

# Project 1 Group 8 Write up

Submitted by: Asia Byrne, Edward Yang, Victor Rincon, Brian Marowsky

## Introduction

Reviewing the datasets available to us from Kaggle (link/citation). One dataset that attracted our attention and curiosity was “insert dataset name”. With over 98% of Americans owning a cellphone in 2024, and more than nine out of 10 of all cell phone owners have a smartphone, . (ConsumerAffairs. “Cell phone statistics 2024 [2024]” ConsumerAffairs.com. Dec. 02, 2024, [https://www.consumeraffairs.com/cell\\_phones/cell-phone-statistics.html](https://www.consumeraffairs.com/cell_phones/cell-phone-statistics.html))

We decided to analyze cell phone data and try to answer some questions about our behavior with a device , which continues to evolve and expand across global markets.

The data set was a collection of 700 simulated users from ages 18-59 which included a variety of data and usage records such as data usage, battery life, phone models, operating systems in addition to gender and age of the users.

This simulated data was based on real world data usage collected from reputable sources such as Pew Research Group and Statista. We searched for datasets on various websites and found them on Kaggle. Our dataset is User Behavior Class in which we can determine the user’s mobile behavior class using various other features. Our dataset link is as follows:

<https://www.kaggle.com/datasets/valakhorasani/mobile-device-usage-and-user-behavior-dataset>

## Data Review and Cleaning

### Initial Data Review

Review of the data (df.info) showed that it was well maintained. It contained enough parameters to develop and answer our research questions. The dataset also had an equal number of rows and no null values which allowed minimal data cleaning.

Out of the 11 columns in our dataset column 10 ‘User Behavior Class’ was the only one we couldn’t interpret properly and determined would not be necessary for our analysis. The description of the dataset didn’t include a detailed explanation of how the ‘User Behavior Class’ scale was developed or calculated. The column had values ranging from 1 (light usage) to 5 (heavy usage) without an explanation of what data determined those values and how they were applied to each user.

```
ub.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 11 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   User ID                               700 non-null    int64
1   Device Model                           700 non-null    object
2   Operating System                       700 non-null    object
3   App Usage Time (min/day)               700 non-null    int64
4   Screen On Time (hours/day)             700 non-null    float64
5   Battery Drain (mAh/day)                700 non-null    int64
6   Number of Apps Installed               700 non-null    int64
7   Data Usage (MB/day)                   700 non-null    int64
8   Age                                    700 non-null    int64
9   Gender                                 700 non-null    object
10  User Behavior Class                    700 non-null    int64
dtypes: float64(1), int64(7), object(3)
memory usage: 60.3+ KB
```

Our dataset contains 700 rows with no null values. Mainly integers. Out of the 11 columns there was one where we thought with advice from one of our TAs to drop. It was the last one named “User Behavior”. According to the description within the data set, the values were ranging from 1 to 5. 1 being “light usage” and 5 being “extreme usage”. It was brought to our attention that there was really no spectrum or scale to actually determine what defines these usage. Removing the column “User Behavior Class” was our only data cleaning required for our dataset as a whole.

```
# Dropping "User Behavior Class"

ub2 = ub.drop("User Behavior Class", axis=1)

# Save the result
ub2.info()
ub2.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 10 columns):
 #   Column                                Non-Null Count  Dtype
---  -
 0   User ID                              700 non-null    int64
 1   Device Model                          700 non-null    object
 2   Operating System                      700 non-null    object
 3   App Usage Time (min/day)              700 non-null    int64
 4   Screen On Time (hours/day)            700 non-null    float64
 5   Battery Drain (mAh/day)               700 non-null    int64
 6   Number of Apps Installed              700 non-null    int64
 7   Data Usage (MB/day)                  700 non-null    int64
 8   Age                                   700 non-null    int64
 9   Gender                                700 non-null    object
dtypes: float64(1), int64(6), object(3)
memory usage: 54.8+ KB
```

We then looked at the unique values, mainly the device models and saw how many we were working with.

```
device_model_counts = ub2["Device Model"].value_counts()
device_model_counts
```

```
Device Model
Xiaomi Mi 11      146
iPhone 12         146
Google Pixel 5    142
OnePlus 9         133
Samsung Galaxy S21 133
Name: count, dtype: int64
```

To wrap things up we wanted to take a look at the statistical numbers for our data with the describe function.

```
ub2.describe()
```

	User ID	App Usage Time (min/day)	Screen On Time (hours/day)	Battery Drain (mAh/day)	Number of Apps Installed	Data Usage (MB/day)	Age
count	700.00000	700.000000	700.000000	700.000000	700.000000	700.000000	700.000000
mean	350.50000	271.128571	5.272714	1525.158571	50.681429	929.742857	38.482857
std	202.21688	177.199484	3.068584	819.136414	26.943324	640.451729	12.012916
min	1.00000	30.000000	1.000000	302.000000	10.000000	102.000000	18.000000
25%	175.75000	113.250000	2.500000	722.250000	26.000000	373.000000	28.000000
50%	350.50000	227.500000	4.900000	1502.500000	49.000000	823.500000	38.000000
75%	525.25000	434.250000	7.400000	2229.500000	74.000000	1341.000000	49.000000
max	700.00000	598.000000	12.000000	2993.000000	99.000000	2497.000000	59.000000

## Color Palette

When choosing a color palette for our presentation, our idea was to find a color palette map that showed a wide variety of bright, unique, colors that would give us the freedom and customization ability to break down our colors into smaller frames when needed in our data set in by using our visuals below is samples as well as data we used to break down our colors. The example below shows the breakdown into five colors which we used for the histogram and box plots as well as other visuals in our presentation although it should be mentioned these are just a small sample of colors that were available using the CMR map palette from Seaborns .

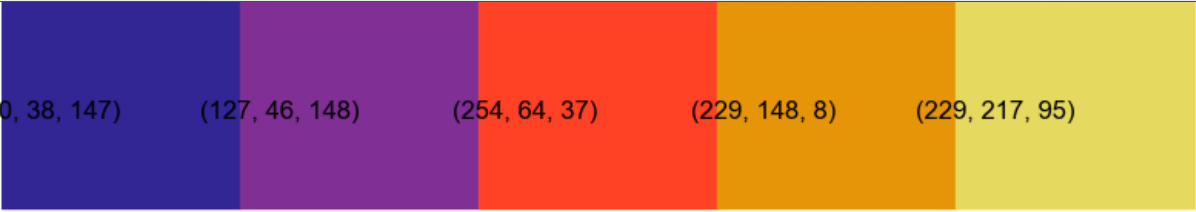
```
[187]:
# Generate the CMRmap color palette with n colors
n_colors = 5
palette = sns.color_palette(palette='CMRmap', n_colors=n_colors)

# Create a figure
plt.figure(figsize=(10, 2))

# Plot each color as a rectangle and annotate with RGB values
for i, color in enumerate(palette):
    plt.fill_between([i, i+1], 0, 1, color=color)
    rgb = tuple([int(x*255) for x in color])
    plt.text(i + 0.5, 0.5, str(rgb), ha='right', va='top', color='black', fontsize=12)

# Remove axes for clarity
plt.gca().set_axis_off()
plt.gca().set_xlim(0, len(palette))

# Show the plot
plt.show()
```



(50, 38, 147) (127, 46, 148) (254, 64, 37) (229, 148, 8) (229, 217, 95)

## Question 1

Does age of the user a factor for what phone model is preferred , affect data usage on any model ?

The focus primarily was to take the age of the users and then only compare it with the columns of Data Usage(MB/day) , and phone models.

We first had to take our cleaned data and pull out the columns we would need , in this first case it would be the age vs Data Usage(MB/day) columns as shown below.

[21]:

	Age	Data Usage (MB/day)
--	-----	---------------------

0	40	1122
---	----	------

1	47	944
---	----	-----

2	42	322
---	----	-----

3	20	871
---	----	-----

4	31	988
---	----	-----

...	...	...
-----	-----	-----

695	22	381
-----	----	-----

696	59	1201
-----	----	------

697	50	457
-----	----	-----

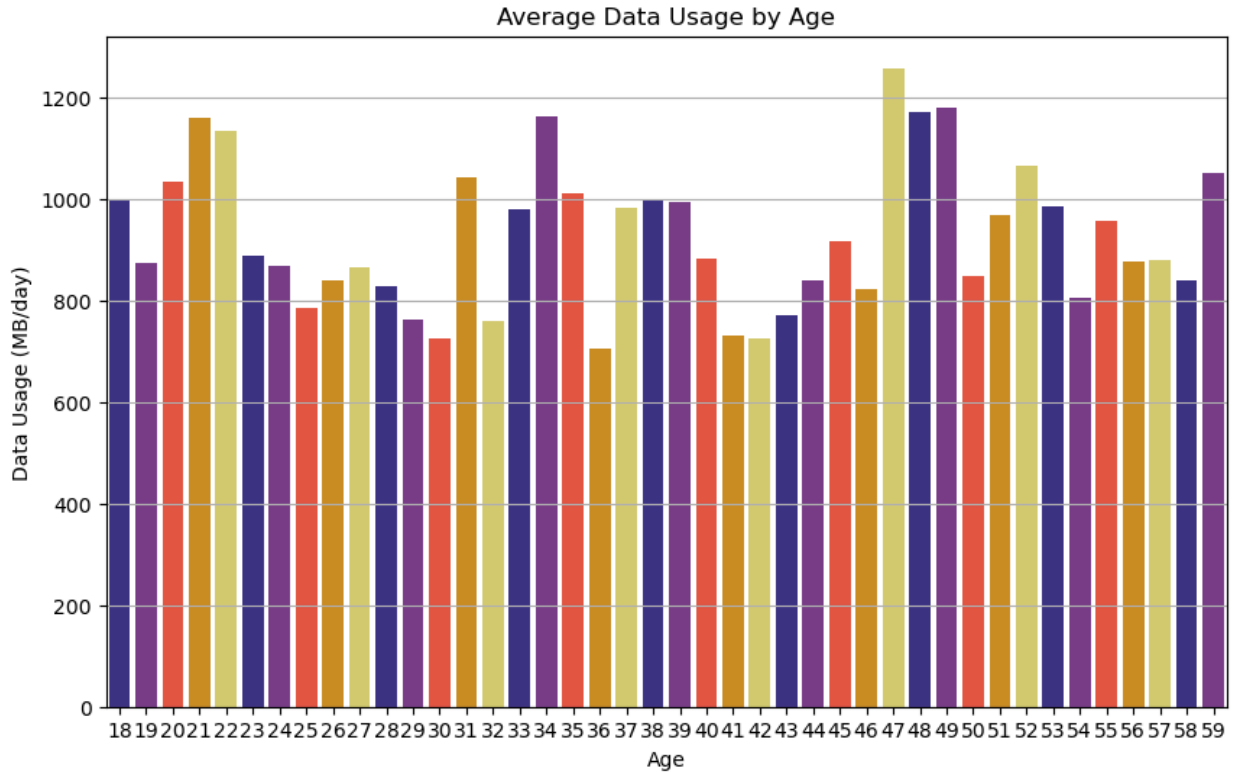
698	44	224
-----	----	-----

699	23	828
-----	----	-----

ages 18-  
]

