Exploring K-Nearest Neighbors (KNN) Algorithm with the Iris Dataset

This homework explores the application of k-nearest neighbors (KNN) algorithm using the famous Iris dataset. Concepts covered are Standardization of features, cross-validation, selecting the optimal k value, visualizing decision boundaries, and evaluating model performance.

KNN1.py explores the different values of k and choosing the optimal value.

KNN2.py visualizes decision boundaries for various values of K combining various features of the dataset.

Steps involved:

1. Loading Dataset:

Import the Iris dataset which contains measurements of sepal length, sepal width, petal length, and petal width for three species of iris flowers (setosa, versicolor, and virginica).

2. Splitting Dataset for Cross-Validation:

Segment the dataset into training and testing subsets to facilitate cross-validation. Splitting was done before scaling to reduce data leakage.

3. Rescaling the Dataset:

Normalize or standardize the dataset to ensure uniformity in feature scales. StandardScaler from scikit-learn was used to scale the features. Standard scaler scales features to have zero mean and unit variance.

4. Experimenting with Different Values of K:

Various values of K (range of 2 - 10) were used to observe their impact on the model's performance. K-fold cross-validation was implemented.

5. Selecting Optimal K:

Evaluate the performance metrics for different K values and determine the most suitable K for the KNN model. Accuracy metric was used, and the best K was chosen as 3.

6. Evaluation Metrics for Test Data:

The performance of the final KNN model on the test data was asessed, utilizing metrics such as accuracy, precision, recall, or other relevant evaluation measures.

7. Visualize the Decision Boundary using various value of K:

Although KNN lacks an explicit formula, you can visualize the decision boundary by plotting regions assigned to different classes. This was done with various values of K.

Conclusion and Findings:

- With a small value of k (e.g., k=1), the decision boundaries tend to be more complex and overfit the training data, resulting in a highly irregular decision boundary.
- As the value of k increases (e.g., k=5 or k=9), the decision boundaries become smoother and more generalized, capturing the overall structure of the data better.
- The choice of k influences the trade-off between overfitting and underfitting. A smaller k may lead to overfitting, while a larger k may result in underfitting.
- Different feature pairs may exhibit different decision boundary patterns, reflecting the separability of classes based on those particular features.