Assignment 1: Machine Learning with Simulated Dataset

Introduction

The primary objective of this assignment was to gain hands-on experience with the workflow of a basic machine learning project using Python. The reference example provided in the template project made use of the well-known **Iris dataset**, which is a real-world dataset widely used in introductory machine learning studies.

To extend this work, I created a **synthetic dataset** using scikit-learn's make_blobs function. The benefit of using a synthetic dataset is that it allows us to control the number of features, the number of classes, and the separability between classes. By doing so, we can simulate real-world classification tasks in a controlled environment and evaluate how different algorithms perform.

In this assignment, the chosen classifier was **K-Nearest Neighbors (KNN)**, which is one of the simplest yet powerful supervised learning methods. The project covered the full pipeline: dataset generation, preprocessing, train/test splitting, model training, evaluation, and visualization of decision boundaries.

Dataset and Preprocessing

The dataset was generated with the following configuration:

- **Number of samples**: 1000
- **Number of features**: 2 (to allow easy visualization in 2D space)
- **Number of classes**: 3 (three distinct cluster centers were defined)
- **Cluster centers**: [2, 4], [6, 6], [1, 9]

Once generated, the dataset was split into training and test subsets:

- **Training set**: 800 samples (80%)
- **Test set**: 200 samples (20%)

This split ensured that the model had sufficient data to learn from, while the test set remained untouched during training, enabling unbiased evaluation of model performance.

A scatter plot of the dataset confirmed that the clusters were fairly well separated, which suggested that classification should be relatively straightforward for distance-based algorithms like KNN.

Model and Method

The **K-Nearest Neighbors (KNN)** algorithm was chosen for this study. KNN is a non-parametric method that makes predictions based on the majority class among the *k* nearest data points in the feature space. Its simplicity makes it an excellent choice for exploring how dataset structure influences model performance.

The model configuration was as follows:

- Number of neighbors (*k*): **3**
- Distance metric: **Euclidean distance (p=2 in Minkowski metric)**
- Weights: **Uniform** (all neighbors contribute equally to classification)

The model was trained on the training set, and then predictions were generated for both the training and test sets. To further illustrate model behavior, a **decision boundary plot** was created by applying the trained KNN model to a fine grid of points across the feature space. This showed how the model partitions the space into different class regions.

