ScreenSense: Kids' Screentime Visualization

Milestone 1: Data Foundation and Cleaning

Prepared by: *Haripriya Mahajan*
Mentor: *Sirisha Arangi*

Week 1: Project Initialization and Dataset Setup

1. Project Goals and Workflow

The main goal of the **ScreenSense: Kids' Screentime Visualization** project is to **analyse and understand children's screentime habits** using data.
Through this analysis, we aim to **find meaningful trends and patterns** that can help **parents, teachers, and policymakers** make better decisions about kids' digital health and learning balance.

Goals:

1. **Understand children's screentime behavior** — how much time they spend on screens each day and what devices they use most.
2. **Identify trends by different factors** such as age, gender, urban/rural background, and type of activity (educational vs recreational).
3. **Visualize key insights** using clear and attractive charts, graphs, and dashboards.
4. **Highlight possible health impacts** like eye strain, poor sleep, or anxiety related to excessive screen time.
5. **Present actionable insights** in a way that's easy to understand for non-technical audiences.

→ Workflow of the Project

1. Collecting and Exploring the Data
I will load the dataset from Kaggle and explore its structure, size, columns, data types, and missing values to understand the information available.

2. Cleaning and Preparing the Data
I will handle missing values, fix inconsistencies, and create new useful features. For example, I will group ages into three categories — **8–11 (Kids), 12–14 (Pre-Teens), and 15–18 (Teens)**

— and simplify the health impact information. I will save the cleaned dataset for reuse in later analysis.

3. Importing Required Libraries
I will import libraries like pandas, numpy, matplotlib, and seaborn for data handling and visualization.

4. Univariate and Bivariate Analysis
I will explore each column, such as age, gender, and device type, individually using histograms, bar charts, and distributions. Then, I will examine combinations — for instance, how average screen time varies by **age and gender** together.

5. Studying Devices, Activities, and Usage Patterns
I will analysee which devices each age group uses most, the types of activities (educational vs recreational), and how screen time differs across **gender** and **urban/rural locations**.

6. Selecting Suitable Graphs
I will choose the most effective visualizations (bar charts, heatmaps, distributions) to clearly present insights for each analysis.

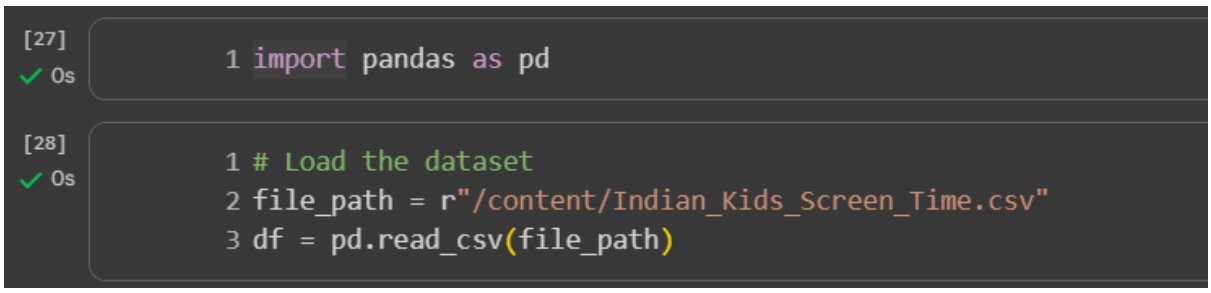
7. Building Visuals and Dashboard
I will combine all insights and charts into a visual **Power BI dashboard**, allowing users to interactively filter by age group, gender, device type, and location to explore trends.

---

2. Load the dataset

I loaded the dataset into the Python environment using the **pandas library** for easy data handling and analysis.

After loading, the dataset was stored in a **pandas DataFrame**, which is a tabular structure (rows and columns) suitable for data analysis and visualization.



