**LOYOLA ACADEMY DEGREE & PG COLLEGE**

**(An Autonomous Degree College affiliated to Osmania University)**

**Alwal, Secunderabad -500010**

**Accredited with “A” grade by NAAC**

**A “College with Potential for Excellence” by UGC**

**2023-2025**

**MINI PROJECT REPORT**

**ON**

# “MOBILE DEVICE USAGE AND USER BEHAVIOR ANALYSIS”

**BY**

**B RAHUL SHARMA 111723720002**

**Department of MSc. Data Science**

# LOYOLA ACADEMY DEGREE & PG COLLEGE

(An autonomous Degree College affiliated to Osmania University)
Alwal, Secunderabad-500010

Accredited with “A” grade by NAAC
A COLLEGE WITH POTENTIAL FOR EXCELLENCE BY UGC

Project Report

This is to certify that this is the bonafide record of the mini project titled

“Mobile Device Usage and User Behavior Analysis”

Done during second year first semester for the completion of

MSc. Data Science Degree by

B RAHUL SHARMA UID:111723720002

Signature of Internal

Signature of HoD

Signature of External

Signature of Principal

# ACKNOWLEDGEMENT

"Task successful" makes everyone happy. But the happiness will gold without glitter if we didn't state the persons who have supported us to make a success.

Success will be crowned to people who made it reality but people who constant guidance and encouragement made it possible crowned first on the eve of success.

This acknowledgement transcends the reality of formality when we would like to express deep gratitude and respect to all those people behind the screen who guided, inspired and helped me for the completion of my project work.

I consider myself lucky enough to get such a project. This mini project would as an asset to my academic profile.

I express my thanks to our beloved principal **Rev Fr Dr N. B. Babu SJ**, who always inspire and motivated us to perform in a better and efficient manner.

I would like to express our thankfulness to my project guide **Mr. N Ashfaq, Head, Department of MSc. Data Science** for his constant motivation and valuable help through the project work, and I express my gratitude to, for his supervision, guidance and cooperation through the project.

B RAHUL SHARMA (111723720002)

# DECLARATION

This is to inform that I **B RAHUL SHARMA** the student of **MSc. Data Science** had completed the mini project work on “Mobile Device Usage and User Behavior Analysis” in the year 2024-25.

I have completed my mini project under the guidance of **Mr. N Ashfaq (HOD),** department of MSc Data Science.

I hereby declare that this mini project report submitted by me is the original work done by a part of my academic course and has not been submitted to any university or institution for the award of any degree or diploma.

B RAHUL SHARMA (UID: 111723720002)

# ABSTRACT

The prediction of user behavior based on mobile device usage is a critical area of research in behavioral analytics, helping businesses and developers understand usage patterns and optimize their services. Accurate forecasting of user behavior is essential for enhancing user experiences and developing personalized solutions. With advancements in machine learning, behavioral prediction models have become more robust and efficient. In this project, we propose a User Behavior Prediction Model leveraging data mining and machine learning techniques. The study is based on a real-world dataset collected from Kaggle containing features like App Usage Time, Screen On Time, Battery Drain, Number of Apps Installed, Data Usage, Age, and Gender, along with the target variable, User Behavior Class. I have applied four classification-based machine learning approaches to predict user behavior: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Support Vector Machine (SVM), KNeighborsClassifier. The performance of the proposed models is evaluated using metrics such as Accuracy, Precision, Recall, F1-Score, and the Confusion Matrix. Among the tested algorithms, Random Forest Classifier achieved the highest accuracy with accuracy score of 99%, demonstrating its capability to handle complex data patterns effectively. The results also indicate that features like App Usage Time**,** Screen on Time**,** and Battery Drain significantly influence user behavior predictions. To enhance usability, the framework includes a Graphical User Interface (GUI) designed using Tkinter, enabling users to input mobile device usage attributes and instantly receive predictions for the user behavior class. This user-friendly interface simplifies the application of the model for non-technical users. This project demonstrates the potential of machine learning techniques in user behavior prediction, offering a practical solution for understanding and forecasting behavioral patterns. Future improvements may include the integration of real-time data, analysis of external factors like user feedback and app categories, and expansion to predict additional behavioral traits such as user engagement and retention.

.

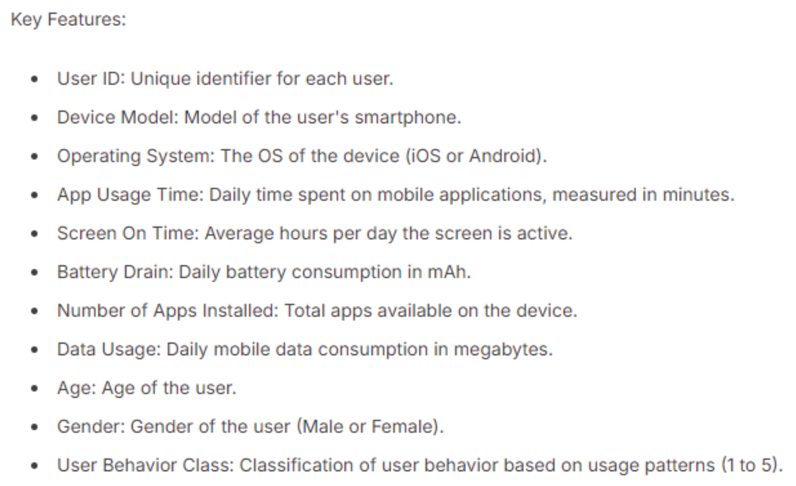
# CONTENTS

# I. INTRODUCTION

User behavior prediction is a compelling area of study due to the increasing reliance on mobile devices and the growing availability of data that captures user interactions and preferences. Understanding and forecasting user behavior can provide valuable insights for businesses and developers, enabling them to design better products, enhance user experiences, and implement targeted marketing strategies. Mobile usage patterns are influenced by various factors, such as app engagement, screen time, battery consumption, and demographic attributes like age and gender. The dynamic nature of user behavior, shaped by emerging technologies, social trends, and individual preferences, makes it a fascinating and challenging domain for predictive modeling.Predicting user behavior presents unique challenges due to the complexity and variability of usage patterns, as well as the influence of external factors like app popularity and technological advancements.

## 1.1. Data Description:

The dataset for this project includes data from Kaggle about Mobile Device Usage and User Behavior Analysis. So, totally I have 11 features with rows.



Key Features:

- User ID: Unique identifier for each user.
- Device Model: Model of the user's smartphone.
- Operating System: The OS of the device (iOS or Android).
- App Usage Time: Daily time spent on mobile applications, measured in minutes.
- Screen On Time: Average hours per day the screen is active.
- Battery Drain: Daily battery consumption in mAh.
- Number of Apps Installed: Total apps available on the device.
- Data Usage: Daily mobile data consumption in megabytes.
- Age: Age of the user.
- Gender: Gender of the user (Male or Female).
- User Behavior Class: Classification of user behavior based on usage patterns (1 to 5).

## 1.2. Important Libraries:

In this project, I have used various libraries to perform the tasks. Few libraries are NumPy, Pandas, Pickle, Sklearn, Matplotlib etc.

## 1.3. Exploratory Data Analysis:

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

## 1.4. Data Preprocessing:

Preprocessing refers to the transformations applied to our data before feeding it to the algorithm.

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. In this project that I have performed only one preprocessing task i.e. feature selection.

## 1.5. Algorithm Application:

The algorithm that I have applied for my project is Random Forest. Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

## 1.6. Pickling the File:

Python pickle module is used for serializing and de-serializing python object structures. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling. We can convert the byte stream (generated through pickling) back into python objects by a process called as unpickling. In real world scenario, the use pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database.

## 1.7. Deployment:

Machine learning deployment is the process of deploying a machine learning model in a live environment. The model can be deployed across a range of different environments and will often be integrated with apps through an API. Deployment is a key step in an organization gaining operational value from machine learning. For this project I deployed the model using tkinter package in python. Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

The dataset for this problem includes data from Kaggle about Mobile Device Usage and User Behavior Analysis . So, the dataset consists of 11 features which are described below.

# II. IMPORTANT LIBRARIES

In this project, I have used various libraries to perform the tasks. Following are the libraries that I have used in this project.

## 2.1. Pandas:

pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labelled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis/manipulation tool available in any language. It is already well on its way toward this goal.

The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2- dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

## 2.2. Numpy:

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open-source project

### 2.3. Scikit learn:

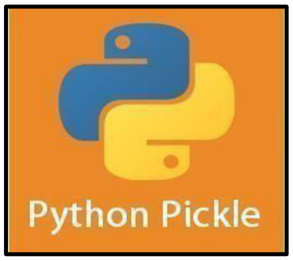
Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPyand Matplotlib.

