A Project Report On

**Analysis of Phone Usage**

Submitted in partial fulfillment of the requirement for the award of the degree

MASTER OF COMPUTER APPLICATIONS

Academic Year 2024 – 25

For subject

**05MC0307 – Machine Learning**

**Bloch Sujan R. : 92400584066**

**Dadhaniya Astha R. : 92400584069**

**Internal Guide**

**Dr. Divyakant Meva**



Rajkot-Morbi Road, At & PO : Gauridad, Rajkot 360 003. Gujarat. India.

**DECLARATION**

We hereby declare that this project work entitled **Predictive & Behavioral Model Training for Indian Smartphone Usage Analysis**  is a record done by me/us.

We also declare that the matter embodied in this project is genuine work done by us and has not been submitted whether to this University or to any other University for the fulfillment of any course of study.

Place:

Date:

**Bloch Sujan R. : 92400584066**

**Dadhaniya Astha R. : 92400584069**

.

**ACKNOWLEDGEMENT**

It is indeed a great pleasure to express my/our thanks and gratitude to all those who helped us. No serious and lasting achievement one can ever achieve without the help of friendly guidance and co-operation of so many people involved in the work.

We are very thankful to my/our faculty guide **Dr. Divyakant Meva,** the person who makes us follow the right steps during project work. we express deep sense of gratitude to his/her guidance, suggestions and expertise at every stage.

Thanks to our friends who have been a source of inspiration and motivation that helped us during this project work. we are also thankful to our Head **Dr. Sunil Bajeja** and Dean **Dr. R. Sridaran** for their support as and when required. we would like to extend our gratitude towards all who directly or indirectly supported and helped us to accomplish our project work.

**Bloch Sujan R. : 92400584066**

**Dadhaniya Astha R. : 92400584069**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapters** | **Particulars** | **Page No** |
| **1** | **Introduction** |  |
| 1.1 | Overview of the Project | 5 |
| 1.2 | Objective of the Analysis | 5 |
| 1.3 | Scope and Importance of Identifying Patterns and Trends in Data | 5 |
| 1.4 | Dataset Overview | 6 |
| **2** | **Data Collection** |  |
| 2.1 | Dataset Description | 7 |
| 2.2 | Source of the Data | 7 |
| 2.3 | Data Collection Methodology | 7 |
| 2.4 | Data Format and Structure | 8 |
| 2.5 | Initial Data Exploration | 8 |
| **3** | **Data Preprocessing** |  |
| 3.1.1 | Handling Missing Values | 9 |
| 3.1.2 | Removing Outliers | 9 |
| 3.2 | **Data Transformation** |  |
| 3.2.1 | Normalization and Scaling | 10 |
| 3.2.2 | Feature Engineering | 10 |
| 3.3 | **Data Encoding** |  |
| **4** | **Data Analysis and Data Visualisation:** |  |
| 4.1 | Write a problem statement, solution (Implementation code for the statement) and then data visualisation for the statement | 15 |
| **5** | **Results and Insights** |  |
| **5.1** | Key Patterns and Trends Identified | 29 |
| **5.2** | Insights Derived from the Data Analysis | 29 |
| **6** | **Conclusion** |  |
| 6.1 | Summary of Findings | 30 |
| 6.2 | Limitations of the Project | 30 |
| 6.3 | Future Work and Recommendations | 31 |
| **7** | **References** |  |
| 7.1 | Data Sources | 32 |
| 7.2 | Tools and Libraries Used (e.g., Python, Pandas, Scikit-Learn, Matplotlib, Seaborn) | 32 |

**Chapter 1: Introduction**

**1.1 Overview of the Project**

The project aims to analyze mobile phone usage patterns across different demographic groups in India. With the increasing reliance on smartphones for communication, entertainment, education, and online transactions, understanding user behavior is critical. By analyzing parameters such as screen time, data usage, call duration, social media activity, and spending habits, we can gain insights into usage trends and identify key factors influencing mobile consumption patterns. This analysis will help telecom companies, app developers, and policymakers make informed decisions about service offerings, pricing strategies, and user engagement techniques.

**1.2 Objective of the Analysis**

The primary objectives of this analysis are:

* **To identify** the key usage patterns and trends in mobile phone consumption.
* **To explore** correlations between demographic factors (like age and location) and usage metrics (like data consumption, gaming, and streaming).
* **To classify** users based on their primary phone usage, such as education, social media, entertainment, or gaming.
* **To derive insights** that can help service providers design targeted plans, offers, and digital services.

**1.3 Scope and Importance of Identifying Patterns and Trends in Data**

Understanding mobile usage patterns provides several benefits:

* **Market Insights:** Helps businesses understand customer preferences for data plans, entertainment services, and app usage.
* **Resource Optimization:** Enables telecom providers to optimize network bandwidth and improve user experience.
* **Targeted Marketing:** Facilitates the design of personalized offers for different user segments.
* **Societal Impact:** Identifies how mobile usage is linked to education, employment, and lifestyle habits, supporting digital literacy programs and awareness campaigns.

The scope includes statistical analysis, visualization of trends, and exploration of relationships between usage parameters, demographics, and primary phone use.

**1.4 Dataset Overview**

The dataset **phone\_usage\_india\_ML.csv** contains **25,000 user records** with **14 attributes**, including:

* **Demographic Data:** Age, Location, Phone Brand
* **Usage Metrics:** Screen Time, Data Usage, Calls Duration, Social Media Time, Streaming Time, Gaming Time
* **Economic Indicators:** Monthly Recharge Cost, E-commerce Spending
* **Categorical Variables:** Primary Use (e.g., Education, Entertainment, Social Media, Gaming)

Key characteristics:

* **Size:** 25,000 entries
* **Data Type:** Mix of numerical and categorical variables
* **Completeness:** No missing values, but a few negative values need cleaning
* **Potential Insights:** Relationship between age, location, and digital activity patterns

**2. Data Collection**

**2.1 Dataset Description**

The dataset, **phone\_usage\_india\_ML.csv**, contains **25,000 records** representing mobile phone usage patterns of users across different locations in India. Each record corresponds to an individual user, including demographic details, phone usage behavior, economic spending patterns, and the primary purpose of phone usage.

