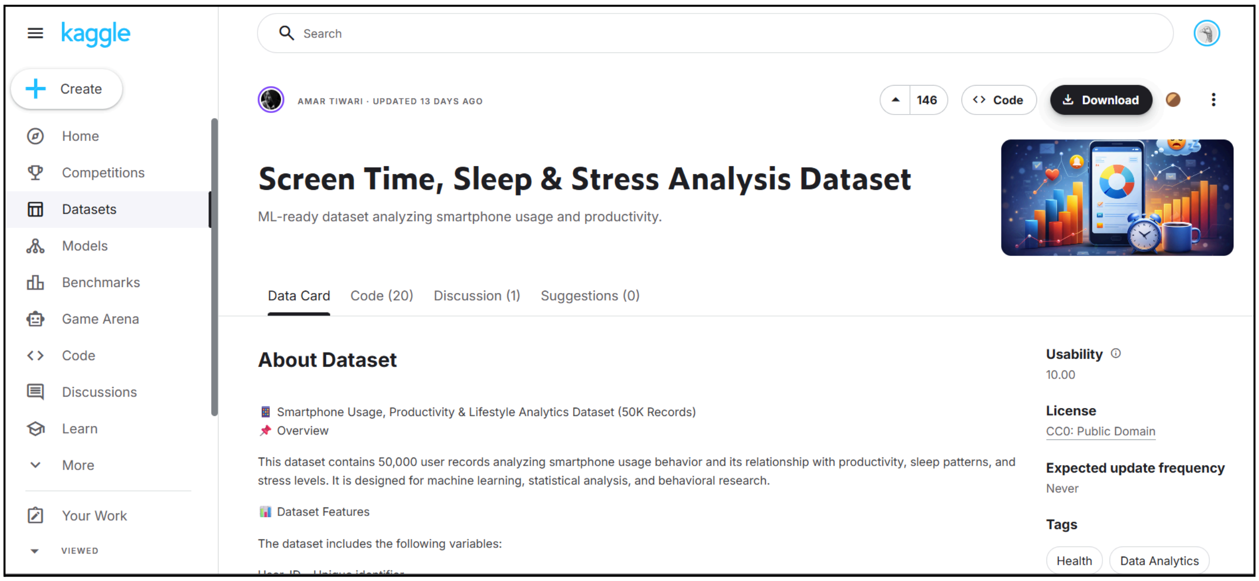
Monica Ahuja **D15B / Batch B / 25**

EXPERIMENT 4

Aim of the Experiment

Implementation of K-Nearest Neighbors (KNN) and Model Evaluation

Theory



Dataset Description

The Smartphone Usage Productivity Dataset contains behavioral and lifestyle features related to smartphone usage patterns and their effect on productivity.

Dataset Size:

- 50,000 records
- 13 features

Features Include:

- User_ID
- Age
- Gender
- Occupation
- Device_Type
- Daily_Phone_Hours
- Social_Media_Hours
- Work_Productivity_Score
- Sleep_Hours

- Stress_Level
- App_Usage_Count
- Caffeine_Intake_Cups
- Weekend_Screen_Time_Hours

Target Variable

Since KNN is a classification algorithm, the continuous variable:

Work_Productivity_Score

was converted into 3 categorical classes using binning:

- 0 → Low Productivity
- 1 → Medium Productivity
- 2 → High Productivity

Thus, the task becomes a multi-class classification problem.

Mathematical Formulation of KNN

K-Nearest Neighbors is a distance-based classification algorithm.

Given a test sample x, KNN computes distance from all training samples.

Euclidean Distance Formula:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

After computing distances:

$$\hat{y} = mode(y_1, y_2, \ldots, y_k)$$

Where:

- K = number of neighbors
- mode = majority voting

Algorithm Limitations

KNN has several limitations:

1. Computationally expensive for large datasets
2. Sensitive to irrelevant features
3. Performance depends heavily on K value
4. Requires feature scaling

5. Struggles when class boundaries overlap
6. Performs poorly when dataset has high dimensionality

In this experiment, overlapping productivity classes reduced accuracy.

Methodology / Workflow

Step 1: Data Collection

- Load dataset into Google Colab

Step 2: Data Preprocessing

- Check missing values
- Convert productivity score into categorical classes
- Drop unnecessary columns (User_ID)
- Encode categorical features
- Perform train-test split (80%-20%)
- Apply StandardScaler

Step 3: Model Training

- Initialize KNN classifier
- Set K = 5
- Train on training dataset

Step 4: Model Evaluation

- Predict on test data
- Calculate:
    - Accuracy
    - Confusion Matrix
    - Precision
    - Recall
    - F1-score

Performance Analysis

Accuracy

Accuracy = **35.97%**

The model achieved moderate performance.

Since the dataset contains 3 classes, random guessing would give ~33% accuracy. The model slightly improves over random baseline.

Confusion Matrix

$$\begin{bmatrix} 2329 & 998 & 659 \\ 1755 & 781 & 480 \\ 1748 & 763 & 487 \end{bmatrix}$$

The confusion matrix shows:

- High misclassification between classes
- Significant overlap between productivity levels
- Poor separation of class boundaries

Classification Report Summary

Class	Precision	Recall	F1-Score
0	0.40	0.58	0.47
1	0.31	0.26	0.28
2	0.30	0.16	0.21

Observations:

- Class 0 performs better
- Class 2 has very low recall
- Model struggles to distinguish medium and high productivity

---

Hyperparameter Tuning

To improve performance, different K values were tested:

K values tested: 1 to 20

A graph of K vs Accuracy was plotted.

Observations:

- Small K → overfitting
- Large K → underfitting
- Optimal K chosen based on maximum accuracy

However, even after tuning, performance improvement was limited due to overlapping class distribution.

Final Conclusion

In this experiment, the K-Nearest Neighbors (KNN) algorithm was implemented on the Smartphone Usage Productivity dataset for multi-class classification.

Although the model achieved an accuracy of 35.97%, performance was limited due to:

- Overlapping class distributions
- Similar behavioral patterns across productivity levels
- High feature interaction complexity

The experiment demonstrates that:

- KNN requires well-separated classes
- Feature scaling is essential
- Hyperparameter tuning is necessary but may not always significantly improve performance

This experiment highlights the importance of dataset structure in determining the effectiveness of distance-based algorithms like KNN.