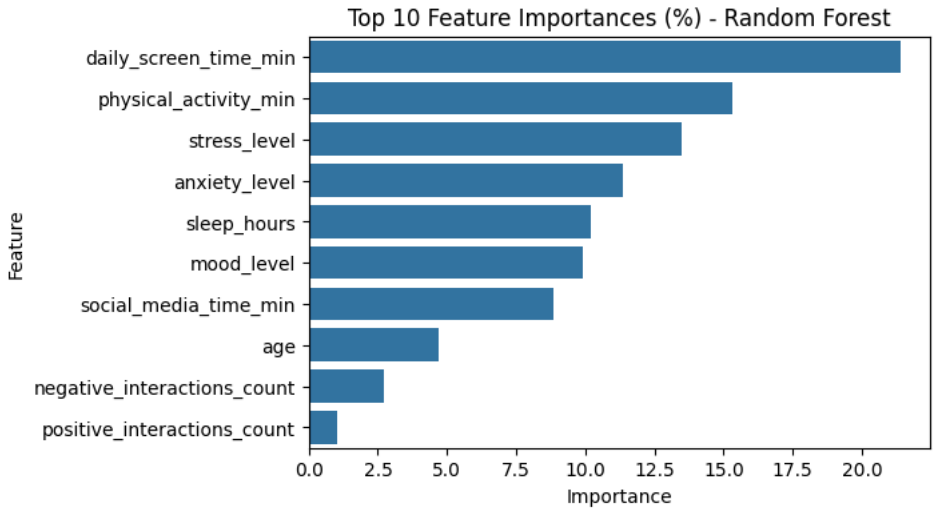



```
# ===== Classification report =====
print(classification_report(
    y_test,
    rf_pred,
    target_names=le_target.classes_
))
```

```
              precision    recall  f1-score   support

     At_Risk       1.00      1.00      1.00        12
     Healthy       1.00      1.00      1.00        68
    Stressed       1.00      1.00      1.00       920

    accuracy                           1.00      1000
   macro avg       1.00      1.00      1.00      1000
weighted avg       1.00      1.00      1.00      1000
```

↑ Perfect scores may indicate very strong feature-target relationships in this dataset

```
# ===== Feature importance =====
# Convert feature importances to percentages for better interpretation
importances = rf_model.feature_importances_*100
feature_names = X.columns

rf_importance = pd.Series(importances, index=feature_names)
rf_importance = rf_importance.sort_values(ascending=False).head(10)

plt.figure(figsize=(6,4))
sns.barplot(x=rf_importance.values, y=rf_importance.index)
plt.title("Top 10 Feature Importances (%) - Random Forest")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.show()
```




↑Daily Screen time and activity levels dominate feature importance in Random Forest

```
# ==========================================
# ===== Second Model LightGBM  model =====
# ==========================================

import lightgbm as lgb

# Initialize LightGBM classifier
lgbm_model = lgb.LGBMClassifier(
    n_estimators=300,
    learning_rate=0.05,
    max_depth=-1,
    random_state=42,
    class_weight="balanced",
    verbose=-1
)
```

```
# ===== Train and predict =====
# Train the model
lgbm_model.fit(X_train, y_train)

# Predict on test set
lgbm_pred = lgbm_model.predict(X_test)
```

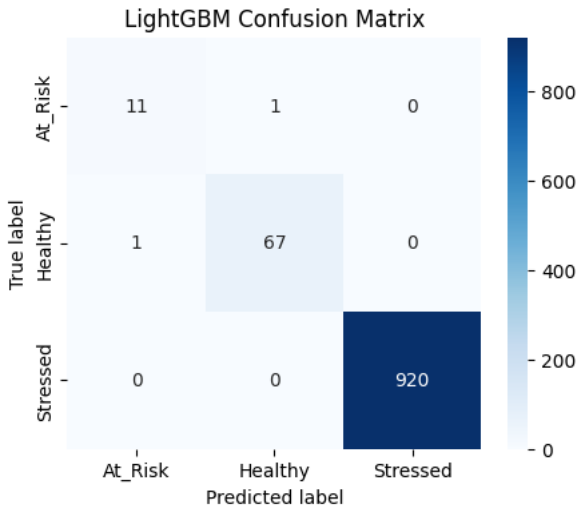
```
# ===== Evaluation metrics =====
lgbm_acc = accuracy_score(y_test, lgbm_pred)
lgbm_f1 = f1_score(y_test, lgbm_pred, average="macro")

print("LightGBM Accuracy:", round(lgbm_acc, 4))
print("LightGBM F1-macro:", round(lgbm_f1, 4))
```

```
LightGBM Accuracy: 0.998
LightGBM F1-macro: 0.9673
```

