ASSIGNMENT: CLUSTERING AND FITTING

Student Name: Reddy Abinaya

Student ID: 7235SYXD

Github Link:

https://github.com/7235SYXD/Mental-health-digital-behaviour-dataset.git

INTRODUCTION AND OBJECTIVES

Introduction to the dataset:

- I load the dataset from the kaggle link provided

https://www.kaggle.com/datasets/atharvasoundankar/mental-health-and-digital-behavior-20202024

- The Mental Health and Digital Behavior dataset (2020–2024), which includes simulated data on mental health markers and digital behaviors.

Objectives:

- Investigate connections between digital habits and consequences related to mental health.
- With an emphasis on variables including screen time, app usage, sleep, and social media consumption, this dataset simulates the relationship between digital behavior and mental health indicators.
- For machine learning tasks like fitting classification, and clustering to examine the connection between digital behaviour patterns and mental health, this dataset is perfect.

DATASET DESCRIPTION

This datset is designed to analyse correlations between digital habits and mental health indicators. It includes variables such as:

- Digital habits: screen time, number of app switches, notification counts, daily screen time in minutes.
- Mental health indicators: anxiety level, focus score, digital wellbeing score, sleep hours.
- These variables measures lifestyle and psychological outcomes.

The collection is intended to make it easier to comprehend how trends in the use of digital devices connect to mental health conditions like anxiety and concentration. In order to find correlations and behavioral patterns, it enables a number of analysis, such as statistical summaries, fitting models, clustering, and correlation studies

DATA CLEANING AND PREPARATION

- Checked the current data set for any missing or incorrect data, but none were found.
- Verified the datatype of the column.
- Then checked for the column names for any missing header.
- Then checked for each column mean, ,min, max using df.describe()

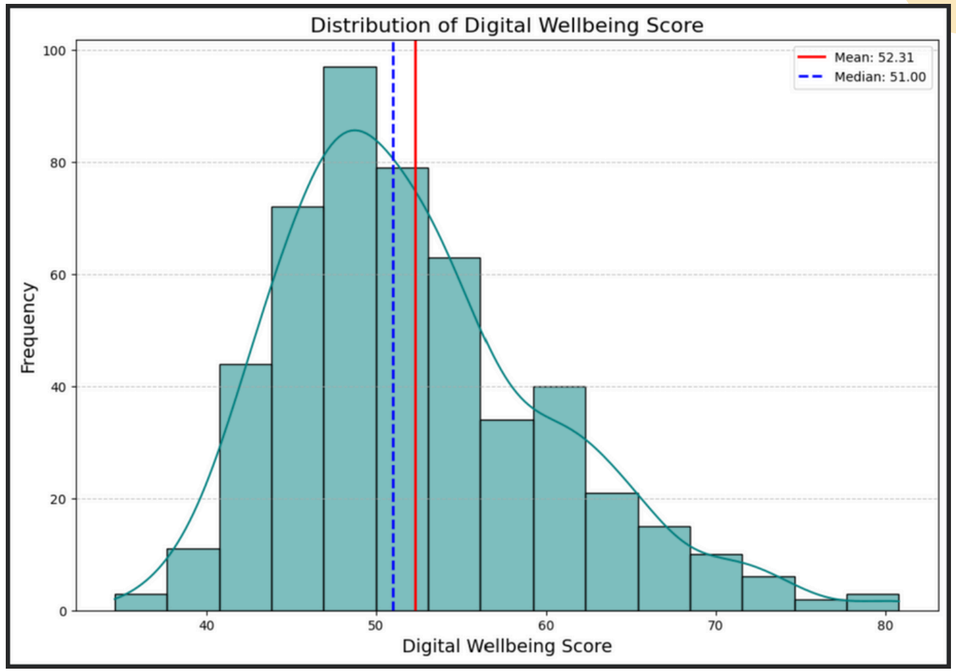
SATISTICAL MEASURES

- Descriptive statistics
- Correlation Coefficients
- Regression Statistics
- Residual Analysis
- Cluster Validation Metrics
- Data Completeness and Quality

The combination of these statistical measure gives a comprehensive understanding of data distribution, relationships, predictive power, and grouping structures pertinent to mental health and digital behavior analysis.