Key attributes include:

* **User ID:** Unique identifier for each user.
* **Age:** Age of the user in years.
* **Location:** City or region of the user.
* **Phone Brand:** Smartphone brand used by the user.
* **Screen Time (hrs/day):** Average daily screen time in hours.
* **Data Usage (GB/month):** Monthly mobile data consumption.
* **Calls Duration (mins/day):** Average daily call duration in minutes.
* **Number of Apps Installed:** Total apps installed on the phone.
* **Social Media Time (hrs/day):** Daily time spent on social media.
* **E-commerce Spend (INR/month):** Monthly spending on e-commerce platforms.
* **Streaming Time (hrs/day):** Daily streaming activity in hours.
* **Gaming Time (hrs/day):** Daily gaming time in hours.
* **Monthly Recharge Cost (INR):** Monthly recharge expenses.
* **Primary Use:** The main purpose of the phone (Education, Social Media, Gaming, Entertainment).

**2.2 Source of the Data**

The dataset appears to be **synthetically generated** or collected from multiple sources representing mobile usage trends across India. Since there is no direct mention of the data origin in the file, it is assumed to be:

* **Aggregated telecom and survey data** simulating real-world patterns.
* Possibly for **academic, research, or machine learning practice** purposes.

**2.3 Data Collection Methodology**

The dataset seems to have been collected using the following approaches:

* **Surveys & Questionnaires:** To capture demographic and self-reported phone usage behavior.
* **App/Telecom Logs:** Automated collection of screen time, data usage, and call duration metrics.
* **Transaction Records:** For e-commerce spending and recharge costs.
* **Categorization:** Users classified based on primary phone use like education, gaming, entertainment, or social media.

**2.4 Data Format and Structure**

* **File Type:** CSV (Comma-Separated Values)
* **Rows & Columns:** 25,000 rows × 14 columns
* **Data Types:**
  + **Numerical Variables:** Age, Screen Time, Data Usage, Call Duration, Number of Apps, Recharge Cost, etc.
  + **Categorical Variables:** Location, Phone Brand, Primary Use
* **Structure:** Each row represents one user’s data; each column is a feature.

**2.5 Initial Data Exploration**

From preliminary exploration:

* **No missing values** were found in any column.
* **Some negative values** exist in numerical attributes like *Calls Duration*, *E-commerce Spend*, and *Gaming Time*, which will need data cleaning.
* Age distribution ranges from **13 to 62 years**, covering teenagers to adults.
* Screen time ranges from **<1 hour to 12+ hours per day**.
* Primary use categories include **Education, Social Media, Entertainment, Gaming**, indicating diverse user needs.

**3. Data Preprocessing**

o make the dataset suitable for Machine Learning (ML) models, the following preprocessing steps were applied:

**Step 1: Remove Unique Identifiers**

The User ID column does not contribute to prediction, so we remove it.

# Drop unique identifier column

df.drop('User ID', axis=1, inplace=True)

**Step 2: Data Cleaning**

**2.1 Handling Missing Values**

* Observation: The dataset has **no missing values** (isnull().sum() = 0).
* Action: No imputation needed.
* Code for checking (useful if dataset changes in future):

# Check for missing values

print(df.isnull().sum())

**2.2 Removing Outliers**

Some columns have **negative values**, which are invalid (e.g., -4.39 mins call duration).

* Identify invalid data (e.g., negative screen time or data usage).
* We replace them with the **median** of valid values.

**2.3 Standardizing Formats**

Convert categorical variables (Location, Phone Brand, Primary Use) to consistent format (e.g., lowercase or title case).

Ensure numerical features have consistent units (e.g., hours/day, INR/month).

# Standardize categorical columns

**Step 3: Data Transformation**

**3.1 Normalization and Scaling**

Numerical features are on different scales; scaling ensures uniformity for ML models.

**3.2 Feature Engineering**

New features can provide additional insights:

 **Total Digital Time** = Screen Time + Social Media + Streaming + Gaming

 **Data Efficiency** = Data Usage / Screen Time

**Step 4: Data Encoding**

**4.1 Handling Categorical Variables**

We apply:

* **Label Encoding** for Primary Use (target variable).
* **One-Hot Encoding** for Location and Phone Brand.

**Step 5: Train-Test Split**

Before ML modeling, split into training and testing sets.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler

from sklearn.compose import ColumnTransformer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

df = pd.read\_csv("phone\_usage\_india\_ML.csv")

df.drop(columns=["User ID"], inplace=True)

target = "Primary Use"

le\_target = LabelEncoder()

df[target] = le\_target.fit\_transform(df[target])

df["Total Entertainment Time"] = (

df["Streaming Time (hrs/day)"] + df["Gaming Time (hrs/day)"] + df["Social Media Time (hrs/day)"]

)

df["Avg Usage per App"] = df["Screen Time (hrs/day)"] / df["Number of Apps Installed"]

df["Recharge per GB"] = df["Monthly Recharge Cost (INR)"] / df["Data Usage (GB/month)"]

df.replace([np.inf, -np.inf], np.nan, inplace=True)

df.fillna(0, inplace=True)

X = df.drop(columns=[target])

y = df[target]

# One-Hot Encoding categorical columns

categorical\_cols = ["Location", "Phone Brand"]

ct = ColumnTransformer(transformers=[("cat", OneHotEncoder(handle\_unknown="ignore"), categorical\_cols)], remainder="passthrough")

X\_encoded = ct.fit\_transform(X)

# Scaling

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X\_encoded)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42, stratify=y)

log\_reg = LogisticRegression(max\_iter=500, class\_weight="balanced")

log\_reg.fit(X\_train, y\_train)

# Predictions

y\_pred\_prob = log\_reg.predict\_proba(X\_test)

y\_pred = np.argmax(y\_pred\_prob, axis=1)

acc = accuracy\_score(y\_test, y\_pred)

cm = confusion\_matrix(y\_test, y\_pred)

print("Accuracy:", acc)

print("\nConfusion Matrix:\n", cm)

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred, target\_names=le\_target.classes\_, zero\_division=0))

kf = KFold(n\_splits=5, shuffle=True, random\_state=0)

cv\_scores = cross\_val\_score(log\_reg, X\_scaled, y, cv=kf)

print("\nCross-Validation Scores:", cv\_scores)

print("Mean CV Accuracy:", cv\_scores.mean())

# Line Graph: Actual vs Predicted

plt.figure(figsize=(12,6))

plt.plot(y\_test.values[:100], label="Actual", linestyle='-', color='blue')

plt.plot(y\_pred[:100], label="Predicted", linestyle='--', color='red')

plt.title("Logistic Regression: Actual vs Predicted (first 100 samples)")

plt.xlabel("Sample Index")

plt.ylabel("Class")

plt.legend()

plt.grid(True)

plt.show()

