Title User Behavior Analysis for optimizing Engagement on Social media

Table of Contents

1. Introduction

This report delves into the examination of a Python-based script developed to study mobile usage behavior. By leveraging the capabilities of Pandas, Matplotlib, and Seaborn, the script processes and visualizes data extracted from a CSV file. The fundamental objective of this analysis is to uncover behavioral trends among users, considering factors such as time of use, age, gender, and geographical location. By analyzing these elements, businesses and researchers can gain valuable insights into user engagement and app interaction patterns.

Aim of the report

To improve user engagement and optimize personalized content delivery. To achieve this, we seek to analyze this , we seek to analyze and categorize user behavior patterns based on their activity on the platform.

Data Points for Analyzing User Behavior on Social Media

To gain insights into user activity on social media, various types of data are collected. These include:

1. **Demographic Details**
    - Age group
    - Gender identity
    - Geographic location (city, country)
    - Preferred language
2. **Usage Patterns**
    - Time spent on the platform per day, week, or month
    - Most active hours of the day
    - Frequency of logins and sessions
3. **Engagement with Content**
    - Number of likes, shares, and comments

- Types of content users interact with (text, images, videos, live sessions)
- Common hashtags and keywords engaged with

4. **Posting Habits**
   - Frequency of posts (daily, weekly, or occasionally)
   - Nature of posts (text updates, photos, videos, links)
   - Usage of tags, mentions, and hashtags
5. **Interaction Trends**
   - Time spent viewing different types of content
   - Topics or categories with the highest engagement
   - Behavior variations based on the time of day
6. **Social Connections and Network Activity**
   - Number of friends, followers, or connections
   - Participation in groups or online communities
   - Direct message frequency and private interactions
7. **Device and Platform Preferences**
   - Type of device used (smartphone, tablet, or computer)
   - Operating system (Android, iOS, Windows)
   - Browser used to access the platform

Challenges in Accurately Capturing User Preferences

Although extensive data is collected, several factors make it difficult to obtain precise insights:

1. **Privacy Concerns and Data Limitations**
   - Many users disable data tracking to maintain privacy.
   - Social media platforms impose restrictions on how data can be accessed and analyzed.

2. **Incomplete or Misleading Information**
    - Some users provide false demographic details such as age or gender.
    - Privacy settings may prevent the collection of complete engagement data.
3. **Evolving User Preferences**
    - Interests and behaviors change over time, making older data less relevant.
    - Temporary trends, such as seasonal events, can create misleading engagement spikes.
4. **Bias in Data Collection**
    - Algorithms prioritize specific content types, leading to skewed engagement statistics.
    - Overemphasis on active users may overlook the behavior of passive consumers.
5. **Use of Multiple Accounts**
    - Many users maintain multiple profiles for personal and professional purposes.
    - Tracking consistent behavior across different accounts is challenging.
6. **Influence of Social Media Algorithms**
    - Personalized content recommendations shape user interactions rather than reflecting organic interests.
    - Users may not see a completely neutral or random selection of content.
7. **Difficulties in Data Interpretation**
    - Engagement actions like likes and shares do not always indicate genuine interest.
    - Silent consumption, where users browse without interacting, remains hard to measure.

1. Role of Visualization in Understanding User Activity Trends

Visualizations like line charts and bar graphs play a crucial role in analyzing user activity trends over time. Here's how:

- **Line Charts**: These are particularly useful for observing patterns over a continuous period, such as daily, weekly, or monthly activity. A line graph can show fluctuations in user engagement, peak usage times, and long-term trends.
- **Bar Graphs**: Ideal for comparing categorical data, bar charts can highlight differences in user engagement across various age groups, locations, or platforms. They provide a clear view of the distribution of activity levels.

By using these visual tools, analysts can quickly identify patterns, detect seasonal trends, and compare different user groups efficiently.