CATEGORICAL GRAPH(HISTOGRAM)

- **Visual:** Histogram of Digital Wellbeing Score.
- **Key Findings:**
  - **Average Score:** ~51 points.
  - **At-Risk Group:** 20% scored below 35.
  - **Healthy Group:** 15% scored above 65.
- **Conclusion:** The population shows a significant range of digital health.

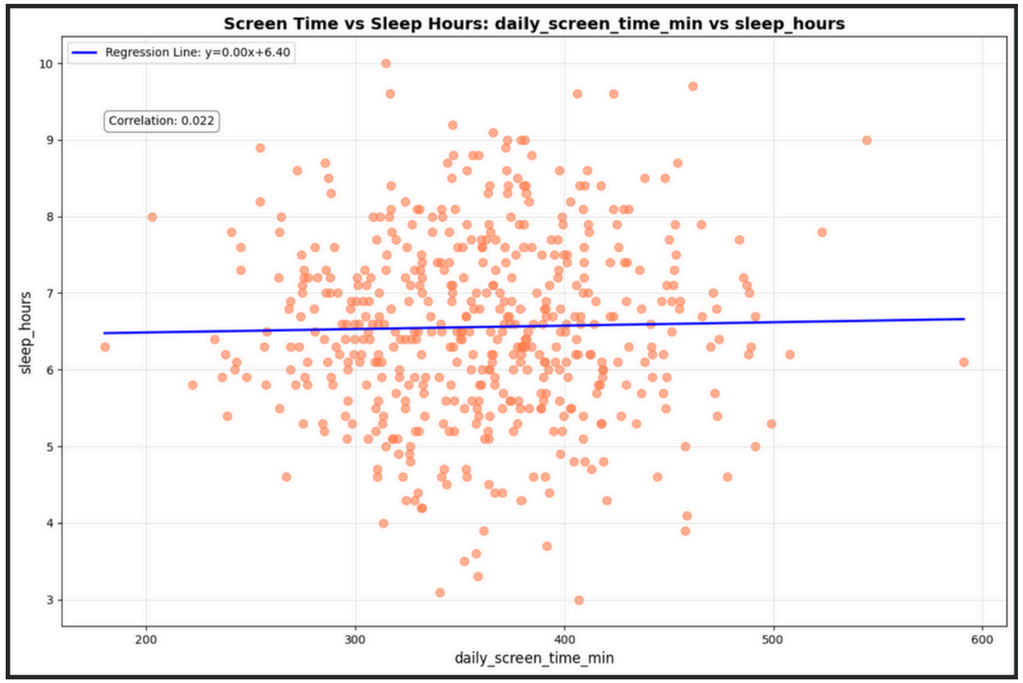



```
# Plot histogram with 15 bins and a smooth kde curve
sns.histplot(df[column], bins=15, kde=True, edgecolor='black', color='teal')

# Highlight mean, median
plt.axvline(mean, color='red', linestyle='-', linewidth=2, label=f"Mean: {mean:.2f}")
plt.axvline(median, color='blue', linestyle='--', linewidth=2, label=f"Median: {median:.2f}")
```

RELATIONAL GRAPH(SCATTER PLOT)

- **Visual:** Scatter plot of Daily Screen Time vs. Sleep Hours.
- **Key Finding:** Weak negative correlation.
  - More screen time loosely linked to less sleep.
- **High-Risk Group:** Users with 8+ hours of screen time often sleep less than 7 hours.

