Predicting User Behavior Class from Phone Usage

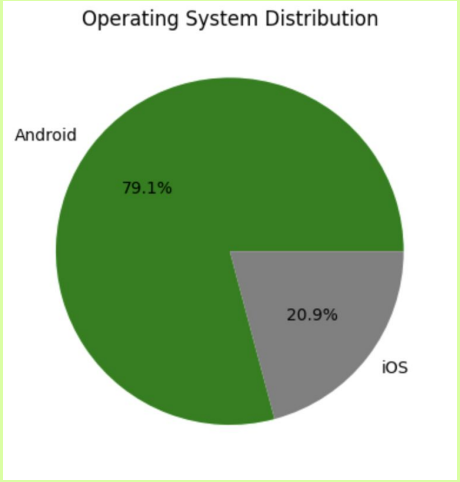
(From Mobile Device Usage and User Behavior on Kaggle.com)

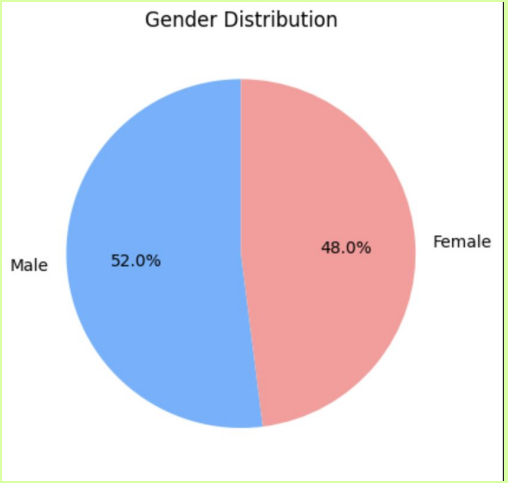
Harsh Bains

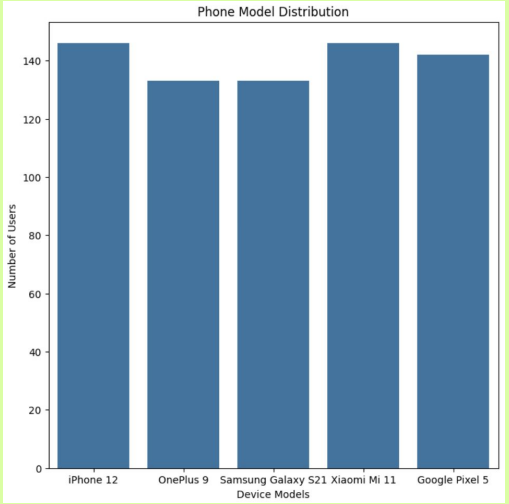
CSCI 1070

1. User ID
2. Device Model (iPhone 12, Google Pixel 5, OnePlus 9, Xiaomi Mi 11, Samsung Galaxy S21)
3. Operating System (Android, iOS)
4. App Usage Time (min/day)
5. Screen On Time (hours/day)
6. Battery Drain (mAh/day)
7. Number of Apps Installed
8. Data Usage (MB/day)
9. Age
10. Gender (Male, Female)
11. User Behavior Class (Target)

Dataset Overview







High-Level Visualization

```python
def dataClean(df: DataFrame = phone_data) -> DataFrame:
    # Loop through categorical columns to clean text data
    for col in df.columns:
        # If the column is categorical, strip any leading or trailing whitespace
        if df[col].dtype == 'object':
            df[col] = df[col].str.strip()
            # Convert all text to lowercase for uniformity
            df[col] = df[col].str.lower()

    return df

phone_data = dataClean(phone_data)
```

Data Cleaning

```python
# Manual encoding for Operating System
def std_os(col: pd.Series) -> pd.Series:
    os = {'ios': 0, 'android': 1}

    # Map the values based on the dictionary
    col = col.map(os)

    return col

phone_data['Operating System'] = std_os(phone_data['Operating System'])

# Manual encoding for Gender
def std_gender(col: pd.Series) -> pd.Series:
    gender_map = {'male': 0, 'female': 1}

    # Map the values based on the dictionary
    col = col.map(gender_map)

    return col

phone_data['Gender'] = std_gender(phone_data['Gender'])
```