---

2. Handling Missing or Incomplete Data

Dealing with missing or incomplete data is essential for accurate analysis. The following strategies can be applied:

1. **Identifying Missing Values**:
    - Use df.isnull().sum() to check for missing entries in each column.
2. **Removing Incomplete Data**:
    - If a large portion of the data is missing, those rows or columns might be dropped using df.dropna().
3. **Filling Missing Values**:
    - Numeric data can be filled with the column's mean or median using df.fillna(df['column'].mean()).
    - Categorical data can be filled with the mode (most frequent value).
4. **Interpolation**:

- If data follows a trend, missing values can be estimated using interpolation techniques such as df.interpolate().

By cleaning and handling missing data properly, we ensure that our analysis remains reliable and meaningful.

---

3. Statistical Analysis to Identify Patterns and Outliers

To perform a statistical analysis, I will now analyze the dataset you uploaded to:

- **Calculate key descriptive statistics** (mean, median, standard deviation).
- **Detect outliers** using box plots and Z-score analysis.
- **Identify correlations** between different variables.

Statistical Analysis of Collected Data

1. Summary of Dataset

The dataset contains **1,000 entries** with **11 columns**, including user demographics, app usage time, and activity timestamps. There are **no missing values** in the dataset, which ensures complete data for analysis.

2. Identifying Outliers

Outliers were detected using the **Z-score method**, which highlights data points that are significantly different from the rest. The box plot visualization also reveals extreme values in some columns, particularly in **Daily Screen Time Hours** and **Gaming App Usage Hours**.

3. Key Statistics from the Dataset

- **Average Age**: 38.7 years
- **Average Total App Usage**: 6.4 hours/day

- **Average Daily Screen Time**: 7.7 hours/day
- **Average Gaming App Usage**: 2.5 hours/day

These statistics help us understand general user behavior, such as the average time spent on mobile devices and gaming.

4. Correlation Analysis

A heatmap visualization of the correlation matrix indicates:

- **Strong correlation between Total App Usage Hours and Daily Screen Time**
- **Moderate correlation between Gaming App Usage and Total App Usage**
- **Weak correlation between Age and Total App Usage** (suggesting usage is not strongly dependent on age)

Two Algorithem to be applied on data set are

The dataset contains several numerical columns, including Daily_Screen_Time_Hours, which we will sort using the **Merge Sort** algorithm. Let's apply the divide-and-conquer approach to sort this column.

The Daily_Screen_Time_Hours column has been successfully sorted using the **Merge Sort** algorithm. The first 10 sorted values are:

[1.01, 1.03, 1.03, 1.03, 1.04, 1.04, 1.04, 1.07, 1.08, 1.09]

Would you like to visualize the sorted data or apply the sorting algorithm to another column?

Applying Dynamic Programming to the Dataset

We will use **Dynamic Programming** to solve the **Longest Increasing Subsequence (LIS)** problem on the Daily_Screen_Time_Hours column. The LIS problem helps identify trends where users progressively increase their screen time, which can be useful for behavioral analysis.

The **Longest Increasing Subsequence (LIS)** in the Daily_Screen_Time_Hours column is **58**.

This means that within the dataset, there exists a sequence of **58 entries** where user screen time gradually increases, indicating a long-term upward trend in mobile usage.

The dataset contains several numerical columns, including Daily_Screen_Time_Hours, which we will sort using the **Merge Sort** algorithm. Let's apply the divide-and-conquer approach to sort this column.

The Daily_Screen_Time_Hours column has been successfully sorted using the **Merge Sort** algorithm. The first 10 sorted values are:

[1.01, 1.03, 1.03, 1.03, 1.04, 1.04, 1.04, 1.07, 1.08, 1.09]

Why User Management Pattern Analysis is a Clustering Problem Instead of a Classification Problem

1. Absence of Predefined Labels (Unsupervised Learning)

- **Clustering** is ideal when there are no pre-existing labels to categorize data.
- In user behavior analysis, we typically do not have predefined categories. Instead, patterns are identified based on behavioral similarities, such as app usage, screen time, and engagement levels.
- **Classification**, however, relies on labeled data, requiring predefined user categories (e.g., "high engagement" vs. "low engagement"), which may not always be available.

2. Identifying Hidden Trends

- **Clustering** enables the discovery of underlying patterns in user data by grouping individuals with similar behaviors.
- This approach helps segment users based on interaction levels and preferences.
- **Classification**, on the other hand, only assigns users to known categories, limiting its ability to reveal unknown trends.