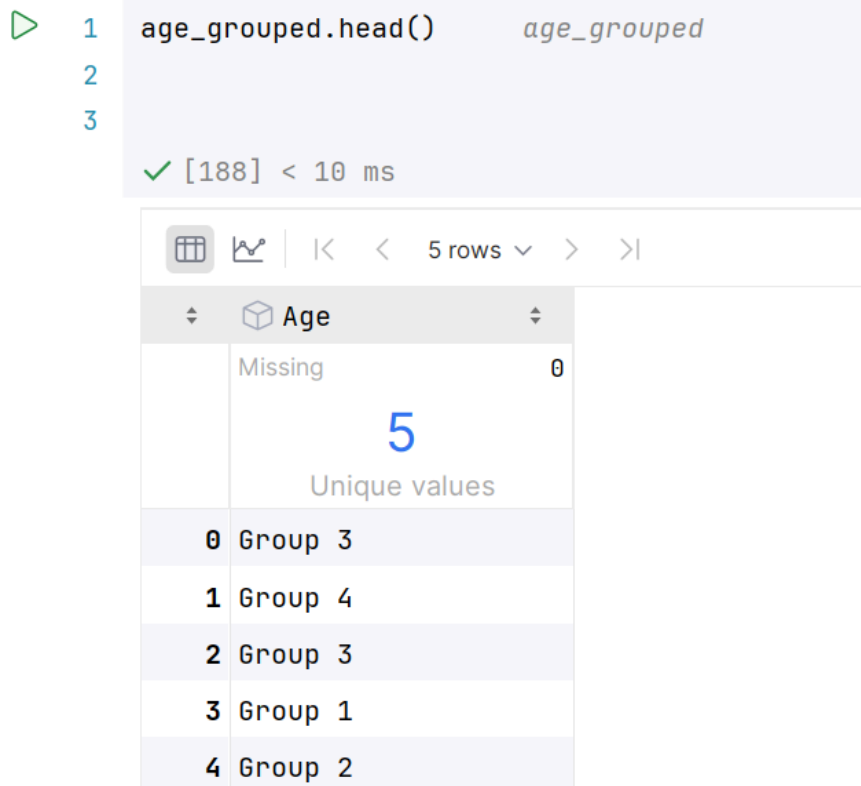
700 rows × 2 columns

```
# Show the plot  
plt.show()
```



With the chart above, even though showing the data across the ages of the users in the set, It was difficult to track the data usage trends across the age groups. In order to simplify the data above into an easier to read data view, we decided to break our ages into 4 age group categories. We had originally set off to do 5 categories but due to the category of 18-20 being so few individuals, it was decided to go into just 4 categories to show the usage of data. We checked our work to see if the ages were divided into bins, taking care to remember that 0 will be ignored in the final set.

Below is the conclusion of that result after confirming the number of groups above were confirmed.



The Graph allowed us to see the correlations as the age groups increase the amount of data used per day with the colors of the bars indicated by the key of their age groups and the lines of their corresponding color show the general trend of the data as a progressed higher in data usage and their frequency. General the trend showed that younger individuals from 18 to 30 were using the most data at least up until 2000 MB /day when age group 30 to 40 surpass The frequency of the number of times the data usage was above 2000 compared to those 18 to 30. when comparing age to device model a similar process was taken to divide the data into only age, device model.

```
[21]: # Plot histogram
#Defin what age_data usage contains
age_data_usage= age_data_usage[['Age', 'Data Usage (MB/day)']]

# Define the number of bins for ages (the first number being a placeholder)
num_bins = 5

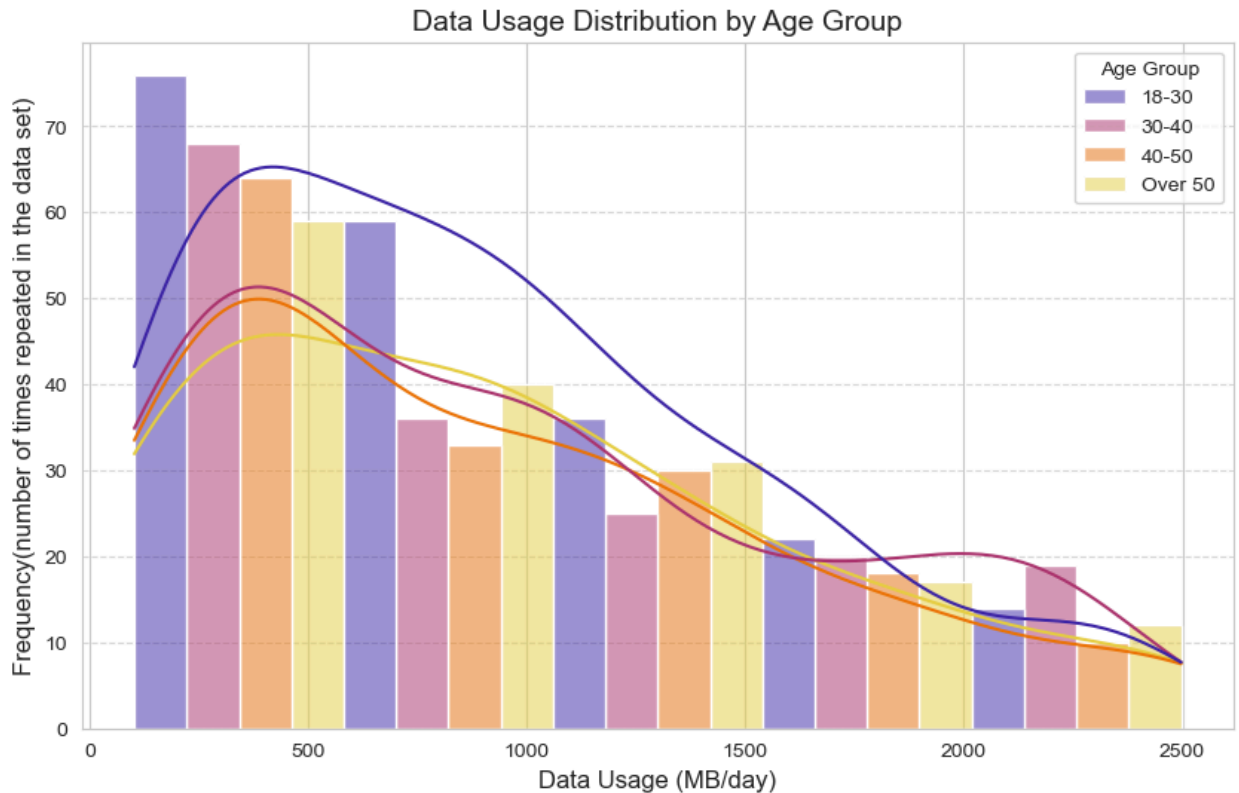
# Create bins for Age box
age_bins = [age_data_usage['Age'].min(), 30, 40, 50, df_new['Age'].max()]
# Example age ranges
age_labels = ['18-30', '30-40', '40-50', 'Over 50'] # Descriptive Labels
age_data_usage['Age Group'] = pd.cut(age_data_usage['Age'], bins=age_bins, labels=age_labels)

plt.figure(figsize=(10, 6))
sns.histplot(
    data=age_data_usage,
    x='Data Usage (MB/day)',
    hue='Age Group',
    bins=num_bins,
    kde=True,
    palette='CMRmap',
    multiple="dodge"
)

# Add titles and Labels
plt.title('Data Usage Distribution by Age Group', fontsize=14)
plt.xlabel('Data Usage (MB/day)', fontsize=12)
plt.ylabel('Frequency(number of times repeated in the data set)', fontsize=12)
plt.grid(True, axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```






The Graph allowed us to see the correlations as the age groups increase the amount of data used per day with the colors of the bars indicated by the key of their age groups and the lines of their corresponding color show the general trend of the data as a progressed higher in data usage and their frequency. General the trend showed that younger individuals from 18 to 30 were using the most data at least up until 2000 MB /day when age group 30 to 40 surpass The frequency of the number of times the data usage was above 2000 compared to those 18 to 30. when comparing age to device model a similar process was taken to divide the data into only age, device model.

# pull apart to only show Age / Data Usage

```
age_device_model_usage = df[['Age', 'Device Model']]
```

```
Age_device_model_usage.head()
```

123 Age			Device Model		
Missing 0			Missing 0		
			4 Unique values		
0	40		Google Pixel 5		
1	47		OnePlus 9		
2	42		Xiaomi Mi 11		
3	20		Google Pixel 5		
4	31		iPhone 12		

From this data we decided to go for a box plot being the fact that we would be comparing age groups with cell phone models to which there were only five in the data set.

```
[185]: #Device model vs Age
#Standard Box Plot
# Define the number of bins for ages
num_bins = 5

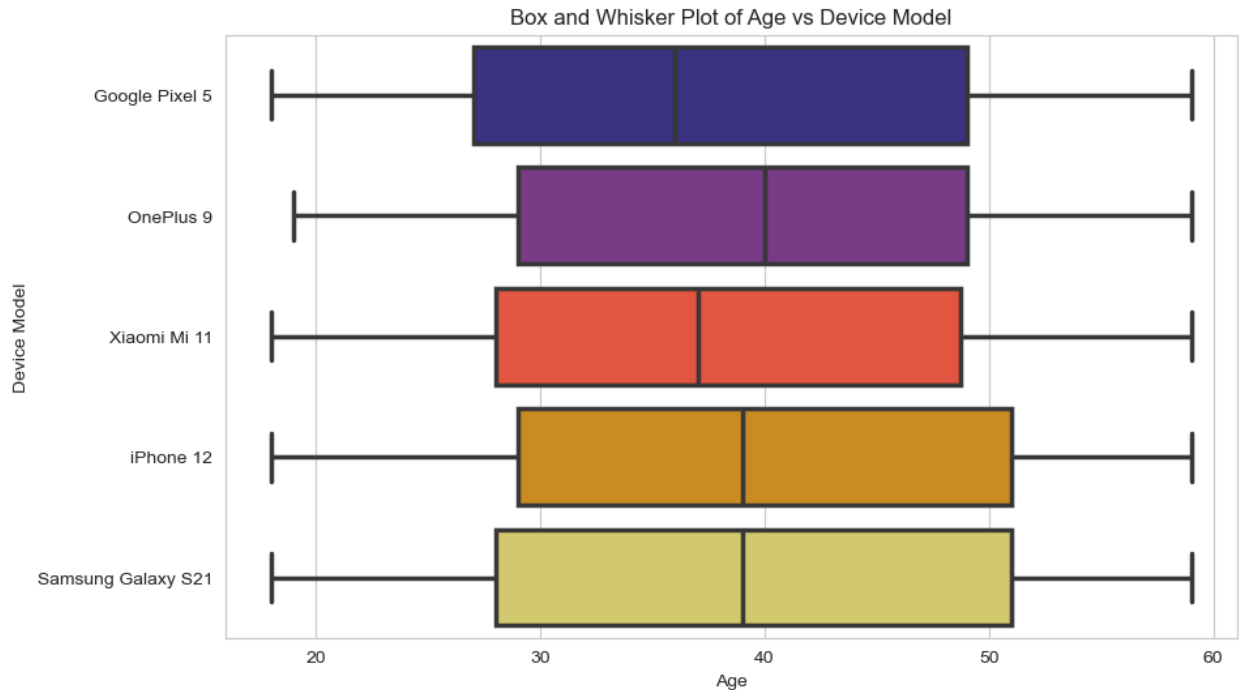
# Create bins
df_new['Age Group'] = pd.cut(df_new['Age'], bins=num_bins, labels=[f'Group {i}' for i in range(1, num_bins + 1)])