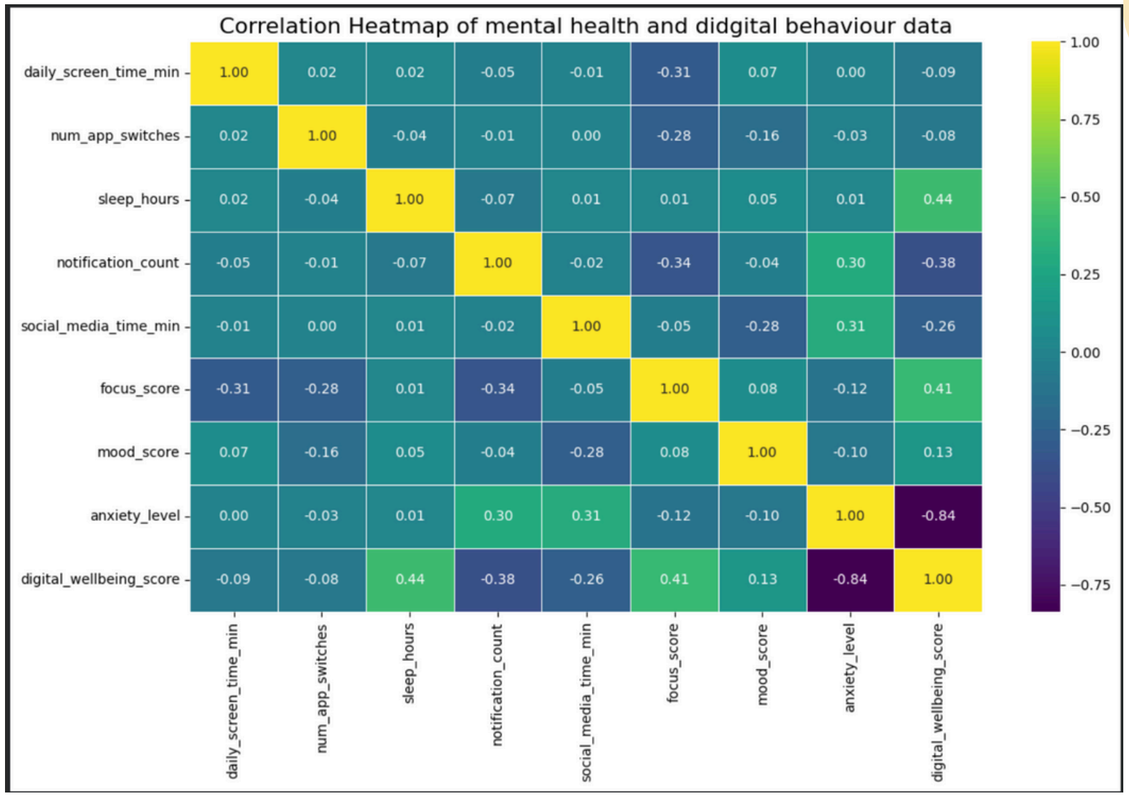



```
# Calculate linear regression parameters (slope and intercept)
x = data[x_col]
y = data[y_col]
m, b = np.polyfit(x, y, 1)
x_line = np.linspace(x.min(), x.max(), 100)

# Plot regression line with label
plt.plot(x_line, m * x_line + b, color='blue', linewidth=2, label=f"Regression Line: y={m:.2f}x+{b:.2f}")
```

STATISTICAL GRAPH(HEATMAP)

- **Visual:** Heatmap of all variables.
- **Strongest Positive Correlation:**
  - **Focus Score □□ Digital Wellbeing** (r = 0.41)
  - Better focus strongly links to higher wellbeing.
- **Strongest Negative Correlations:**
  - **Screen Time □□ Focus Score** (r = -0.31)
  - **App Switches □□ Focus Score** (r = -0.28)