↑ High accuracy but lower macro F1 due to minority class errors ↑ # Macro F1 balances performance across all classes, not dominated by majority

```
# =====  Confusion matrix =====
cm_lgbm = confusion_matrix(y_test, lgbm_pred)

plt.figure(figsize=(5,4))
sns.heatmap(
    cm_lgbm,
    annot=True,
    fmt="d",
    cmap="Blues",
    xticklabels=le_target.classes_,
    yticklabels=le_target.classes_
)
```

```
plt.title("LightGBM Confusion Matrix")
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.show()
```



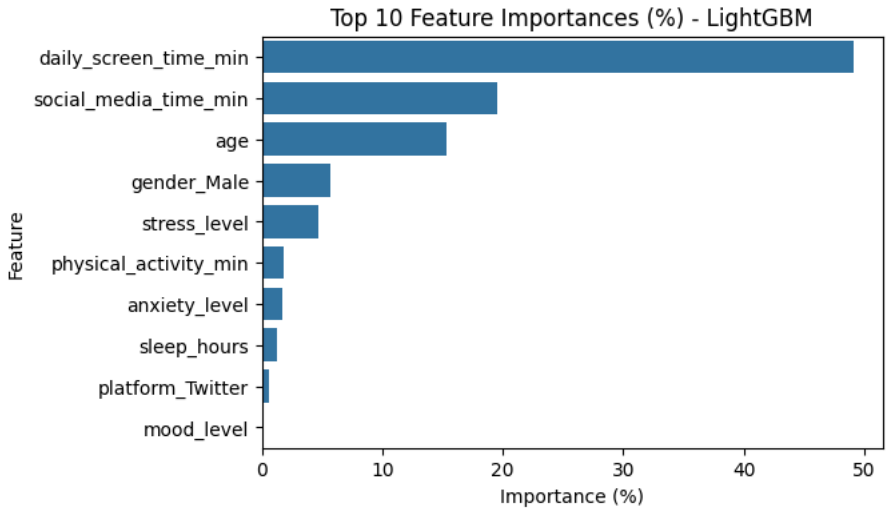

↑ # Confusion matrix shows how predictions compare against true labels

↑ Most errors occur between At_Risk and Healthy classes

↑ Majority class Stressed is classified almost perfectly

↑ Minority class mistakes strongly affect macro F1 score

```
# ===== Feature importance (LightGBM) =====

# Feature importance converted to percentages for easier interpretation
lgbm_importances = lgbm_model.feature_importances_.astype(float)
lgbm_importances = (lgbm_importances / lgbm_importances.sum()) * 100

lgbm_top10 = (
    pd.Series(lgbm_importances, index=X.columns)
      .sort_values(ascending=False)
      .head(10)
)

plt.figure(figsize=(6,4))
sns.barplot(x=lgbm_top10.values, y=lgbm_top10.index)
plt.title("Top 10 Feature Importances (%) - LightGBM")
plt.xlabel("Importance (%)")
plt.ylabel("Feature")
plt.show()
```




```
# LightGBM emphasizes screen time variables more strongly than Random Forest
# Daily and social media usage dominate boosted tree decisions
# Emotional variables play a secondary role in this model
# Different importance patterns reflect model-specific learning behavior
```

```
!pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.2.8-cp312-cp312-manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: graphviz in /usr/local/lib/python3.12/dist-packages (from catboost) (0.21)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from catboost) (3.10.0)
Requirement already satisfied: numpy<3.0,>=1.16.0 in /usr/local/lib/python3.12/dist-packages (from catboost) (2.0.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.12/dist-packages (from catboost) (2.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages (from catboost) (1.16.3)
Requirement already satisfied: plotly in /usr/local/lib/python3.12/dist-packages (from catboost) (5.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.12/dist-packages (from catboost) (1.17.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (2025.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (3.2.5)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.12/dist-packages (from plotly->catboost) (9.1.2)
Downloading catboost-1.2.8-cp312-cp312-manylinux2014_x86_64.whl (99.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.2/99.2 MB 8.2 MB/s eta 0:00:00
Installing collected packages: catboost
Successfully installed catboost-1.2.8
```

```
# ==========================================
# ===== Third Model CatBoost  model =====
# ==========================================

from catboost import CatBoostClassifier

# Initialize CatBoost classifier with manual class weights
cat_model = CatBoostClassifier(
    iterations=500,
    learning_rate=0.05,
    depth=6,
    loss_function="MultiClass",
    class_weights=[10, 2, 1],
    random_seed=42,
    verbose=False
)
```