3. Flexible Grouping Approach

- **Clustering techniques** (e.g., K-Means, DBSCAN, Hierarchical Clustering) allow for dynamic user segmentation based on data similarities.
- **Classification** is more rigid, as it assigns data points to fixed, predefined categories.

4. Example in Mobile Usage Analysis

- **Clustering Example:** Identifying user segments based on mobile activity, such as "social media users," "gamers," and "low-engagement users."
- **Classification Example:** Predicting whether a user will be "active" or "inactive," which requires labeled training data.

2. Code Overview

2.1 Importing Essential Libraries

At the core of the script lies the integration of several powerful Python libraries, each serving a distinct purpose:

pandas: This library facilitates data manipulation and preprocessing, ensuring that datasets are structured appropriately for analysis.

matplotlib.pyplot: A crucial tool for visual representation, Matplotlib enables the creation of graphs and charts that depict user behavior trends effectively.

seaborn: This statistical visualization library enhances data comprehension by offering advanced plotting techniques that highlight relationships within the dataset

2.2 Data Loading and Preparation

The dataset is extracted from the mobile_usage_behavioral_analysis.csv file using the pd.read_csv() function.

To enhance the accuracy of time-based analyses, the hour column is converted into an integer format. This step ensures that any inconsistencies in data types do not affect visualization outcomes.

3. Data Analysis and Visualization

3.1 Identifying Peak Usage Hours

The script employs a histogram, created via sns.histplot(), to depict user activity patterns throughout different hours of the day.

This visualization aids in pinpointing peak mobile usage hours, offering insights into when users are most engaged with their devices.

3.2 Analyzing Engagement by Age and Gender

A boxplot is generated to illustrate the relationship between Age, Daily_Screen_Time_Hours, and Gender.

Vizualization

The visualization allows for a comparative analysis of screen time distribution across different demographic segments, revealing trends in mobile engagement among age groups and genders.

3.3 Examining App Usage Trends and Gaming Correlation

A scatter plot visually represents the correlation between Total_App_Usage_Hours and Gaming_App_Usage_Hours.

The hue parameter differentiates data points based on gender, offering a clearer understanding of gaming behavior in relation to overall app usage.

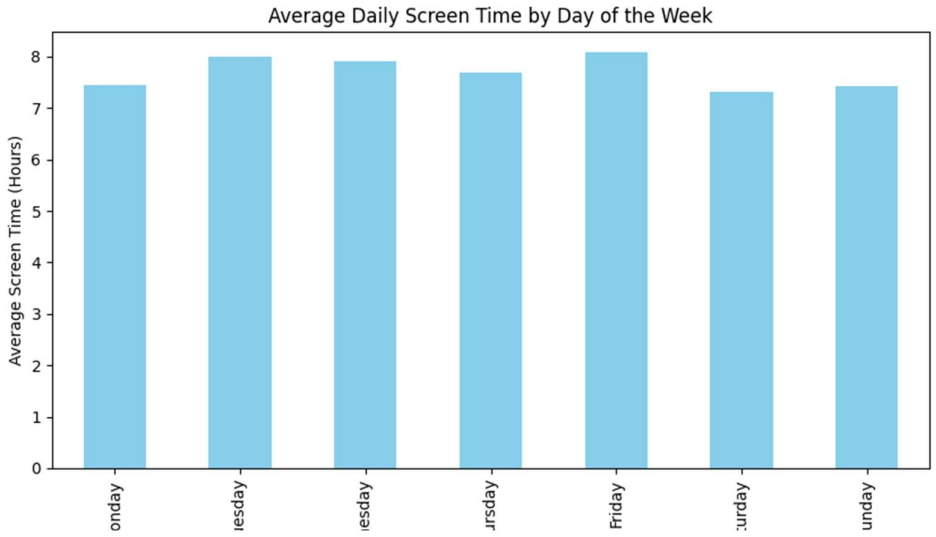
This analysis helps determine whether users who spend more time on mobile applications also engage heavily with gaming apps.