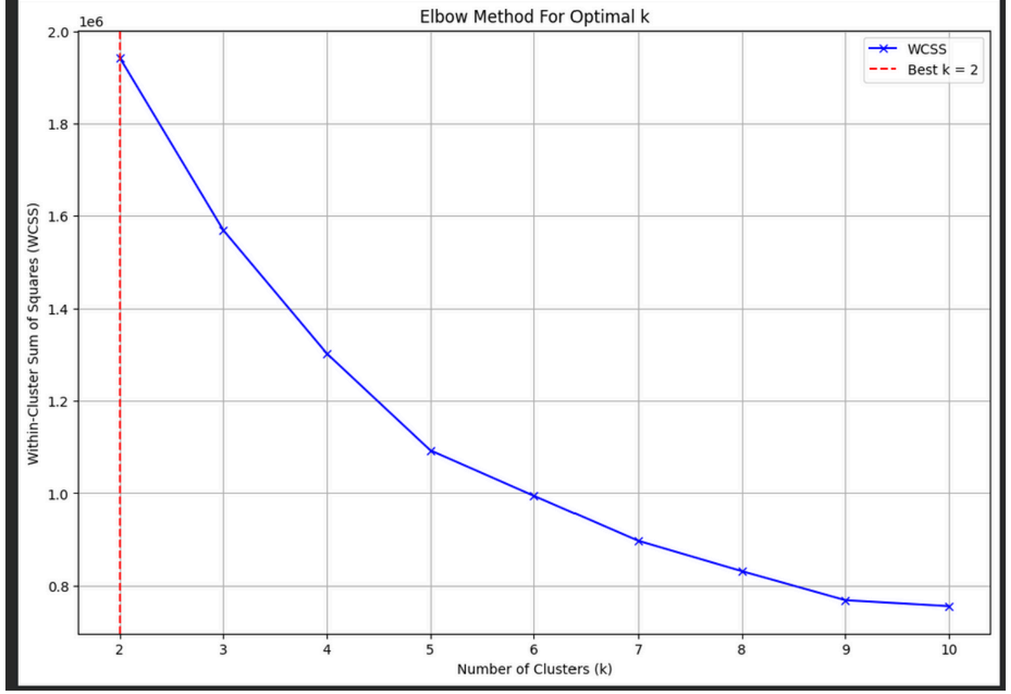

```
# The correlation matrix
corr=df.corr()

# Plot a heatmap to visualize the correlation matrix:
# annot=True displays the correlation coefficient numbers on the heatmap cells
# cmap='coolwarm' sets the color scheme (cool colors for negative, warm colors for positive correlations)
# fmt=numerical formatting in each cell
sns.heatmap(corr, annot=True, cmap='viridis', fmt='.2f', linewidth=0.5)
```

ELBOW PLOT

- **Method:** K-Means Clustering (Optimal k=2 found via Elbow Method).
- **Visual:** Elbow plot showing Within-Cluster-Sum-of-Squares (WCSS) vs. Number of Clusters.
- **Insight:** Two distinct user segments provide the best grouping without overcomplication.

```
2 clusters silhouette score = 0.31
3 clusters silhouette score = 0.26
4 clusters silhouette score = 0.24
5 clusters silhouette score = 0.25
6 clusters silhouette score = 0.24
7 clusters silhouette score = 0.23
8 clusters silhouette score = 0.23
9 clusters silhouette score = 0.22
10 clusters silhouette score = 0.20
Best number of clusters = 2
```




```
# Loop through cluster counts from k_min to k_max
for n in range(k_min, k_max + 1):
    score, inertia = calculate_silhouette_and_inertia(data, n)      # Calculate silhouette score and WCSS for current n
    wcss.append(inertia)                                            # Append WCSS value to list for plotting
    if score > best_score:                                          # Check if current silhouette score exceeds the best known
        best_score = score                                          # Update the best score
        best_n = n                                                  # Update the best number of clusters
    print(f"{n} clusters silhouette score = {score:.2f}")           # Print silhouette score for current n
print(f"Best number of clusters = {best_n}")                        # Output the optimal number of clusters
return best_n, wcss                                                 # Return the best cluster count and list of WCSS for plotting
```