### 2.4. Matplotlib:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

### 2.5. Pickle:

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.



## 2.6. Tkinter:

Python offers tkinter module to create graphic programs. The tkinter represents 'toolkit interface' for GUI. This is an interface for Python programmers that enable them to use the classes of TK module of TCL/TK language. Let's see what this TCL/TK is. The TCL (Tool Command Language) is a powerful dynamic programming language, suitable for web and desktop applications, networking, administration, testing and many more. It is open source and hence can be used by any one freely. TCL language uses TK (Tool Kit) language to generate graphics. TK provides standard GUI not only for TCL but also for many other dynamic programming languages like Python. Hence, this TK is used by Python programmers in developing GUI applications through Python's tkinter module.

Tkinter

## 2.7. Seaborn:

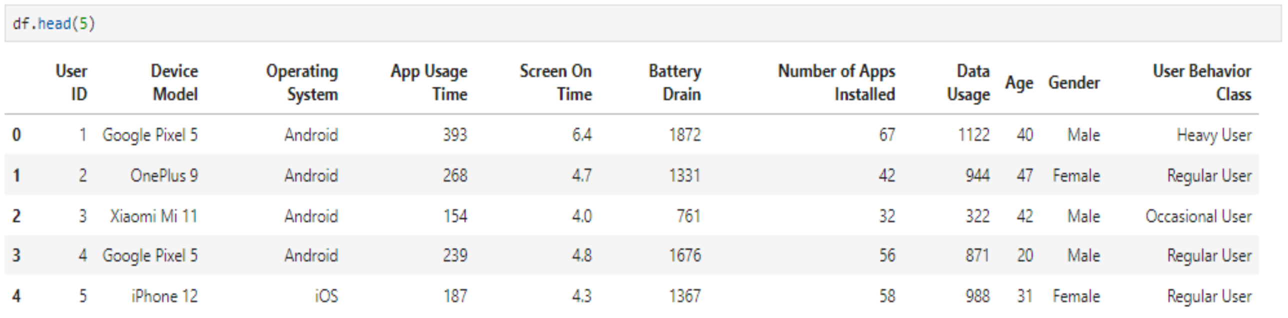
Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and colour palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representationsfor same variables for better understanding of dataset.

seaborn

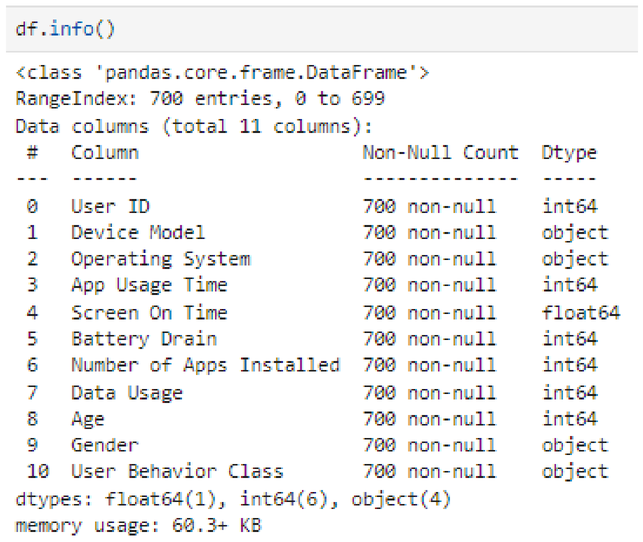
# III. DATASET DESCRIPTION

This dataset is taken from the Kaggle website. he objective of the dataset is to classify users into specific behavior categories, enabling a deeper understanding of how individuals interact with mobile devices. Several factors were considered when selecting the data, including daily app usage time, screen on time, battery consumption, the number of apps installed, and data usage. The dataset includes multiple independent variables (such as App Usage Time**,** Screen On Time**,** Battery Drain**,** Number of Apps Installed, and Data Usage) and one dependent variable (User Behavior Class), which serves as the target variable for classification.



```
df.head(5)
```

| | User ID | Device Model | Operating System | App Usage Time | Screen On Time | Battery Drain | Number of Apps Installed | Data Usage | Age | Gender | User Behavior Class |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Google Pixel 5 | Android | 393 | 6.4 | 1872 | 67 | 1122 | 40 | Male | Heavy User |
| **1** | 2 | OnePlus 9 | Android | 268 | 4.7 | 1331 | 42 | 944 | 47 | Female | Regular User |
| **2** | 3 | Xiaomi Mi 11 | Android | 154 | 4.0 | 761 | 32 | 322 | 42 | Male | Occasional User |
| **3** | 4 | Google Pixel 5 | Android | 239 | 4.8 | 1676 | 56 | 871 | 20 | Male | Regular User |
| **4** | 5 | iPhone 12 | iOS | 187 | 4.3 | 1367 | 58 | 988 | 31 | Female | Regular User |



```
df.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              700 non-null    int64  
 4   Screen On Time              700 non-null    float64
 5   Battery Drain               700 non-null    int64  
 6   Number of Apps Installed    700 non-null    int64  
 7   Data Usage                  700 non-null    int64  
 8   Age                         700 non-null    int64  
 9   Gender                      700 non-null    object 
 10  User Behavior Class         700 non-null    object 
dtypes: float64(1), int64(6), object(4)
memory usage: 60.3+ KB
```

# IV. EXPLORATORY DATA ANALYSIS

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

EDA is primarily used to see what data can reveal beyond the formal modelling or hypothesis testing task and provides a provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate. Originally developed by American mathematician John Tukey in the 1970s, EDA techniques continue to be a widely used method in the data discovery process today.

## 4.1. Why is exploratory data analysis important in data science?

The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

Data scientists can use exploratory analysis to ensure the results they produce are valid and applicable to any desired business outcomes and goals. EDA also helps stakeholders by confirming they are asking the right questions. EDA can help answer questions about standard deviations, categorical variables, and confidence intervals. Once EDA is complete and insights are drawn, its features can then be used for more sophisticated data analysis or modelling, including machine learning.

## 4.2. Data Visualization:

Data visualization is defined as a graphical representation that contains the information and the data. By using visual elements like charts, graphs, and maps, data visualization techniques provide an accessible way to see and understand trends, outliers, and patterns in data.

In modern days we have a lot of data in our hands i.e., in the world of Big Data, data visualization tools, and technologies are crucial to analyze massive amounts of information and make data- driven decisions.

It is used in many areas such as:

- To model complex events.
- Visualize phenomenon's that cannot be observed directly, such as weather patterns, medical conditions, or mathematical relationships.

## 4.3. Bar plots:

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other axis represents the measured values corresponding to those categories.

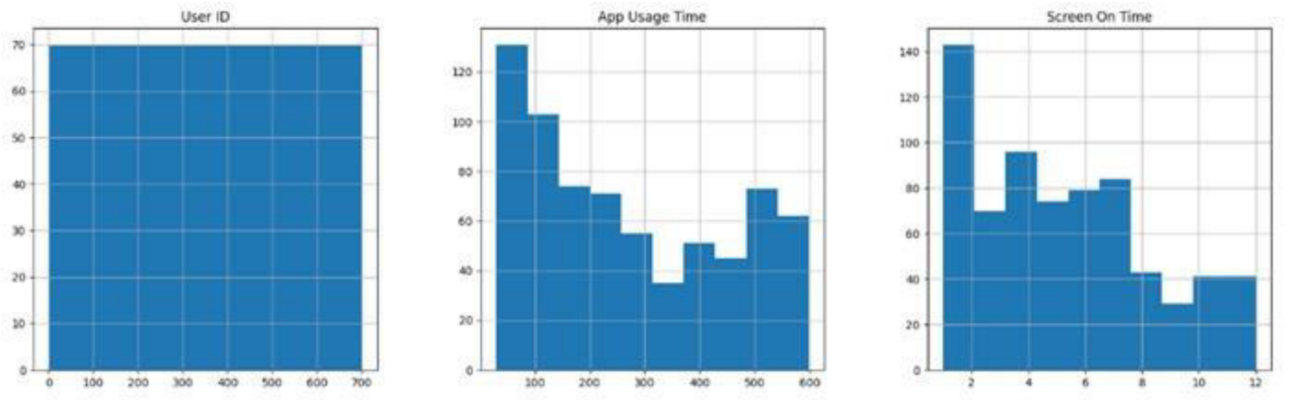
## 4.4. Bar plots Representation

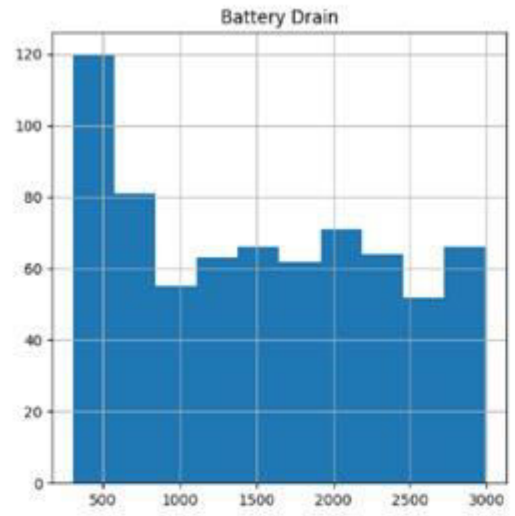
Bar graphs are the pictorial representation of data (generally grouped), in the form of vertical or horizontal rectangular bars, where the length of bars are proportional to the measure of data. They are also known as bar charts. Bar graphs are one of the means of data handling in statistics.