---

3. Explore schema, data types, size, and nulls

(a) Dataset Size

```
[29]
✓ 0s
```

```python
#Dataset size (rows x columns)
print("Dataset Shape:", df.shape)
```

```
Dataset Shape: (9712, 8)
```

(b) Column Names

```
[49]
✓ 0s
```

```python
# Column Names vertically
print("Column Names:\n")
for col in df.columns:
    print(col)
```

```
Column Names:

Age
Gender
Avg_Daily_Screen_Time_hr
Primary_Device
Exceeded_Recommended_Limit
Educational_to_Recreational_Ratio
Health_Impacts
Urban_or_Rural
```

(c) Data types of each column

[43]
✓ 0s

```
# Data types of each column
print("Data Types:\n\n", df.dtypes)
```

```
Data Types:

 Age                                int64
Gender                             object
Avg_Daily_Screen_Time_hr          float64
Primary_Device                     object
Exceeded_Recommended_Limit           bool
Educational_to_Recreational_Ratio  float64
Health_Impacts                     object
Urban_or_Rural                     object
dtype: object
```

(d) Summary statistics for numeric columns:

[40]
✓ 0s

```
# Summary statistics for numeric columns
print("Numeric Summary:\n")
display(df.describe())
```

```
Numeric Summary:
```

	Age	Avg_Daily_Screen_Time_hr	Educational_to_Recreational_Ratio
count	9712.000000	9712.000000	9712.000000
mean	12.979201	4.352837	0.427226
std	3.162437	1.718232	0.073221
min	8.000000	0.000000	0.300000
25%	10.000000	3.410000	0.370000
50%	13.000000	4.440000	0.430000
75%	16.000000	5.380000	0.480000
max	18.000000	13.890000	0.600000

(e) Summary Statistics for categorical columns:

[44] ✓ 0s

```
# Summary statistics for categorical columns
print("Categorical Summary:\n\n")
display(df.describe(include='object'))
```

```
Categorical Summary:
```

	Gender	Primary_Device	Health_Impacts	Urban_or_Rural
count	9712	9712	6494	9712
unique	2	4	15	2
top	Male	Smartphone	Poor Sleep	Urban
freq	4942	4568	2268	6851

(f) Check for null values:

[47] ✓ 0s

```
# Check for null/missing values
print("Missing Values per Column:\n\n", df.isnull().sum())
```

```
Missing Values per Column:

 Age                                0
Gender                              0
Avg_Daily_Screen_Time_hr            0
Primary_Device                      0
Exceeded_Recommended_Limit          0
Educational_to_Recreational_Ratio   0
Health_Impacts                   3218
Urban_or_Rural                      0
dtype: int64
```

---

4. Capture initial notes on quality and assumptions

(a) **Dataset Size & Structure**

- The dataset has **9712 rows and 8 columns**, making it a medium-sized dataset suitable for analysis.
- Columns include a mix of **numeric** (Age, Avg_Daily_Screen_Time_hr, Educational_to_Recreational_Ratio), **categorical** (Gender, Primary_Device, Urban_or_Rural, Health_Impacts), and **boolean** (Exceeded_Recommended_Limit) types.

(b) **Missing Values**

- Only the Health_Impacts column has missing values (**3218 out of 9712**, ~33%).
- All other columns are complete and reliable for analysis.
- **Assumption:** missing Health_Impacts likely means **no health issues reported**, so these can be filled with "Healthy" or "None".

(c) **Data Types & Consistency**

- Numeric columns (Age, Avg_Daily_Screen_Time_hr, Educational_to_Recreational_Ratio) are appropriately stored as int or float.
- Categorical columns appear consistent (e.g., Gender has Male/Female, Urban_or_Rural has Urban/Rural).
- Primary_Device is consistent with 4 distinct types (Smartphone, Laptop, TV, and Tablet).
- Health_Impacts has multiple comma-separated values and will need to be split for detailed analysis.

