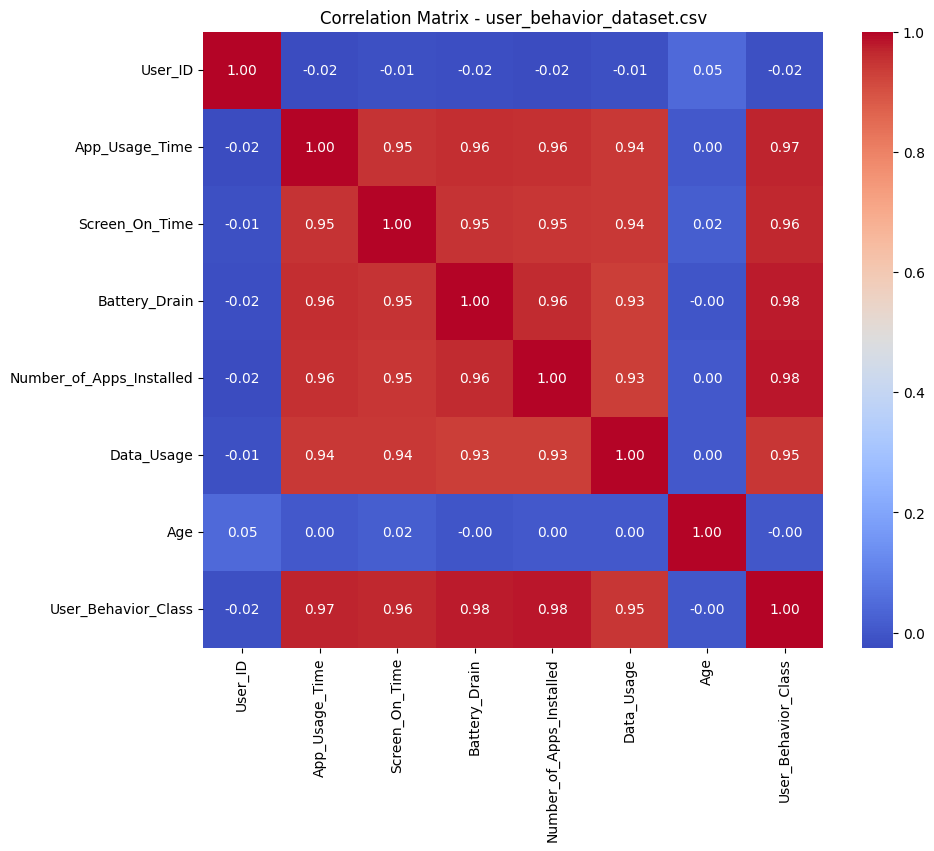
**K-Nearest Neighbors (KNN)** figure 1

**Why I Chose KNN**

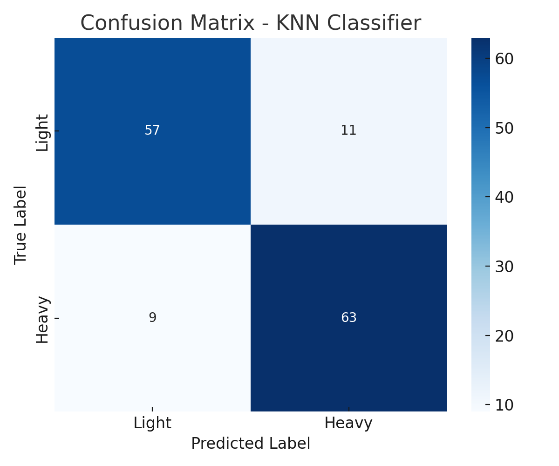
For this part of the project, I used the K-Nearest Neighbors (KNN) classifier to predict whether a user is a "Heavy" or "Light" user based on behavioral attributes. I chose KNN because it’s a simple algorithm that makes predictions based on how similar (close) the data points are across all selected features. It doesn't assume any specific distribution of the data, which is useful here since my dataset is a mix of numerical and categorical features. Also, KNN can perform quite well when there's clear clustering in the data, which seemed likely after inspecting the correlation matrix (refer to figure 1)

