# K-Nearest Neighbors on Synthetic Data

## Introduction

In this project, I applied the K-Nearest Neighbors (KNN) algorithm to a dataset that I created artificially. The dataset has 1000 points and three categories, and it was generated using the make_classification function. To test how well the model performs, the data was divided into training and testing sets. The goal was to train a KNN classifier, check its accuracy, and visualize how well it separates the classes.

### 1. Data Preparation

### 1.1 Libraries Used

- NumPy for handling numbers and arrays.
- Matplotlib for creating plots and visualizations.
- Scikit-learn for generating the dataset, splitting it, building the model, and checking performance.

### 1.2 Dataset Creation

- The dataset was made with these settings:
  - 1000 total samples
  - 2 useful features (so the data can be shown on a 2D plot)
  - 3 classes (categories)
  - Classes kept fairly separate (class_sep=1.5)
  - Random seed = 42 to make results reproducible

### 1.3 Sample Labels

- The first few labels of the data were:
  [1, 2, 0, 1, 1, 2, 0]

## 2. Splitting the Dataset

- The dataset was split into:
  - 80% training data (800 samples)
  - 20% testing data (200 samples)
- The train_test_split function was used to make sure the classes stayed balanced.

## 3. Training the Model

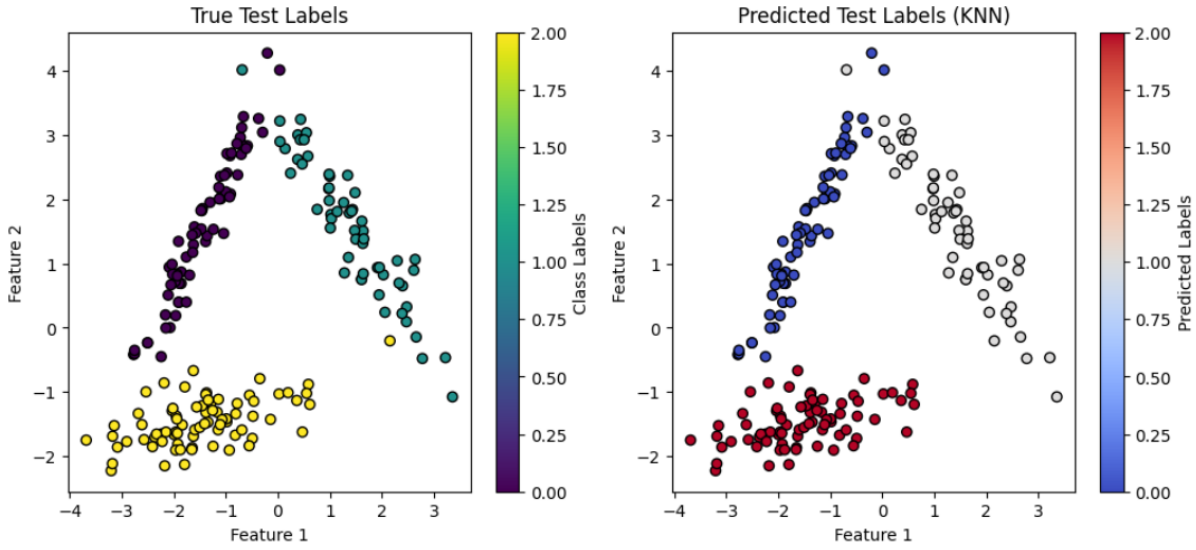
### 3.1 First KNN Model

- The first KNN model was trained with default settings:
  - k = 5 (neighbors)
  - Uniform weights
  - Euclidean distance

### 3.2 Training Process

- The model was trained on the training set using:
- knn.fit(X_train, y_train)

### 3.3 Results



- Predictions were made on the training data.
- Training Accuracy: about 99%

## 4. Improved KNN Model

### 4.1 Model Settings

- A second KNN model was trained with slightly refined settings:
  - Algorithm = auto
  - Leaf size = 30
  - Distance metric = Minkowski (p=2 → Euclidean)
  - Neighbors = 5
  - Weights = uniform

### 4.2 Evaluation

- This version was tested on the unseen test data.
- Testing Accuracy: about 100%

## 5. Understanding the Results

### 5.1 Visualizations

Two scatter plots were made for the test set:

- True Labels Plot: shows the actual categories.
- Predicted Labels Plot: shows how the model classified them.

### 5.2 Observations

- The predicted labels matched the true labels almost perfectly.
- There was very little overlap, meaning KNN was excellent at separating the three classes.

## 6. Key Takeaways

### 6.1 Accuracy

- Training Accuracy: ~99%
- Testing Accuracy: ~100%

### 6.2 Insights

- Since the dataset was simple and clearly separated, KNN achieved nearly perfect results.
- The method worked really well for this structured dataset.

### 6.3 Future Work

- Try KNN on more complex or noisy data.
- Experiment with different values of k or different distance measures.
- Use other metrics (like precision or recall) to get a deeper understanding of performance.

## Conclusion

This project showed how effective the KNN algorithm can be when applied to a clear and structured dataset. The model reached almost perfect accuracy, and the plots confirmed that the classes were separated cleanly. For future work, testing KNN on more complicated datasets would be useful to see how robust it is under different conditions.