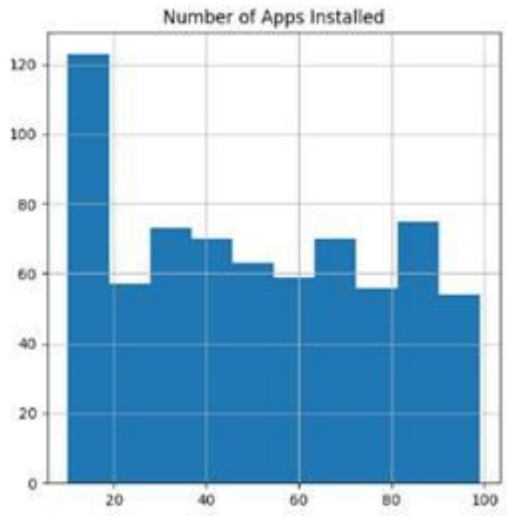
Usually, Histograms represent organized data into groups.

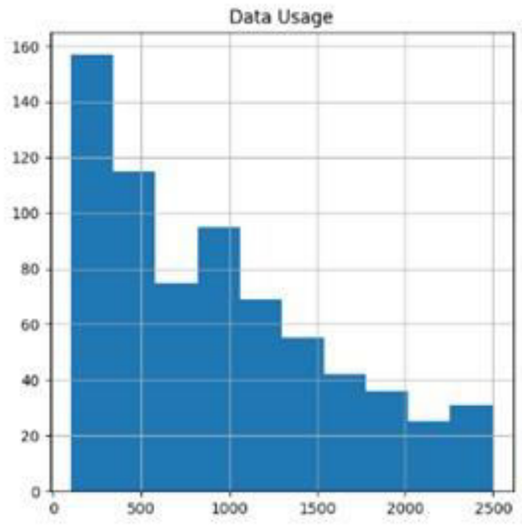
- Firstly, the X-axis represents bin ranges
- Secondly, the Y-axis represents frequency.

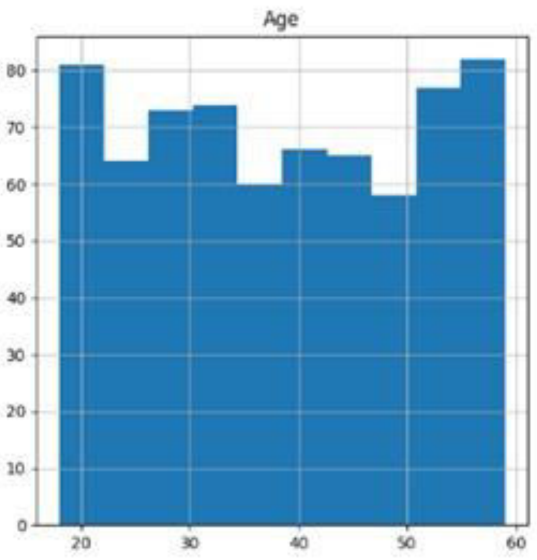
## 4.5. Visualization using Bar plots:




Battery Drain
0
20
40
60
80
100
120
500
1000
1500
2000
2500
3000


Number of Apps Installed
0
20
40
60
80
100
120
20
40
60
80
100


Data Usage
0
20
40
60
80
100
120
140
160
0
500
1000
1500
2000
2500


Age
0
10
20
30
40
50
60
70
80
20
30
40
50
60

## V. DATA PREPROCESSING

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data preprocessing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

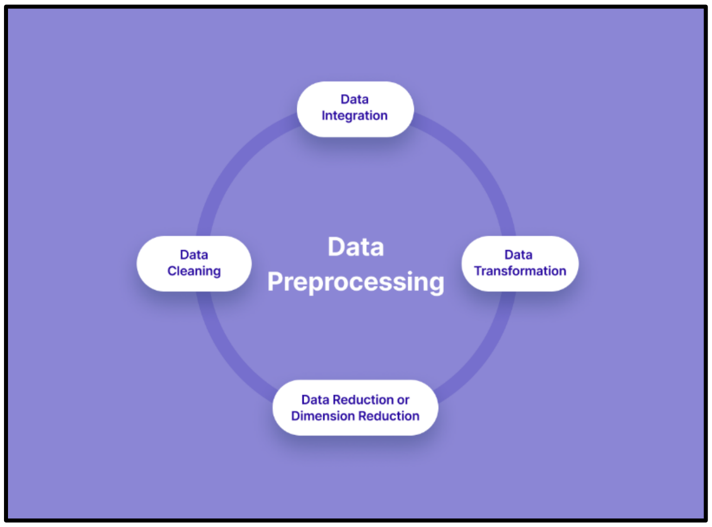
The majority of the real-world datasets are highly susceptible to missing, inconsistent, and noisy data due to their heterogeneous origin.

Applying data mining algorithms on this noisy data would not give quality results as they would fail to identify patterns effectively. Data Processing is, therefore, important to improve the overall data quality.

Duplicate or missing values may give an incorrect view of the overall statistics of data.

Outliers and inconsistent data points often tend to disturb the model’s overall learning, leading to false predictions.

Quality decisions must be based on quality data. Data Preprocessing is important to get this quality data, without which it would just be a Garbage In, Garbage Out scenario.



## 5.1. Feature Selection:

A feature is an attribute that has an impact on a problem or is useful for the problem, and choosing the important features for the model is known as feature selection. Each machine learning process depends on feature engineering, which mainly contains two processes; which are Feature Selection and Feature Extraction. Although feature selection and extraction processes may have the same objective, both are completely different from each other. The main difference between them is that feature selection is about selecting the subset of the original feature set, whereas feature extraction creates new features. Feature selection is a way of reducing the input variable for the model by using only relevant data in order to reduce overfitting in the model.

So, we can define feature Selection as, "It is a process of automatically or manually selecting the subset of most appropriate and relevant features to be used in model building." Feature selection is performed by either including the important features or excluding the irrelevant features in the dataset without changing them.

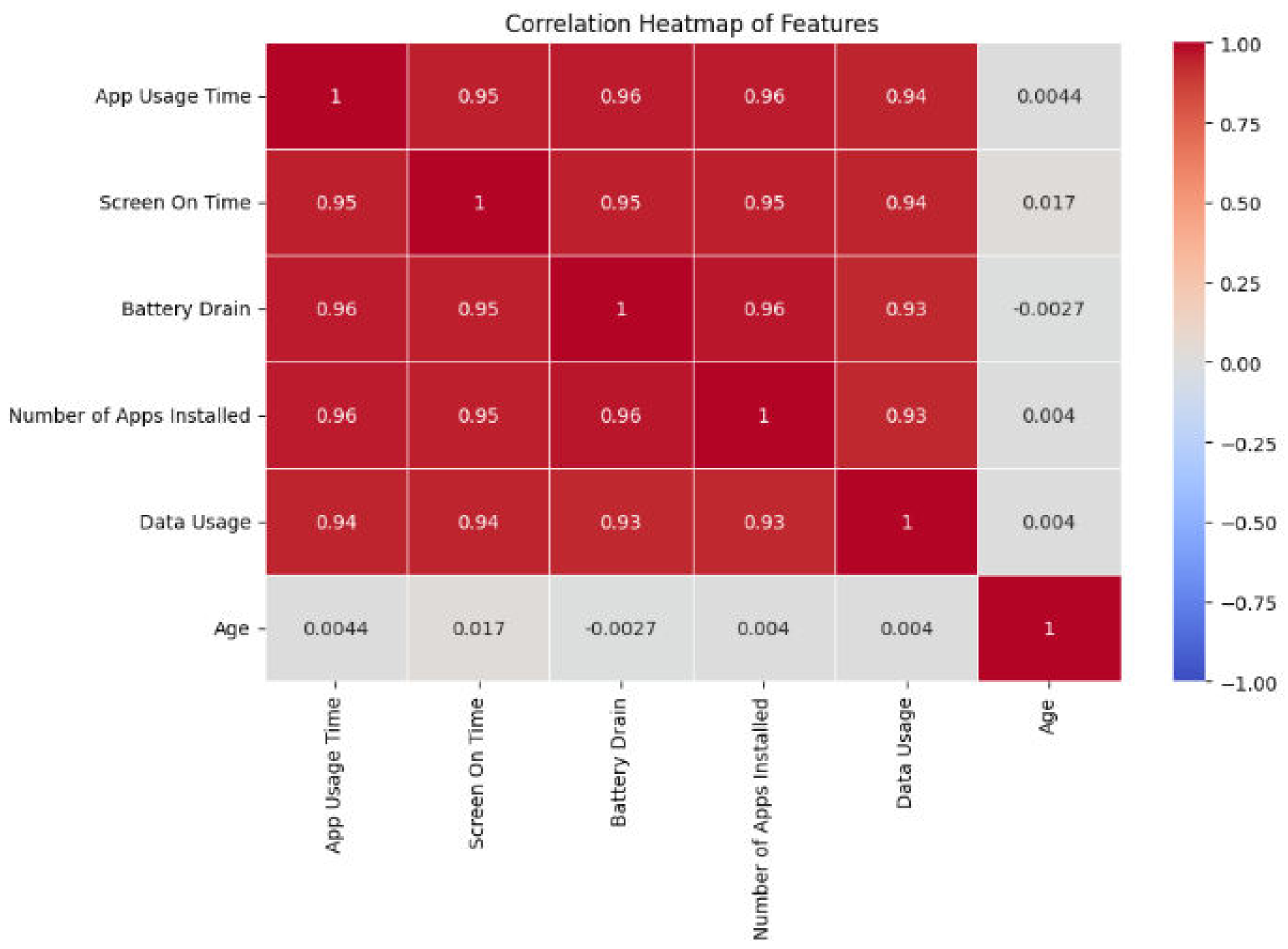
Among Different methods to apply feature selection, we have selected the below method are discussed below