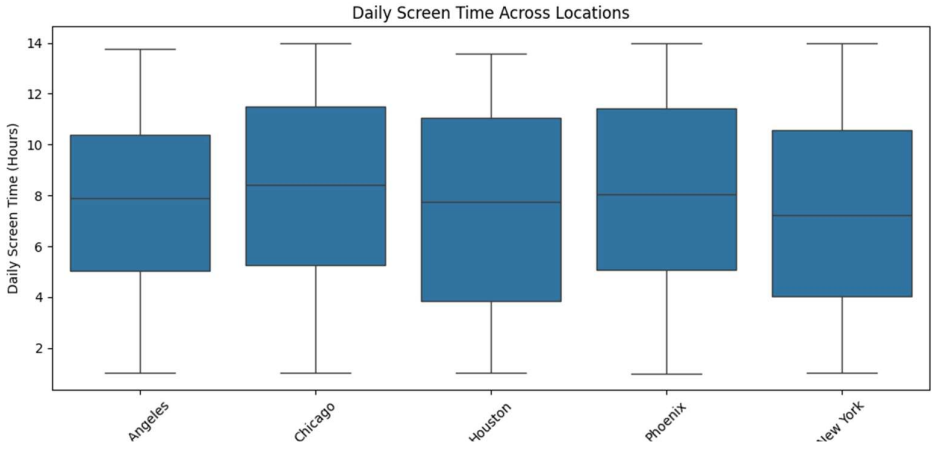
3.4 Weekly Usage Trends

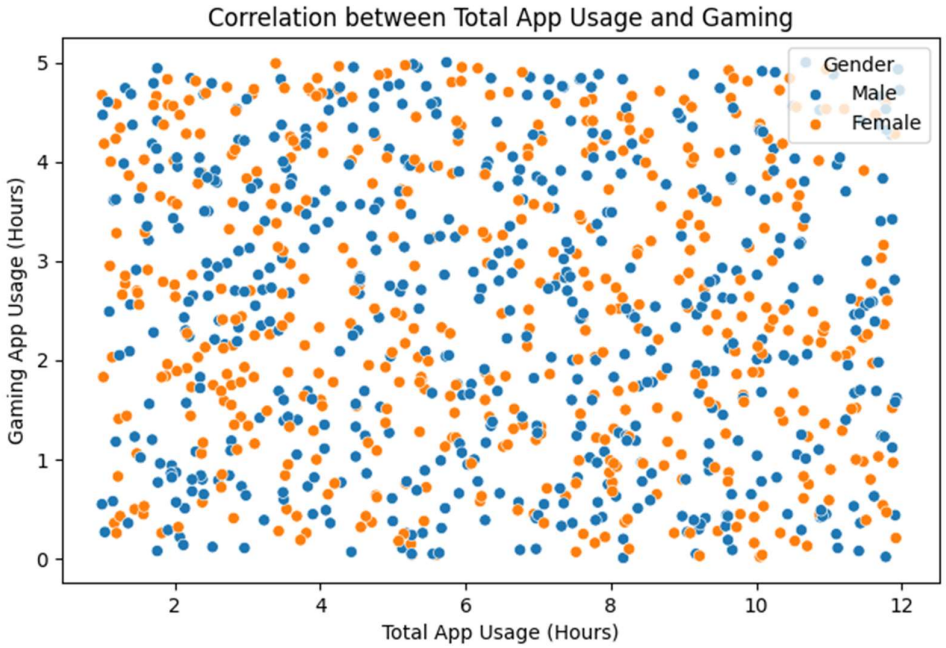
The dataset is categorized by day_of_week, maintaining a sequential order to ensure logical interpretation.