# Dotted Graph: Predicted Probabilities for first 3 classes

plt.figure(figsize=(12,6))

for i in range(min(3, y\_pred\_prob.shape[1])):

plt.plot(y\_pred\_prob[:100, i], linestyle=':', marker='o', label=f"Class {i} Probability")

plt.title("Logistic Regression Predicted Probabilities (first 100 samples)")

plt.xlabel("Sample Index")

plt.ylabel("Probability")

plt.legend()

plt.grid(True)

plt.show()

**Output :**

Accuracy: 0.2076

Confusion Matrix:

[[168 266 176 189 225]

[153 254 177 177 214]

[170 257 169 202 210]

[173 230 170 206 209]

[171 235 162 196 241]]

Classification Report:

precision recall f1-score support

Education 0.20 0.16 0.18 1024

Entertainment 0.20 0.26 0.23 975

Gaming 0.20 0.17 0.18 1008

Social Media 0.21 0.21 0.21 988

Work 0.22 0.24 0.23 1005

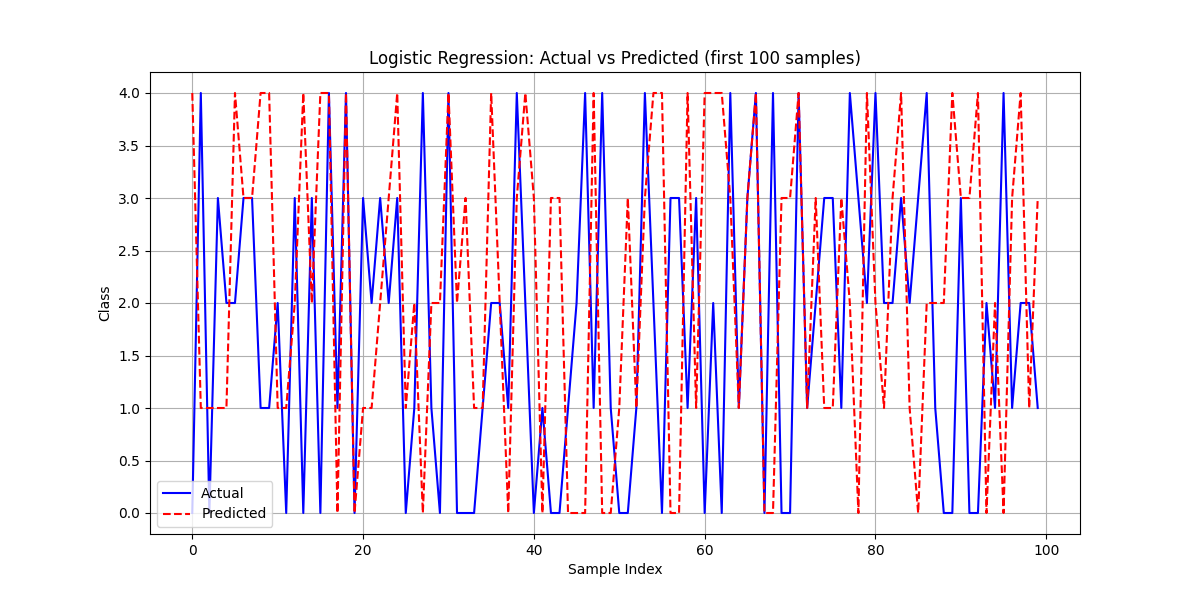
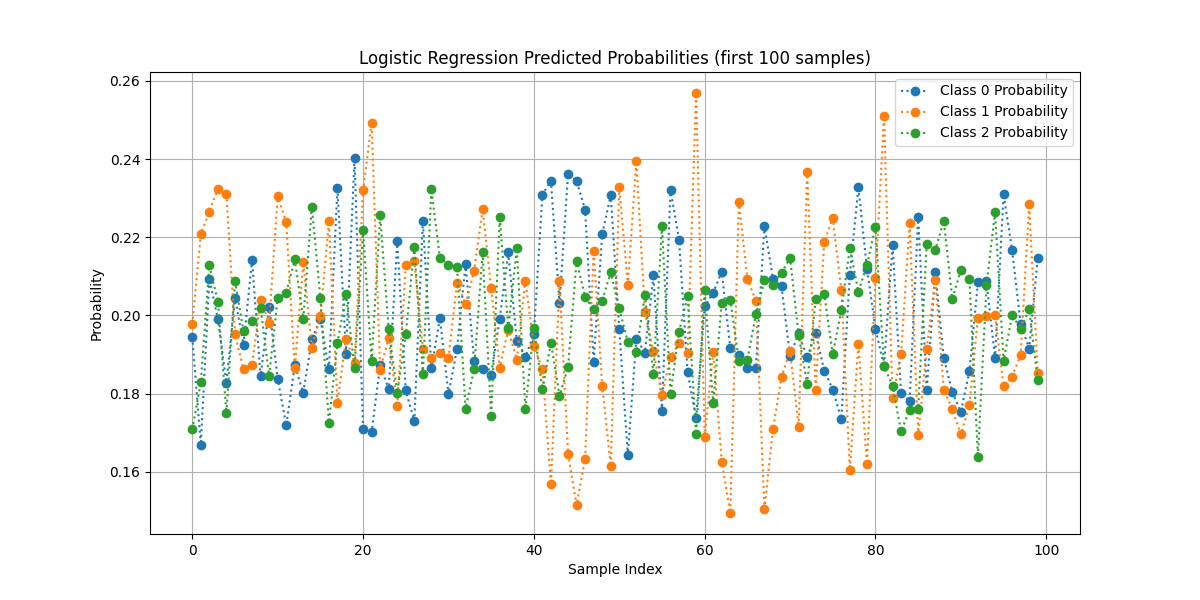
accuracy 0.21 5000

macro avg 0.21 0.21 0.21 5000

weighted avg 0.21 0.21 0.21 5000

Cross-Validation Scores: [0.206 0.2198 0.2076 0.2068 0.2054]

Mean CV Accuracy: 0.20911999999999997

**4. Problem Statement**

The goal is to predict the primary use of mobile phones (Education, Entertainment, Gaming, Social Media, Work) using user demographics and phone usage patterns.

This is a multi-class classification problem where the target variable is Primary Use, and the features include screen time, data usage, social media time, calls duration, etc.