## 5.2. Filter methods

Filter methods pick up the intrinsic properties of the features measured via univariate statistics instead of cross-validation performance. These methods are faster and less computationally expensive than wrapper methods. When dealing with high-dimensional data, it is computationally cheaper to use filter methods.

Methods that come under this are

- Information Gain
- Chi-square Test
- Fisher’s Score
- Correlation Coefficient
- Variance Threshold
- Mean Absolute Difference (MAD)
- Dispersion ratio



|  | App Usage Time | Screen On Time | Battery Drain | Number of Apps Installed | Data Usage | Age |
|---|---|---|---|---|---|---|
| App Usage Time | 1 | 0.95 | 0.96 | 0.96 | 0.94 | 0.0044 |
| Screen On Time | 0.95 | 1 | 0.95 | 0.95 | 0.94 | 0.017 |
| Battery Drain | 0.96 | 0.95 | 1 | 0.96 | 0.93 | -0.0027 |
| Number of Apps Installed | 0.96 | 0.95 | 0.96 | 1 | 0.93 | 0.004 |
| Data Usage | 0.94 | 0.94 | 0.93 | 0.93 | 1 | 0.004 |
| Age | 0.0044 | 0.017 | -0.0027 | 0.004 | 0.004 | 1 |

# VI. ALGORITHM APPLICATION

Algorithm application is an important part for any machine learning projects. Choosing the right algorithm based on the problem statement and the data gathered is one of the major steps to be taken. As a beginner we might not know the right algorithm at once. So, we need to try different algorithms on our data. The algorithm which performs good and produces the good accuracy score can be chosen.

So, in my project the algorithm that I have chosen is Random Forest, which is a supervised learning algorithm and can be applied for both regression and classification type.

Before we apply any algorithm to our data, we need to split the data into training and testing part. It is an essential step for all supervised algorithms.

Let us get to know what does actually training and testing data means.

## 6.1. Train-Test Split Evaluation:

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

- **Train Dataset:** Used to fit the machine learning model.
- **Test Dataset:** Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values.

The train-test procedure is appropriate when there is a sufficiently large dataset available.

## 6.2. How to Configure the Train-Test Split

The procedure has one main configuration parameter, which is the size of the train and test sets. This is most commonly expressed as a percentage between 0 and 1 for either the train or 62 | Pag e test datasets. For example, a training set with the size of 0.67 (67 percent) means that the remainder percentage 0.33 (33 percent) is assigned to the test set.

There is no optimal split percentage.

You must choose a split percentage that meets your project's objectives with considerations that include:

- Computational cost in training the model.
- Computational cost in evaluating the model.
- Training set representativeness.
- Test set representativeness.

Nevertheless, common split percentages include:

- Train: 80%, Test: 20%
- Train: 67%, Test: 33%
- Train: 50%, Test: 50%

Now let us get into the algorithm part, which is the Random Forest.

## 6.3. Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

**The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**

## 6.4. How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

**Step-1:** Select random K data points from the training set.

**Step-2:** Build the decision trees associated with the selected data points (Subsets).

**Step-3:** Choose the number N for decision trees that you want to build.

**Step-4**: Repeat Step 1 & 2.

**Step-5:** For new data points, find the predictions of each decision tree, and assign the new datapoints to the category that wins the majority votes.

## 6.5. Applications of Random Forest

There are mainly four sectors where Random Forest mostly used:

**1. Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.

**2. Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.

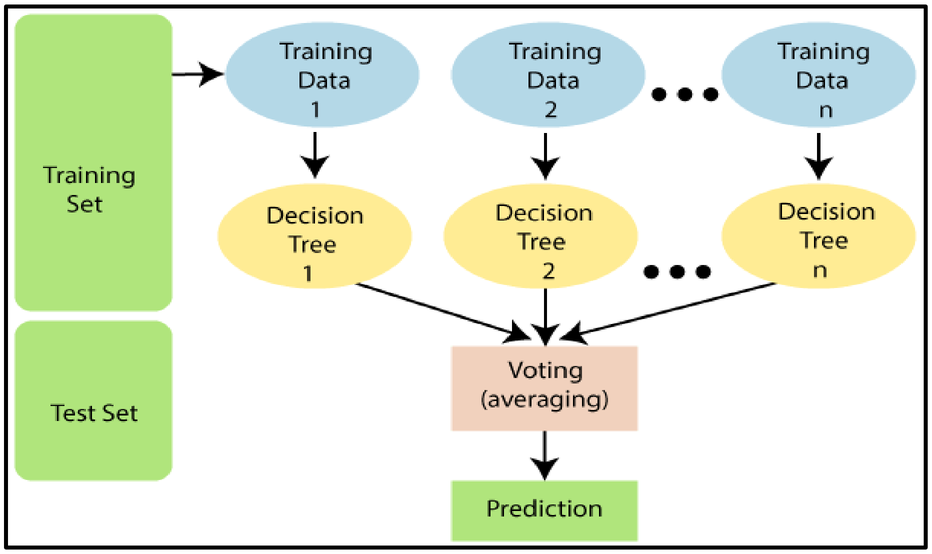
**3. Land Use:** We can identify the areas of similar land use by this algorithm.

**4. Marketing:** Marketing trends can be identified using this algorithm.

## 6.6. Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations



# VII. PICKLING AND UNPICKLING THE FILE

Python pickle module is used for serializing and de-serializing python object structures. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling. We can convert the byte stream (generated through pickling) back into python objects by a process called as unpickling.

## Why Pickle?

In real world scenario, the use pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database.

Precaution: It is advisable not to unpickle data received from an untrusted source as they may pose security threat. However, the pickle module has no way of knowing or raise alarm while pickling malicious data.

Only after importing pickle module, we can do pickling and unpickling. Importing pickle can be done using the following command –

```
import pickle
```

To pickle a file, you can use the following code

```
with open(‘file.pkl’,’wb’) as fh:
    pickle.dump(model,fh)
```

To unpickle the file you can use the following code

```
with open(‘file.pkl’,’rb’) as fh:
    ver=pickle.load(fh)
```

# VIII. DEPLOYMENT

Machine learning deployment is the process of deploying a machine learning model in a live environment. The model can be deployed across a range of different environments and will often be integrated with apps through an API. Deployment is a key step in an organization gaining operational value from machine learning.

Machine learning models will usually be developed in an offline or local environment, so will need to be deployed to be used with live data. A data scientist may create many different models, some of which never make it to the deployment stage. Developing these models can be very resource intensive. Deployment is the final step for an organization to start generating a return on investment for the organization.

However, deployment from a local environment to a real-world application can be complex. Models may need specific infrastructure and will need to be closely monitored to ensure ongoing effectiveness. For this reason, machine learning deployment must be properly managed so it's efficient and streamlined.