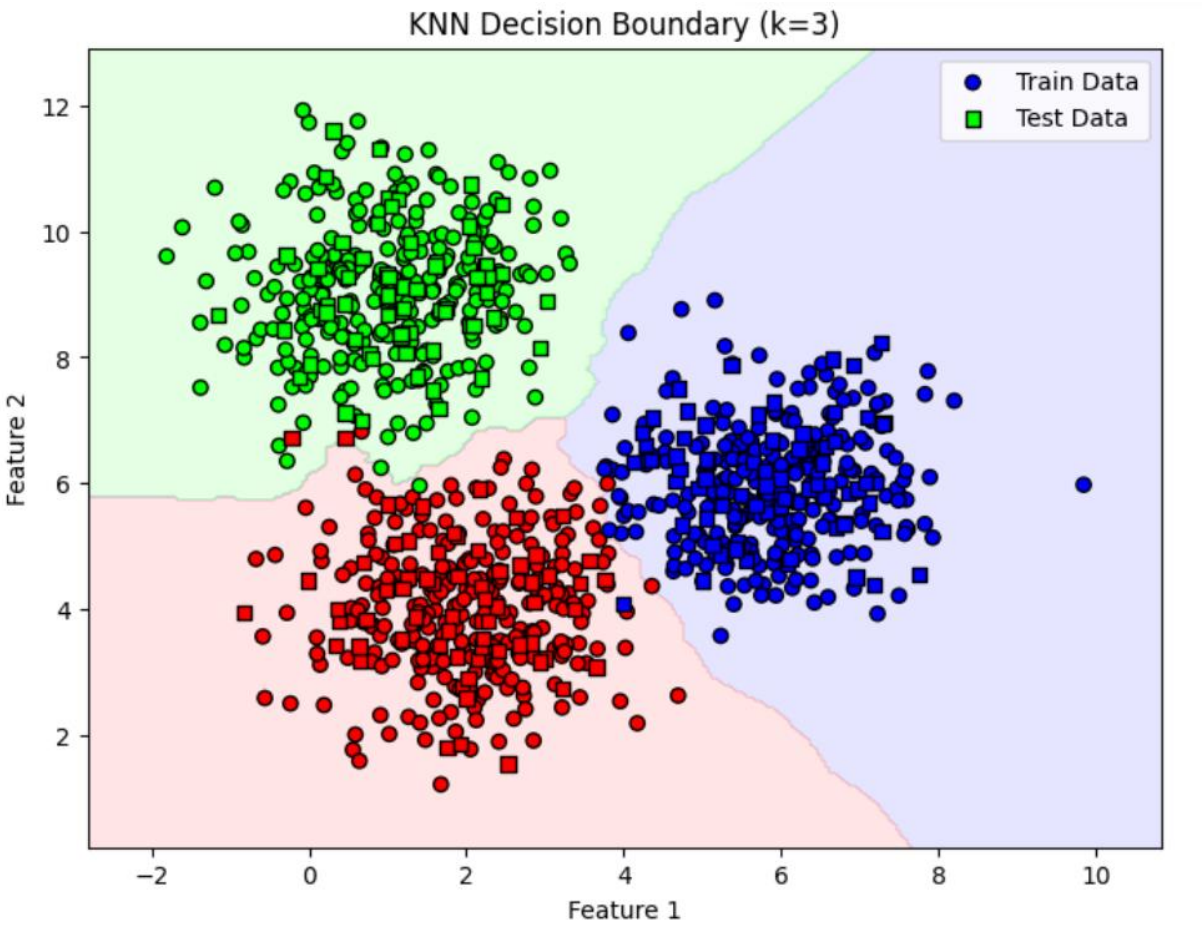
Results

The KNN model produced strong results on both the training and test sets:

- **Training Accuracy**: 0.99
- **Test Accuracy**: 0.98

The decision boundary visualization highlighted that the three clusters were separated into distinct regions, with only minimal overlap at the edges where class boundaries intersected. These overlapping areas are expected to produce the few misclassifications observed.

Overall, the classifier generalized very well, as the training and test accuracy values were nearly identical.



Discussion

Several observations can be drawn from these results:

1. **Dataset Characteristics**
   The high accuracy values can be attributed to the clear separation between clusters in the synthetic dataset. Since the clusters were well-defined with little noise, the classification task was relatively easy.
2. **KNN Performance**
   KNN performed exceptionally well under these conditions. Its reliance on local neighborhoods made it well-suited to the structure of the data. However, it is important to note that KNN can struggle when:
   - The data has high dimensionality (curse of dimensionality).
   - Classes are heavily overlapping.
   - The dataset is very large (because KNN requires storing all training data and computing distances at prediction time).
3. **Comparison to Real Data**
   Unlike real-world datasets such as the Iris dataset, synthetic datasets often lack noise, measurement errors, or overlapping features. This explains why the performance here is higher than what is typically observed with real-world data.

4. **Parameter Sensitivity**
   The choice of *k* is crucial in KNN. A smaller *k* (like 3) can capture fine-grained decision boundaries but risks overfitting. Larger *k* values (e.g., 7 or 9) smooth out the boundaries but may underfit. Exploring different *k* values could further demonstrate this trade-off.

Conclusion

This assignment successfully demonstrated the application of a basic machine learning workflow using Python. By moving from the Iris dataset to a simulated dataset, I gained insights into how data structure affects model performance.

Key takeaways include:

- KNN is highly effective on well-separated data with low noise.
- Synthetic datasets provide a controlled environment for testing algorithms, but may not fully reflect the challenges of real-world data.
- Visualization of decision boundaries is a valuable tool for understanding classifier behavior, particularly with 2D datasets.

Overall, this exercise reinforced the importance of understanding both the dataset and the model, as their interaction determines the ultimate performance of a machine learning system.