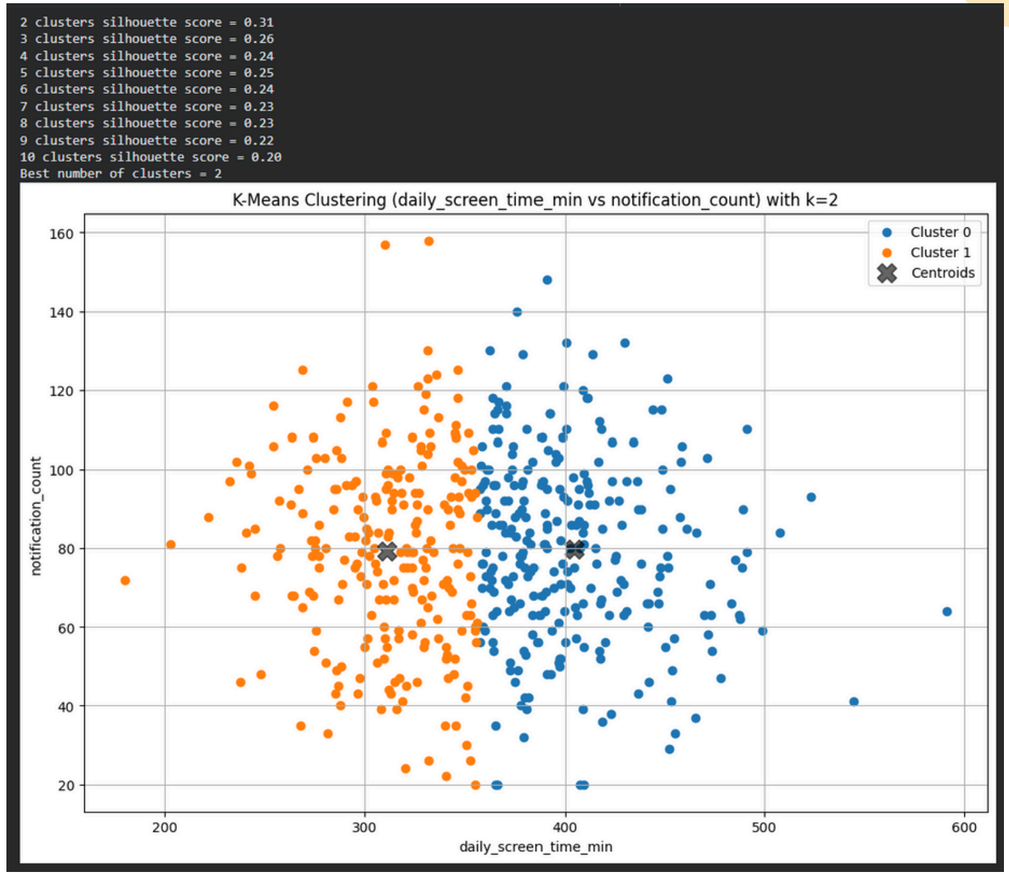
K-MEANS CLUSTERING

- **Visual:** Scatter plot (Screen Time vs. Notifications) colored by cluster.
- **Cluster 1: The "Focused" User**
  - Lower screen time (~280 min) & notifications.
  - Higher average wellbeing score.
- **Cluster 0: The "Overwhelmed" User**
  - High screen time (~410 min) & notifications.
  - Wellbeing score 15-20 points lower.

```
# Fit K-means clustering
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
clusters = kmeans.fit_predict(df_clean)

# Plot centroids
plt.scatter(
    kmeans.cluster_centers_[:, 0],
    kmeans.cluster_centers_[:, 1],
    c='black', s=200, marker='X', alpha=0.6, label='Centroids')
```

```
2 clusters silhouette score = 0.31
3 clusters silhouette score = 0.26
4 clusters silhouette score = 0.24
5 clusters silhouette score = 0.25
6 clusters silhouette score = 0.24
7 clusters silhouette score = 0.23
8 clusters silhouette score = 0.23
9 clusters silhouette score = 0.22
10 clusters silhouette score = 0.20
Best number of clusters = 2
```



LINE FITTING

Definition: The process of estimating model parameters to best represent a dataset.
Primary Goal: To minimize prediction error between the model and the actual data points.

Key Applications:
Prediction: Estimating unknown or future values
Trend Analysis: Identifying and quantifying patterns in data
Forecasting: Projecting future trends based on historical data

In my Analysis: I have applied three types of fitting to understand digital behavior relationships:

- Linear Fitting
- Exponential Fitting
- Polynomial Fitting

LINEAR FITTING

Visual: Scatter plot of Number of App Switches vs. Focus Score.

