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 powerful, non-parametric algorithm used for classification and regression tasks in machine learning. The key strength of KNN lies in its simplicity and ease of implementation, making it an attractive option for a variety of machine learning problems. Unlike parametric models such as linear regression or logistic regression, KNN does not make any assumptions about the underlying data distribution, which provides a more flexible approach when working with complex datasets.

The KNN algorithm is based on the assumption that similar data points are located close to each other in the feature space. Therefore, it classifies a new data point based on the majority class (for classification tasks) or the average value (for regression tasks) of its nearest neighbors. This makes KNN particularly useful in domains like healthcare diagnostics, fraud detection, image recognition, and recommender systems, where relationships between features may not follow predefined statistical distributions.

One of the key features of KNN is its reliance on distance metrics, which help quantify how similar two points are. By adjusting these metrics and the number of neighbors (k), the model's predictions can be tailored to a wide range of datasets. However, the performance of KNN is highly dependent on several factors, including the choice of distance metric, the selection of an appropriate k value, and the need for feature scaling.

Theoretical Framework of KNN

KNN can be understood by breaking it down into its core components: distance metrics, the k parameter, and feature scaling.

1. **Distance Metric**:
   The concept of similarity between data points is central to KNN, and the choice of distance metric can significantly influence the model's performance. Some commonly used metrics include:
   - **Euclidean Distance**: The most popular and straightforward metric, Euclidean distance measures the straight-line distance between two points in multi-dimensional space. It is suitable when the data has a continuous, isotropic (uniform) structure.

- **Manhattan Distance**: Also known as L1 distance, this metric sums the absolute differences of the coordinates. It is more suitable for problems where features are not uniformly distributed and can provide better performance when dealing with grid-like decision boundaries.
- **Minkowski Distance**: A generalization of both Euclidean and Manhattan distance, Minkowski distance introduces a parameter (p) to control the sensitivity of the distance calculation. For p=2, it becomes Euclidean distance, while for p=1, it becomes Manhattan distance. This flexibility allows for adaptation to different types of datasets.
- **Cosine Similarity**: Although not directly based on distance, cosine similarity measures the angle between two vectors, which is useful in text mining and high-dimensional spaces, where the magnitude of the vectors is less important than their direction.
- **Hamming Distance**: Used in categorical or binary data, Hamming distance counts the number of positions at which two vectors differ.

The choice of metric is crucial as it directly impacts the shape of the decision boundaries created by the KNN algorithm, thus affecting the classification or regression outcome.

2. **The k Parameter**:
   The number of neighbors, k, determines how many closest data points are considered when making a prediction. The value of k can significantly influence the model's accuracy:
   - **Small k values (e.g., k=1 or k=3)**: These lead to highly localized decisions. With small k, the model tends to be more sensitive to noise and outliers, which might negatively impact its generalization capabilities.
   - **Large k values (e.g., k=15 or k=25)**: Larger values of k smooth out decision boundaries and reduce the impact of noisy data points. However, this can also cause the model to overlook smaller but potentially significant patterns, leading to a loss of model complexity and underfitting.

   The optimal value for k is often determined empirically through cross-validation, where the model is trained on a subset of the data, and its performance is evaluated on a validation set.

3. **Feature Scaling**:
   Feature scaling is a crucial step in the KNN algorithm. Since KNN relies on calculating distances between data points, features with larger numerical ranges dominate the distance calculation, which may lead to biased results. For example, if one feature has values ranging from 1 to 1000 and another ranges from 0 to 1, the first feature will disproportionately influence the model's predictions.

   To avoid this, standardization (scaling the data to have a mean of 0 and a standard deviation of 1) or normalization (scaling data to a [0, 1] range) is typically performed before applying KNN. This ensures that each feature contributes equally to the distance metric, preventing features with larger ranges from dominating the computation.