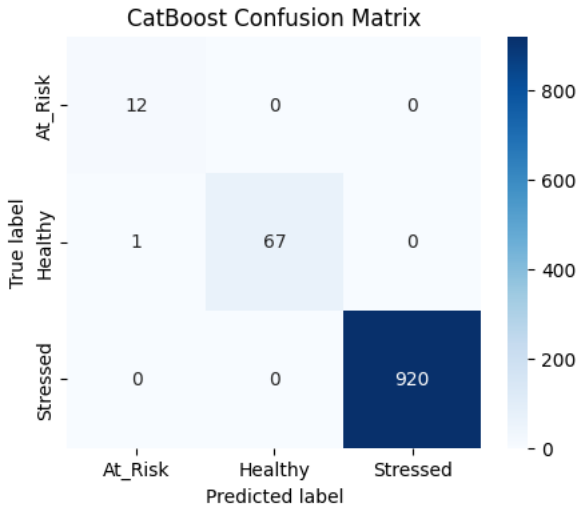
```
# ===== Train and predict =====
cat_model.fit(X_train, y_train)

cat_pred = cat_model.predict(X_test).ravel()
```

```
# ===== Evaluation metrics =====
cat_acc = accuracy_score(y_test, cat_pred)
cat_f1 = f1_score(y_test, cat_pred, average="macro")

print("CatBoost Accuracy:", round(cat_acc, 4))
print("CatBoost F1-macro:", round(cat_f1, 4))
```

```
CatBoost Accuracy: 0.999
CatBoost F1-macro: 0.9842
```

```
# CatBoost balances accuracy and macro F1 better than previous models
```

```
# =====Confusion matrix =====
cm_cat = confusion_matrix(y_test, cat_pred)

plt.figure(figsize=(5,4))
sns.heatmap(
    cm_cat,
    annot=True,
    fmt="d",
    cmap="Blues",
    xticklabels=le_target.classes_,
    yticklabels=le_target.classes_
)

plt.title("CatBoost Confusion Matrix")
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.show()
```




CatBoost shows strong classification performance across all mental state classes

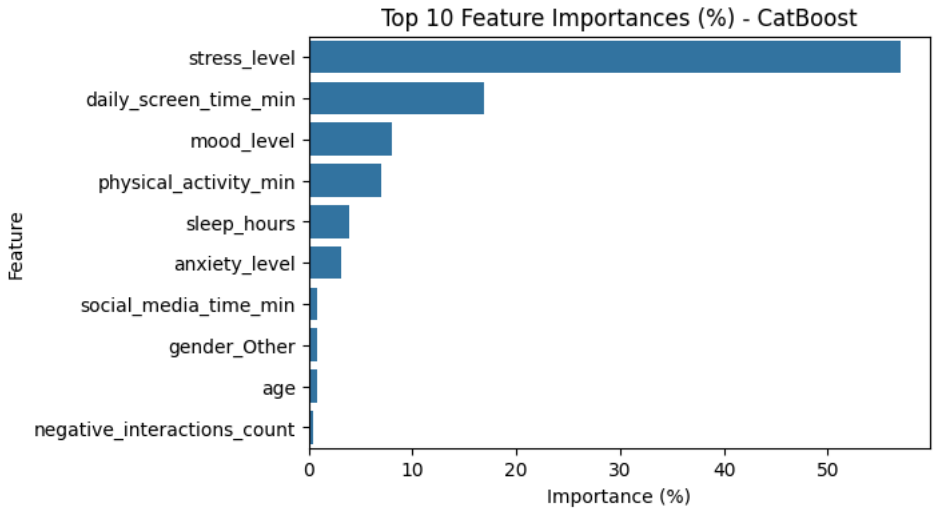
Minority class At_Risk is fully detected without misclassification