# Plot size
plt.figure(figsize=(10, 6)) # Adjust the size as needed

# Create the box plot
sns.boxplot(data=age_device_model_usage, x='Age', y='Device Model', palette='CMRmap', linewidth=2.5)

# Add titles and labels
plt.title('Box and Whisker Plot of Age vs Device Model')
plt.xlabel('Age')
plt.ylabel('Device Model')

# Show the plot
plt.show()
```



While we were happy with the look of the box plot and the data it gave was very straightforward it still leaves much room for the outliers discussion as much of our data particularly in the younger age groups of 20 to 30 and 50 to 60 ages were outside of the Lower quartiles and upper quartiles of the main median box plot where the main data was shown. After further research on Seaborn visual types we stumbled and agreed upon a box and whisker plot which in seaborns would be able to show us multiple upper quadrilateral and lower collateral and median's of the categories outside of the main box plot.

```
186]: #Device model vs Age
#Box and Whisker plot
# Define the number of bins for ages
num_bins = 5

# Create bins of equal width and Label them

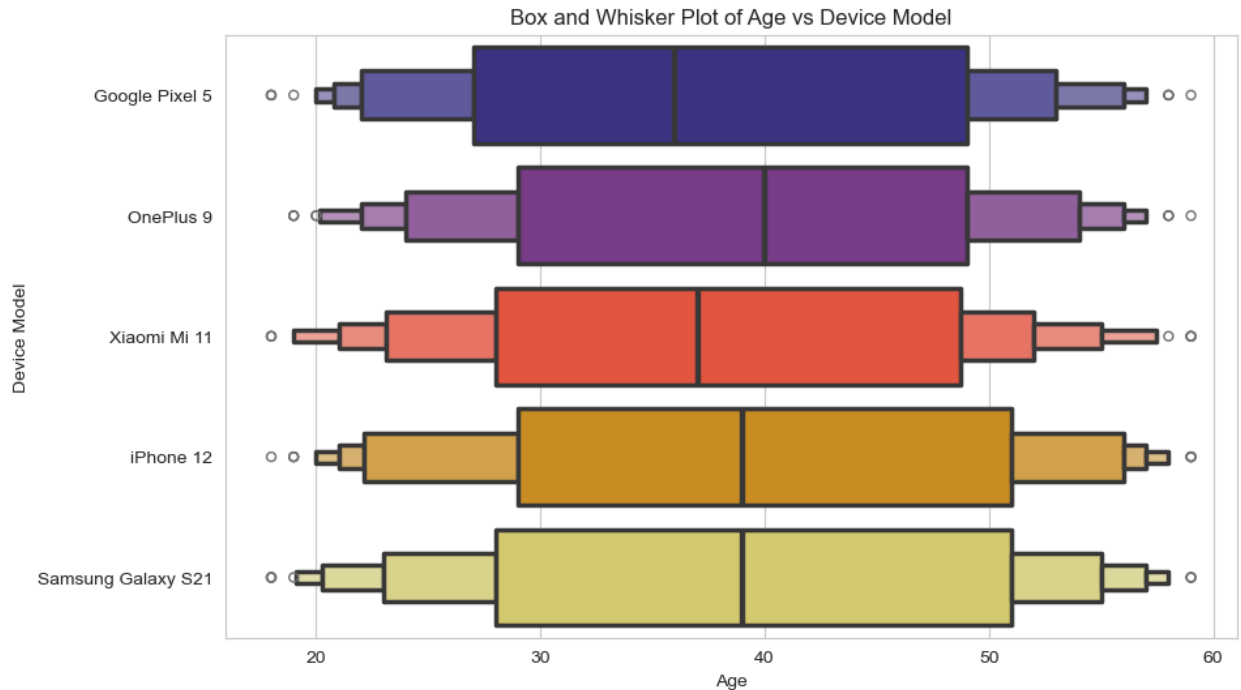
df_new['Age Group'] = pd.cut(df_new['Age'], bins=num_bins, labels=[f'Group {i}' for i in range(1, num_bins + 1)])

# Plot size
plt.figure(figsize=(10, 6)) # Adjust the size as needed

# Create the box whisker plot
sns.boxenplot(data=age_device_model_usage, x='Age', y='Device Model', palette='CMRmap', linewidth=2.5)

# Add titles and Labels
plt.title('Box and Whisker Plot of Age vs Device Model')
plt.xlabel('Age')
plt.ylabel('Device Model')

# Show the plot
plt.show()
```



Now with the box and whisker plot, we were able to visually see the mean as well as the lower and the upper quadrilaterals for each of the age categories as shown through their individual boxes on the whiskers. Looking at the main box we were generally looking at a trend of Ages 30 to 50 being the most represented through the mean. For example we can see the Google pixel 5 and Samsung Galaxy S21 having the greatest distance from the lower to the upper quadrilateral meaning it's the most age range for the device models for its popularity. Well not as great as our main plot we can see a significant number of those younger Between 20 and 30 had a greater age range for the Iphone 12 vs the other models as shown. The status show slightly there is preferential choice depending on your age for the phone models as shown above with younger people gravitating more towards Google pixel 5 being that the lower quadrilateral is starting around age 27 carrying all the way to age 50.

## Question 2

Is there any correlation with the amount of app usage measured in min/day and battery drain? Does having more apps installed affect battery drain?

The main focus here was battery drain and how it is affected by apps installed and the usage of the apps. The hypothesis we came up with was straightforward. More apps and more app usage time would result in more battery drain. The correlation in mind would be a positive regression. Before doing all that though we wanted to create two new data frames that we could

work with. The first was app usage time and battery drain. And the second being apps installed and battery drain.

```
battery_app_usage = ub2[["App Usage Time (min/day)", "Battery Drain (mAh/day)"]]  
battery_app_usage
```

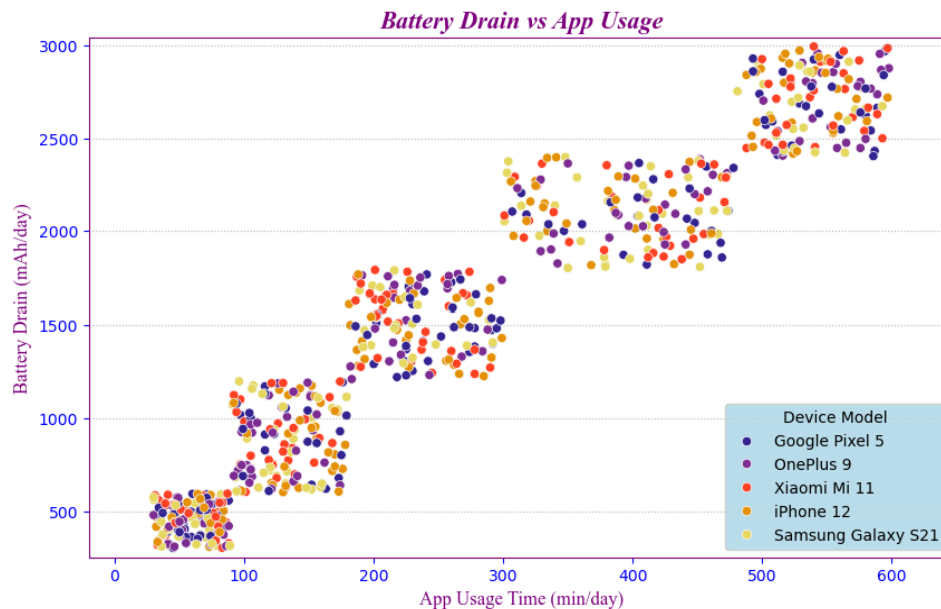
	App Usage Time (min/day)	Battery Drain (mAh/day)
0	393	1872
1	268	1331
2	154	761
3	239	1676
4	187	1367
...	...	...
695	92	1082
696	316	1965
697	99	942
698	62	431
699	212	1306

```
battery_app_installed = ub2[["Number of Apps Installed", "Battery Drain (mAh/day)"]]  
battery_app_installed
```

	Number of Apps Installed	Battery Drain (mAh/day)
0	67	1872
1	42	1331
2	32	761
3	56	1676
4	58	1367
...	...	...
695	26	1082
696	68	1965
697	22	942
698	13	431
699	49	1306

With these two data frames made we were now able to proceed with creating visualizations.

To start with was app usage time vs battery drain. Using a scatter plot as shown below we can see that it was quite uniformed at specific intervals and would keep its shape more or less when increasing app usage.