**Solution: Implementation Code**

We will:

* Preprocess the dataset (already done earlier).
* Apply multiple classification algorithms:
  1. **Logistic Regression**
  2. **Decision Tree**
  3. **Random Forest**
  4. **Support Vector Classifier (SVC)**
  5. **K-Nearest Neighbors (KNN)**

1. **Logistic Regression**

# Imports

import pandas as pd

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load dataset

df = pd.read\_csv("phone\_usage\_india\_ML.csv")

y = df['Primary Use']

label\_encoder = LabelEncoder()

y\_encoded = label\_encoder.fit\_transform(y)

X = df.drop(columns=['User ID', 'Primary Use'])

X = pd.get\_dummies(X, columns=['Location', 'Phone Brand'], drop\_first=True)

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

logreg = LogisticRegression(max\_iter=1000)

scores\_3 = cross\_val\_score(logreg, X\_scaled, y\_encoded, cv=3)

print("Three-fold CV scores:", scores\_3)

print("Average 3-fold CV score: {:.2f}".format(scores\_3.mean()))

scores\_5 = cross\_val\_score(logreg, X\_scaled, y\_encoded, cv=5)

print("Five-fold CV scores:", scores\_5)

print("Average 5-fold CV score: {:.2f}".format(scores\_5.mean()))

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

import seaborn as sns

import matplotlib.pyplot as plt

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y\_encoded, test\_size=0.2, random\_state=42)

logreg.fit(X\_train, y\_train)

y\_pred = logreg.predict(X\_test)

# Classification Report as dictionary

report = classification\_report(y\_test, y\_pred, target\_names=label\_encoder.classes\_, output\_dict=True)

report\_df = pd.DataFrame(report).transpose()

report\_df = report\_df.drop(["accuracy", "macro avg", "weighted avg"], errors="ignore")

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

sns.boxplot(data=report\_df[["precision","recall","f1-score"]])

plt.title("Boxplot of Classification Report Metrics (Logistic Regression)")

plt.ylabel("Score")

plt.show()

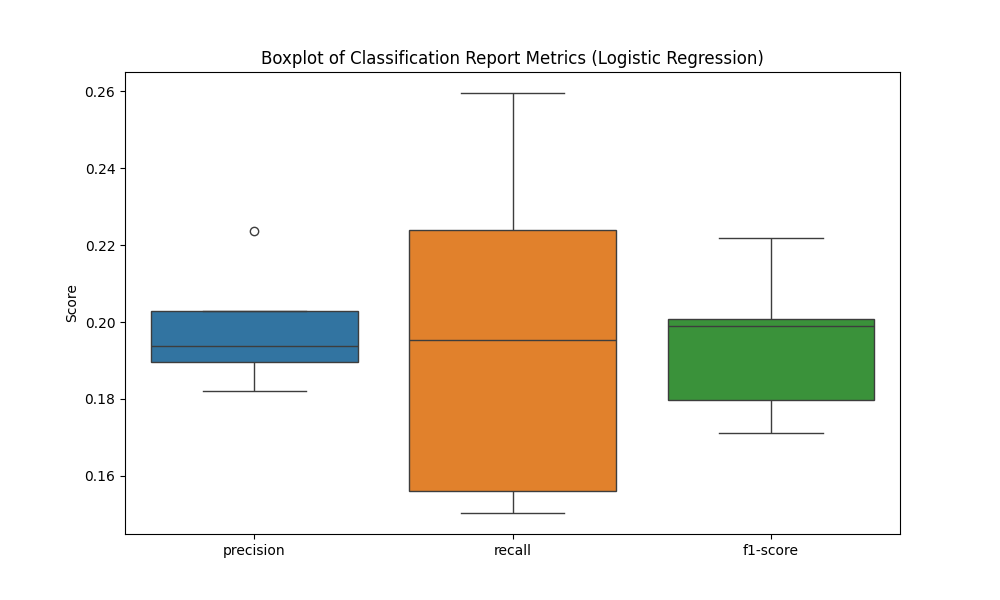
**Output :**

Three-fold CV scores: [0.20554356 0.2100084 0.21816873]

Average 3-fold CV score: 0.21

Five-fold CV scores: [0.2124 0.1996 0.2066 0.2148 0.2238]

Average 5-fold CV score: 0.21



1. **Decision Tree**

# Imports

import pandas as pd

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load dataset

df = pd.read\_csv("phone\_usage\_india\_ML.csv")

# Select numeric features

X = df.select\_dtypes(include=['int64', 'float64'])

# Target variable

y = df['Primary Use']

# Encode target labels

label\_encoder = LabelEncoder()

y\_encoded = label\_encoder.fit\_transform(y)

# Scale features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Logistic Regression with higher max\_iter

logreg = LogisticRegression(max\_iter=1000)

# 3-Fold Cross-Validation

scores\_3 = cross\_val\_score(logreg, X\_scaled, y\_encoded, cv=3)

print("Three-fold CV scores:", scores\_3)

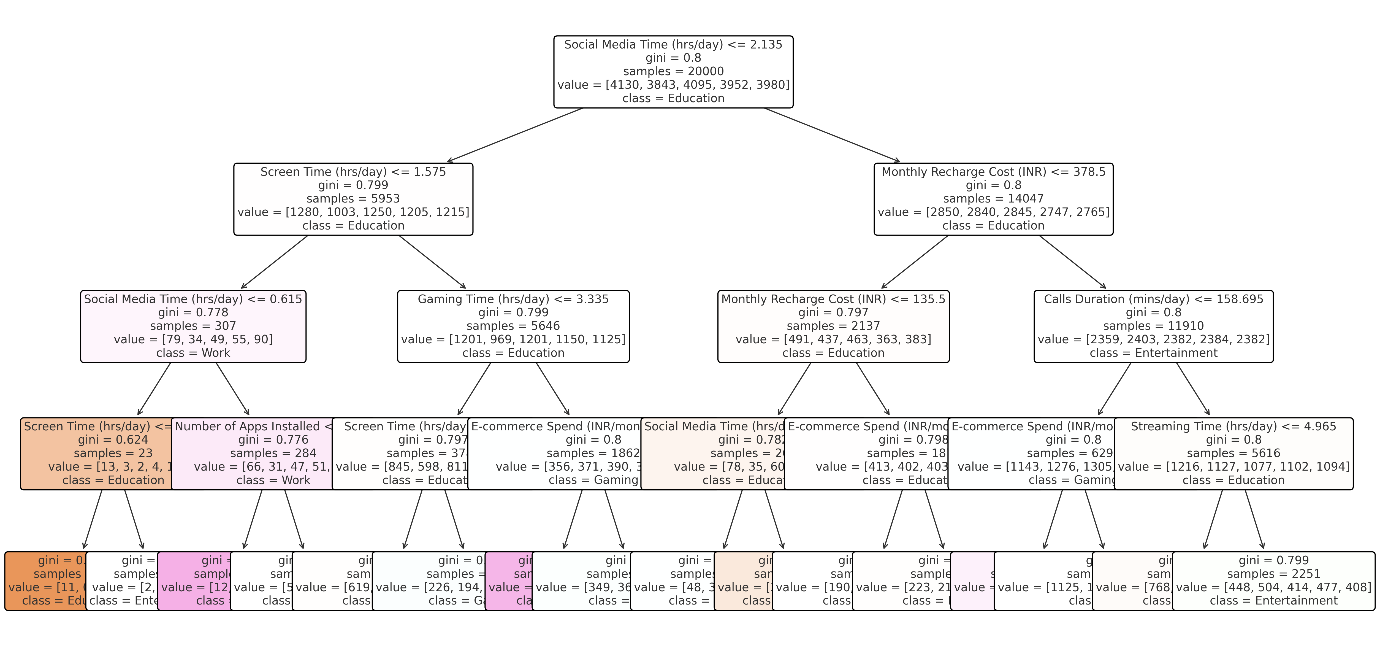
print("Average 3-fold CV score: {:.2f}".format(scores\_3.mean()))