Small confusion appears between Healthy and At_Risk, expected due to similarity

High accuracy is supported by balanced performance across classes

```
# ===== Feature importance (percentages) =====

# Convert CatBoost importances to percentages for interpretation
cat_importances = cat_model.get_feature_importance()
cat_importances = (cat_importances / cat_importances.sum()) * 100

cat_top10 = (
    pd.Series(cat_importances, index=X.columns)
      .sort_values(ascending=False)
      .head(10)
)

plt.figure(figsize=(6,4))
sns.barplot(x=cat_top10.values, y=cat_top10.index)
```

```
plt.title("Top 10 Feature Importances (%) - CatBoost")
plt.xlabel("Importance (%)")
plt.ylabel("Feature")
plt.show()
```




Stress level dominates predictions, strongly defining mental state categories

Screen time remains a key behavioral signal influencing model decisions

Emotional and behavioral features jointly explain mental health outcomes

CatBoost relies on fewer, stronger features compared to other models

```
# ========================================
# ===== Final model comparison table =====
# ========================================

results_df = pd.DataFrame({
    "Model": ["Random Forest", "LightGBM", "CatBoost"],
    "Accuracy": [rf_acc, lgbm_acc, cat_acc],
    "F1_macro": [rf_f1, lgbm_f1, cat_f1]
})

results_df
```

| | Model | Accuracy | F1_macro |
|---|---|---|---|
| **0** | Random Forest | 1.000 | 1.000000 |
| **1** | LightGBM | 0.998 | 0.967320 |
| **2** | CatBoost | 0.999 | 0.984198 |

Random Forest achieves perfect scores, indicating possible overfitting to this dataset

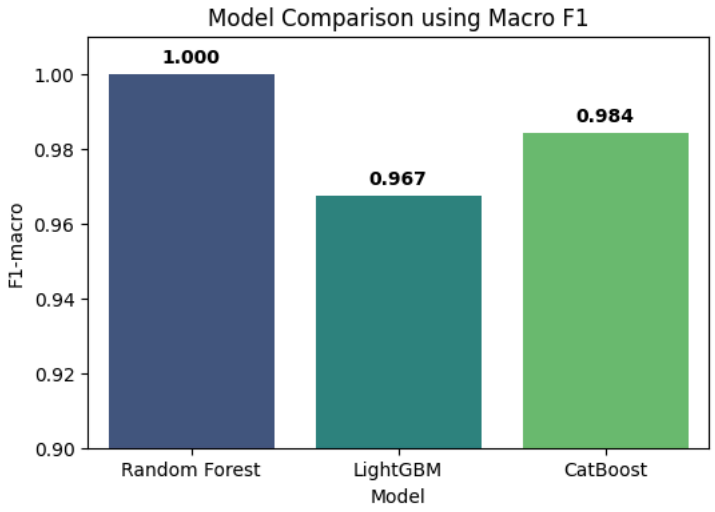
LightGBM shows lower macro F1, mainly affected by minority class performance

CatBoost maintains high accuracy with improved balance across classes

```
# =====  Model comparison plot =====
plt.figure(figsize=(6,4))
sns.barplot(
    data=results_df,
    x="Model",
    y="F1_macro",
    palette="viridis"
)

plt.title("Model Comparison using Macro F1")
plt.ylabel("F1-macro")
plt.ylim(0.9, 1.01)
```

```
for i, v in enumerate(results_df["F1_macro"]):
    plt.text(i, v + 0.003, f"{v:.3f}", ha="center", fontweight="bold")

plt.show()
```

```
/tmp/ipython-input-4139687771.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False`

  sns.barplot(
```