## 8.1 How to deploy machine learning models

Machine learning deployment can be a complex task and will differ depending on the system environment and type of machine learning model. Each organization will likely have existing DevOps processes that may need to be adapted for machine learning deployment. However, the general deployment process for machine learning models deployed to a containerized environment will consist of four broad steps.

The four steps to machine learning deployment include:

- Develop and create a model in a training environment.
- Test and clean the code ready for deployment.
- Prepare for container deployment.
- Plan for continuous monitoring and maintenance after machine learning deployment.

For this project I deployed the model using tkinter package in python.

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

## 8.2 To create a tkinter app:

1. Importing the module – tkinter

2. Create the main window (container)

3. Add any number of widgets to the main window

4. Apply the event Trigger on the widgets.

# IX. CODE FOR IMPLEMENYING THE PROJECT

## #Importing Packages

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.linear_model import Ridge

from sklearn.linear_model import Lasso

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score,classification_report

import seaborn as sns

import matplotlib.pyplot as plt
```

## #Importing the dataset

```
df=pd.read_csv('user_behavior_dataset.cs

v')

df.shape
```

Output:

```
(700,11)
```

## #Data Exploration

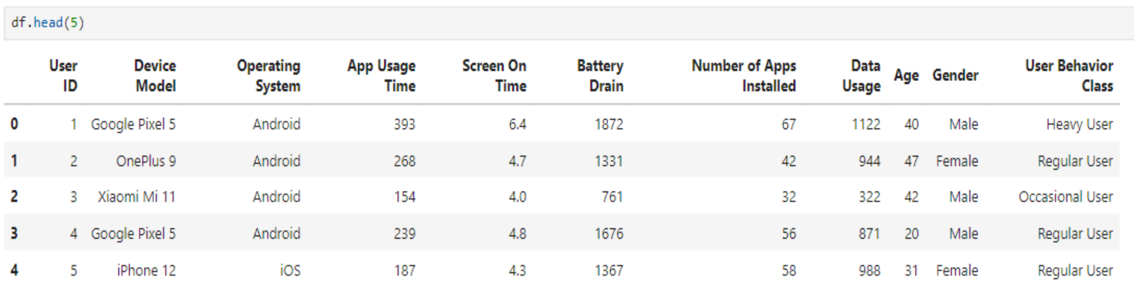
```
df.columns
```

**Output:**

```
   Index(['User ID','Device Model', 'Operating System', 'App Usage Time', 'Screen
 on Time', 'Battery Drain', 'Number of Apps Installed',
[‘Data Usage,’Age’,’Gender’,’User Behavior Class’] dtype='object')
```

df.head()

**Output:**

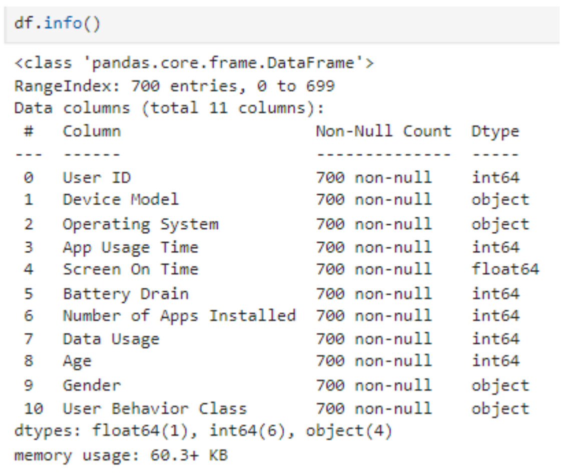


```
df.head(5)
```

| | User ID | Device Model | Operating System | App Usage Time | Screen On Time | Battery Drain | Number of Apps Installed | Data Usage | Age | Gender | User Behavior Class |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Google Pixel 5 | Android | 393 | 6.4 | 1872 | 67 | 1122 | 40 | Male | Heavy User |
| **1** | 2 | OnePlus 9 | Android | 268 | 4.7 | 1331 | 42 | 944 | 47 | Female | Regular User |
| **2** | 3 | Xiaomi Mi 11 | Android | 154 | 4.0 | 761 | 32 | 322 | 42 | Male | Occasional User |
| **3** | 4 | Google Pixel 5 | Android | 239 | 4.8 | 1676 | 56 | 871 | 20 | Male | Regular User |
| **4** | 5 | iPhone 12 | iOS | 187 | 4.3 | 1367 | 58 | 988 | 31 | Female | Regular User |

df.info()

**Output:**

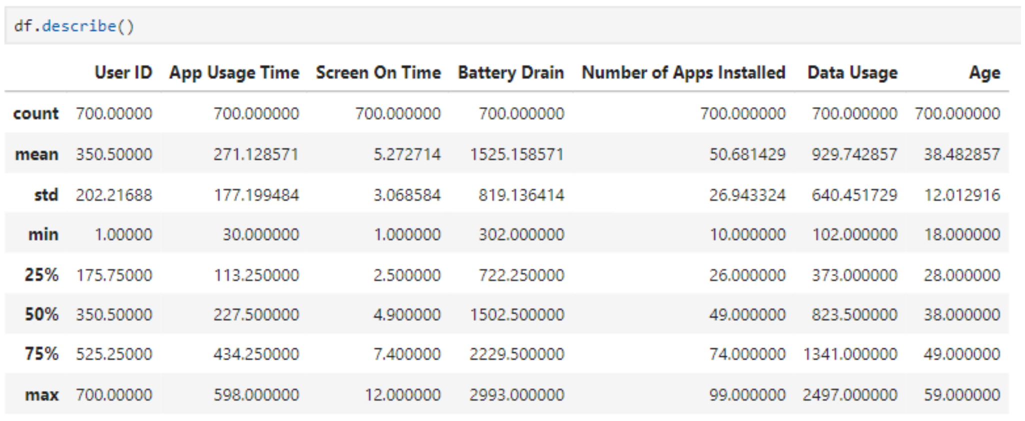


```
df.info()
```

```
<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              700 non-null    int64  
 4   Screen On Time              700 non-null    float64
 5   Battery Drain               700 non-null    int64  
 6   Number of Apps Installed    700 non-null    int64  
 7   Data Usage                  700 non-null    int64  
 8   Age                         700 non-null    int64  
 9   Gender                      700 non-null    object 
 10  User Behavior Class         700 non-null    object 
dtypes: float64(1), int64(6), object(4)
memory usage: 60.3+ KB
```

data.describe()

**Output:**



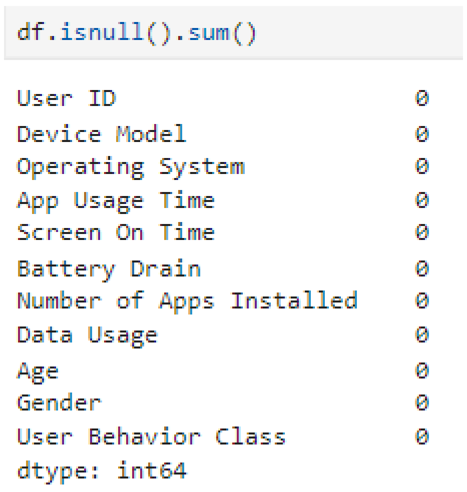
```
df.describe()
```

| | User ID | App Usage Time | Screen On Time | Battery Drain | Number of Apps Installed | Data Usage | 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 |

➢ Now let’s check that if our dataset have null values or not

df.isnull().sum()

**<u>Output:</u>**



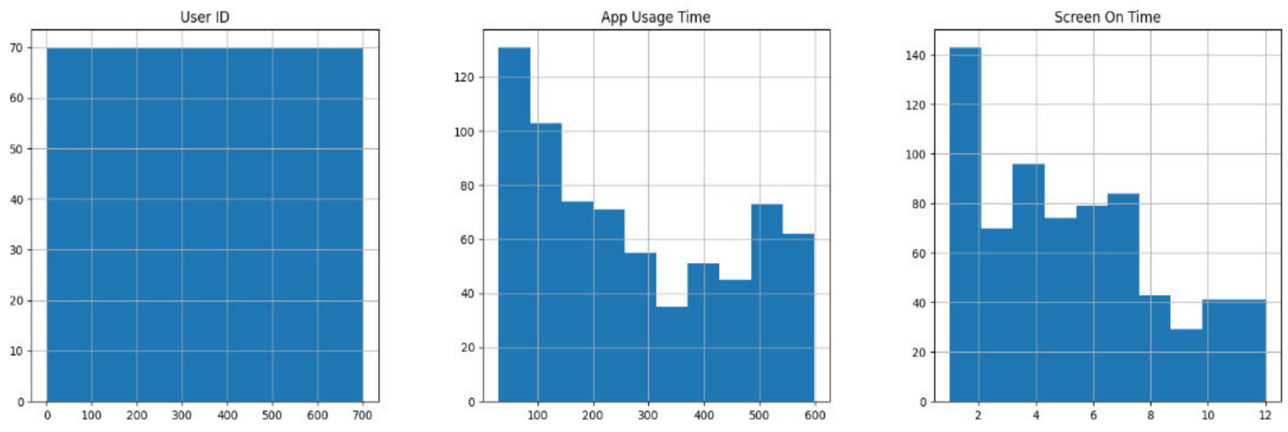
```
df.isnull().sum()