```
# Create scatterplot
plt.figure(figsize=(10, 6))
sns.scatterplot(x='App Usage Time (min/day)', y='Battery Drain (mAh/day)', data=ub2, palette='CMRmap', hue="Device Model")

# grid lines customization
plt.grid(True, axis='y', linestyle=':', color='gray', alpha=0.5)

# titles and labels
plt.title('Battery Drain vs App Usage',color="purple", fontsize=16, fontweight='bold', family='Times New Roman', style="italic")
plt.xlabel('App Usage Time (min/day)', color="purple", fontsize=12, family="Times New Roman")
plt.ylabel('Battery Drain (mAh/day)', color="purple", fontsize=12, family="Times New Roman")
plt.grid(True, axis='y')

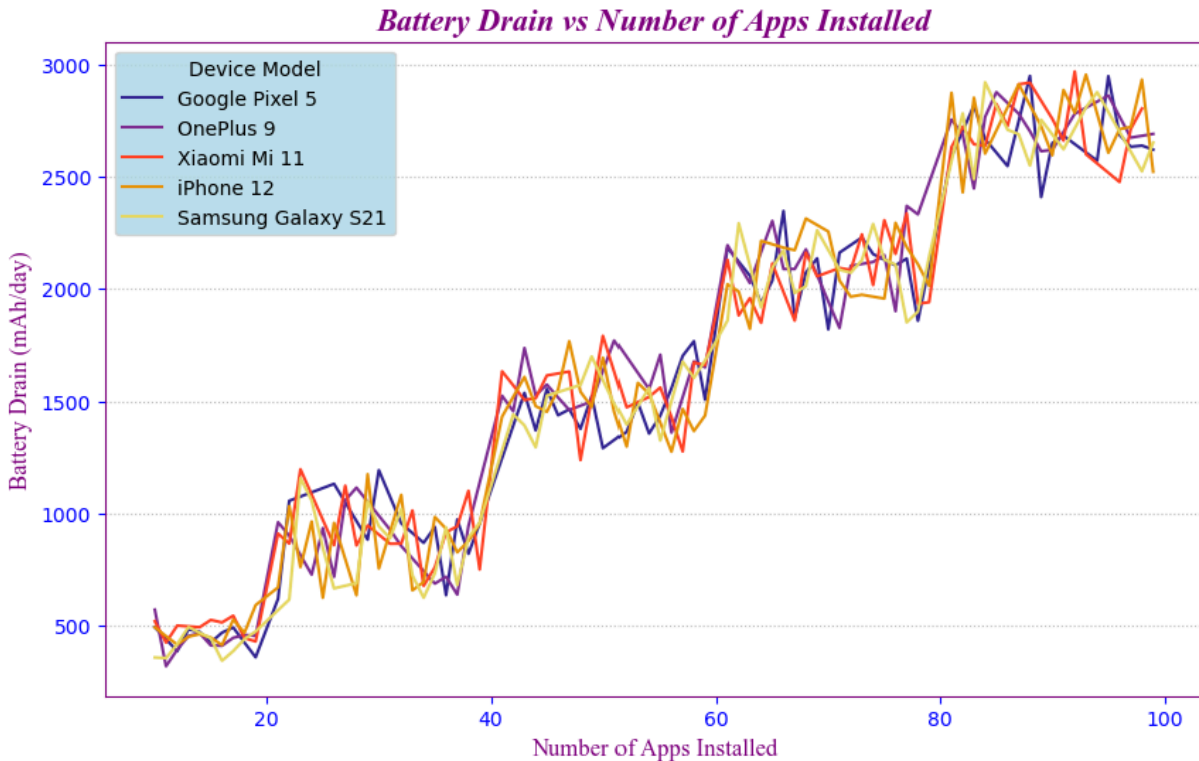
# axis colors
plt.tick_params(axis='x', labelcolor='blue', colors='blue')
plt.tick_params(axis='y', labelcolor='blue', colors='blue')

# axis border colors
plt.gca().spines['top'].set_color('purple')
plt.gca().spines['right'].set_color('purple')
plt.gca().spines['bottom'].set_color('purple')
plt.gca().spines['left'].set_color('purple')

plt.xlim(min(ub2['App Usage Time (min/day)'] - 50), max(ub2['App Usage Time (min/day)'] + 50))
plt.ylim(min(ub2['Battery Drain (mAh/day)'] - 50), max(ub2['Battery Drain (mAh/day)'] + 50))

# moving Legend hue
plt.legend(title="Device Model", loc='lower right', frameon=True, facecolor='lightblue')
```

Next up we had battery drain vs apps installed. With this we decided to use a linear plot.



```
# linear chart
plt.figure(figsize=(10, 6))
sns.lineplot(x='Number of Apps Installed', y='Battery Drain (mAh/day)', data=ub2, palette='CMRmap', hue="Device Model", errorbar=None, linewidth=1.5)

# grid lines customization
plt.grid(True, axis='y', linestyle=':', color='gray', alpha=0.5)

# color of axis
plt.tick_params(axis='x', labelcolor='blue', colors='blue')
plt.tick_params(axis='y', labelcolor='blue', colors='blue')

# color of axis border
plt.gca().spines['top'].set_color('purple')
plt.gca().spines['right'].set_color('purple')
plt.gca().spines['bottom'].set_color('purple')
plt.gca().spines['left'].set_color('purple')

# titles and labels
plt.title('Battery Drain vs Number of Apps Installed', color="purple", fontsize=16, fontweight='bold', family='Times New Roman', style="italic")
plt.xlabel('Number of Apps Installed', color="purple", fontsize=12, family="Times New Roman")
plt.ylabel('Battery Drain (mAh/day)', color="purple", fontsize=12, family="Times New Roman")
plt.grid(True, axis='y')

# moving legend hue
plt.legend(title="Device Model", loc='upper left', frameon=True, facecolor='lightblue')
```

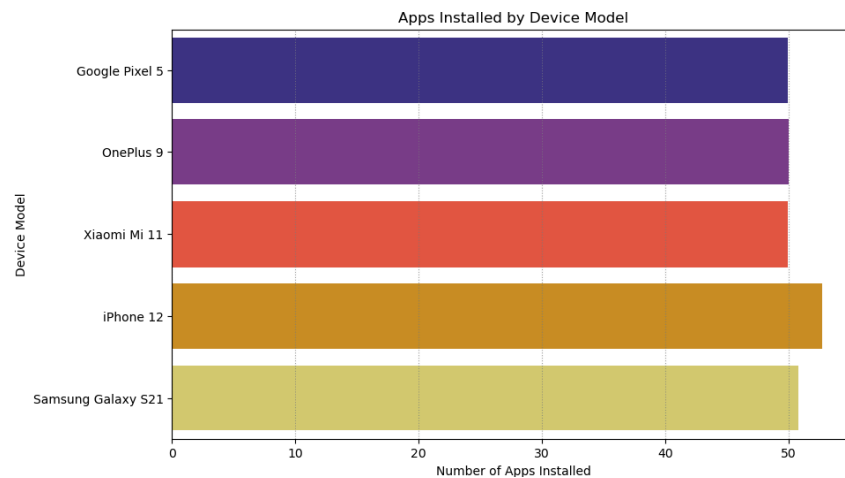
Again similar to the scatter plot the linear plot also shows the same trend, an increasing correlation.

Wanting to go deeper into the specifics and visualizations of device models with the apps installed and app usage we created two horizontal bar charts.

```
plt.figure(figsize=(10, 6))

plt.grid(True, axis='x', linestyle=':', color='gray', alpha=0.75)

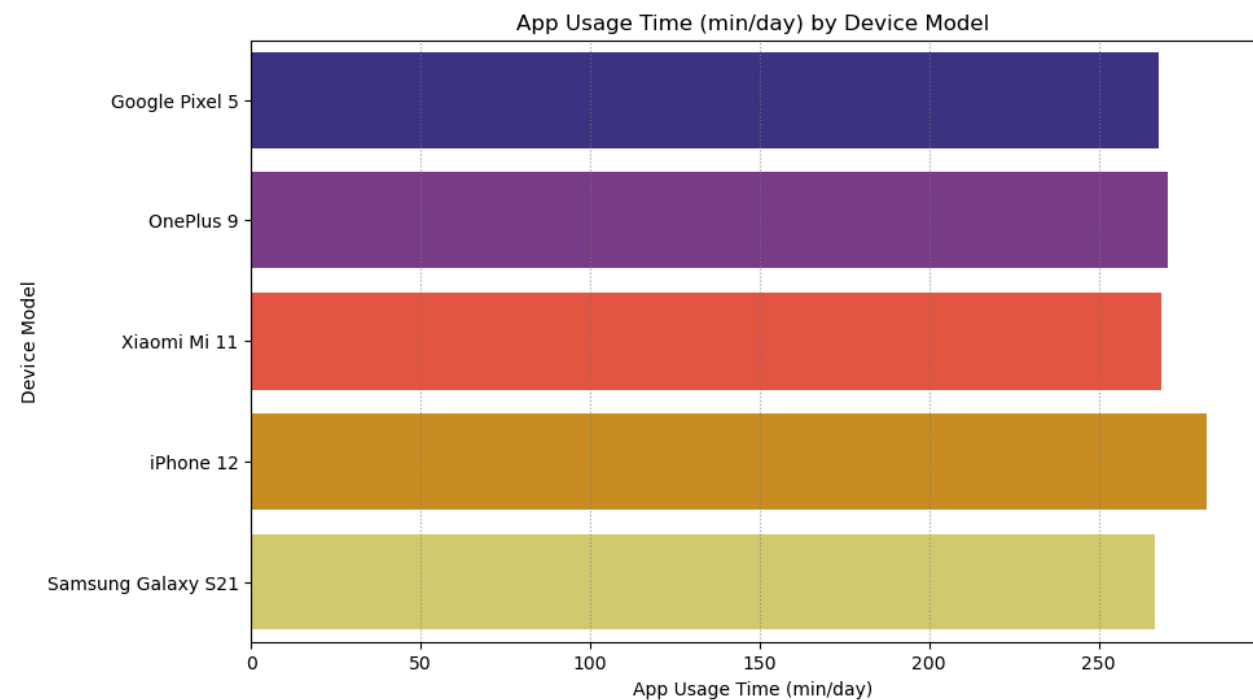
sns.barplot(y='Device Model', x="Number of Apps Installed", data=ub2, palette='CMRmap', errorbar=None, hue='Device Model', legend=False)
plt.title('Apps Installed by Device Model')
plt.xlabel('Number of Apps Installed')
plt.ylabel('Device Model')
plt.show
```