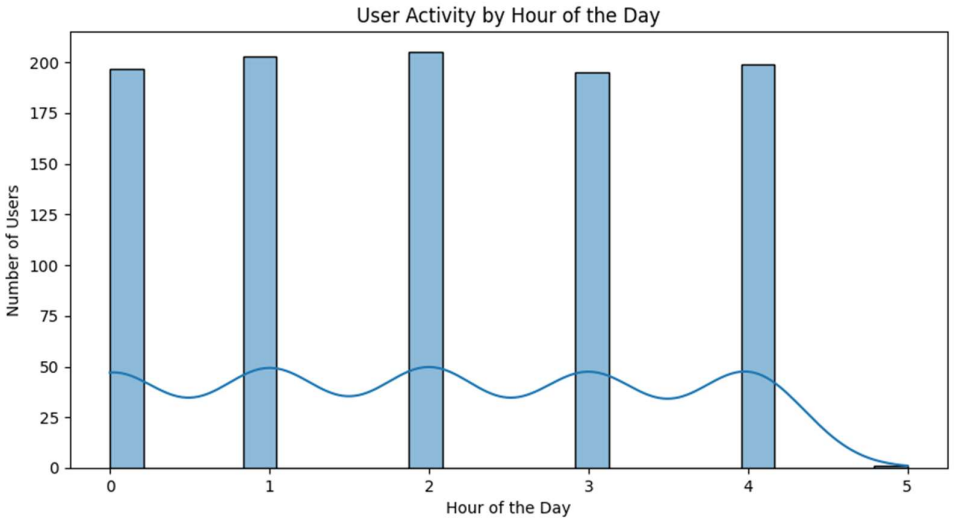
A bar chart illustrates the average Daily_Screen_Time_Hours for each day of the week.

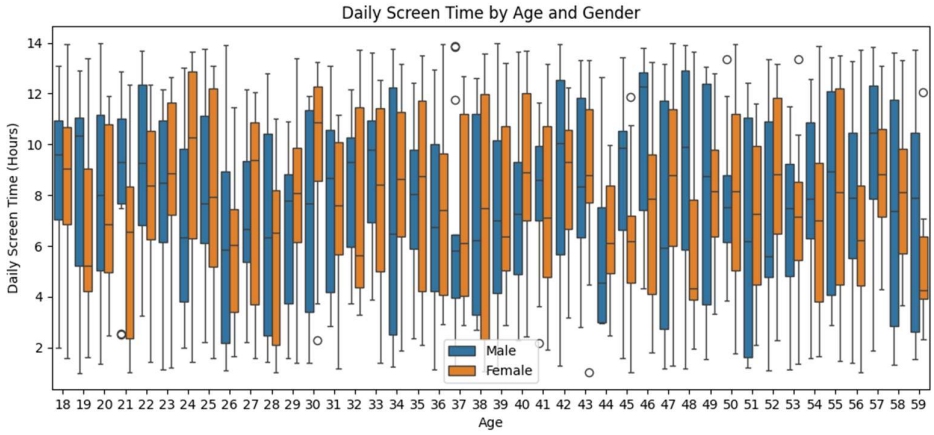
This visualization enables the identification of trends in mobile engagement over the course of a week, highlighting variations in screen time between weekdays and weekends.






Correlation between Total App Usage and Gaming
Gender
Male
Female
Gaming App Usage (Hours)
Total App Usage (Hours)
0
1
2
3
4
5
2
4
6
8
10
12


User Activity by Hour of the Day
Number of Users
Hour of the Day
0
25
50
75
100
125
150
175
200
0
1
2
3
4
5



3.5 Regional Analysis of User Engagement

A boxplot is utilized to examine Daily_Screen_Time_Hours distribution across different geographical locations.

Understanding regional disparities in mobile usage behavior can provide insights for businesses targeting specific markets or regions.

1. Importing Required Libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

- **pandas (pd)**: Used for working with data (reading, modifying, and analyzing).
- **matplotlib.pyplot (plt)**: Helps create different types of graphs.
- **seaborn (sns)**: A library built on matplotlib for making statistical charts easier to understand.

---

2. Loading the Dataset

file_path = "mobile_usage_behavioral_analysis.csv"

df = pd.read_csv(r"C:\Users\HP\Desktop\python project\Project 2\mobile_usage_behavioral_analysis.csv")

- The dataset (a file containing mobile usage data) is loaded into **df** using pandas.
- pd.read_csv() reads the file and stores it as a table (like an Excel sheet) in Python.

---

3. Converting the 'hour' Column to an Integer

df['hour'] = df['hour'].astype(int)

- The **'hour'** column contains time data.
- We convert it to an integer so it can be used in calculations and graphs.