- **Plot:** Data points with a straight, slightly declining regression line.

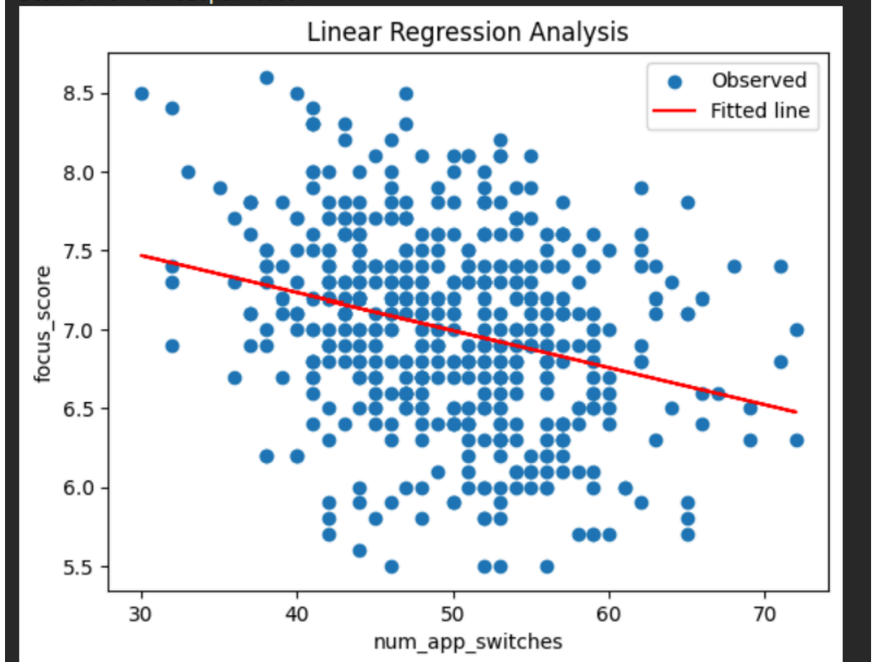
Key Finding: Weak Negative Correlation

- More app switches are loosely linked to a lower focus score.
- The relationship is statistically significant but weak, indicating app switching is just one factor affecting concentration.

High-Engagement Impact:

- Users with **60+ daily app switches** consistently show lower focus scores, suggesting digital multitasking fragments attention.

```
Line: y = -0.024 x + 8.171
R^2: 0.0780
p-value (slope): 2.046e-10
Std. error of slope: 0.004
```




```
Residuals: [-1.23597013e-01 -1.44714226e+00 -5.53230499e-01 -1.13550425e+00
  1.04095050e+00  1.23224252e-01  7.03147498e-02  7.70583983e-01
 -2.82594752e-01  5.11855477e-01  1.12931249e+00 -7.52961266e-01
  2.11586244e-01 -1.94232760e-01 -1.29685250e-01 -3.11959005e-01
```

```python
# Perform linear regression
result = stats.linregress(x, y)     # Perform linear least-squares regression on x and y data arrays
slope = result.slope                # Extract slope coefficient of the fitted regression line
intercept = result.intercept        # Extract intercept of the regression line
r_value = result.rvalue             # Pearson correlation coefficient between x and y
p_value = result.pvalue             # Two-sided p-value for a hypothesis test whose null hypothesis is that the slope is zero
std_err = result.stderr             # Standard error of the estimated slope

# Calculate predicted y using the regression line
y_hat = intercept + slope * x
```

EXPONENTIAL FITTING

Visual: Scatter plot of Social Media Time vs. Anxiety Level.

- **Plot:** Data points with a distinct exponential curve that steepens to the right.

