ASSIGNMENT 1: A Machine Learning Project with Python

Nishanth Gurujala
09-08-02025

Objective:

In this project, K-Nearest Neighbors (KNN) algorithm is utilized to make classifications of a synthetic dataset with three different classes. The main objective is to properly classify the data and test the output of the model. We start with creating the data by using make_blobs of Scikit-Learn that makes sure that the clusters are separated. Once we split the data into training and tests, we then combine an Euclidean distance based KNN classifier. The accuracy scores and a confusion matrix are used to determine model effectiveness. At last, we visualize the dataset distribution and the performance of the classifier.

Dataset Creation:

The artificial data set has a size of 150, having 150 observations that are of one out of three classes and where two numerical characteristics characterize them. The centers of classes are characterized as:

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

Data Preparation:

The dataset is split into training (80%) and testing (20%) subsets using train_test_split.

Model Technique- KNN Classifier:

For this project, we use the K-Nearest Neighbors (KNN) classifier with the following parameters:

- Number of neighbors (n_neighbors): 3
- p=2 (Euclidean distance metric)
- weights='uniform' (all neighbors have equal influence)
- leaf_size=30 (controls tree search efficiency)
- n_jobs=1 (single parallel process for computation)

KNN works by classifying a new data point to the most frequently occurring class among its nearest neighbors. Given that our dataset has clearly separated classes, we expect KNN to perform well.

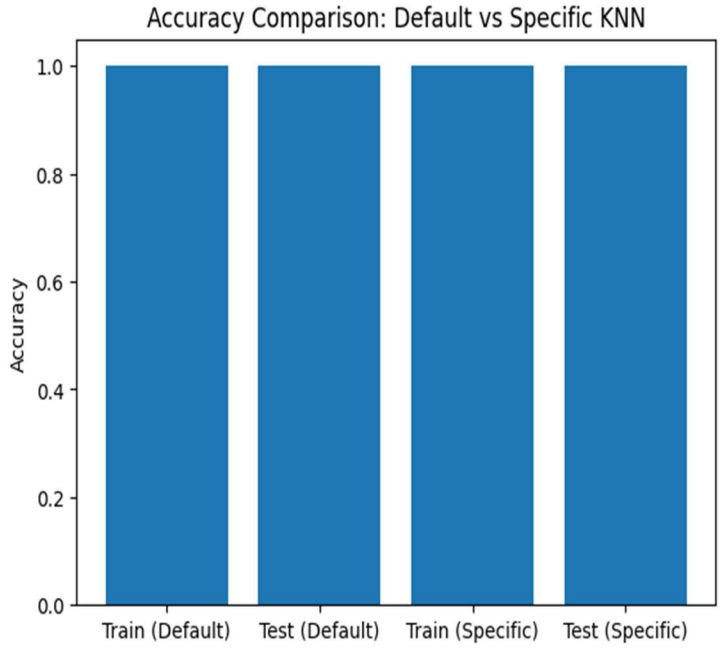
Model Performance and Evaluation:

- Training Accuracy: 1.0
- Test Accuracy: 1.0

The default and the parameter-specified KNN models had both attained a perfect classification both on training and testing sets indicating excellent generalization on unknown data.

Data Visualization:

Scatter plots demonstrate that there is evident division between the classes of both training and test set, which proves that the dataset is appropriate in the KNN classification. Performance is further confirmed with a confusion matrix and a heatmap that indicate a 100 percent and no mislabels diagonal classification.



Conclusion:

- KNN is a suitable classifier because the dataset can be separated linearly.
- The accuracy of the model was immaculate, which proves its reliability.
- KNN was both effective and efficient with a small dataset.
- To handle more data, a scalable model, e.g. a SVM or Decision Tree could be more realistic.
- The trade-off between bias and variance was satisfactory with the use of k= 3.
- The exploration in the future might consider trying other k values to optimize performance.

Future Recommendations:

- As the dataset is the simplest and artificial, its clarity resulted in an ideal accuracy. The kind of real-world data will have noise, imbalance, and missing values that would complicate classification.
- A more reliable estimation of performance could be made through the use of cross-validation and to minimize the chances of overfitting.
- Using KNN on more complicated or noisy data would help us better understand its practical usefulness.
- Other hyperparameter optimization (varying the number of neighbors or using distance measures not based on simple count) would further optimize the results.