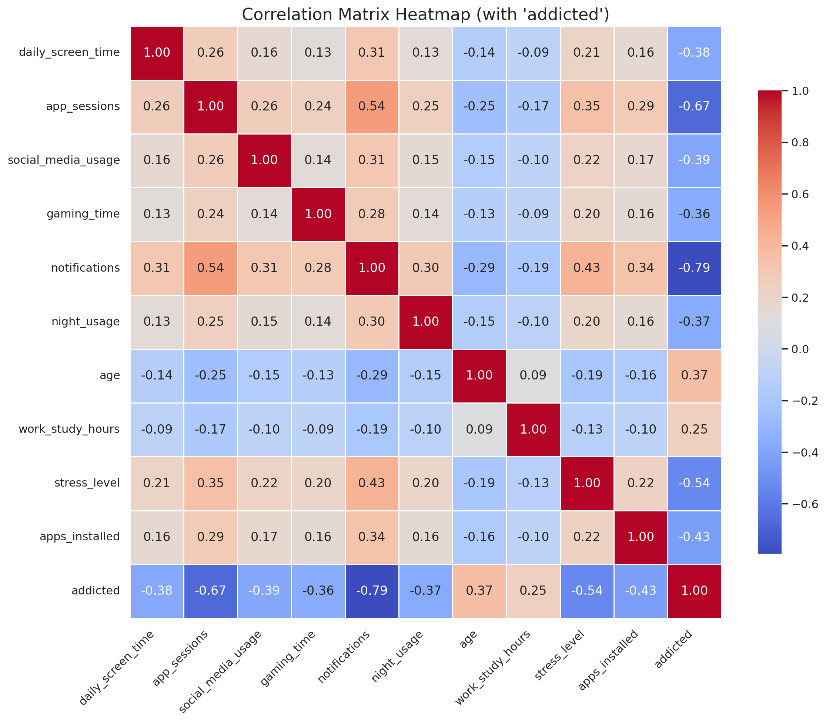
**K-Nearest Neighbors (KNN)** figure 2.1

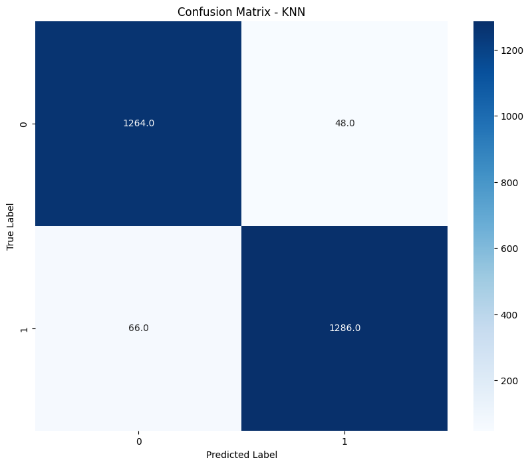
**Why I Chose KNN**

For this part of the project, I used the K-Nearest Neighbors (KNN) classifier to predict whether a user is "Addicted" or "Not Addicted" 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 by calculating the Euclidean distance. It is non-parametric which means it doesn't assume any specific distribution of the data. Since my dataset (mobile\_addiction\_cleaned.csv) is made up of clean numerical features (e.g. daily screen time, app sessions, notifications), it fits KNN very well because these types of models work best with numerical, scaled data. Also, KNN is sensitive to the structure of the data, so it was a good candidate for exploring whether natural clusters of "addicted" vs "not addicted" users exist based on mobile usage behavior.

**Dataset Insights & Preprocessing**

The dataset mobile\_addiction\_cleaned.csv contains usage metrics like Daily screen time, App sessions, social media usage, and gaming time. From the correlation matrix (Figure 1), it’s clear that many of the features are highly correlated with the target label (e.g., notifications vs addicted label ~ 0.79). I removed the work\_study\_hours feature as it did not have significant correlation with the target label. The target label ‘addicted’ was label-encoded, and all features were standardized using StandardScaler to ensure fair distance calculations.

**Model Performance & Metrics** figure 2.2

After training KNN with k=5 and applying standard scaling, the model achieved an accuracy of 97.52% on the test set and an AUC score of 0.99. While accuracy alone isn't enough, the F1 scores were also balanced:

* F1 Score for *Addicted Users*: 0.98
* F1 Score for *Not Addicted Users*: 0.97

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 "Addicted" and "Not Addicted" categories, with relatively few false positives or negatives. This supports the balanced F1 scores discussed earlier.