```
plt.figure(figsize=(10, 6))

plt.grid(True, axis='x', linestyle=':', color='gray', alpha=0.75)

sns.barplot(y='Device Model', x="App Usage Time (min/day)", data=ub2, palette='CMRmap', errorbar=None, hue='Device Model', legend=False)
plt.title('App Usage Time (min/day) by Device Model')
plt.xlabel('App Usage Time (min/day)')
plt.ylabel('Device Model')
plt.show
```





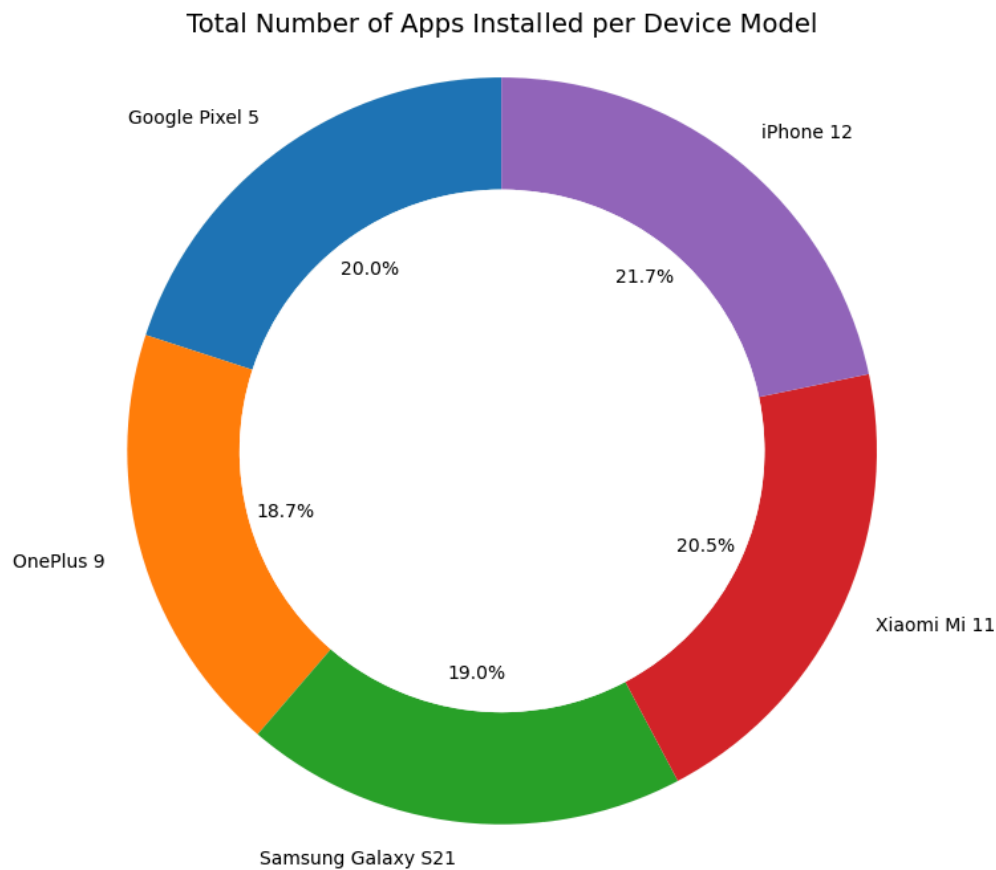
What we can see from above is that all the devices relatively were close in apps installed and app usage. The iPhone 12 notably was a bit more than the rest. Visualizing further using donut charts with average percentages we can also see down below how closely similar the device models were.

```
# Group by 'Device Model' and sum 'Number of Apps Installed'
app_installed = ub2.groupby('Device Model')['Number of Apps Installed'].sum()

# donut chart
plt.figure(figsize=(8, 8))
plt.pie(app_installed, labels=app_installed.index, autopct='%1.1f%%', startangle=90, wedgeprops={'width': 0.3})

centre_circle = plt.Circle((0,0), 0.70, color='white', fc='white', linewidth=0)
plt.gca().add_artist(centre_circle)

# title
plt.title('Total Number of Apps Installed per Device Model', fontsize=14)
plt.axis('equal')
plt.show()
```

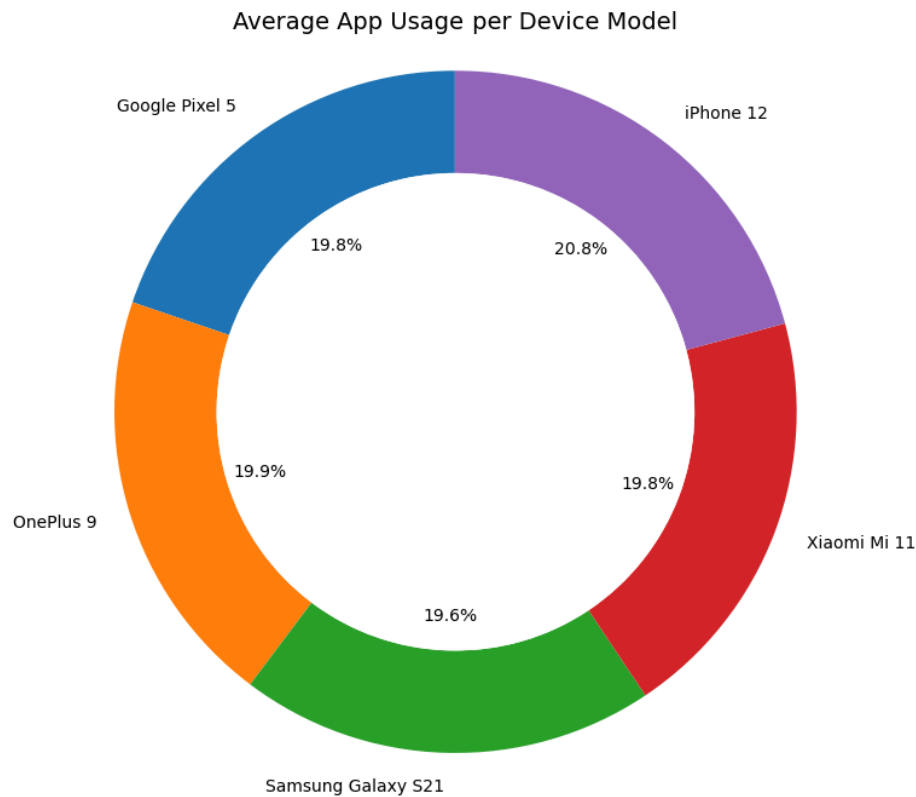


```
# Group by 'Device Model' and sum 'Number of Apps Installed'
app_usage = ub2.groupby('Device Model')['App Usage Time (min/day)'].mean().round()

# donut chart
plt.figure(figsize=(8, 8))
plt.pie(app_usage, labels=app_usage.index, autopct='%1.1f%%', startangle=90, wedgeprops={'width': 0.3})

centre_circle = plt.Circle((0,0), 0.70, color='white', fc='white', linewidth=0)
plt.gca().add_artist(centre_circle)

plt.title('Average App Usage per Device Model', fontsize=14)
plt.axis('equal')
plt.show()
```



While working on the average percentages with the donut charts we wanted to also add in a leaderboard of sorts that would show the means of not just the two targets in the question but all the columns we had.

	Rank	Device Model	App Usage Time (min/day)	Screen On Time (hours/day)	Battery Drain (mAh/day)	Number of Apps Installed	Data Usage (MB/day)	Age
0	1	Samsung Galaxy S21	266.4	5.3	1504.57	51.0	931.87	39.0
1	2	Google Pixel 5	267.8	5.1	1475.68	50.0	897.70	38.0
2	3	Xiaomi Mi 11	268.5	5.3	1528.88	50.0	940.16	38.0
3	4	OnePlus 9	270.4	5.2	1523.85	50.0	911.12	39.0
4	5	iPhone 12	282.0	5.4	1589.51	53.0	965.51	39.0

```

ub3 = pd.read_csv("user_behavior_data.csv")

# grouping by device model and calculating means
numeric_columns = ub3.select_dtypes(include=["number"]).columns
average_leaderboard = ub3.groupby("Device Model")[numeric_columns].mean()

# sorting by app usage time
average_leaderboard = average_leaderboard.sort_values("App Usage Time (min/day)", ascending=True)

# Reset index to make 'Device Model' a regular column
average_leaderboard.reset_index(inplace=True)

# adding rank column
average_leaderboard["Rank"] = average_leaderboard.index + 1

# saving leaderboard to new csv file
average_leaderboard.to_csv("average_leaderboard.csv", index=True)

pd.set_option("display.max_columns", None)
pd.set_option("display.width", 1000)
pd.set_option("display.max_colwidth", None)

# Dropping USER ID column
average_leaderboard = average_leaderboard.drop(columns=["User ID"])

# moving rank column before device model
columns_order = ['Rank'] + [col for col in average_leaderboard.columns if col != 'Rank']
average_leaderboard = average_leaderboard.reindex(columns=columns_order)

# rounding to nearest integer, 2 decimal or 1 decimal
average_leaderboard["Age"] = average_leaderboard["Age"].round()
average_leaderboard["Number of Apps Installed"] = average_leaderboard["Number of Apps Installed"].round()
average_leaderboard["Battery Drain (mAh/day)"] = average_leaderboard["Battery Drain (mAh/day)"].round(2)
average_leaderboard["Data Usage (MB/day)"] = average_leaderboard["Data Usage (MB/day)"].round(2)
average_leaderboard["Screen On Time (hours/day)"] = average_leaderboard["Screen On Time (hours/day)"].round(1)
average_leaderboard["App Usage Time (min/day)"] = average_leaderboard["App Usage Time (min/day)"].round(1)

display(average_leaderboard)

```

Pictured above we created a brand new data frame just showing the 5 device models and their averages across the board. With this we had a better understanding of just how close the means were with the devices.

### Question 3: Is there any correlation between the number of installed apps by gender and age?

The Null Hypothesis would be that there is no significant difference between Number of Apps Installed and gender or age.

With the easy availability, large selection and diversity of apps, and obvious socio-economic and cultural differences between age and gender it would be possible for there to be a significant difference in the number of installed apps for either set.

## Data Review

Some minor data engineering was performed of the main Data Frame which consisted of creating one new data frame with the only columns required for this analysis; 'Number of Apps Installed', 'Gender' and 'Age'

Figure 3.1 Apps, Gender Age Data Frame