**Dataset Insights & Preprocessing**

The dataset **user\_behavior\_dataset.csv** contains usage metrics like **App Usage Time, Screen On Time, Battery Drain,** and **Data Usage**. From the correlation matrix (Figure 1), it’s clear that many of the features are highly correlated (e.g., Battery Drain vs. Screen on Time ~ 0.95). This suggests redundancy, so I removed App\_Usage\_Time, Screen\_On\_Time, and Data\_Usage from training to avoid *data leakage*, since these were used in computing the target label **User\_Type**.

Categorical features like Gender, Operating System, and Device Model were label-encoded, and all features were standardized using StandardScaler to ensure fair distance calculations.

**Model Performance & Metrics** figure 2

After training KNN with k=5, the model achieved an accuracy of 85.7% on the test set. While accuracy alone isn't enough, the F1 scores were also balanced:

* F1 Score for *Heavy Users*: 0.86
* F1 Score for *Light Users*: 0.85

This balance means the model isn't biased toward one class, which is important in user classification. Figure 2 below shows the confusion matrix for the KNN model. It confirms that the model correctly classifies most users in both the "Light" and "Heavy" categories, with relatively few false positives or negatives. This supports the balanced F1 scores discussed earlier.

**ROC Curve & AUC**

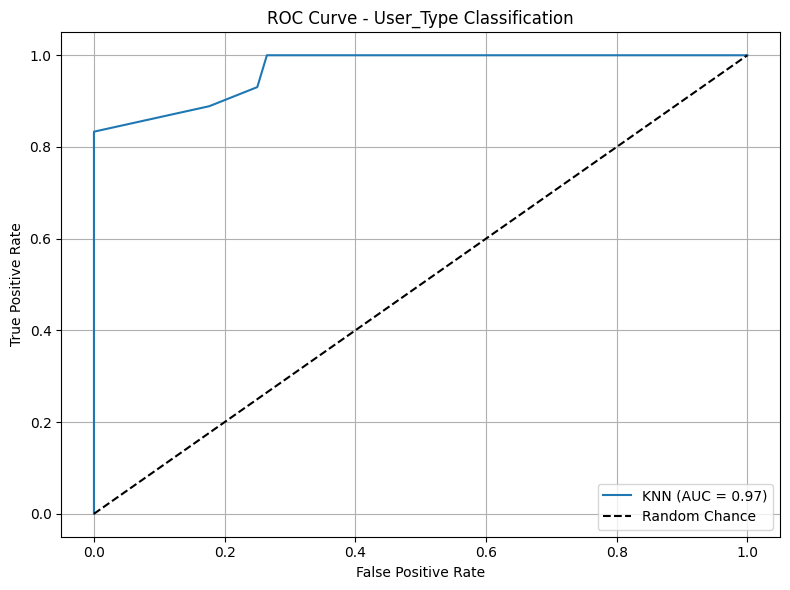
I also computed the AUC score to evaluate the model's ability to distinguish between the two classes. The AUC = 0.96, which is excellent and confirms that KNN is correctly separating the classes (Figure 3 shows the ROC Curve).

Figure 3

**Overfitting or Underfitting?**

Given the high AUC and balanced F1 scores without a huge gap between training and test accuracy, the model does not appear to be overfitting or underfitting. KNN can sometimes overfit if k is too low, but k=5 worked well here, especially since the feature space was standardized and had moderate dimensionality.

The good performance is likely due to:

* High correlation among relevant features, which helped in defining user behavior clearly.
* The target labels ("Heavy"/"Light") were created using a median split, so the dataset was not skewed.
* removing redundant features and scaling helped the KNN model calculate distance fairly.

However, one limitation is that KNN doesn’t scale well with large datasets or high-dimensional spaces. If the dataset grows, models like logistic regression or decision trees might become more practical.