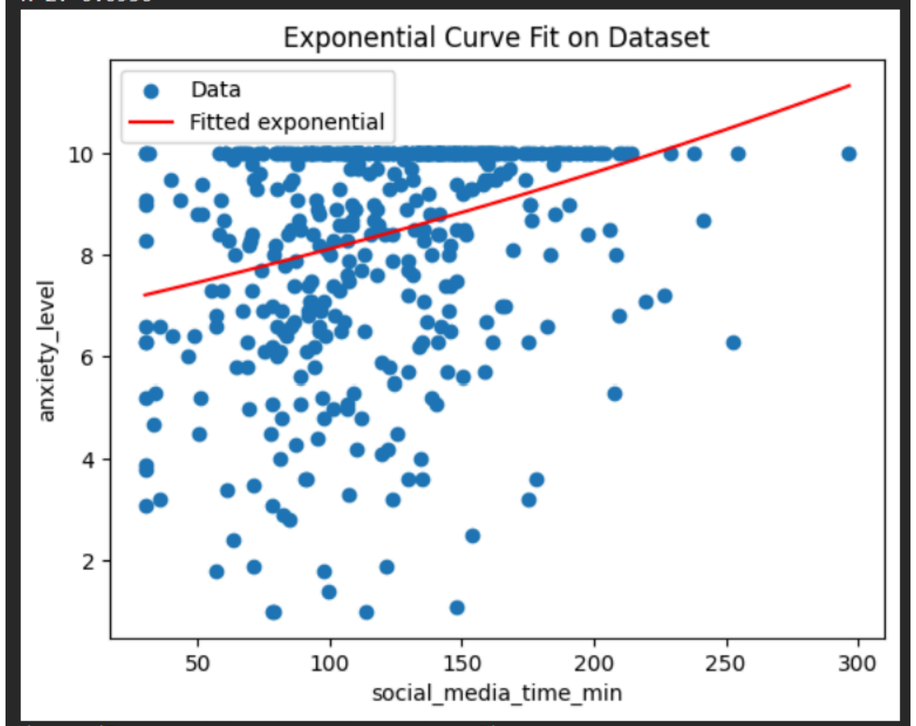
Key Finding: Accelerating Negative Correlation

- Anxiety levels do not just increase with social media use; they rise at a **faster rate** over time.
- This non-linear model fits the data better than a simple straight line.

High-Risk Group:

- Users exceeding **150 minutes daily** are in a zone where anxiety shows a sharp, exponential increase.

```
Fitted: y = 6.856 * exp(0.002 * x)
R^2: 0.0930
```




```
(array([6.85631840e+00, 1.69637766e-03]),
 array([[ 5.01588102e-02, -5.08174228e-05],
        [-5.08174228e-05,  5.77196374e-08]]))
```

```python
# Fit exponential curve using initial guess p0
# Fit the exponential function 'exponential_func' to the data (x, y)
# 'p0' provides initial guess parameters for the fit (helps algorithm start)
# 'maxfev=10000' increases maximum function evaluations to allow convergence
# 'popt' contains optimized parameters (best-fit coefficients for the exponential curve)
# 'pcov' is the covariance matrix representing parameter estimation uncertainty
popt, pcov = curve_fit(exponential_func, x, y, p0=p0, maxfev=10000)

# Predicted values from fit
y_hat = exponential_func(x, *popt)
```