```
# Extract seperate DF for Number of apps installed, age and gender
genderappdf = df[["Number of Apps Installed", "Age", "Gender"]]
genderappdf
```

	Number of Apps Installed	Age	Gender
0	67	40	Male
1	42	47	Female
2	32	42	Male
3	56	20	Male
4	58	31	Female
...	...	...	...
695	26	22	Male
696	68	59	Male
697	22	50	Female
698	13	44	Male
699	49	23	Female

700 rows × 3 columns

Three other data Frames were created afterwards for separate analysis of Number of apps installed by gender and another for analysis of Number of Apps installed by Age.

Figure 3.2 Number of Apps and Gender-Female

Number of Apps Installed	Age	Gender
335	99	45 Female
122	99	49 Female
80	99	51 Female
538	98	38 Female
165	98	49 Female
...	...	...
637	10	32 Female
458	10	58 Female
464	10	26 Female
583	10	43 Female
56	10	42 Female

336 rows × 3 columns

Figure 3.3 Number of Apps and Gender- Male

Number of Apps Installed	Age	Gender
381	99	39 Male
504	99	47 Male
434	99	58 Male
378	99	57 Male
125	99	50 Male
...	...	...
167	10	58 Male
213	10	22 Male
563	10	31 Male
289	10	50 Male
263	10	57 Male

364 rows × 3 columns

	Number of Apps Installed	Age
80	99	51
335	99	45
504	99	47
252	99	22
378	99	57
...	...	...
289	10	50
292	10	36
527	10	57
464	10	26
637	10	32

Figure 3.4 Number of Apps and Age

### Initial Analysis

A quick glance analysis was conducted by generating multiple charts for all of the main variables that will be analyzed, Number of Apps Installed, Gender (Male/Female) and Age. Three charts were created, A grouped bar chart (Fig. 4.1), a line chart (Fig. 4.2) and a Scatter plot (fig 1.2)

None of the charts were able to demonstrate any correlation and the scatter plot showed evenly spread out data points for all genders across all ages.

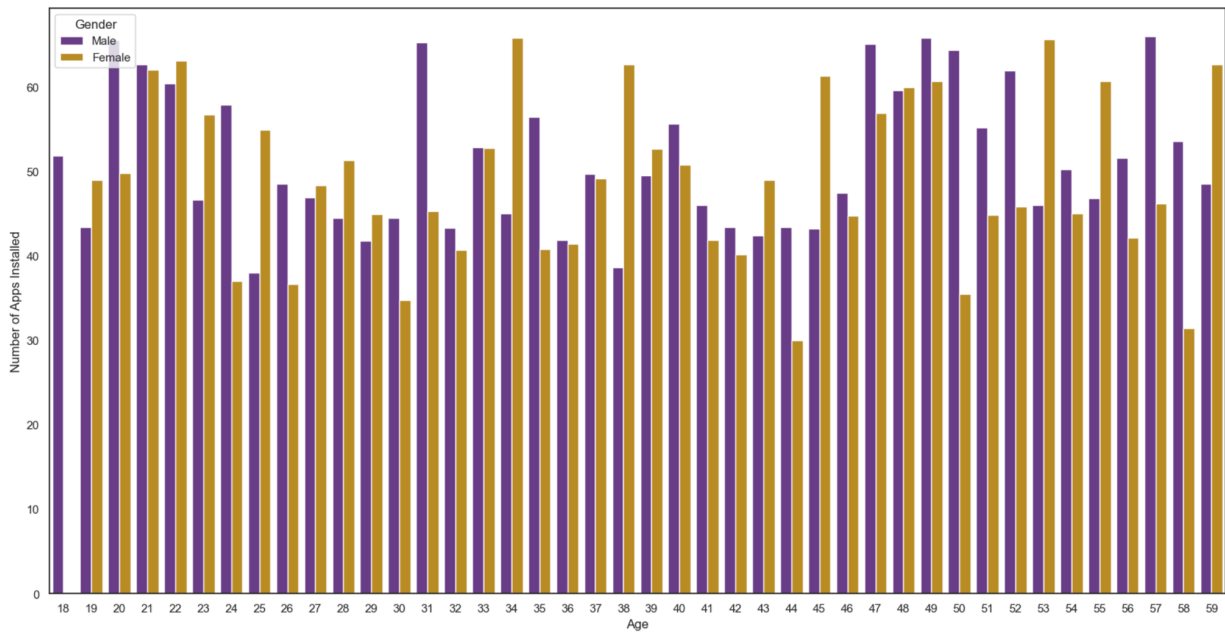


Figure 3.5 Grouped Bar Chart

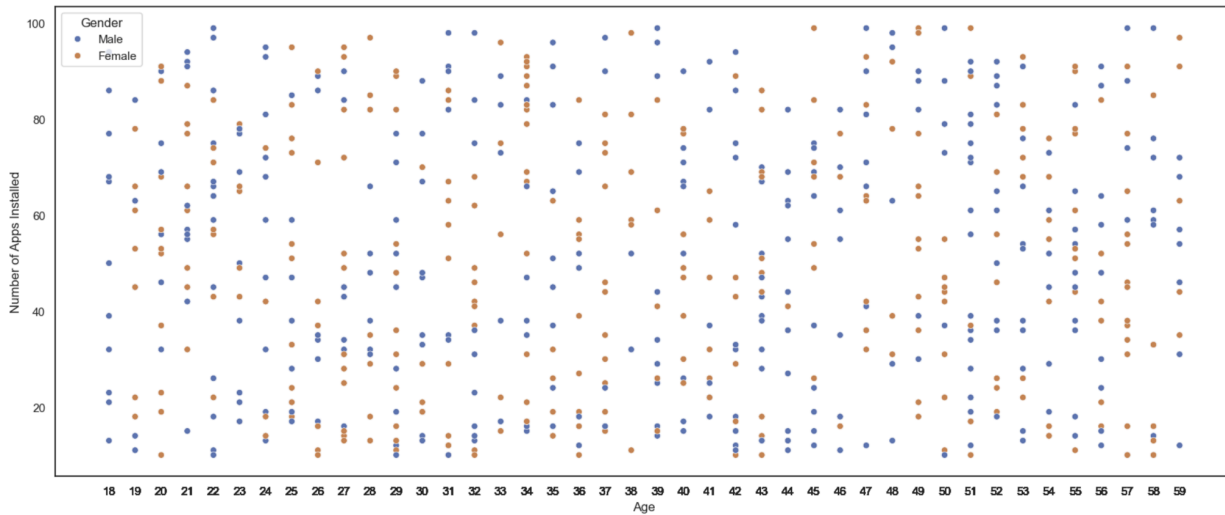


Figure 3.6 Scatterplot

The line chart did show an interesting pattern of inverse relationship in the number of apps installed by gender. Meaning, when males had a large

number of apps installed, females had a smaller number of apps installed and vice-versa. There was also an “Age” lag in the number of apps installed by gender. This means that when one gender had a high number of Apps installed, the other gender had a similar amount of apps installed but at an older age.

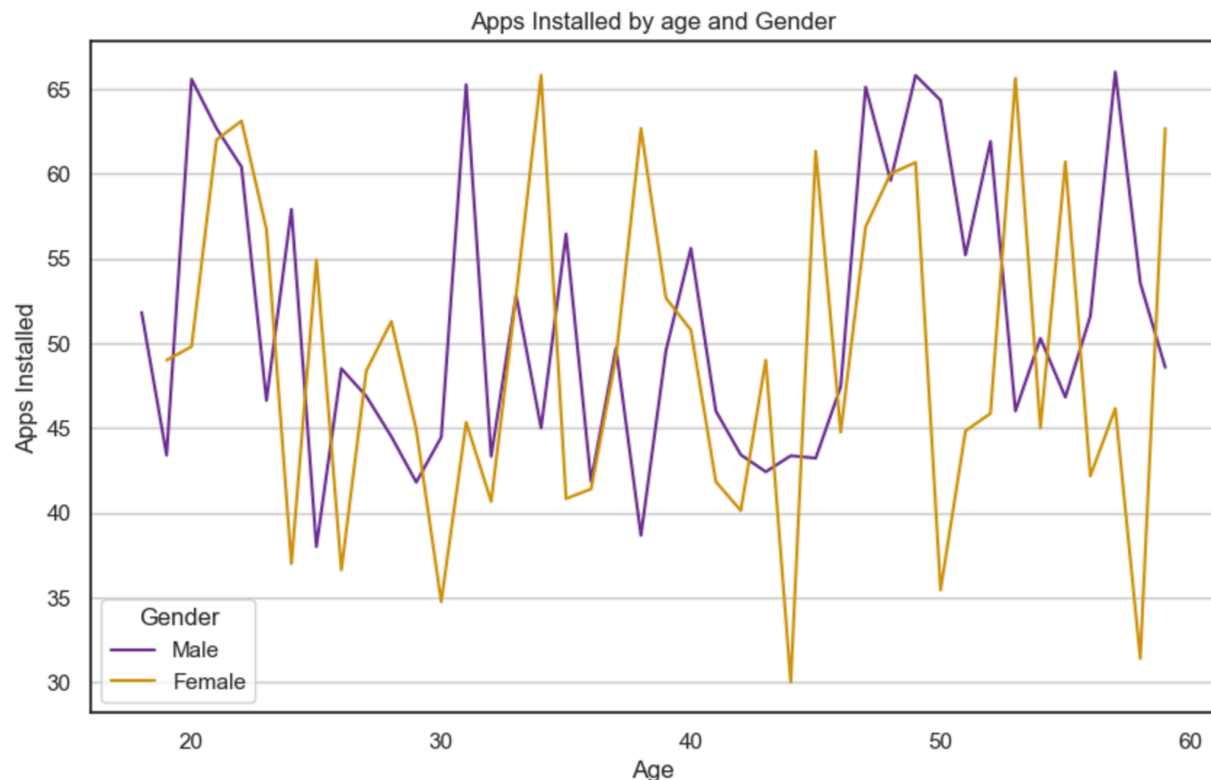


Figure 3.7 Line Chart

### 3.a Is there a correlation between the Number of Apps installed and Gender?

Based on the initial analysis using the previous charts, the answer is no; with several calculations performed to verify the conclusion. First a sum of the number of apps installed by each gender was done; with males having a total of 18601 and females a total of 16876 apps installed with a difference of 1725 more apps for males. Even though males had a higher number of apps installed, it doesn't determine if they use more apps overall than females since there were a higher number of males in the data set (364 males vs. 336 females, Fig 3.2 and fig 3.3) possibly



skewing the results. A mean calculation was also performed for each gender and number of apps with the results being:

Males: 51.10164835164835

Females: 50.226190476190474

With a percentage difference being 1.73%.

A P value was also performed with the result of: 0.667354455172253 or 66.7 %

Which was likely due to the small sample size and leading to no significant conclusion for the hypothesis.