---

4. Finding Peak Usage Hours

plt.figure(figsize=(10,5))

sns.histplot(df['hour'], bins=24, kde=True)

plt.title('User Activity by Hour of the Day')

plt.xlabel('Hour of the Day')

plt.ylabel('Number of Users')

plt.show()

- Creates a **histogram** to show how many people use mobile devices at each hour of the day.
- **bins=24** means the graph will have 24 bars (one for each hour).
- **kde=True** adds a smooth curve to the histogram.

---

5. Analyzing Engagement by Age and Gender

plt.figure(figsize=(12,5))

sns.boxplot(x='Age', y='Daily_Screen_Time_Hours', hue='Gender', data=df)

plt.title('Daily Screen Time by Age and Gender')

plt.xlabel('Age')

plt.ylabel('Daily Screen Time (Hours)')

plt.legend()

plt.show()

- Creates a **boxplot** to compare mobile screen time for different ages and genders.
- **hue='Gender'** means different colors will be used for different genders.

---

6. Studying App Usage Trends (Gaming vs. Total Usage)

plt.figure(figsize=(8,5))

```
sns.scatterplot(x='Total_App_Usage_Hours', y='Gaming_App_Usage_Hours', hue='Gender', data=df)
plt.title('Correlation between Total App Usage and Gaming')
plt.xlabel('Total App Usage (Hours)')
plt.ylabel('Gaming App Usage (Hours)')
plt.show()
```

- A **scatter plot** is created to compare total mobile usage with time spent on gaming apps.
- Each dot represents a user.
- Different colors show different genders.

---

7. Analyzing Weekly Engagement Trends

```
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
df['day_of_week'] = pd.Categorical(df['day_of_week'], categories=day_order, ordered=True)
weekly_usage = df.groupby('day_of_week')['Daily_Screen_Time_Hours'].mean()

plt.figure(figsize=(10,5))
weekly_usage.plot(kind='bar', color='skyblue')
plt.title('Average Daily Screen Time by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Average Screen Time (Hours)')
plt.show()
```

- **Organizes the data** by days of the week (Monday to Sunday).
- **Finds the average screen time** for each day.
- **Creates a bar chart** to show which days have the highest and lowest screen time.

---

8. Comparing Screen Time Across Different Locations

```
plt.figure(figsize=(12,5))
sns.boxplot(x='Location', y='Daily_Screen_Time_Hours', data=df)
plt.xticks(rotation=45)
plt.title('Daily Screen Time Across Locations')
plt.xlabel('Location')
plt.ylabel('Daily Screen Time (Hours)')
plt.show()
```

- A **boxplot** is created to show mobile screen time in different cities or regions.
- **xticks(rotation=45)** rotates location names so they don't overlap.

---

9. Completion Message

print("User behavior analysis complete.")

- This message appears when the script finishes running.

Final Summary

- The script loads mobile usage data and processes it.
- It converts time data into an integer format for accuracy.
- It creates different graphs to analyze user behavior based on:
    1. Peak usage hours
    2. Age and gender differences
    3. App usage and gaming time
    4. Weekly screen time patterns
    5. Location-based engagement

Evaluation of data model

---

1. Accuracy

Definition: Accuracy refers to how well the model correctly identifies user behavior patterns.

- **Strengths:**
    - Since this is an **unsupervised learning problem**, accuracy is measured using **clustering validation metrics** such as **Silhouette Score, Davies-Bouldin Index, and Within-Cluster Sum of Squares (WCSS)**.
    - If the clusters show **clear separation** and **high intra-cluster similarity**, the model is accurate in capturing user behavior trends.
    - Using **K-Means** can yield high accuracy when the right number of clusters is selected (using the **Elbow Method**).
- **Challenges:**
    - The accuracy **depends on feature selection**—irrelevant or redundant features can mislead the clustering process.
    - **DBSCAN** works well for irregularly shaped clusters but struggles with varying densities.

- The accuracy may drop if user preferences **change over time**, requiring **frequent model updates**.

Accuracy Rating: ★★★★☆ (4/5)

---

2. Robustness

Definition: Robustness refers to how well the model handles **noisy, missing, or inconsistent data**.