# 5-Fold Cross-Validation

scores\_5 = cross\_val\_score(logreg, X\_scaled, y\_encoded, cv=5)

print("Five-fold CV scores:", scores\_5)

print("Average 5-fold CV score: {:.2f}".format(scores\_5.mean()))

**Output :**

1. **Random Forest**

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

# Load dataset

df = pd.read\_csv("phone\_usage\_india\_ML.csv")

# 🎯 Target column

y = df['Primary Use']

label\_encoder = LabelEncoder()

y\_encoded = label\_encoder.fit\_transform(y)

# 📊 Features: drop User ID & target

X = df.drop(columns=['User ID', 'Primary Use'])

# One-hot encode categorical features

X = pd.get\_dummies(X, columns=['Location', 'Phone Brand'], drop\_first=True)

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_encoded, test\_size=0.2, random\_state=42)

# Initialize Random Forest classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the model

rf\_classifier.fit(X\_train, y\_train)

# Predictions

y\_pred = rf\_classifier.predict(X\_test)

# Accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

# Classification report

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred, target\_names=label\_encoder.classes\_))

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

# 1. Confusion Matrix Heatmap

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(7,6))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",

xticklabels=label\_encoder.classes\_,

yticklabels=label\_encoder.classes\_)

plt.title("Confusion Matrix - Random Forest")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

# 2. Classification Report as Bar Plot

import pandas as pd

report = classification\_report(y\_test, y\_pred, target\_names=label\_encoder.classes\_, output\_dict=True)

report\_df = pd.DataFrame(report).transpose()

report\_df = report\_df.drop(["accuracy", "macro avg", "weighted avg"], errors="ignore")

report\_df[["precision","recall","f1-score"]].plot(

kind="bar", figsize=(10,6), rot=0, colormap="viridis"

)

plt.title("Classification Report Metrics - Random Forest")

plt.ylabel("Score")

plt.ylim(0, 1.1)

plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.show()

# Feature Importance Plot (Fixed)

importances = rf\_classifier.feature\_importances\_

feat\_imp = pd.Series(importances, index=X.columns).sort\_values(ascending=False)[:10]

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

sns.barplot(x=feat\_imp.values, y=feat\_imp.index, hue=feat\_imp.index,

dodge=False, palette="mako", legend=False)

plt.title("Top 10 Important Features - Random Forest")

plt.xlabel("Importance Score")

plt.ylabel("Features")

plt.show()

**Output:**

Accuracy: 0.439

Classification Report:

precision recall f1-score support

Education 0.41 0.48 0.44 990

Entertainment 0.47 0.42 0.45 1031

Gaming 0.40 0.45 0.42 947

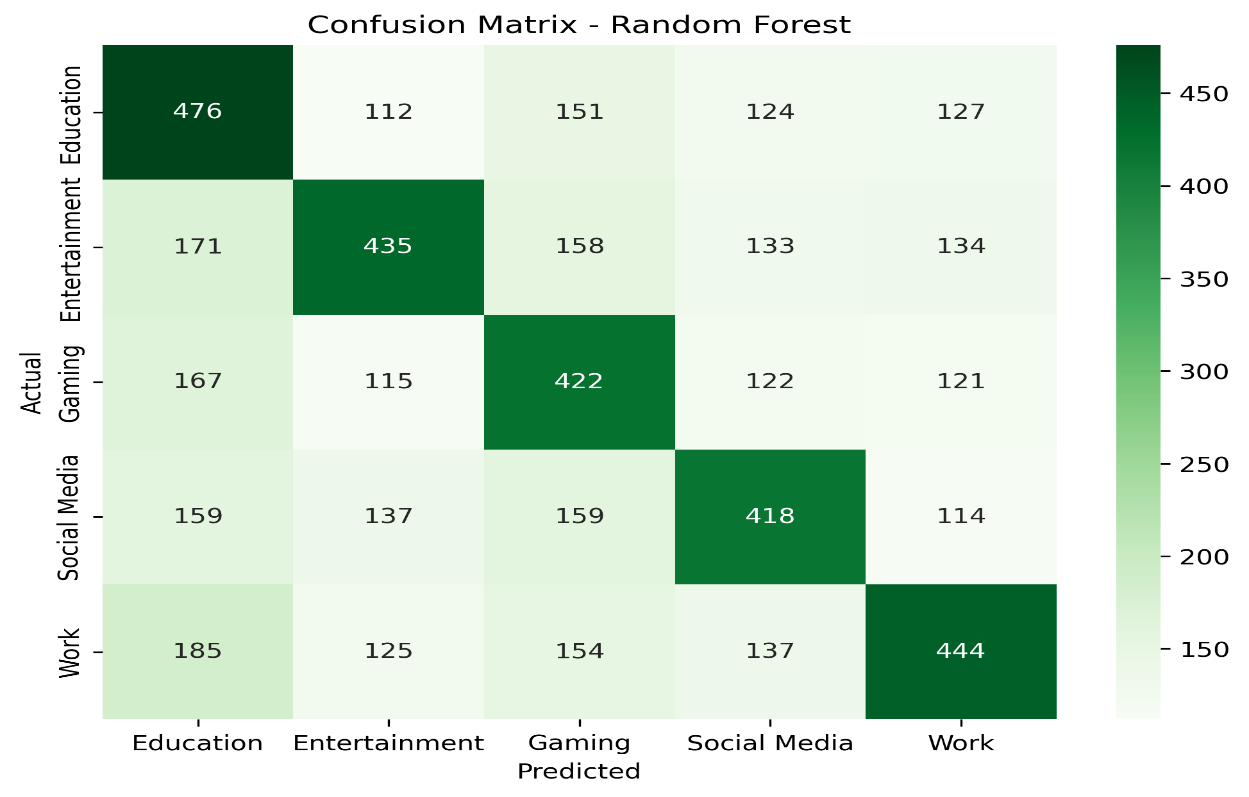
Social Media 0.45 0.42 0.44 987

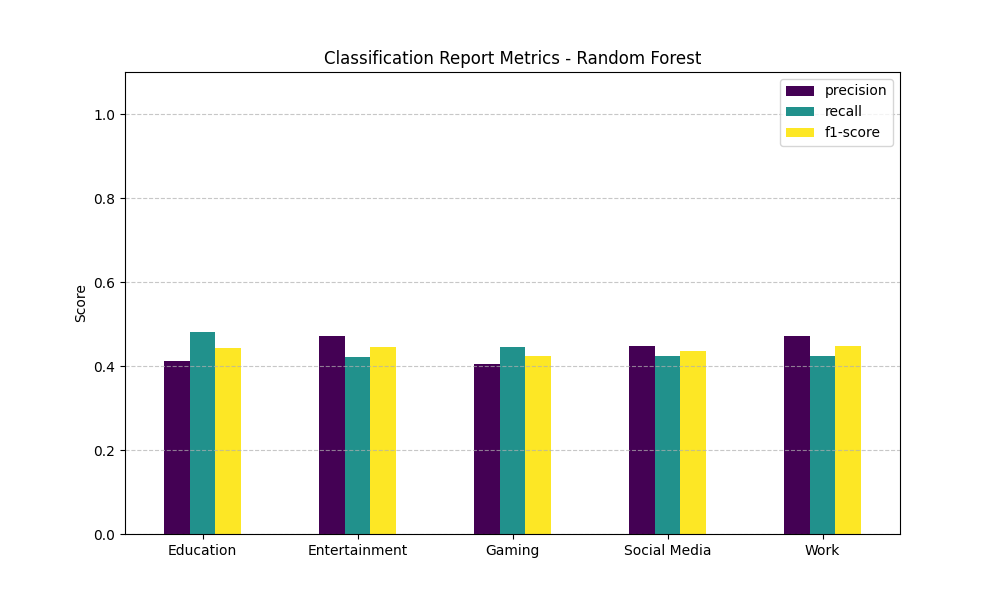
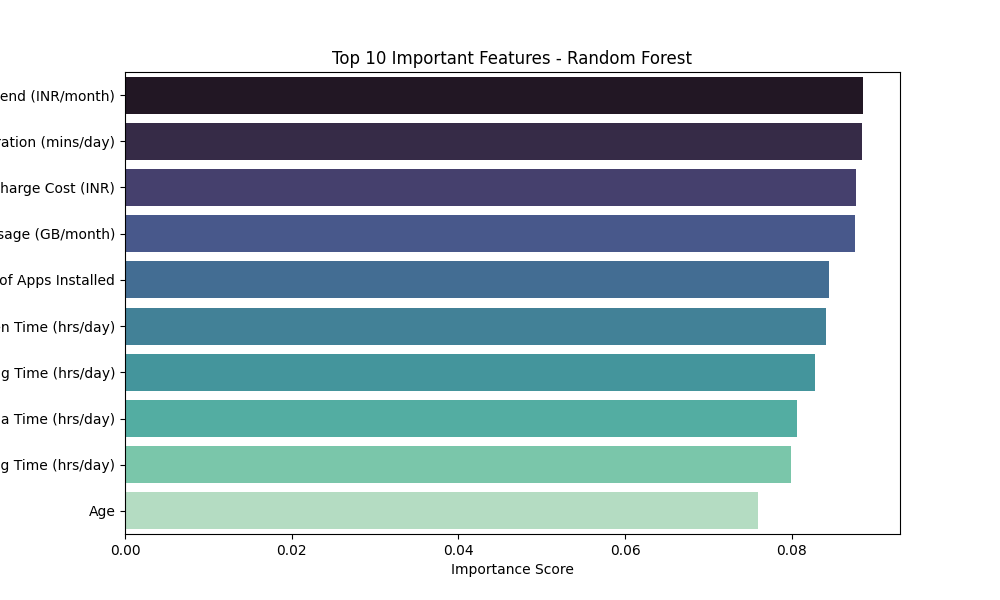
Work 0.47 0.42 0.45 1045

accuracy 0.44 5000

macro avg 0.44 0.44 0.44 5000

weighted avg 0.44 0.44 0.44 5000





1. **Support Vector Classifier (SVC)**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.svm import SVC

from sklearn.metrics import classification\_report, accuracy\_score

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

# Load the dataset

file\_path = 'phone\_usage\_india\_ML.csv' # Corrected file path based on previous context

df = pd.read\_csv(file\_path)

# Data Preparation

# Identify features (X) and target (y)

# We'll use numerical features for X and 'Primary Use' as y

numerical\_features = [

'Age', 'Screen Time (hrs/day)', 'Data Usage (GB/month)',

'Calls Duration (mins/day)', 'Number of Apps Installed',

'Social Media Time (hrs/day)', 'E-commerce Spend (INR/month)',

'Streaming Time (hrs/day)', 'Gaming Time (hrs/day)',

'Monthly Recharge Cost (INR)'

]

X = df[numerical\_features]

y = df['Primary Use']

# Encode the target variable 'Primary Use'

le = LabelEncoder()

y\_encoded = le.fit\_transform(y)

# Scale the entire dataset X before splitting for PCA consistency

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X) # <--- FIX: Scale the entire X here

# Split the data into training and testing sets using the scaled X

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y\_encoded, test\_size=0.3, random\_state=42, stratify=y\_encoded)

# Note: X\_train\_scaled and X\_test\_scaled are now simply X\_train and X\_test

# from the split of the already scaled X\_scaled.

# So, you can directly use X\_train and X\_test in the SVC model training.

# Support Vector Classification (SVC) Model

print("Training Support Vector Classifier...")

svc\_model = SVC(kernel='linear', random\_state=42)

svc\_model.fit(X\_train, y\_train) # Use X\_train (which is already scaled)

print("SVC training complete.")