```
# T test and p value
# Separate the groups
group_a = genderappdf[genderappdf["Gender"] == "Male"]["Number of Apps Installed"]
group_b = genderappdf[genderappdf["Gender"] == "Female"]["Number of Apps Installed"]

# Perform independent t-test
t_stat, p_value = sc.ttest_ind(group_a, group_b, equal_var=False)

print("t-statistic:", t_stat)
print("p-value:", p_value)

# Convert p value to percentage
p_value_percentage = p_value * 100
print("p-value (percentage): {:.2f}%".format(p_value_percentage))

t-statistic: 0.4299643757641013
p-value: 0.667354455172253
p-value (percentage): 66.74%
```

Figure 3.8 P and T test

```
# Sum number of Apps Installed by each gender
genderapp_sums = df.groupby('Gender')['Number of Apps Installed'].sum()
genderapp_sums
```

Gender	
Female	16876
Male	18601
Name: Number of Apps Installed, dtype: int64	

Figure 3.9 Sum of Apps by Gender

### 3.b Is there a correlation between Number of Installed Apps and Age?

Because of cultural and personal changes in peoples' lifestyle there is the possibility of changes in the number of installed apps vs person's age. This could be due to the need of certain apps during certain periods in a person's life or the amount of time they would have available to use more apps.

Unfortunately, analyzing the pair scatter plot chart and calculating the  $r$  value of installed apps with age no correlation was detected.

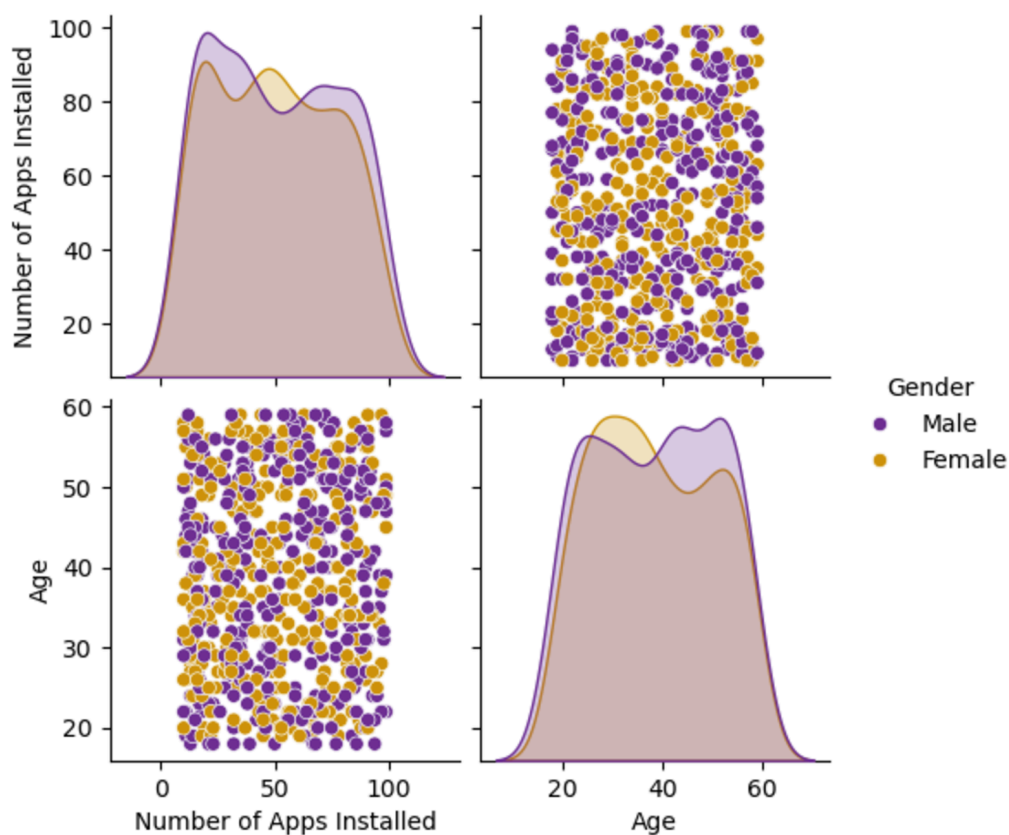


Figure 3.10 Pair Scatter Plot, Num. Apps vs Age

Viewing the scatter plot its visually clear that the Number of Apps Installed was pretty even throughout all ages, even through genders. The density map did show some differences between genders and age, older males having more apps than females and younger females having more apps than younger males, the total number of Apps installed was pretty even.

Figure 3.11 R value for correlation of Apps vs Age

A correlation calculation was performed between the number of Apps installed and age with a result of .4 % which means no correlation.

### 3. Conclusion

Several factors would have contributed to the observed results that led to the conclusion that there's no correlation in the number of apps installed and age or whether males or females have a significant difference in the Number of Installed apps. The small sample size for the data set and the slight uneven number of males vs females in the data set.

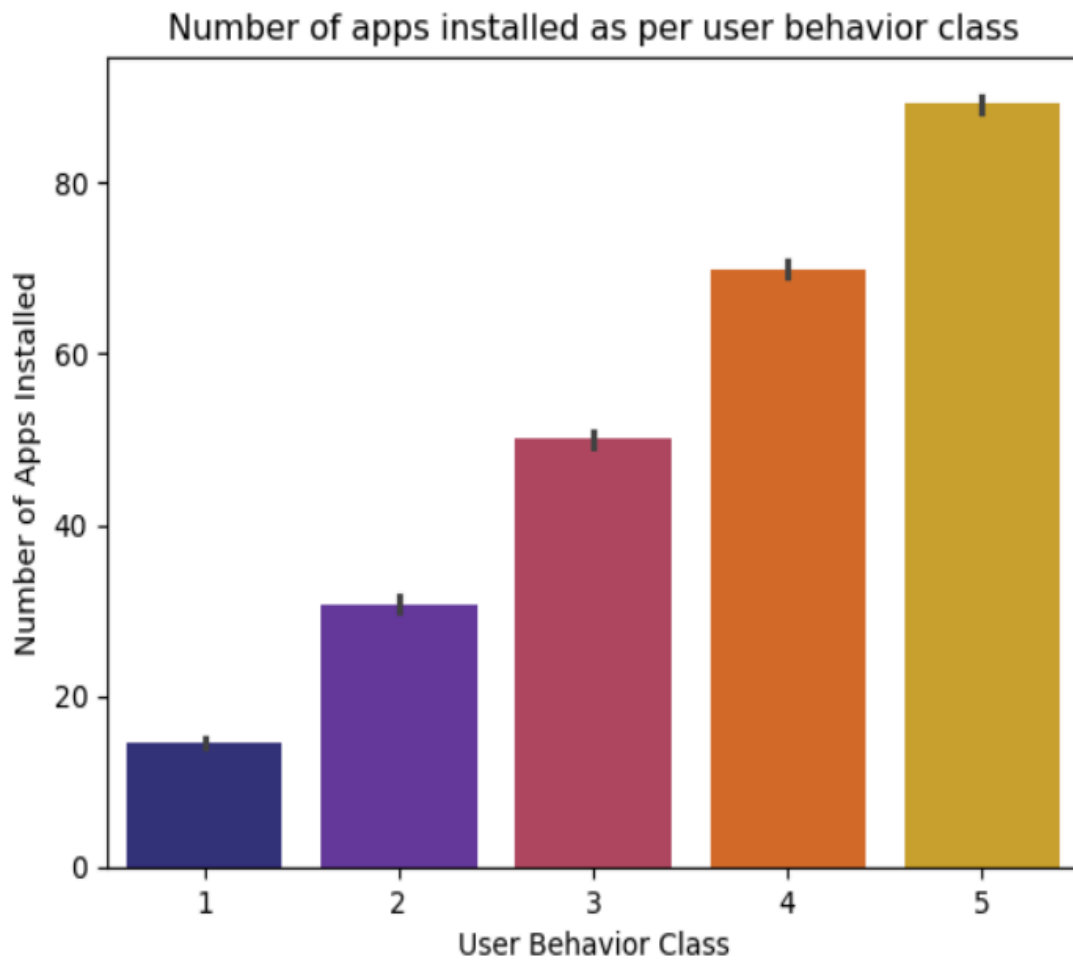
A larger data set with an even sample between gender would be needed to successfully answer these questions.

### User Behavior Class Undiscovered Data And More

As previously mentioned we did remove the User Behavior column due to lack of specificity. However, it is still quite interesting to see what could have been and what can come out of such limited data so here is a look.

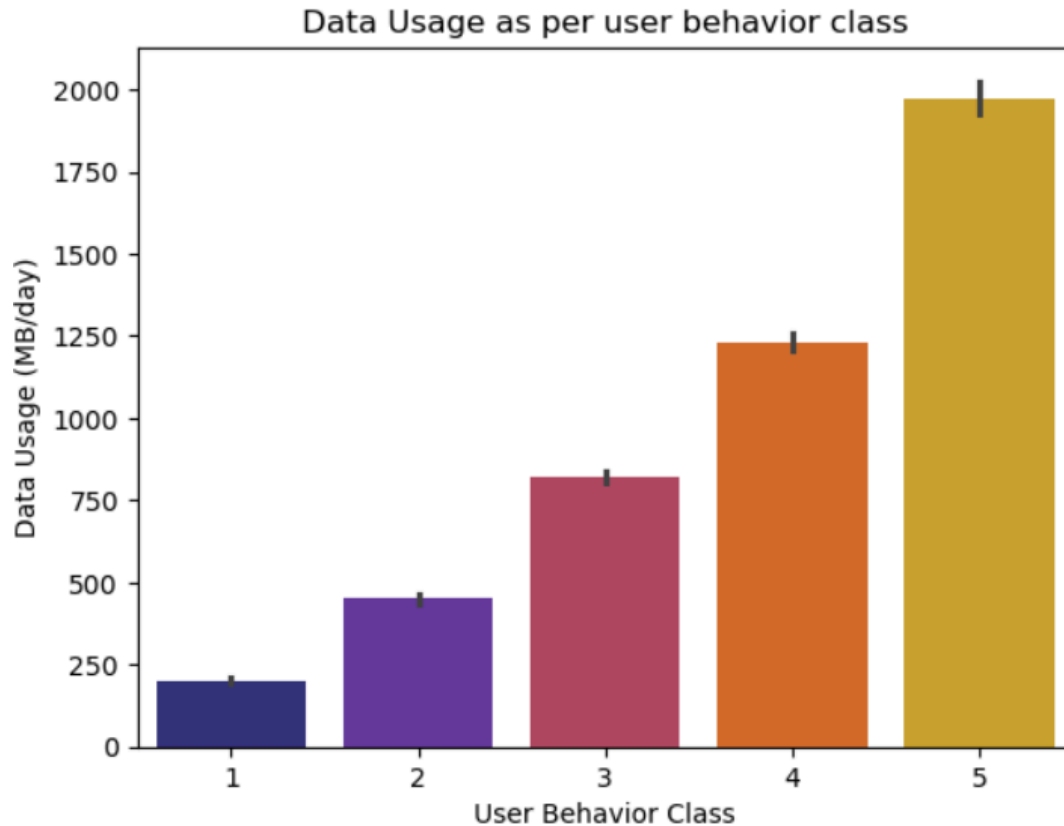
As the dataset's name suggests User Behavior Class, we tried to predict user behavior class based on the Number of Apps installed and Data Usage. Using column charts, we analyzed how user behavior class is classified based on the number of apps installed and Data Usage.

```
# Number of apps installed as per user behavior class
sns.barplot(x=df['User Behavior Class'],y=df['Number of Apps Installed'],palette=color
            )
plt.title('Number of apps installed as per user behavior class')
```



It is evident from the graph that the more the number of apps installed, the higher will be the user behavior class. Later, we also analyzed if the Data Usage feature affects User Behavior Class classification.