User ID                     0
Device Model                0
Operating System            0
App Usage Time              0
Screen On Time              0
Battery Drain               0
Number of Apps Installed    0
Data Usage                  0
Age                         0
Gender                      0
User Behavior Class         0
dtype: int64
```

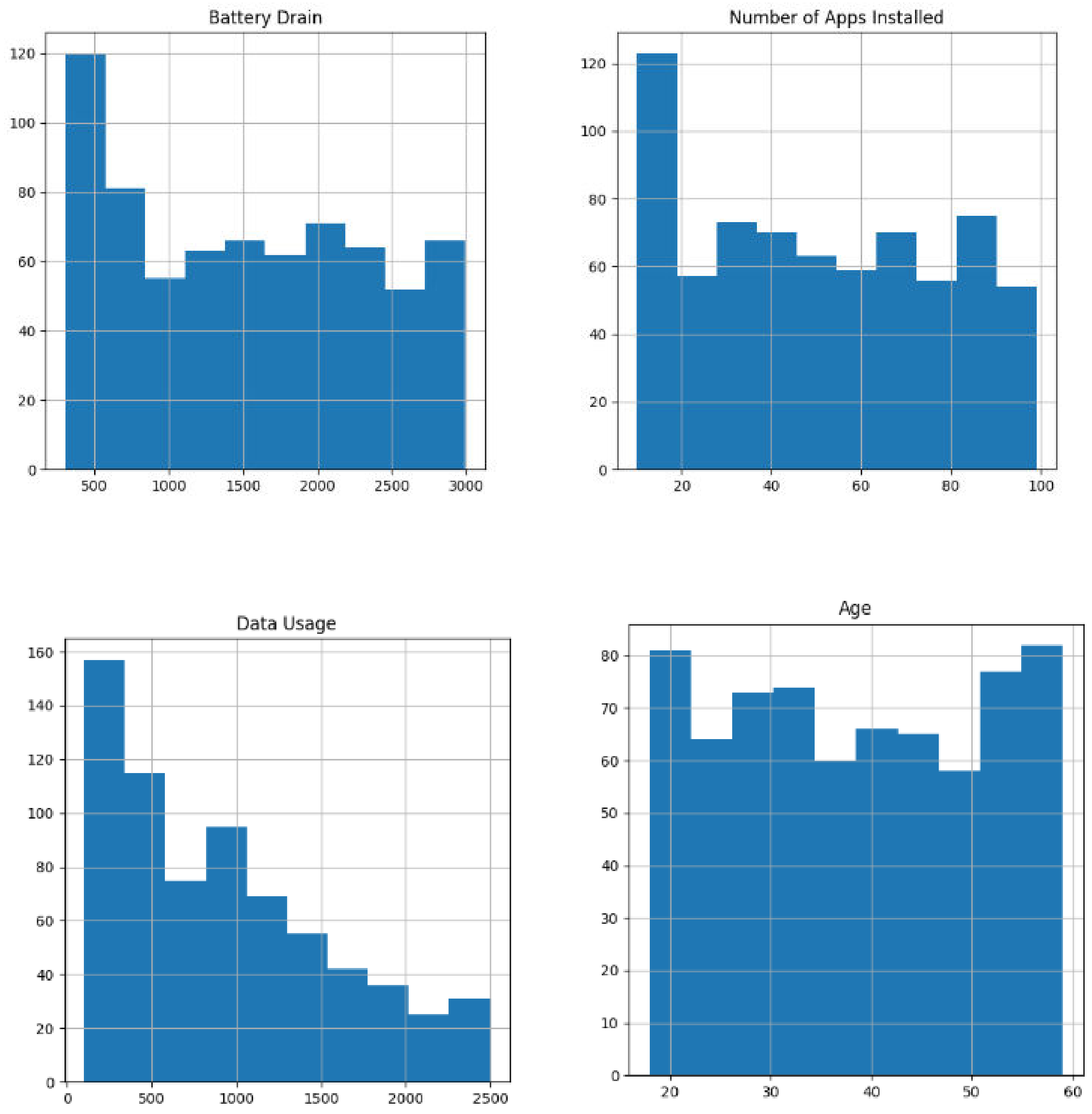
## Data Visualization

➢ Plotting the data distribution plots before removing null values

data.hist(figsize = (20,20))

**<u>Output:</u>**

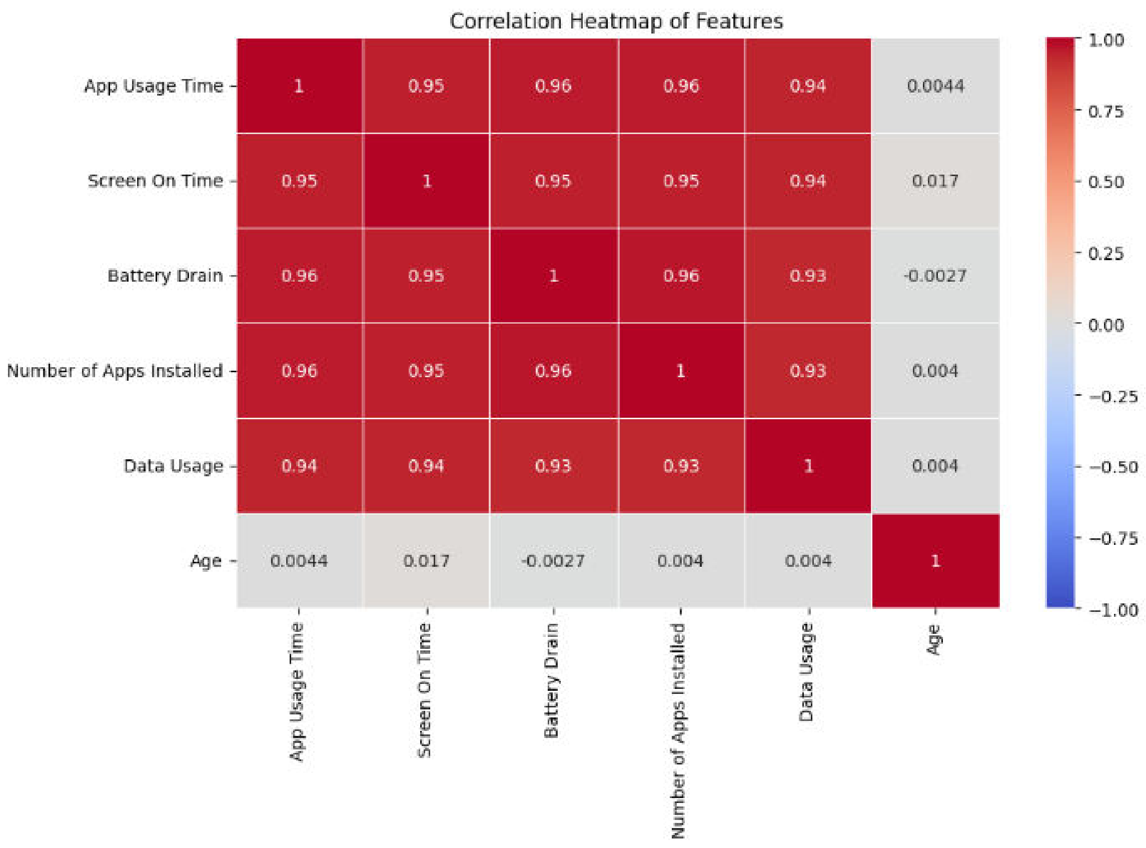



Battery Drain
0
20
40
60
80
100
120
500
1000
1500
2000
2500
3000
Number of Apps Installed
0
20
40
60
80
100
120
20
40
60
80
100
Data Usage
0
20
40
60
80
100
120
140
160
0
500
1000
1500
2000
2500
Age
0
10
20
30
40
50
60
70
80
20
30
40
50
60

## #Correlation between all the features