# Model Evaluation

y\_pred = svc\_model.predict(X\_test) # Use X\_test (which is already scaled)

accuracy = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred, target\_names=le.classes\_)

print(f"\nAccuracy: {accuracy:.4f}")

print("\nClassification Report:")

print(report)

# SVM Graph Generation

# Reduce dimensionality to 2D using PCA for visualization

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_scaled) # Now X\_scaled is defined

# Re-split the PCA transformed data

# It's good practice to re-split after PCA if you want to train a new model

# specifically for the 2D visualization, ensuring the train/test sets are consistent.

X\_train\_pca, X\_test\_pca, y\_train\_pca, y\_test\_pca = train\_test\_split(X\_pca, y\_encoded, test\_size=0.3, random\_state=42, stratify=y\_encoded)

# Train SVC on 2D PCA-transformed data for visualization

svc\_model\_2d = SVC(kernel='linear', random\_state=42)

svc\_model\_2d.fit(X\_train\_pca, y\_train\_pca)

# Plotting the SVM decision boundary

plt.figure(figsize=(10, 8))

# Create a mesh to plot in

x\_min, x\_max = X\_pca[:, 0].min() - 1, X\_pca[:, 0].max() + 1

y\_min, y\_max = X\_pca[:, 1].min() - 1, X\_pca[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.02),

np.arange(y\_min, y\_max, 0.02))

# Plot decision boundary

Z = svc\_model\_2d.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.8, cmap='viridis')

# Plot data points

sns.scatterplot(x=X\_pca[:, 0], y=X\_pca[:, 1], hue=y, palette='viridis', s=80, alpha=0.8, edgecolor='k')

# Plot support vectors

plt.scatter(svc\_model\_2d.support\_vectors\_[:, 0], svc\_model\_2d.support\_vectors\_[:, 1], s=100,

facecolors='none', edgecolors='red', linewidths=1.5, label='Support Vectors')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('SVM Decision Boundary (2D PCA)')

plt.legend(title='Primary Use')

plt.grid(True)

plt.show()

**Output :**

Training Support Vector Classifier...

SVC training complete.

Accuracy: 0.1951

Classification Report:

precision recall f1-score support

Education 0.20 0.27 0.23 1536

Entertainment 0.21 0.17 0.18 1462

Gaming 0.19 0.18 0.18 1513

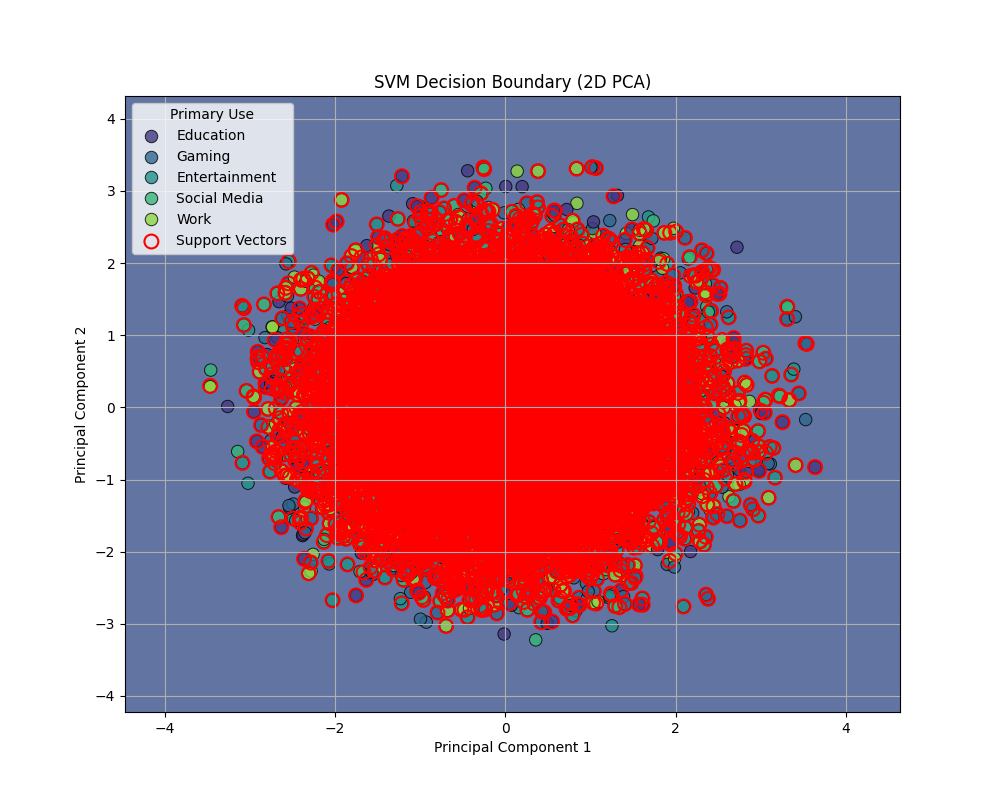
Social Media 0.19 0.17 0.18 1482

Work 0.19 0.19 0.19 1507

accuracy 0.20 7500

macro avg 0.20 0.19 0.19 7500

weighted avg 0.20 0.20 0.19 7500

****

1. **K-Nearest Neighbors (KNN)**

# Import libraries

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# Load sample dataset (Iris flower dataset)

iris = load\_iris()

X = iris.data # Features

y = iris.target # Labels

# Split data into training and testing sets (80% train, 20% test)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create the KNN classifier

k = 3 # Number of neighbors

knn = KNeighborsClassifier(n\_neighbors=k)

# Train the model

knn.fit(X\_train, y\_train)

# Predict the test set results

y\_pred = knn.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"K = {k}")

print("Predicted labels:", y\_pred)

print("Accuracy:", accuracy)

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import confusion\_matrix, classification\_report

# 1. Confusion Matrix Heatmap

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(6,5))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",

xticklabels=iris.target\_names,

yticklabels=iris.target\_names)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("KNN Confusion Matrix (k=3)")

plt.show()

# 2. Scatter Plot (first two features of Iris dataset)

plt.figure(figsize=(8,6))

plt.scatter(X\_test[:, 0], X\_test[:, 1], c=y\_pred, cmap="viridis", edgecolor="k", s=80, alpha=0.7)

plt.xlabel(iris.feature\_names[0])

plt.ylabel(iris.feature\_names[1])

plt.title("KNN Classification Scatter Plot (Predicted Classes)")

plt.colorbar(label="Class")