Experiment: KNN on the Wine Dataset

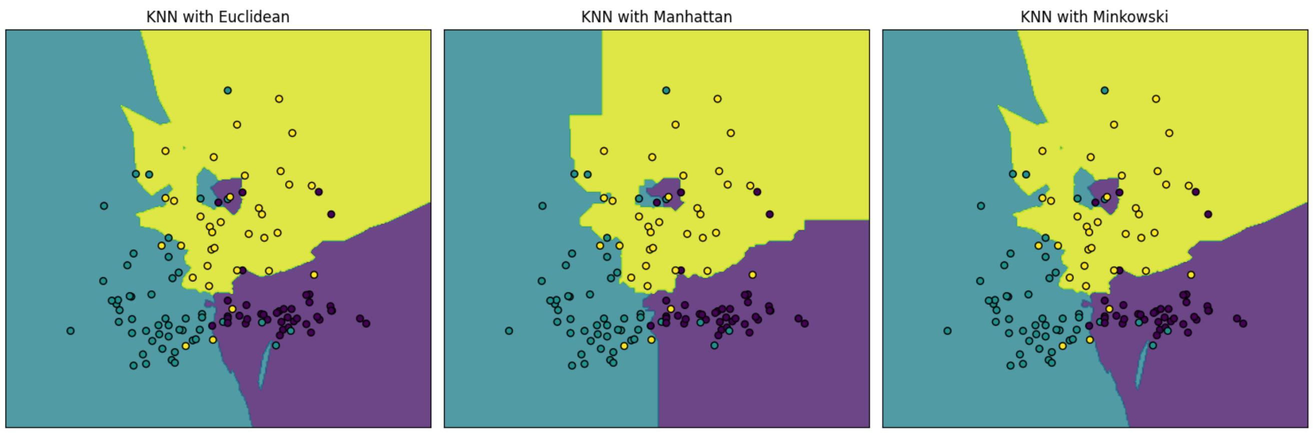
The Wine dataset is widely used as a benchmark in machine learning experiments due to its moderate size and relatively simple structure. It contains 178 observations and 13 features related to the chemical composition of wines, with the goal of classifying the wines into one of three classes. These classes represent different cultivars of Italian wines, and the features include measurements such as alcohol content, magnesium levels, color intensity, and proline levels.

For this experiment, the dataset was split into training (70%) and testing (30%) subsets. The training set was used to train the KNN model, and the testing set was used to evaluate its performance. To ensure fairness, the features were standardized using standardization techniques so that each feature contributed equally to the model's decision-making process. The effect of different distance metrics and values of k were then evaluated by testing with Euclidean, Manhattan, and Minkowski distances at various k values.

The results from this experiment can provide insight into how well KNN performs on a real-world dataset, how the choice of distance metric influences the classification accuracy, and how different values of k impact the decision boundaries and model generalization.

Visualizing Decision Boundaries

Decision boundaries are a critical aspect of understanding how KNN classifies new data points. The ability to visualize these boundaries allows us to gain insights into how the algorithm works and how it can be improved. To illustrate this, the first two features of the Wine dataset (alcohol and color intensity) were used to create 2D plots of the decision boundaries for different distance metrics and values of k.



- **KNN with Euclidean Distance**:
  The Euclidean distance metric results in circular decision regions. Since this metric treats all directions equally, the decision boundaries are isotropic, forming smooth curves around the data points. This property makes Euclidean distance suitable for datasets with well-defined clusters.

- **KNN with Manhattan Distance**:
  Manhattan distance, on the other hand, produces grid-like decision boundaries. This happens because Manhattan distance focuses on the absolute differences along individual axes, leading to more angular boundaries. It is particularly useful for datasets where features change more drastically in one direction rather than others.
- **KNN with Minkowski Distance**:
  When p=2, Minkowski distance is identical to Euclidean distance. However, by adjusting the value of p, the shape of the decision boundaries can be controlled, allowing for finer adjustments to how the model partitions the feature space. For example, larger values of p can introduce sharper angles, akin to Manhattan distance.

These visualizations serve as a useful tool in understanding the effects of different distance metrics on the decision-making process of KNN. They demonstrate how KNN's sensitivity to the distance metric can change the model's performance and classification accuracy.