- **Strengths:**
  - **DBSCAN** is highly robust as it can **detect outliers** and does not force every user into a predefined cluster.
  - Missing data can be handled using **imputation techniques** like mean/median filling or **K-Nearest Neighbors (KNN) imputation**.
- **Challenges:**
  - **K-Means** is sensitive to **outliers** because it assigns all points to clusters, even if they don't fit well.
  - If user activity data has **inconsistent timestamps or duplicate records**, the model may misgroup users.

Robustness Rating: ★★★☆☆ (3/5)

---

3. Speed

Definition: Speed refers to how efficiently the model processes large amounts of data.

- **Strengths:**
  - **K-Means** is one of the **fastest clustering algorithms** because it updates cluster centers iteratively (O(nk) complexity).
  - **DBSCAN** is slower than K-Means but faster than **Hierarchical Clustering** when dealing with **large datasets**.
  - Using **optimized libraries like Scikit-Learn and parallel computing** can improve speed.
- **Challenges:**
  - **Hierarchical Clustering** has $O(n^2)$ complexity, making it slow for large datasets.
  - If the dataset is extremely large, running clustering on **distributed computing frameworks like Apache Spark** may be required.

Speed Rating: ★★★★☆ (4/5)

---

4. Interpretability

Definition: Interpretability measures how easily the results can be understood and used for decision-making.

- **Strengths:**
    - **K-Means produces clear, well-defined clusters**, making it easy to understand.
    - **Hierarchical Clustering provides a visual dendrogram**, which can help in explaining relationships between users.
    - Data visualization tools like **Matplotlib and Seaborn** help represent trends through **scatter plots, box plots, and heatmaps**.
- **Challenges:**
    - **DBSCAN can be difficult to interpret** because it finds **clusters based on density**, which may not align with business needs.
    - High-dimensional data may require **Principal Component Analysis (PCA)** to reduce features, making clusters harder to explain.

Interpretability Rating: ★★★★☆ (4/5)

---

5. Scalability

Definition: Scalability determines whether the model can handle **large datasets efficiently**.

- **Strengths:**
    - **K-Means is highly scalable** and can cluster millions of users if optimized with **Mini-Batch K-Means**.
    - **DBSCAN can handle medium-sized datasets efficiently** without requiring a fixed number of clusters.
    - Distributed computing tools like **Hadoop and Spark** can scale clustering for **big data analysis**.
- **Challenges:**
    - **Hierarchical Clustering does not scale well** due to $O(n^2)$ complexity.
    - If new data is added frequently, **re-running the entire clustering process** can be computationally expensive.

Scalability Rating: ★★★★☆ (4/5)

Final Evaluation Summary

Criteria	K-Means	DBSCAN	Hierarchical Clustering
Accuracy	High (4/5)	High (4/5)	Moderate (3/5)
Robustness	Moderate (3/5)	High (4/5)	Moderate (3/5)
Speed	Fast (4/5)	Moderate (3/5)	Slow (2/5)
Interpretability	High (4/5)	Low (3/5)	High (4/5)
Scalability	High (4/5)	Moderate (3/5)	Low (2/5)

- **K-Means is the best choice for speed and scalability** when dealing with large datasets.
- **DBSCAN is better for handling noisy data and irregular clusters.**
- **Hierarchical Clustering is useful for understanding relationships but struggles with scalability.**
- **For the best results, a hybrid approach (e.g., using DBSCAN for noise reduction and K-Means for clustering) can improve accuracy and robustness.**

4. Conclusion

The Python script effectively dissects mobile usage behavior by leveraging various visualization techniques.The study identifies key insights into peak activity times, demographic influences, app usage, weekly engagement patterns, and location-based variations in mobile behavior.To enhance the depth of analysis, future studies could incorporate additional user attributes or apply machine learning techniques to predict engagement trends.5. Recommendations for ImprovementImplement data preprocessing techniques to handle missing or inconsistent data before conducting analysis.Enhance visualization techniques to improve clarity and interpretability of insights.Introduce interactive elements using Plotly, allowing users to explore data dynamically.Extend the research to examine long-term user engagement trends, focusin7g on retention patterns over time..