Macro F1 comparison highlights differences beyond raw accuracy values

CatBoost outperforms LightGBM in handling minority mental state classes

Random Forest dominance suggests strong feature separability in the dataset

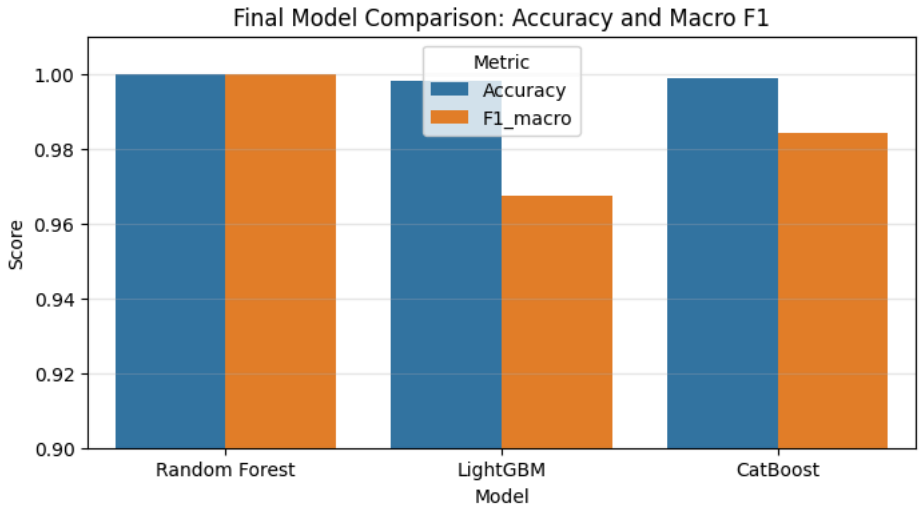
```
# =====  Final model selection =====
best_model_name = results_df.sort_values("F1_macro", ascending=False).iloc[0]["Model"]
best_model_name
```

```
'Random Forest'
```

```
# ===== Build plot_df for grouped bar chart =====
plot_df = results_df.melt(id_vars="Model", var_name="Metric", value_name="Score")
plot_df.head()
```

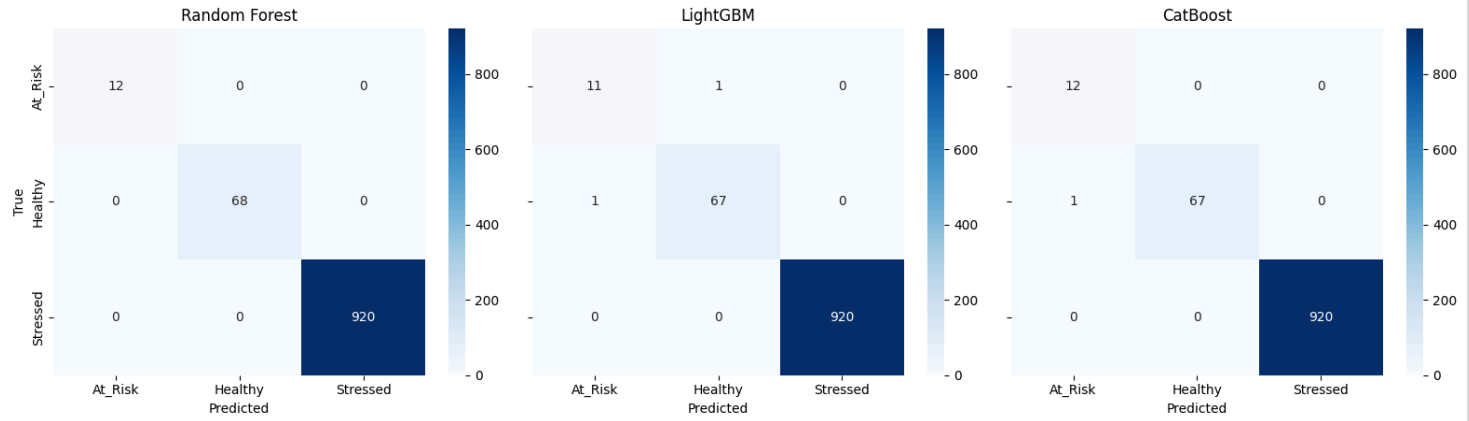
| | Model | Metric | Score |
|---|---|---|---|
| **0** | Random Forest | Accuracy | 1.00000 |
| **1** | LightGBM | Accuracy | 0.99800 |
| **2** | CatBoost | Accuracy | 0.99900 |
| **3** | Random Forest | F1_macro | 1.00000 |
| **4** | LightGBM | F1_macro | 0.96732 |

```
# Plot grouped bar chart for final model comparison
plt.figure(figsize=(8,4))
sns.barplot(data=plot_df, x="Model", y="Score", hue="Metric")
plt.title("Final Model Comparison: Accuracy and Macro F1")
plt.ylim(0.9, 1.01)
plt.ylabel("Score")
plt.grid(axis="y", alpha=0.3)
plt.show()
```