Manual Encoding

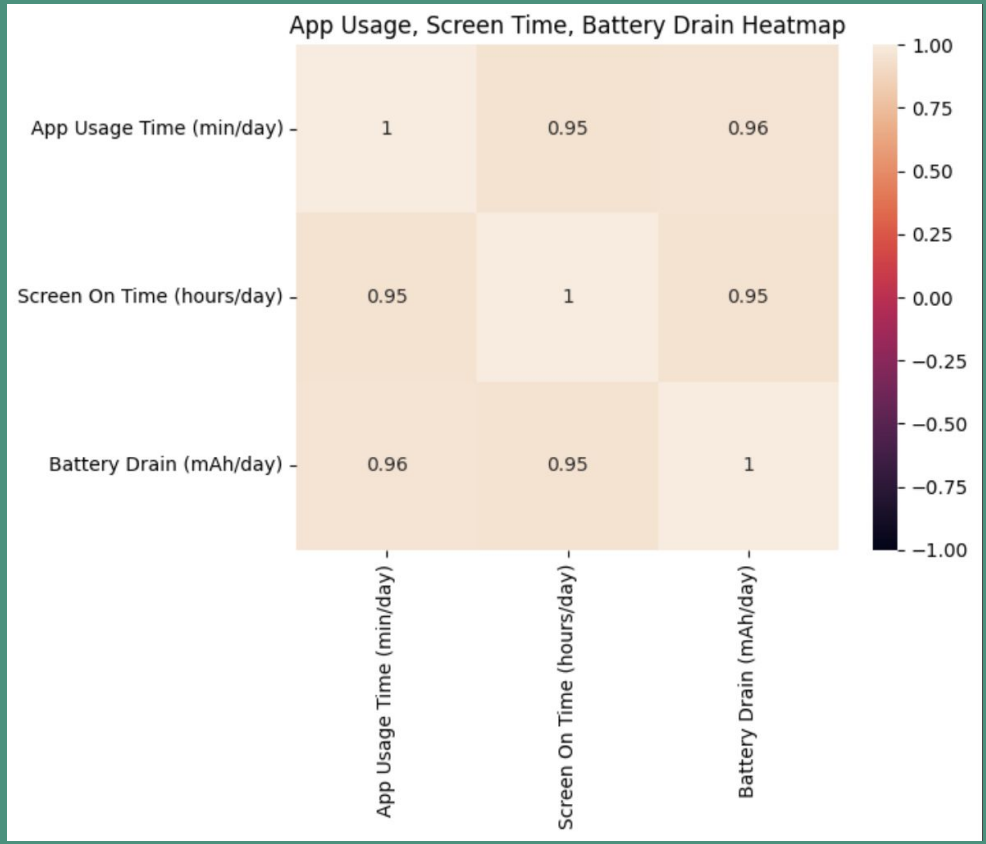
```python
# One hot encode all categorical string columns
def encode(df: pd.DataFrame = phone_data) -> pd.DataFrame:
    # Select only categorical string columns
    for col in df.columns:
        # If the column is categorical, strip any leading or trailing whitespace
        if df[col].dtype == 'object':
            # Encode
            encoded = pd.get_dummies(df)

    return encoded

phone_data = encode(phone_data)
```

One-Hot Encoding

Preprocessing

Why KNN & Decision Trees?

Multicollinearity!



Simplicity
KNN is adaptable and easy to use to learn the correlation in the data.

Robust
It is simply a matter of trial-and-error to find the optimal K-value.

KNN

Mixed Data
We have different data types, and decision trees handle those well.

Feature Importance
By creating a tree graph, we can look at what features are most important when predicting behavior class.

Decision Trees

KNN

99.7% accuracy!

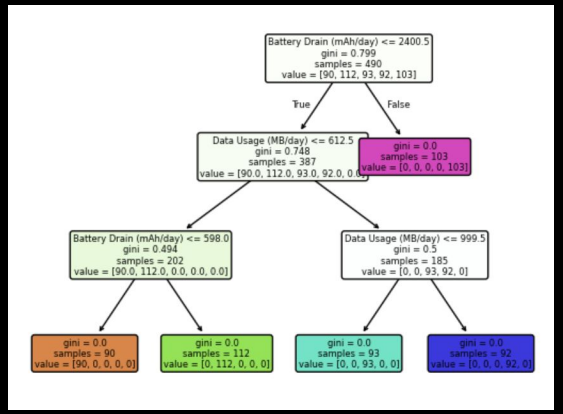
BUT...

But this doesn't prove much as the model might be overfitted, so let's look at decision tree performance.

Model Performance

Decision Trees

	precision	recall	f1-score	support
1	1.00	1.00	1.00	46
2	0.97	1.00	0.99	34
3	1.00	0.98	0.99	50
4	1.00	1.00	1.00	47
5	1.00	1.00	1.00	33
accuracy			1.00	210
macro avg	0.99	1.00	1.00	210
weighted avg	1.00	1.00	1.00	210



Takeaways

1. Only two important features (battery drain and data usage)!
2. KNN might be overfit (requires a bigger dataset to verify)
3. High multicollinearity requires custom, manual preprocessing to ensure acceptable feature selection
4. Begin with elementary techniques when approaching complex problems

Thank you!