KNearest Neighbors (KNN) Classification with the Wine Dataset

Student Name: Avinash Angilikam

Student ID: 23037971

Git Hub link:
https://github.com/AVINASHANGILIKAM/KNearestNeighborsKNNClassificationwiththeWineDataset

---

Introduction to K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a foundational algorithm in machine learning, popular for its intuitive approach to both classification and regression tasks. This algorithm is nonparametric, meaning it makes no assumptions about the underlying data distribution. Instead, KNN classifies new data points based on their proximity to labeled instances in the feature space, making it versatile and effective for datasets with complex distributions.

Key Concepts in KNN

1. *Distance Metric*:
   KNN determines the similarity between data points using a distance metric. The most common distance metrics are:
   - **Euclidean Distance**: Measures the straight-line distance between two points.
   - **Manhattan Distance**: Calculates the distance by summing the horizontal and vertical movements needed to connect two points.
   - **Minkowski Distance**: A generalized distance metric that can emulate both Euclidean and Manhattan distances depending on its parameter.
2. *Hyperparameter k*:
   The parameter **k** defines the number of nearest neighbors considered for classification or regression.
   - A smaller value of k (e.g., 1 or 3) makes the model sensitive to noise, leading to overfitting.
   - A larger value of k results in smoother decision boundaries but might underfit the data, losing important local patterns.

How KNN Works

KNN operates in four simple steps:

1. *Choose k*: Decide the number of nearest neighbors to consider.
2. *Compute Distances*: Calculate the distances between the query point and all other points in the dataset.
3. *Identify Neighbors*: Select the k closest data points.
4. *Predict*:
   - For classification, assign the majority class label among the k neighbors.
   - For regression, compute the average value of the neighbors' responses.

Theoretical Foundations of KNN

KNN is a type of lazy learning algorithm, meaning that it defers most computations until the prediction phase. This stands in contrast to other models, such as decision trees or neural networks, which require a significant training phase to construct a predictive model.

Advantages of KNN:

- Simple to implement and understand.
- No explicit training phase, reducing the initial computational cost.
- Effective for smaller datasets with low noise.

Disadvantages of KNN:

- Computationally intensive during prediction, especially for large datasets.
- Sensitive to irrelevant or unscaled features, which can skew the distance calculations.
- Prone to performance degradation in high-dimensional spaces (curse of dimensionality).

Wine Dataset Overview

The Wine dataset is a benchmark dataset commonly used for classification tasks. It contains chemical features of wine derived from three cultivars and is ideal for demonstrating the capabilities of KNN.

Dataset Description:

- *Features*: The dataset consists of 13 continuous variables, such as alcohol content, malic acid, and color intensity.
- *Target Variable*: A categorical variable representing three wine classes (0, 1, and 2).
- *Size*: The dataset contains 178 instances, making it manageable for experimentation.

Preparing the Dataset for KNN

1. *Data Splitting*: To evaluate the algorithm's performance, the dataset is divided into training and testing subsets. This allows the model to be tested on unseen data.
2. *Feature Scaling*: Standardization is critical for KNN since distance metrics are sensitive to feature scales. Scaling transforms features to have a mean of 0 and a standard deviation of 1, ensuring no single feature dominates the distance calculations.