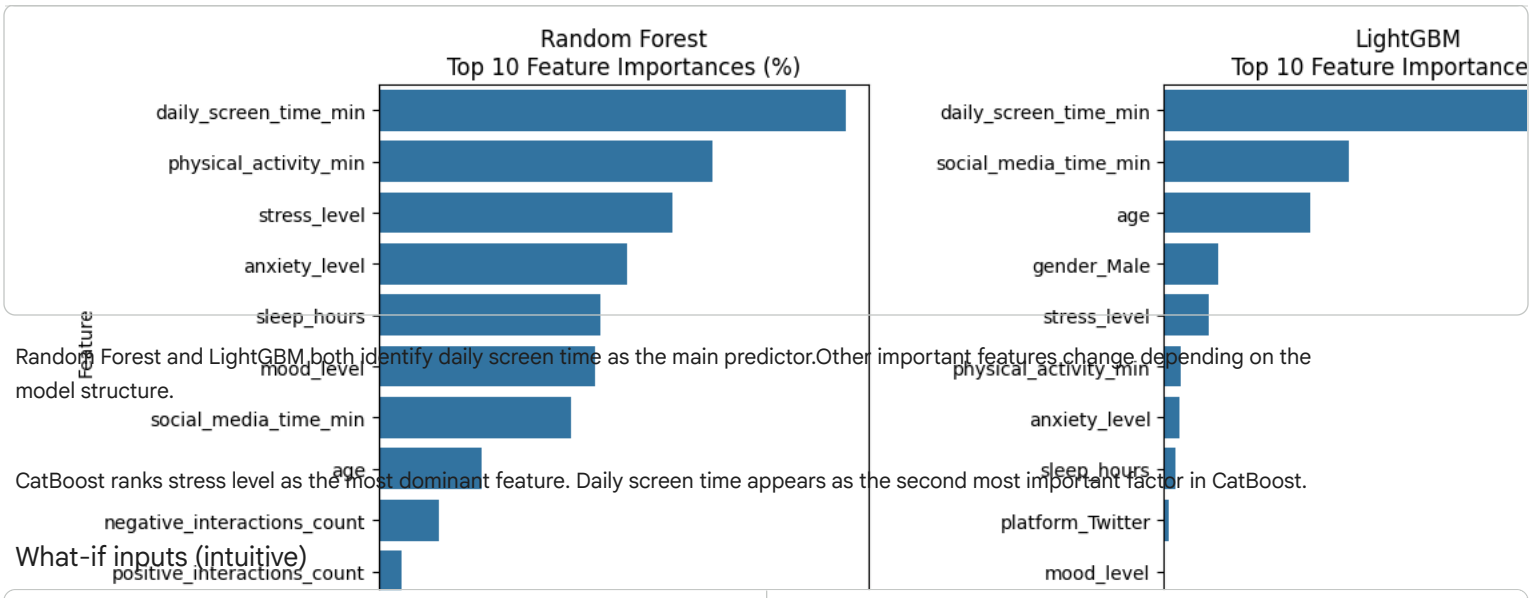



Double-click (or enter) to edit

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Compute confusion matrices
cm_rf = confusion_matrix(y_test, rf_model.predict(X_test))
cm_lgbm = confusion_matrix(y_test, lgbm_model.predict(X_test))
cm_cat = confusion_matrix(y_test, cat_model.predict(X_test))

labels = le_target.classes_

# Plot aligned heatmaps
fig, axes = plt.subplots(1, 3, figsize=(16, 5), sharey=True)

sns.heatmap(
    cm_rf, annot=True, fmt="d", cmap="Blues",
    xticklabels=labels, yticklabels=labels, ax=axes[0]
)
axes[0].set_title("Random Forest")
axes[0].set_xlabel("Predicted")
axes[0].set_ylabel("True")

sns.heatmap(
    cm_lgbm, annot=True, fmt="d", cmap="Blues",
    xticklabels=labels, yticklabels=labels, ax=axes[1]
)
axes[1].set_title("LightGBM")
axes[1].set_xlabel("Predicted")

sns.heatmap(
    cm_cat, annot=True, fmt="d", cmap="Blues",
    xticklabels=labels, yticklabels=labels, ax=axes[2]
)
axes[2].set_title("CatBoost")
axes[2].set_xlabel("Predicted")

plt.suptitle("Confusion Matrices Comparison Across Models", fontsize=14)
plt.tight_layout()
plt.show()
```




```
# ===== Top 10 Feature Importances (%) - All Models (normalized) =====

def top10_percent(series, top_n=10):
    s = series.astype(float)
    s = (s / s.sum()) * 100
    return s.sort_values(ascending=False).head(top_n)