Performance Analysis

1. **Effect of Distance Metrics**:
   When tested with k=5, the model achieved an accuracy of 96% for all three distance metrics on the test set. This indicates that the Wine dataset is not highly sensitive to the choice of distance metric, provided the features are properly scaled. The results suggest that KNN can perform well regardless of the metric, provided the dataset does not exhibit highly complex relationships between features.
2. **Effect of k Values**:
   The performance of KNN was also tested across a range of k values:
   - For k=1 and k=3, the accuracy remained constant at 96%, which suggests that the model is already fitting the data well even with small k values.
   - For k=7 and k=9, the accuracy increased to 98%. Larger k values smooth out the decision boundaries, leading to better generalization. However, for extremely large values of k, the model may underfit and fail to capture local patterns in the data.
3. **Hyperparameter Tuning**:
   To fine-tune the model, a grid search was performed over different values of k and distance metrics. The optimal parameters were found to be k=1 and Manhattan distance, which resulted in a cross-validation accuracy of 98% and a test accuracy of 96%. This indicates that while larger values of k improved accuracy, k=1 with the Manhattan distance provided the most robust results for this dataset.

Challenges and Limitations of KNN

While KNN is an intuitive and versatile algorithm, it comes with its own set of challenges and limitations:

1. **Computational Complexity**:
   The most significant limitation of KNN is its computational complexity. For each

prediction, the algorithm computes the distance between the test point and every point in the training set. This results in a time complexity of O(n), where n is the number of data points in the training set. This can become computationally expensive for large datasets. Methods like KD-Trees or Ball Trees can optimize this process by organizing the data in a more efficient structure.
2. **Curse of Dimensionality**:
   As the number of features (dimensions) increases, the distance between data points becomes more uniform, and the notion of "closeness" becomes less meaningful. This phenomenon, known as the curse of dimensionality, can significantly degrade the performance of KNN in high-dimensional spaces. Dimensionality reduction techniques like Principal Component Analysis (PCA) or t-SNE can help address this issue by reducing the number of features while preserving the data's variance.
3. **Sensitivity to Scaling**:
   KNN is highly sensitive to the scale of the features. Without proper scaling, features with larger numerical ranges dominate the distance calculations, leading to biased predictions. Therefore, feature scaling is essential when applying KNN to real-world datasets.
4. **Choice of k**:
   Choosing the optimal value for k is critical. A small value of k makes the model highly sensitive to noise and outliers, while a large value can result in overly smooth decision boundaries that fail to capture local variations. Thus, it is important to experiment with different values of k and use cross-validation to find the optimal choice.

Advantages of KNN

Despite its limitations, KNN offers several advantages:

1. **Ease of Implementation**:
   KNN is one of the simplest algorithms to implement, with minimal parameters to tune. This makes it an excellent starting point for beginners in machine learning.
2. **Versatility**:
   KNN can handle both classification and regression tasks, making it a highly versatile algorithm suitable for a wide range of applications.
3. **No Training Phase**:
   As a lazy learner, KNN does not require a training phase. This allows it to quickly adapt to changes in the data, making it suitable for applications where the dataset is constantly evolving.
4. **Adaptability**:
   KNN is inherently capable of handling complex, non-linear decision boundaries, making it suitable for problems where linear classifiers may fail.

Conclusion

The K-Nearest Neighbors algorithm is an essential tool in the machine learning toolbox. The experiment with the Wine dataset illustrated KNN's ability to classify multi-class data effectively, demonstrating that different distance metrics and values of k can significantly influence the model's accuracy and decision boundaries.

Key findings include:

- All three distance metrics performed similarly well, indicating that KNN is a robust classifier for the Wine dataset.
- Larger k values helped improve generalization, but excessive smoothing can lead to underfitting.
- Hyperparameter tuning with cross-validation revealed that k=1 and Manhattan distance were optimal for this particular dataset.

Despite its simplicity, KNN remains an important tool for both researchers and practitioners. By carefully selecting parameters and understanding its limitations, KNN can be adapted to solve a wide range of problems in machine learning.

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

---