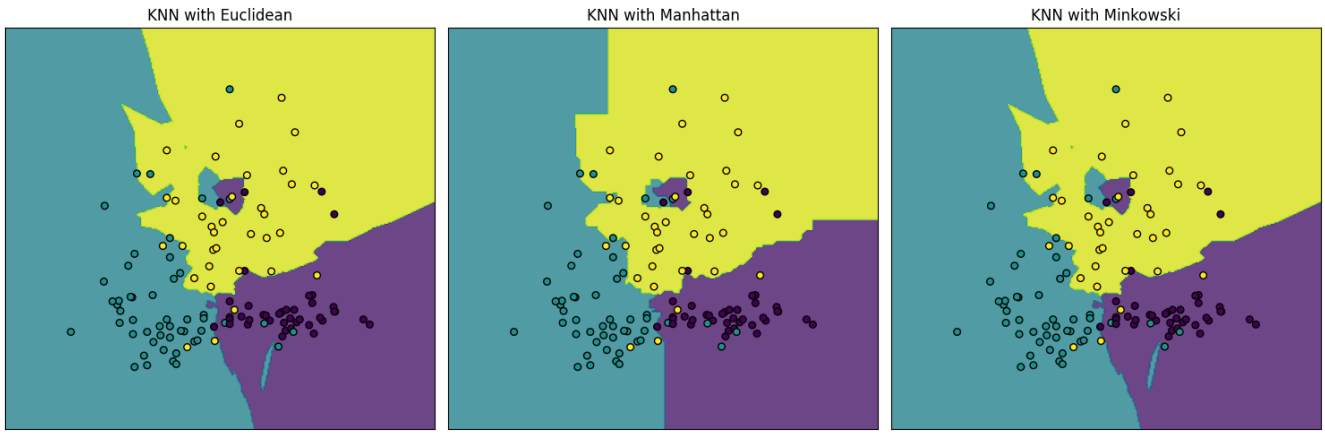
KNN Implementation and Observations

By experimenting with the Wine dataset, the following observations can be made:

1. *Impact of Distance Metrics*:
   Using different distance metrics such as Euclidean, Manhattan, and Minkowski provides consistent results with a test accuracy of 96% for k = 5.
2. *Effect of k Values*:
   Testing the algorithm with different values of k shows:
   - Small values (e.g., k = 1 or 3) achieve an accuracy of 96%.
   - Larger values (e.g., k = 7 or 9) increase accuracy to 98%.

Challenges and Considerations in Using KNN

1. *Choice of k*: Selecting the optimal k is crucial. A small k leads to overfitting, while a large k might underfit the data. Cross-validation is a reliable method to determine the best k value.
2. *Computational Cost*: For large datasets, KNN's prediction phase becomes computationally expensive due to distance calculations with every training instance. Efficient data structures like KD-Trees or Ball Trees can help optimize this process.
3. *Curse of Dimensionality*: In high-dimensional spaces, data points tend to become equidistant, reducing the relevance of distance metrics. Techniques like Principal Component Analysis (PCA) or feature selection can mitigate this issue.
4. *Feature Scaling*: Without scaling, features with larger ranges dominate the distance metric, leading to biased predictions. Standardization or normalization is mandatory for KNN.
5. *Distance Metric Selection*: Experimenting with different distance metrics helps identify the most effective one for a given problem.



Hyperparameter Tuning for KNN

Optimizing the hyperparameters of KNN involves:

- *k Value*: Testing different values of k to balance bias and variance.
- *Distance Metric*: Selecting metrics like Euclidean or Manhattan based on dataset characteristics.

Grid Search with Cross-Validation:
Grid search automates the search for the best combination of hyperparameters. A 5-fold cross-validation ensures robust evaluation.

- For the Wine dataset, the best parameters were Manhattan distance with k = 1, achieving a cross-validation accuracy of 98%.

Visualizing Decision Boundaries

Decision boundaries reveal how KNN partitions the feature space. By reducing the dataset to two dimensions using PCA, these boundaries can be visualized. The plots demonstrate how the distance metric and k influence the classification regions.

Insights from Visualization:

- Smaller values of k produce more detailed and irregular boundaries, capturing local variations in the data.
- Larger values of k result in smoother and more generalized boundaries.

Conclusion

The KNN algorithm is a powerful yet simple tool for machine learning tasks. Key takeaways include:

- The necessity of feature standardization to ensure fair distance calculations.
- The importance of selecting appropriate hyperparameters such as k and the distance metric to achieve optimal performance.
- Visualization techniques can provide valuable insights into the decision-making process of KNN.

KNN's effectiveness, combined with its straightforward implementation, makes it a valuable algorithm for classification and regression tasks, especially on smaller, well-prepared datasets. By addressing its limitations, such as computational cost and sensitivity to high-dimensional spaces, KNN can be adapted to a wide range of applications.

References

1. Scikitlearn Documentation: KNeighborsClassifier. Available at: https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

2. KNN Algorithm Overview. Towards Data Science. Available at: https://towardsdatascience.com/knearestneighborsknnfromscratchinpython9f6894b17b13

3. Wine Dataset Documentation. Scikitlearn. Available at: https://scikitlearn.org/stable/modules/generated/sklearn.datasets.load_wine.html