(d) **Value Distributions & Observations**

- **Age** ranges from 8–18, covering children and teens.
- **Avg_Daily_Screen_Time_hr** ranges from 0 to 13.89 hours/day, with most kids spending around 3–5 hours/day.
- **Educational_to_Recreational_Ratio** ranges from 0.3–0.6, with a median of 0.43, suggesting slightly more educational than recreational time on average.
- **Health_Impacts** top reported issue is “Poor Sleep,” but many rows have multiple impacts (e.g., Poor Sleep, Eye Strain).

(e) **Assumptions for Analysis**

- Missing Health_Impacts → treated as "Healthy".
- Boolean Exceeded_Recommended_Limit is clean and can be used directly.
- Numeric and categorical columns are accurate and consistent, so no major cleaning needed apart from standardizing labels and creating derived features.
- Age can be grouped into **3 categories** (Kids 8–11, Pre-teens 12–14, Teens 15–18) for cohort analysis.

Week 2: Preprocessing and Feature Engineering

(a) Handling Missing Values

- The **Health_Impacts** column had missing values in around 33% of rows.
- To make this data more interpretable:
    - If a child **exceeded the recommended screen limit** and the health impact was missing, I labeled it as **"Potential Risk."**
    - If the child **did not exceed the limit** and the health impact was missing, I labeled it as **"Healthy."**

```
# Case 1: If limit exceeded and Health_Impacts is missing → label as 'Potential Risk'
df.loc[
    (df['Health_Impacts'].isna()) & (df['Exceeded_Recommended_Limit'] == True),
    'Health_Impacts'
] = 'Potential Risk'

# Case 2: If limit not exceeded and Health_Impacts is missing → label as 'Healthy'
df.loc[
    (df['Health_Impacts'].isna()) & (df['Exceeded_Recommended_Limit'] == False),
    'Health_Impacts'
] = 'Healthy'

# Check remaining missing values (if any)
df['Health_Impacts'].isna().sum()
```

```
np.int64(0)
```

- After applying this logic, no missing values remained in the column.

(b) Fixing Inconsistent Categories

To ensure uniformity, categorical fields were standardized:

- **Gender**, **Primary_Device**, and **Urban_or_Rural** columns were cleaned by removing extra spaces and converting all text to title case (e.g., *male* → *Male*, *urban* → *Urban*).
- This prevents duplication of categories during grouping or visualization.

```
# Fixing Inconsistent Categories
df['Gender'] = df['Gender'].str.strip().str.title()
df['Primary_Device'] = df['Primary_Device'].str.strip().str.title()
df['Urban_or_Rural'] = df['Urban_or_Rural'].str.strip().str.title()
```

(c) Creating Derived Features

1. **Age Groups**
    - Children were divided into three logical cohorts for better demographic analysis:
        - **Kids:** 8–11 years
        - **Pre-Teens:** 12–14 years
        - **Teens:** 15–18 years
    - This feature will help compare patterns like screen time or device preference across age categories.

```python
# Creating age groups
def categorize_age(age):
    if 8 <= age <= 11:
        return 'Kids'
    elif 12 <= age <= 14:
        return 'Pre-Teens'
    else:
        return 'Teens'

df['Age_Group'] = df['Age'].apply(categorize_age)

# Checking Age Group distribution
print("Age Group Distribution:\n", df['Age_Group'].value_counts(), "\n")
```

---

2. **Screen Time Categories**
    - Based on average daily screen hours, I grouped users as:
        - **Low Usage:** < 2 hours
        - **Moderate Usage:** 2–5 hours
        - **High Usage:** 5–8 hours
        - **Very High Usage:** > 8 hours
    - This categorization provides an easier way to visualize digital exposure levels.

```python
# Creating screen time groups
bins = [0, 2, 5, 8, 24]  # 24 just to include all possible values
labels = ["Low Usage", "Moderate Usage", "High Usage", "Very High Usage"]

df["ScreenTime_Category"] = pd.cut(df["Avg_Daily_Screen_Time_hr"], bins=bins, labels=labels, right=False)

# Check the distribution
print(df["ScreenTime_Category"].value_counts())
```

```
ScreenTime_Category
Moderate Usage     5588
High Usage         3123
Low Usage           826
Very High Usage     175
Name: count, dtype: int64
```