```
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True,
cmap='coolwarm', linewidths=0.5, vmin=-1,
vmax=1)
plt.title('Correlation Heatmap of Features')
plt.show()
```

**Output:**



| | App Usage Time | Screen On Time | Battery Drain | Number of Apps Installed | Data Usage | Age |
|---|---|---|---|---|---|---|
| App Usage Time | 1 | 0.95 | 0.96 | 0.96 | 0.94 | 0.0044 |
| Screen On Time | 0.95 | 1 | 0.95 | 0.95 | 0.94 | 0.017 |
| Battery Drain | 0.96 | 0.95 | 1 | 0.96 | 0.93 | -0.0027 |
| Number of Apps Installed | 0.96 | 0.95 | 0.96 | 1 | 0.93 | 0.004 |
| Data Usage | 0.94 | 0.94 | 0.93 | 0.93 | 1 | 0.004 |
| Age | 0.0044 | 0.017 | -0.0027 | 0.004 | 0.004 | 1 |

## #Model Building

**splitting the dataset**

```
x = df.loc[:,['App Usage Time', 'Screen On Time', 'Number of Apps Installed', 'Data
Usage','Age','Operating System']]
 y df.loc[:,["User Behavior Class"]]
```

- Now we will split the data into training and testing data using the train_test_split function

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.9, random_state=6)

 print("Training set shape:", x_train.shape)
 print("Testing set shape:", x_test.shape)
```

**Output:**

```
Training set shape: (70, 6)

Testing set shape: (630, 6)
```

## #Scaling the data

```
scaler = StandardScaler()
x_train_std = scaler.transform(x_train)
x_test_std = scaler.transform(x_test)
x_train_std.std()
```

**Output:**

```
0.999999999
```

```
X_test_std.std()
```

**Output:**

```
1.005792470801565
```

## #Algorithm Application

```
Model4=RandomForestClassifier()
y_predict4=model.predict(x_test)
accuracy_score(y_predict,y_test)*100
```

**Output:**

```
100
```

## GRAPHICAL USER INTERFACE

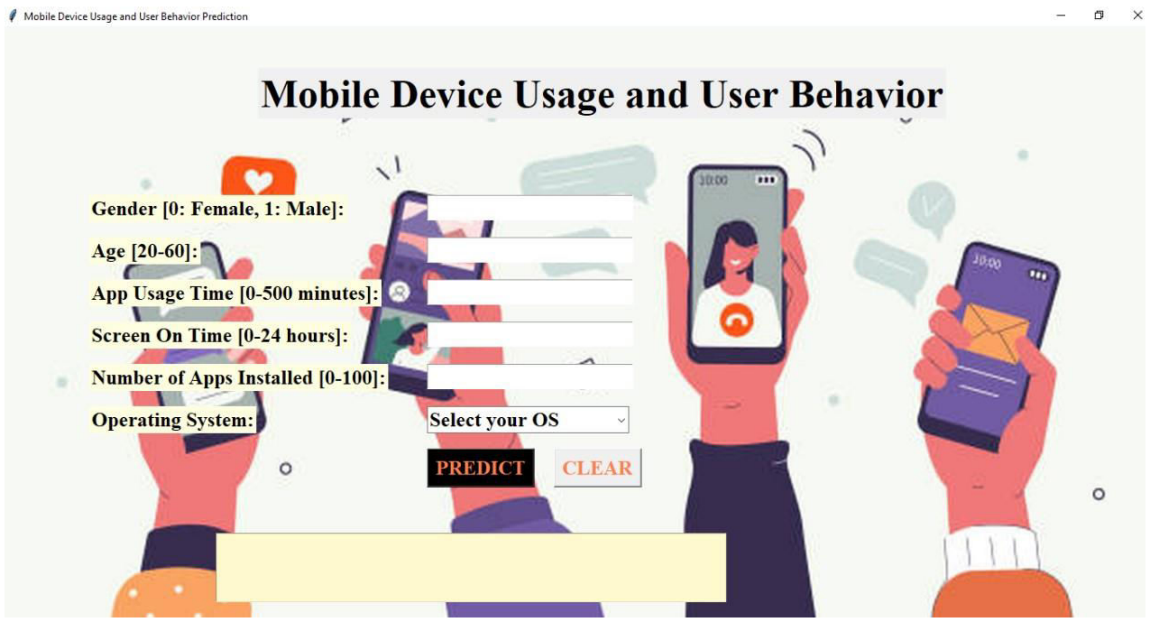
### #Pickel file

```
import tkinter as tk
from tkinter import messagebox
import numpy as np
import pickle
from PIL import Image, ImageTk

filename=" Rahul Minor project"
pickle.dump(model4,open(filename,'wb'))
model04=pickle.load(open(filename,'rb'))

with open('model04.pkl','wb') as files:
    pickle.dump(model04,files)

with open('model04.pkl','rb') as file:
    Result=pickle.load(file)
def load_model():
    return pickle.load(open('model04.pkl', 'rb'))

# Load your model
Result = load_model()

# Validation function
def validate_input(entry, min_val, max_val, data_type):
    try:
        value = data_type(entry.get())
        if not (min_val <= value <= max_val):
            raise ValueError(f'The input field should be between {min_val} and {max_val}')
        return value
    except ValueError as ve:
        messagebox.showerror('Error', str(ve))
        return None

# Prediction function
def predict():
    try:
        # Get the selected gender value
        Gender = gender_var.get()
        if Gender not in [0, 1]:  # Ensure a valid selection
            messagebox.showerror('Error', 'Please select a gender')
            return

        # Validate other input values
        Age = validate_input(entry2, 20, 60, int)  # Age: 20 to 60
        App_Usage_Time = validate_input(entry3, 0, 500, int)  # App Usage Time: 0 to 500 minutes
        Screen_On_Time = validate_input(entry4, 0, 24, float)  # Screen On Time: 0 to 24 hours
        Num_Apps_Installed = validate_input(entry5, 0, 100, int)  # Number of Apps Installed: 0 to 100

        Operating_System = dropdown_os.get()
        if Operating_System == 'Android':
            Operating_System = 0
        elif Operating_System == 'iOS':
            Operating_System = 1
        else:
            messagebox.showerror('Error', 'Please select a valid Operating System')
            return

        # Check if any input validation failed
        if any(value is None for value in [Age, App_Usage_Time, Screen_On_Time,
Num_Apps_Installed]):
            return  # Exit if any input is invalid

        # Prepare input for prediction
        inp = np.array([App_Usage_Time, Screen_On_Time, Num_Apps_Installed, Age, Gender,
Operating_System])

        # Make the prediction
        prediction = int(Result.predict(inp.reshape(1, -1)))

        # Display the prediction result
        if prediction == 0:
            text = 'The user is an Occasional User'
        elif prediction == 1:
            text = 'The user is a Regular User'
        else:
            text = 'The user is a Heavy User'
        t.delete('1.0', 'end')
        t.insert('1.0', text)
    except Exception as ep:
        messagebox.showerror('Error', f'Prediction error: {str(ep)}')

# Reset function to clear inputs and output
def reset():
    for entry in [entry2, entry3, entry4, entry5]:
        entry.delete('0', 'end')
    gender_var.set(-1)  # Reset gender selection
    dropdown_os.set("Select your OS")
    t.delete('1.0', 'end')

# Create the main application window
root = tk.Tk()
root.title('Mobile Device Usage and User Behavior Prediction')

# Load and resize the image to fit the window
image_path = r"C:\Users\brahu\OneDrive\Desktop\IMAGE.jpg"  # Change this to your image path

try:
    image1 = Image.open(image_path)  # Load the image
    window_width = root.winfo_screenwidth()
    window_height = root.winfo_screenheight()
    image1 = image1.resize((window_width, window_height), Image.LANCZOS)
    background = ImageTk.PhotoImage(image1)
    label = tk.Label(root, image=background)
    label.image = background  # Store a reference to avoid garbage collection
    label.place(x=0, y=0, relwidth=1, relheight=1)
except FileNotFoundError:
    messagebox.showerror('Error', f'Image file not found: {image_path}')
    root.destroy()
except Exception as e:
    messagebox.showerror('Error', f'Failed to load image: {e}')
    root.destroy()

# Title Label
label1 = tk.Label(root, text="Mobile Device Usage and User Behavior", font=("Times new roman", 35,
"bold"), fg="black", highlightbackground='grey')
label1.place(x=300, y=50)

# Feature 1: Gender (Radio Buttons)
label2 = tk.Label(root, text="Gender:", font=("Times new roman", 18, "bold"), fg="black",
highlightbackground='grey', bg="light yellow")
label2.place(x=100, y=200)
```

```
gender_var = tk.IntVar(value=-1)  # Variable to store the selected gender (0: Female, 1: Male)

radio1 = tk.Radiobutton(root, text="Female", variable=gender_var, value=0, font=("Times new roman", 
18))
radio1.place(x=500, y=200)

radio2 = tk.Radiobutton(root, text="Male", variable=gender_var, value=1, font=("Times new roman", 
18))
radio2.place(x=650, y=200)

# Feature 2: Age
label3 = tk.Label(root, text="Age [20-60]:", fg="black", font=("Times new roman", 18, "bold"), 
highlightbackground='grey', bg="light yellow")
label3.place(x=100, y=250)
entry2 = tk.Entry(root, width=20, font=("Times new roman", 18, "bold"), fg="black")
entry2.place(x=500, y=250)

# Feature 3: App Usage Time
label4 = tk.Label(root, text="App Usage Time [0-500 minutes]:", fg="black", font=("Times new roman", 
18, "bold"), highlightbackground='grey', bg="light yellow")
label4.place(x=100, y=300)
entry3 = tk.Entry(root, width=20, font=("Times new roman", 18, "bold"), fg="black")
entry3.place(x=500, y=300)

# Feature 4: Screen On Time
label5 = tk.Label(root, text="Screen On Time [0-24 hours]:", fg="black", font=("Times new roman", 18, 
"bold"), highlightbackground='grey', bg="light yellow")
label5.place(x=100, y=350)
entry4 = tk.Entry(root, width=20, font=("Times new roman", 18, "bold"), fg="black")
entry4.place(x=500, y=350)

# Feature 5: Number of Apps Installed
label6 = tk.Label(root, text="Number of Apps Installed [0-100]:", fg="black", font=("Times new roman", 
18, "bold"), highlightbackground='grey', bg="light yellow")
label6.place(x=100, y=400)
entry5 = tk.Entry(root, width=20, font=("Times new roman", 18, "bold"), fg="black")
entry5.place(x=500, y=400)

# Feature 6: Operating System (Dropdown)
label7 = tk.Label(root, text="Operating System:", fg="black", font=("Times new roman", 18, "bold"), 
highlightbackground='grey', bg="light yellow")
label7.place(x=100, y=450)

options = ['Android', 'iOS']
dropdown_os = ttk.Combobox(root, values=options, width=18, font=("Times new roman", 18, "bold"))
dropdown_os.place(x=500, y=450)
dropdown_os.set("Select your OS")
```

```
# Button to trigger prediction
button1 = tk.Button(root, text='PREDICT', highlightbackground='grey', fg="coral", command=predict,
font=("Times new roman", 18, "bold"), bg="black")
button1.place(x=500, y=500)

# Text area to display the result
t = tk.Text(root, highlightbackground='green', fg="dark blue", bg="lemon chiffon", height=3, width=50,
font=("Times new roman", 18, "bold"))
t.place(x=250, y=600)

# Clear button
button2 = tk.Button(root, text='CLEAR', fg="coral", command=reset, font=("Times new roman", 18,
"bold"))
button2.place(x=650, y=500)

# Start the main event loop
root.mainloop()
```