# Random Forest
rf_series = pd.Series(rf_model.feature_importances_, index=X.columns)
rf_top10 = top10_percent(rf_series)

# LightGBM (needs normalization!)
lgbm_series = pd.Series(lgbm_model.feature_importances_, index=X.columns)
lgbm_top10 = top10_percent(lgbm_series)

# CatBoost (needs normalization!)
cat_series = pd.Series(cat_model.get_feature_importance(), index=X.columns)
cat_top10 = top10_percent(cat_series)

fig, axes = plt.subplots(1, 3, figsize=(18, 5))

sns.barplot(x=rf_top10.values, y=rf_top10.index, ax=axes[0])
axes[0].set_title("Random Forest\nTop 10 Feature Importances (%)")
axes[0].set_xlabel("Importance (%)")
axes[0].set_ylabel("Feature")

sns.barplot(x=lgbm_top10.values, y=lgbm_top10.index, ax=axes[1])
axes[1].set_title("LightGBM\nTop 10 Feature Importances (%)")
axes[1].set_xlabel("Importance (%)")
axes[1].set_ylabel("")

sns.barplot(x=cat_top10.values, y=cat_top10.index, ax=axes[2])
axes[2].set_title("CatBoost\nTop 10 Feature Importances (%)")
axes[2].set_xlabel("Importance (%)")
axes[2].set_ylabel("")

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




Random Forest and LightGBM both identify daily screen time as the main predictor.Other important features change depending on the model structure.

CatBoost ranks stress level as the most dominant feature. Daily screen time appears as the second most important factor in CatBoost.

## What-if inputs (intuitive)

```
## =====================================================
# What-If Simulator Instructions
# -----------------------------------------------------
# Fill realistic daily values using hours + minutes fields
# Minutes fields must be between 0 and 59
# Click "Run" to see predicted mental state and probabilities
# =====================================================

#@title What-if inputs (intuitive)
age = 20  #@param {type:"integer",min:18,max:70,step:1}

screen_hours = 10  #@param {type:"integer",min:0,max:12,step:1}
screen_minutes = 2  #@param {type:"integer",min:0,max:59,step:1}

social_hours = 4  #@param {type:"integer",min:0,max:10,step:1}
social_minutes = 50  #@param {type:"integer",min:0,max:59,step:1}

sleep_hours = 12  #@param {type:"number",min:0,max:12,step:0.5}

activity_hours = 4  #@param {type:"integer",min:0,max:3,step:1}
activity_minutes = 20  #@param {type:"integer",min:0,max:59,step:1}

stress_1_to_10 = 5  #@param {type:"integer",min:1,max:10,step:1}
anxiety_1_to_4 = 4  #@param {type:"integer",min:1,max:4,step:1}
mood_1_to_10 = 3  #@param {type:"integer",min:1,max:10,step:1}

negative_interactions = 2  #@param {type:"integer",min:0,max:5,step:1}
positive_interactions = 1  #@param {type:"integer",min:0,max:10,step:1}

gender = "Female"  #@param ["Male","Female","Other"]
platform = "Twitter"  #@param ["Instagram","Snapchat","Facebook","TikTok","WhatsApp","Twitter"]


# ===== Input validation =====
# Minutes fields should be 0-59
if screen_minutes > 59 or social_minutes > 59 or activity_minutes > 59:
    print("Warning: minutes fields must be between 0 and 59.")

# Convert to minutes (matches training units)
daily_screen_time_min = (screen_hours * 60) + screen_minutes
social_media_time_min = (social_hours * 60) + social_minutes
physical_activity_min = (activity_hours * 60) + activity_minutes

# Print converted values to confirm inputs
print("Screen minutes:", daily_screen_time_min)
print("Social minutes:", social_media_time_min)
print("Activity minutes:", physical_activity_min)

# Sanity check: social time should not exceed total screen time
if social_media_time_min > daily_screen_time_min:
    print("Warning: Social media time exceeds total screen time.")
```

age: 20
screen_hours: 10
screen_minutes: 2
social_hours: 4
social_minutes: 50
sleep_hours: 12
activity_hours: 4
activity_minutes: 20
stress_1_to_10: 5
anxiety_1_to_4: 4
mood_1_to_10: 3
negative_interactions: 2
positive_interactions: 1
gender: Female
platform: Twitter

Hide code