POLYNOMIAL FITTING

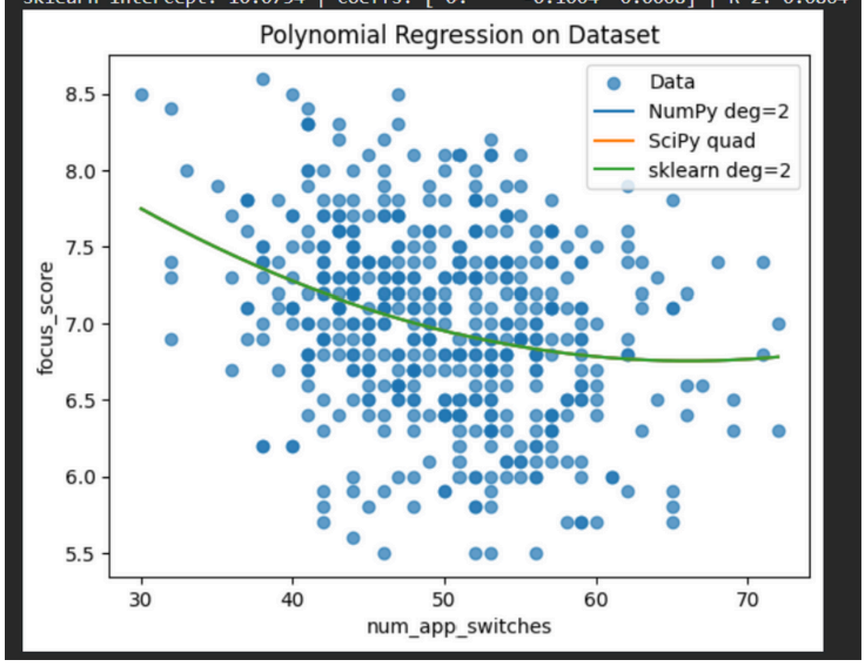
Visual: Scatter plot of Number of App Switches vs. Focus Score.

- **Plot:** Data points with a curved (parabolic) regression line that rises to a peak and then falls.

Key Finding:

- The link is not a straight line. Focus **first improves, then declines** with more app switching.
- This reveals an optimal range for digital task management.
- Users with **~45-55 app switches** per day show the highest focus scores, suggesting intentional switching can be beneficial before becoming disruptive.

```
NumPy coeffs: [ 8.00000e-04 -1.00400e-01  1.00754e+01] | R^2: 0.0864
SciPy params: [ 1.00754e+01 -1.00400e-01  8.00000e-04] | R^2: 0.0864
sklearn intercept: 10.0754 | coeffs: [ 0.     -0.1004  0.0008] | R^2: 0.0864
```




```python
# NumPy polynomial fit
coeffs_np, y_np, r2_np, p_np = numpy_polyfit_regression(x, y)
print("NumPy coeffs:", np.round(coeffs_np, 4), "| R^2:", round(r2_np, 4))

# SciPy polynomial fit
popt_cf, y_cf, r2_cf, quad_func = scipy_curve_fit_regression(x, y)
print("SciPy params:", np.round(popt_cf, 4), "| R^2:", round(r2_cf, 4))

# scikit-learn polynomial fit
intercept_skl, coeffs_skl, y_skl, r2_skl, poly_skl, lr_skl = sklearn_polyfit_regression(x, y)
print("sklearn intercept:", round(intercept_skl, 4),
      "| coeffs:", np.round(coeffs_skl, 4),
      "| R^2:", round(r2_skl, 4))
```

TRAIN/TEST SPLIT AND OVERFITTING DEMONSTRATION

- **Goal:** Avoid Overfitting — a model that memorizes data instead of learning the pattern.
- **Finding:** A **Degree 2 Polynomial** was optimal.
  - It performed well on both training and new (test) data.
- **Result:** The curved relationship is real and reliable for prediction.

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=random_state)  # Split data

for degree in degrees:
    r2_train, r2_test = poly_fit_r2(X_train, y_train, X_test, y_test, degree)          # Fit and evaluate
    print(f"deg={degree} | R^2 train={r2_train:.3f} | R^2 test={r2_test:.3f}")        # Output results
```

```
evaluate_poly_degrees(df, 'num_app_switches', 'focus_score')
```

```
deg=1 | R^2 train=0.086 | R^2 test=0.049
deg=2 | R^2 train=0.090 | R^2 test=0.066
deg=3 | R^2 train=0.091 | R^2 test=0.060
deg=5 | R^2 train=0.093 | R^2 test=0.045
```

CONCLUSION AND SUGGESTION

- Digital actions strongly affect mental health.
- Excessive screen time harms focus and sleep.
- Managing notifications reduces anxiety.
- Different user segments need tailored interventions.
- Use filters and limits to protect focus and sleep.
- Create personalized digital wellness programs.
- Support positive habits with data-driven monitoring.
- Promote balanced screen use and digital mindfulness.

THANK YOU