3. **Educational-to-Recreational Ratio Groups**
    - Based on the ratio values, I defined three groups:
        - **Mostly Recreational:** 0.3–0.4
        - **Balanced Usage:** 0.4–0.5
        - **Mostly Educational:** 0.5–0.6
    - This feature helps in identifying students who use screens more for learning vs entertainment.

```python
# Create Educational vs Recreational ratio groups
bins = [0.3, 0.4, 0.5, 0.61]
labels = ["Mostly Recreational", "Balanced Usage", "Mostly Educational"]

df["EduRec_Category"] = pd.cut(df["Educational_to_Recreational_Ratio"], bins=bins, labels=labels, right=False)

# Check the distribution
print(df["EduRec_Category"].value_counts())
```

```
EduRec_Category
Balanced Usage         4726
Mostly Recreational    3389
Mostly Educational     1597
Name: count, dtype: int64
```

(d) One-Hot Encoding for Multi-Label Health Impacts

- The **Health_Impacts** column contained multiple comma-separated values (e.g., *"Eye Strain, Poor Sleep"*).
- Each health condition was separated and converted into individual binary columns using **MultiLabelBinarizer**.
- For example, new columns like HealthImpact_Eye Strain and HealthImpact_Poor Sleep were created with 1/0 values indicating whether each issue applied to the child.
- This transformation will make it possible to analyze which health issues are most common across different groups.

```python
# One-Hot Encoding for Health_Impacts column

# Split comma-separated values into lists
df["Health_Impacts_List"] = df["Health_Impacts"].apply(lambda x: [i.strip() for i in x.split(",")])

# Use MultiLabelBinarizer to create one-hot encoded columns
from sklearn.preprocessing import MultiLabelBinarizer

mlb = MultiLabelBinarizer()
health_encoded = pd.DataFrame(
    mlb.fit_transform(df["Health_Impacts_List"]),
    columns=[f"HealthImpact_{c}" for c in mlb.classes_]
)

# Combine the new one-hot columns with the original dataframe
df = pd.concat([df, health_encoded], axis=1)

# Drop the helper column
df.drop(columns=["Health_Impacts_List"], inplace=True)
```

(e) Final Checks and Saving Processed Data

- After all preprocessing steps, I checked the dataset for remaining null values — none were found.

```python
# Check for remaining nulls
print(df.isnull().sum(), "\n")
```

```
Age                                0
Gender                             0
Avg_Daily_Screen_Time_hr           0
Primary_Device                     0
Exceeded_Recommended_Limit         0
Educational_to_Recreational_Ratio  0
Health_Impacts                     0
Urban_or_Rural                     0
Age_Group                          0
ScreenTime_Category                0
EduRec_Category                    0
HealthImpact_Anxiety               0
HealthImpact_Eye Strain            0
HealthImpact_Healthy               0
HealthImpact_Obesity Risk          0
HealthImpact_Poor Sleep            0
HealthImpact_Potential Risk        0
dtype: int64
```

- The cleaned and enriched dataset was saved as:

  data/processed/indian_kids_screentime_cleaned.csv

```
# Creating the directory
os.makedirs("data/processed", exist_ok=True)

# Save the cleaned data to a new processed file
df.to_csv("data/processed/indian_kids_screentime_cleaned.csv", index=False)
```

(f) Outcomes

- All missing and inconsistent entries were resolved.
- New derived features (Age_Group, ScreenTime_Category, and EduRec_Category) were successfully created.
- Health impacts were standardized and one-hot encoded for flexible visualization.
- The dataset is now clean, structured, and analysis-ready for univariate and bivariate visual exploration.