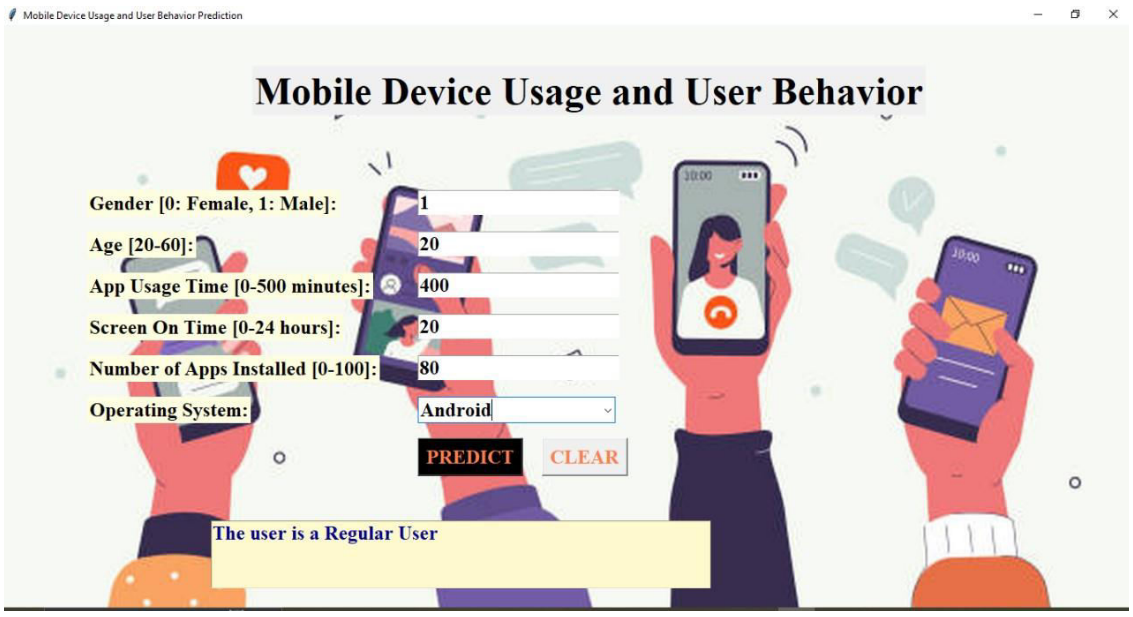
## X. SCREENSHOT'S OF GUI



Mobile Device Usage and User Behavior Prediction

Mobile Device Usage and User Behavior

| | |
|---|---|
| Gender [0: Female, 1: Male]: | |
| Age [20-60]: | |
| App Usage Time [0-500 minutes]: | |
| Screen On Time [0-24 hours]: | |
| Number of Apps Installed [0-100]: | |
| Operating System: | Select your OS |



Mobile Device Usage and User Behavior Prediction

Mobile Device Usage and User Behavior

| | |
|---|---|
| Gender [0: Female, 1: Male]: | 1 |
| Age [20-60]: | 20 |
| App Usage Time [0-500 minutes]: | 400 |
| Screen On Time [0-24 hours]: | 20 |
| Number of Apps Installed [0-100]: | 80 |
| Operating System: | Android |

The user is a Regular User

# XI. CONCLUSION

The user behavior analysis project successfully demonstrates the potential of data analysis and machine learning in understanding user patterns and categorizing behavior. By leveraging features such as app usage time, screen-on time, battery drain, number of apps installed, data usage, age, and gender, we were able to classify user behavior into distinct categories: Heavy User, Regular User, and Occasional User. This classification helps in identifying the factors that influence mobile usage behavior.

Through data visualization, significant trends and patterns were uncovered, providing valuable insights for designing user-centric applications and services. The project emphasizes the importance of feature engineering and data preprocessing in creating effective models that accurately capture user behavior.

This project serves as a foundation for advanced user behavior analysis, highlighting the role of data-driven insights in enhancing user experience and personalization. It also showcases the potential to extend this analysis to include real-time monitoring, enabling businesses to adapt to user needs dynamically. By bridging the gap between raw data and actionable insights, this project demonstrates the impactful role of machine learning in solving real-world problems in technology and consumer behavior.

# XII. FUTURE SCOPE

The user behavior analysis project holds immense potential for further development. Incorporating advanced machine learning models, such as clustering algorithms, deep learning techniques, or reinforcement learning, can improve the ability to uncover complex patterns and predict user behavior with greater precision. Including additional features like app categories, social media activity, and contextual factors such as time of day or location could further enhance the depth and accuracy of the analysis.

Developing a real-time behavior monitoring system that dynamically updates user classifications based on live data streams can make the tool more actionable. Integrating sentiment analysis from app reviews or user feedback could provide richer insights into user preferences and satisfaction. Expanding the analysis to cover group behavior trends or peer comparison can offer businesses a broader understanding of user engagement and retention.

Additionally, incorporating predictive analytics to anticipate future behavior, such as churn likelihood or app adoption, can support businesses in proactive decision-making. Developing a dashboard or application with intuitive visualizations and reports will make the tool accessible to diverse stakeholders, ranging from app developers to marketing teams.

By enhancing the project with these advancements, it can evolve into a comprehensive platform for user behavior analysis, catering to diverse industries such as mobile technology, digital marketing, and user experience design. This evolution would significantly contribute to the field of consumer behavior analytics and empower businesses to make data-driven decisions that improve user engagement and satisfaction.

## XIII. REFERENCES

- https://www.digitalocean.com/community/tutorials/exploratory-data-analysis-python
- https://www.w3schools.com/python/python_intro.asp
- https://www.geeksforgeeks.org/introduction-machine-learning/
- https://www.w3schools.com/python/pandas/default.asp
- https://builtin.com/data-science/random-forest-algorithm
- https://www.w3schools.com/python/numpy/numpy_intro.asp
- https://www.activestate.com/resources/quick-reads/what-is-scikit-learn-in-python/
- https://www.datacamp.com/tutorial/matplotlib-tutorial-python
- https://www.simplilearn.com/tutorials/python-tutorial/python-seaborn
- https://stackoverflow.com/questions/7501947/understanding-pickling-in-python
- https://blog.hubspot.com/website/python-pickle