```
# Data Usage as per user behavior class
sns.barplot(x=df['User Behavior Class'], y=df['Data Usage (MB/day)'], palette=color)
plt.title('Data Usage as per user behavior class')
```

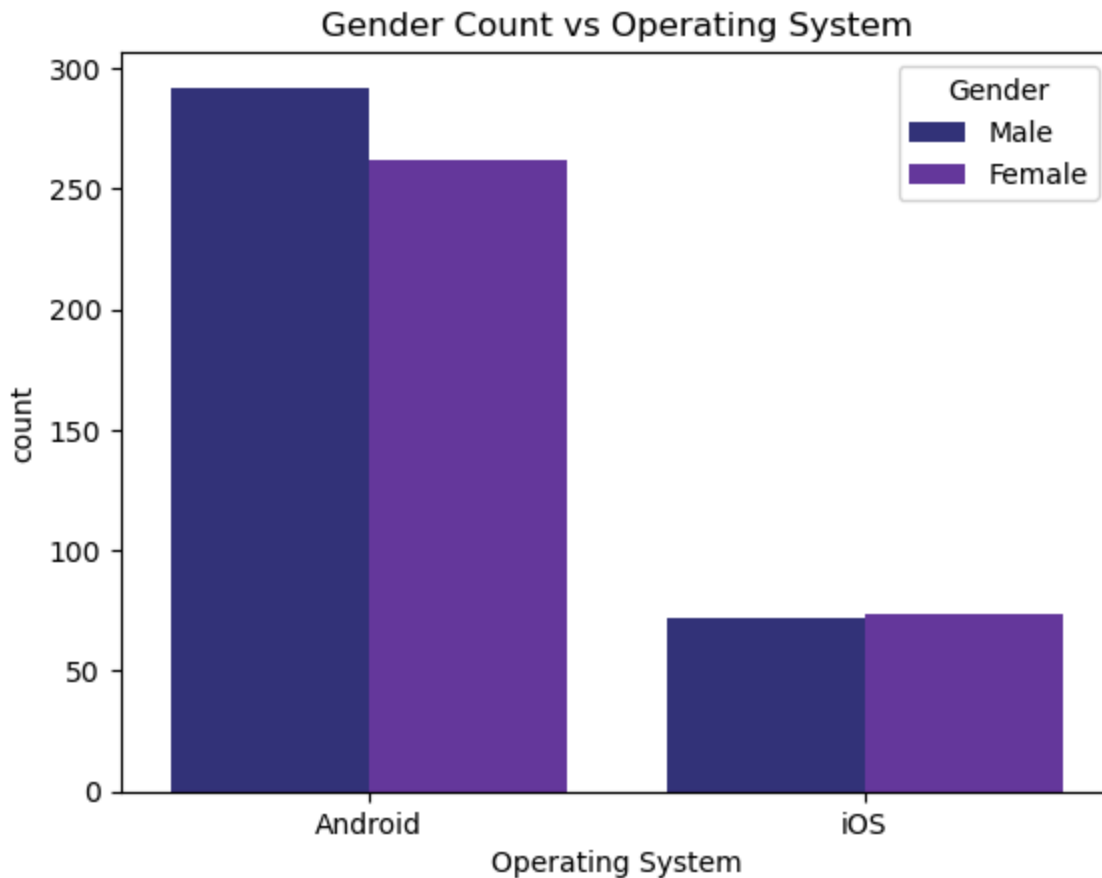


From the above graph, again it is visible that the higher the Data Usage, the higher will be User Behavior Class classification. Hence, from all our analysis it is evident that we can easily identify User Behavior Classes from various features of the dataset.

In conclusion these were some brief data visuals that came out of using User Behavior Class data moving forward we can focus our attention on Count Of Gender For Operating systems.

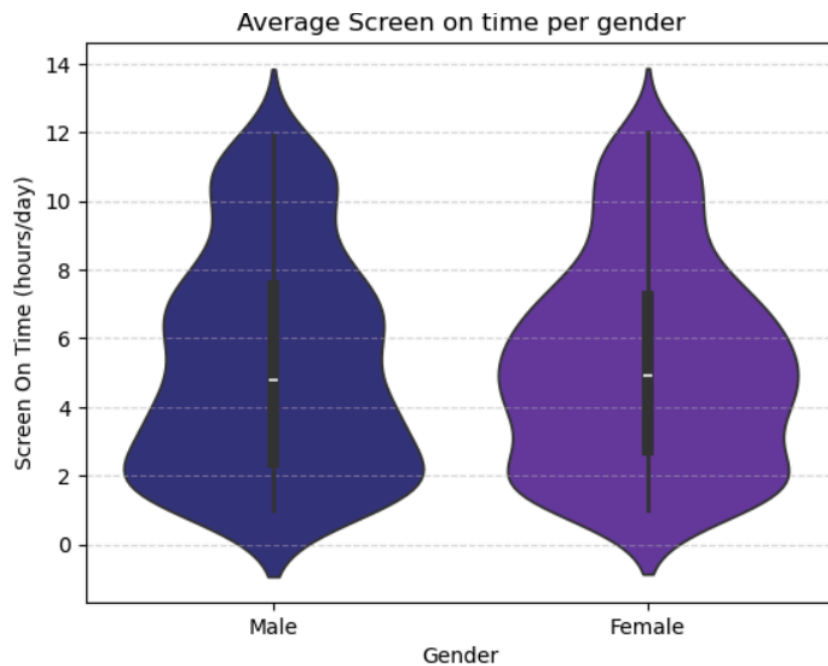
What we can begin to discover with this chart is the comparison between male and female and the operating systems they prefer and with that showcasing Android's user base is greater with both male, female compared to IOS. I have to add here is when the data got even more interesting if you look you can see for both Android and IOS that male users took the lead.

```
# Count of gender for operating systems
color=sns.color_palette(palette='CMRmap')
sns.countplot(data=df,x="Operating System",hue="Gender",palette=color)
plt.title("Gender Count vs Operating System")
```

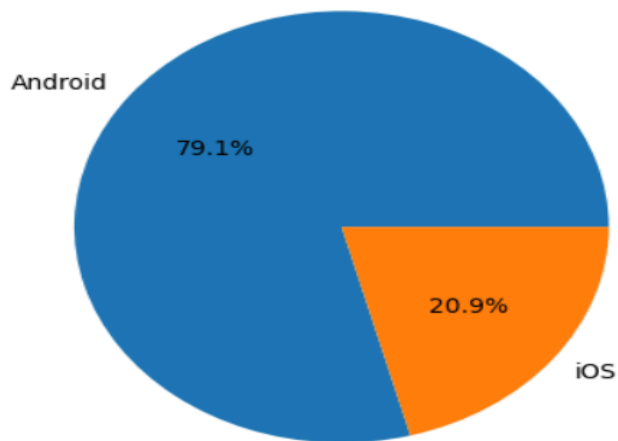


I want to acknowledge that during this time spent studying and understanding our dataset we have been able to come up with many combinations and correlations to provide a broader and more comprehensive way to view mobile device usage and user behavior. These two data displays will give just that of what I briefly talked about at the start of this minute publication.

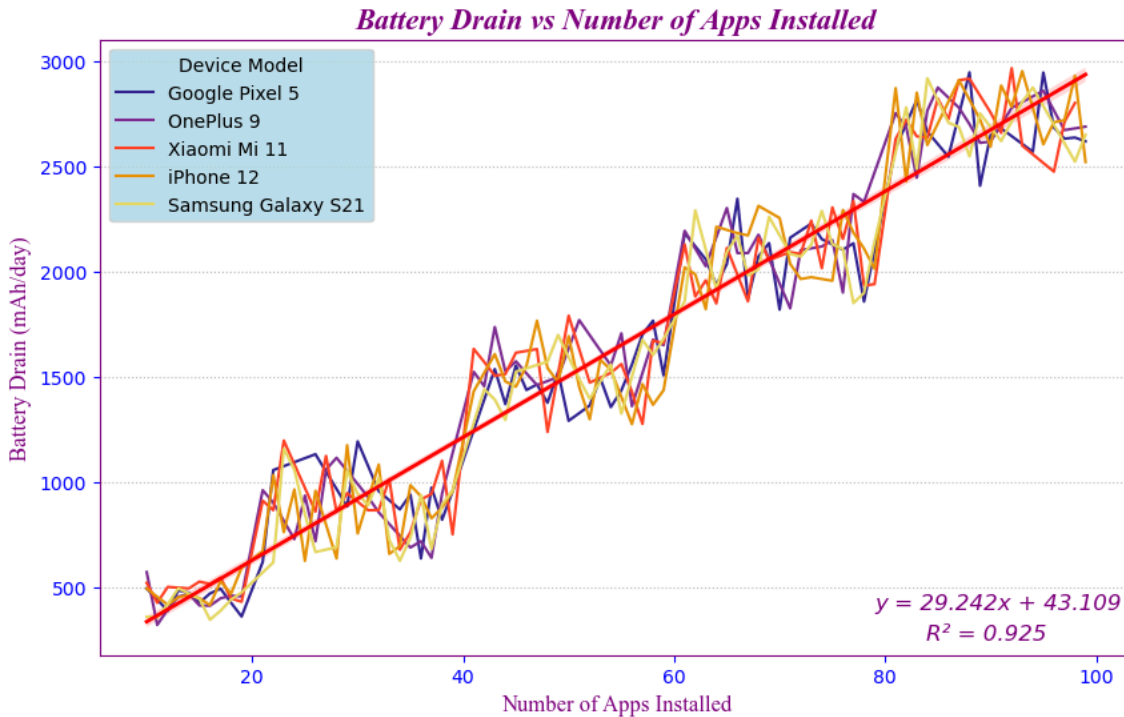
```
# Average screen on time per gender
sns.violinplot(x=df['Gender'],y=df['Screen On Time (hours/day)'],palette=color)
plt.title('Average Screen on time per gender')
plt.grid(True, axis='y', linestyle='--', alpha=0.5)
```



```
# Count of operating system
labels = df['Operating System'].value_counts().index
sizes = df['Operating System'].value_counts()
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.show()
```



What we have gathered from the current violin plot above is that it is trying to tell us android is the dominating primary software in this data is Android.



Pictured above we had also done a regression line on one of our previous charts. Shown is a positive regression and a strong relationship between the independent and dependent variable.

## Possible Future Research

Ways that we could improve this data which would have allowed us to expand and further dive deeper into mobile data usage is to track trends as new phone models with updated OS software become more efficient. We also would have wanted to see some data on the actual apps installed. Or the most common ones. This would better help with differences between variables such as gender and age. Grabbing a larger sample size or using sample sizes from multiple regions could also open data viewing from a map perspective. One final addition would be the capacity of batteries. If that was given some more specific data analysis could have been done when comparing the battery drain variable to others.