**Reasons for High Accuracy**

1. **Balanced Dataset:**The target variable addicted is nearly evenly split between the two classes, which helps prevent bias. there were:

* not addicted 6735
* addicted 6583

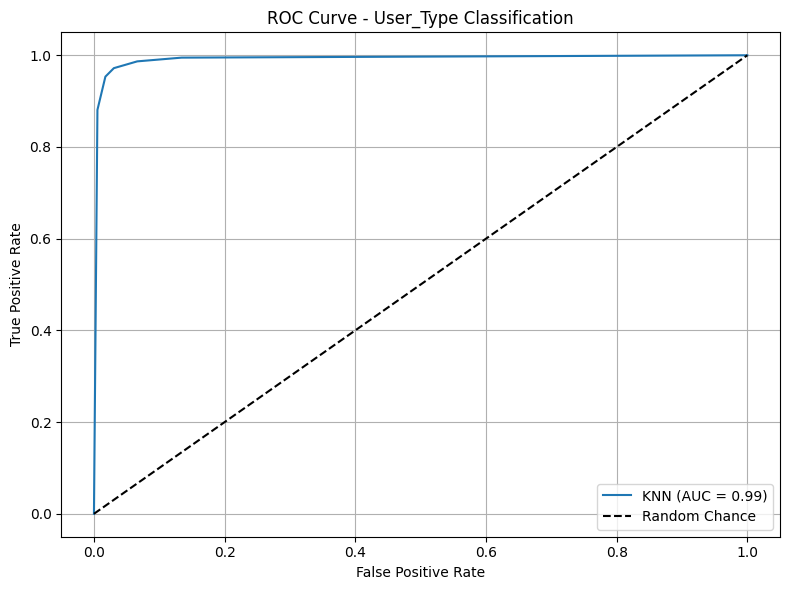
1. **Strong Feature Correlations:**  
   The correlation matrix (see Figure 2.1) shows that features like notifications, app\_sessions, and stress\_level are strongly correlated with the addiction label:

* notifications: -0.79
* app\_sessions: -0.67
* stress\_level: -0.54

These features provide meaningful signals for the classifier.

1. **Numerical and Scaled Features:**KNN is a distance-based algorithm, and it’s very sensitive to the scale of the features. By applying StandardScaler, I ensured all features contributed equally to the distance calculation, improving model accuracy.
2. **Low Noise in the Data:**The features appear to have clear, distinct boundaries between classes, which is ideal for KNN. This helps explain the high precision and recall for both classes.

**ROC Curve & AUC**

I also computed the AUC score to evaluate the model's ability to distinguish between the two classes. The AUC = 0.99, which is excellent and confirms that KNN is correctly separating the classes and not by chance (Figure 3 shows the ROC Curve).

**Overfitting or Underfitting?**

I checked for overfitting by comparing the test AUC score (0.992) with the cross-validated mean AUC score from 5-fold CV (0.989). Since these values are very close, the model generalizes well and does not overfit the training data.

Also, it is not underfitting because:

* The accuracy is high.
* The confusion matrix (see Figure 2.2) shows low misclassification.
* F1 scores and AUC are both strong.

**Decision Tree Classifier**

**Why I Chose the Decision Tree Classifier**

In this part of the project, I applied a Decision Tree Classifier to the mobile addiction dataset to explore how well it can distinguish between "addicted" and "not addicted" users based on various behavioral features. A decision tree is a supervised machine learning algorithm that splits the dataset into subsets based on feature values, creating a tree-like structure where each parent node represents a feature decision, each branch represents an outcome, and each leaf node represents a class label (in this case, not addicted = 0 or addicted = 1).

**Why Decision Tree?**

I chose a Decision Tree because:

* It visually represents how decisions are made.
* It is non-parametric meaning it makes no assumptions about data distribution, which is ideal given that many of the features in the dataset are not normally distributed.
* It is effective in modelling complex feature interactions like those observed between social media use, notifications, and stress level.

