Assignment 1: Machine Learning Project in Python using K-Nearest Neighbors Classification

Objective

This project applies the K-Nearest Neighbors (KNN) algorithm to classify data generated from a synthetic dataset. The task is to separate the data into three distinct classes and measure how accurately the model performs. The dataset is created using Scikit-Learn's make_blobs(), then divided into training and test sets. The KNN classifier, based on Euclidean distance, is trained and evaluated through accuracy scores and a confusion matrix, with visualizations included to better understand results.

Synthetic Dataset Creation

The dataset consists of 150 observations distributed across three classes. Each observation has two numerical variables. The class centers are located at:

- Class 1: (2, 4)
- Class 2: (6, 6)
- Class 3: (1, 9)

The spread of points around these centers ensures clear class separation.

Data Preparation: Splitting the Dataset

The dataset is split into training (80%) and testing (20%) portions using the train_test_split function. This allows the model to be trained on one subset and tested on unseen data, ensuring a fair evaluation of its performance.

KNN Classifier

The classifier is configured with the following settings:

- Neighbors (n_neighbors = 5): The prediction is based on the five nearest data points.
- Distance metric (p=2): Euclidean distance is used to measure closeness.
- Weights (uniform): Each neighbor contributes equally to the classification.
- Leaf size (30): Determines the number of points checked at each node when using tree-based search.
- Jobs (n_jobs=1): A single processor core is used for faster computations.

The KNN approach assigns new data points to the class most frequently represented among their nearest neighbors. With well-separated clusters, strong performance is expected.

Model Evaluation

- Training Accuracy: **1.0**
- Test Accuracy: **1.0**

Both the default model and the version with specific parameters achieved perfect accuracy on training and testing data. This indicates that the classifier not only fits the training set but also performs perfectly on the unseen test set.

Visualization

The bar chart showed that both the **default KNN** and the **specific KNN (with tuned parameters)** achieved **perfect accuracy (100%)** on both training and testing data. There is **no difference in performance** between the default and specific KNN models in this case, as both yield equally high accuracy

The confusion matrix, visualized as a heatmap, contains values only on the diagonal. This reflects that every test observation was classified into the correct category, with no misclassifications.

Conclusion

- The dataset is clearly separable, making KNN a suitable method for classification.
- The classifier achieved perfect accuracy, showing its effectiveness on this dataset.
- The small dataset makes KNN computationally efficient.
- For larger or more complex datasets, alternatives like Decision Trees or SVMs may scale better.
- The chosen k = 5 value strikes a reasonable balance between bias and variance.
- Further experiments could test different values of k for optimization.

Future Work

- Apply KNN to larger and more complex datasets that include noise, imbalance, or missing values.
- Use cross-validation to evaluate the model more rigorously and tune the hyper-parameters.
- Experiment with different distance metrics and weighting schemes to see their effect on performance.
- Extend hyperparameter tuning beyond n_neighbors to improve model robustness for larger and data with for features.