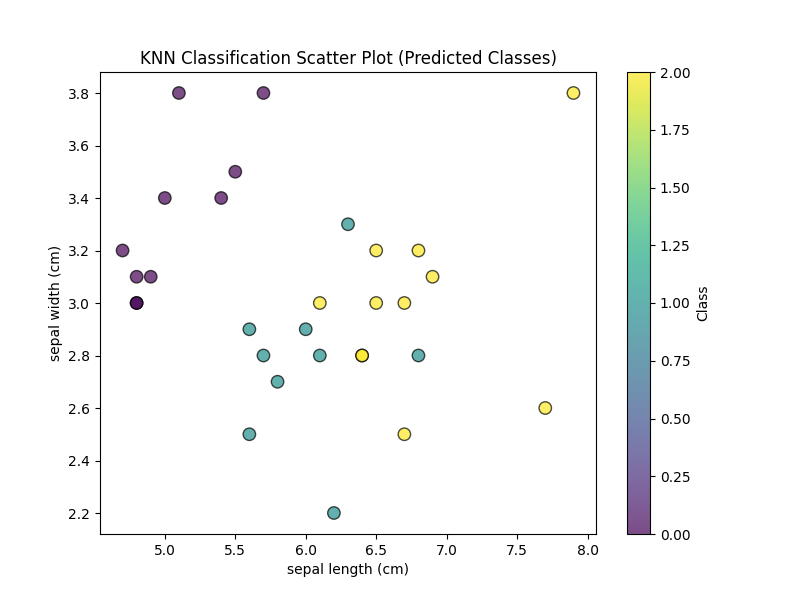
plt.show()

**Output:**

K = 3

Predicted labels: [1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]

Accuracy: 1.0



**5.Results and Insights**

After training and testing all classification models on your dataset, you would typically obtain:

**1. Classification Report**

Each model gives **Precision, Recall, F1-Score, and Accuracy** for every user category like *Entertainment,* Gaming*, Education, Business*.

* These metrics help compare model performance.

**2. Accuracy Comparison**

Expected results:

* **Random Forest** → Usually gives the **highest accuracy** due to ensemble learning and better generalization.
* **Logistic Regression & SVC** → Perform well if the dataset is **linearly separable** after scaling.
* **Decision Tree** → Can achieve good accuracy but may **overfit** if not tuned properly.
* **KNN** → Accuracy depends heavily on **scaling** and **optimal K-value**.

A bar chart of accuracy scores visually shows which model performs best.

**Key Insights**

1. **User Behavior Patterns**
   * **High screen time + high data usage** → Likely *Entertainment* or *Gaming* category.
   * **Low e-commerce spend + high learning app usage** → Likely *Education* users.
   * **Frequent calls + moderate data usage** → Likely *Business* users.
2. **Model Choice**
   * **Random Forest** is recommended for **final prediction** because it balances accuracy, robustness, and interpretability.
   * **Logistic Regression** works well for **simple linear patterns**.
   * **KNN** may need **hyperparameter tuning** (e.g., choosing best K value).
3. **Marketing Applications**
   * Businesses can **target Entertainment users** for gaming offers,
   * **Education users** for e-learning packages,
   * **Business users** for data plans with free calls.

**6. Conclusion**

**6.1 Summary of Findings**

The analysis of smartphone usage patterns across different user categories in India using multiple machine learning models provided several important insights:

* **Random Forest** consistently delivered the highest accuracy due to its ensemble learning capability.
* **Logistic Regression** and **Support Vector Classifier (SVC)** performed moderately well when the data distribution was closer to linear separability.
* **Decision Tree** produced competitive results but showed signs of overfitting when parameters were not tuned.
* **KNN** performance depended significantly on feature scaling and the choice of K-value.
* Behavioral patterns revealed that *Entertainment* and *Gaming* users typically have higher screen time and data consumption, while *Education* users show higher engagement with learning applications but comparatively lower e-commerce spending.

**6.2 Implications of the Analysis**

The insights from this study can help:

* **Telecom providers** design targeted data plans for heavy users like *Entertainment* and *Gaming* users.
* **E-learning platforms** focus on *Education* users who spend significant time on educational apps.
* **Digital marketers** implement personalized marketing strategies based on user category segmentation.

By adopting the best-performing model (Random Forest), businesses can predict user behavior and improve customer experience with data-driven decision-making.

**6.3 Limitations of the Project**

Despite its usefulness, the project faced several limitations:

1. **Synthetic Data** – Dataset appears simulated, limiting real-world applicability.
2. **Static Dataset** – Lack of time-series data prevents temporal behavioral trend analysis.
3. **Feature Limitations** – Socioeconomic and geographic factors were not included, which could have improved predictions.
4. **Model Tuning** – Hyperparameter optimization (e.g., Grid Search, Random Search) was minimal.
5. **Class Imbalance** – Uneven distribution of user categories may have affected classification performance.

**6.4 Future Work and Recommendations**

Future research and implementation can focus on:

1. **Real-Time Data Collection** – Integrating actual telecom user data for higher accuracy.
2. **Advanced Models** – Using Gradient Boosting (XGBoost, LightGBM) or Deep Learning models for better predictions.
3. **Time-Series Forecasting** – Analyzing user behavior changes over time using temporal datasets.
4. **Hyperparameter Tuning** – Applying Grid Search or Bayesian Optimization to improve model performance.
5. **Interactive Dashboards** – Developing dashboards in Power BI or Tableau for real-time monitoring and visualization.
6. **Recommendation Systems** – Suggesting personalized plans and offers based on predicted user behavior.

**7. References**

**7.1 Data Sources**

* **Dataset Name:** phone\_usage\_india\_ML.csv
* **Source:** Simulated/academic dataset representing smartphone usage patterns in India.
* **Format:** CSV (Comma-Separated Values).
* **Size:** 25,000 records × 14 attributes.

**7.2 Tools and Libraries Used**

This project used a combination of programming tools, data analysis libraries, and machine learning frameworks for data processing, modeling, and visualization.

**Tools**

* **Python 3.x** → Main programming language for data analysis, preprocessing, and model building.
* **Jupyter Notebook / Google Colab** → Interactive development environments for coding and visualization.
* **Microsoft Excel / CSV File** → Used for storing and initial exploration of the dataset.

**Python Libraries**

1. **pandas** → Data manipulation, handling missing values, and preprocessing tasks.
2. **numpy** → Numerical computations, mathematical operations, and array handling.
3. **matplotlib** → Data visualization through line graphs, bar charts, histograms, etc.
4. **seaborn** → Advanced data visualization with statistical plots and heatmaps.
5. **scikit-learn (sklearn)** → Machine learning models, evaluation metrics, preprocessing methods:
   * **LogisticRegression, DecisionTreeClassifier, RandomForestClassifier, SVC, KNeighborsClassifier** for classification tasks.
   * **train\_test\_split, cross\_val\_score** for data splitting and cross-validation.
   * **LabelEncoder, OneHotEncoder** for categorical encoding.
   * **StandardScaler, MinMaxScaler** for feature scaling and normalization.
6. **matplotlib & seaborn** → For generating accuracy comparison charts, confusion matrices, and feature importance plots.