The core concept behind a decision tree is selecting the best feature or best split to split the data at each node. This is done using impurity measures like Information Gain (Entropy). For example, Entropy is calculated as:

A black and white image of a symbol

AI-generated content may be incorrect.where pi is the probability of class i. The algorithm calculates the entropy for each node (parent and children) then calculates the information gain using the following equation:

A black text on a white background

AI-generated content may be incorrect.where wi is the weight of the ith child, it is calculated by:

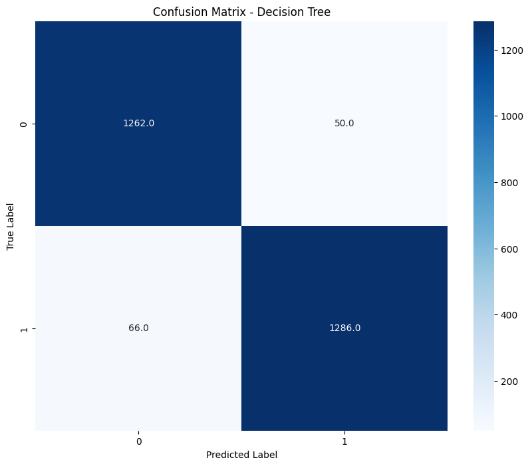
A screenshot of a math equation

AI-generated content may be incorrect.The algorithm tries every possible split and chooses the split with the highest information gain. NOTE: a “pure” state (all samples belong to one class) has an entropy of 0, while an entropy of 1 indicates equal probabilities of each class (in binary classification, each class would have a probability of 0.5)

**Dataset Insights & Preprocessing**

For this classifier, I didn’t use the standard scaler since it was not needed. I also encoded the addicted label.

**Model Performance & Metrics** figure 2.2

The model achieved an accuracy of 95.65% on the test set and an AUC score of 0.96. the F1 scores were once again balanced:

* F1 Score for *Addicted Users*: 0.96
* F1 Score for *Not Addicted Users*: 0.96

Figure 2 below shows the confusion matrix for the Decision tree model.

**Reasons for High Accuracy**

Like KNN, the balance in classes is a reason for the high accuracy of this model. Low noise in data is another factor.

**ROC Curve & AUC**

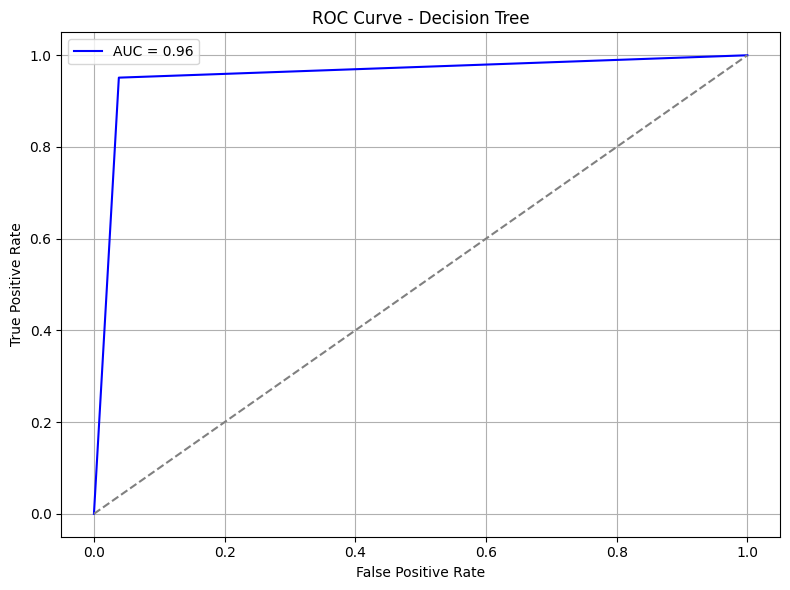
The model achieved an AUC score of 0.96, which suggests that the model has a very high ability to distinguish between the two classes

Figure 2.3

**Overfitting or Underfitting?**

To test the model’s generalization ability, I applied 5-fold cross-validation using the AUC metric. The mean AUC across folds was:

* Mean Cross-validated AUC: 0.956

This consistency across folds confirms the model is generalizing well and is not overfitting to the training data.

**Model Fit & Data Properties**

Compared to the KNN model, the decision tree had slightly lower performance metrics but was still very strong. One possible reason is that KNN is sensitive to local patterns, while Decision Trees may overfit to noise if not pruned. However, in this case, the tree seems to be well-balanced. The features in the dataset are moderately correlated (as seen in the correlation matrix), so the tree can effectively pick key splits without being overwhelmed by multicollinearity.

There’s no sign of underfitting, since the accuracy and AUC are high both in training and cross-validation. Also, no significant drop in performance indicates it is not overfitting either.