Optimizing k-Nearest Neighbors Classifier: A Comprehensive Study

Zahir AHMAD [*1]

Abstract

This study evaluates and optimizes the k-Nearest Neighbors (k-NN) classifier, a machine learning algorithm that can find patterns in large, high-dimensional datasets. We use a waveform dataset with **5000** instances and **21** features as a case study. We determine the optimal number of neighbors ($k = \mathbf{58}$) through cross-validation, achieving a mean accuracy of **85.78%** with a standard deviation of **0.63%**. We also explore data reduction techniques, such as the Condensed Nearest Neighbor method and Principal Component Analysis, to reduce the data size and complexity. Furthermore, we examine the impact of using k-dimensional trees to speed up the 1-Nearest Neighbor search. The results provide significant insights into balancing accuracy, computational efficiency, and complexity in the k-NN classifier, and contribute to its application in real-world problems.

1. Tune the best k of a kNN classifier by Cross-Validation

The waveform dataset, containing 5000 samples with 21 features and 3 balanced classes, was analyzed using k-nearest neighbors (kNN). Cross-validation identified the optimal $k = \mathbf{58}$, achieving a mean accuracy of **85.78%** and a standard deviation of **0.63%**. The overall accuracy of **85.9%** reflects a robust model, while precision and recall metrics reveal its varying efficacy in classifying each class, with some lower recall rates noted. High F1 scores suggest a well-balanced performance, and the confusion matrix provides insights into specific classification challenges, especially between classes 0 and 1, indicating areas for potential improvement. (See: Firgure 1, Table 1, Table 2)

Figure 1. Best k-value

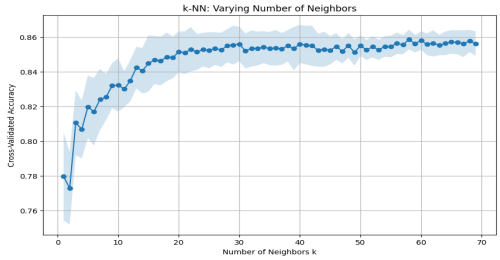

Table 1. Results

Metric/Class	Class 0	Class 1	Class 2	Overall
Accuracy	-	-	-	85.9%
Precision	0.91	0.84	0.84	-
Recall	0.74	0.91	0.92	-
F1 Score	0.82	0.87	0.88	-

Table 2. Confusion Matrix

	Predicted 0	Predicted 1	Predicted 2
True 0	248	47	38
True 1	14	323	17
True 2	10	15	288

2. Data Reduction

Data reduction methods address the computational and memory demands of the k-NN algorithm, particularly beneficial for large, high-dimensional datasets like the waveform dataset with 5000 examples and 21 features. These methods aim to maintain essential data structure while boosting the k-NN's speed and accuracy. The Condensed Nearest Neighbor (CNN) method selects a subset from the original set, reducing it from 4000 to 1826 examples (54.35% reduction) but slightly drops the 1-NN classifier's accuracy from 0.79 to 0.763. This indicates a trade-off between complexity and accuracy, as CNN might discard vital information or add noise. On the other hand, Principal Component Analysis (PCA) reduces feature count by projecting the data onto a lower-dimensional space, cutting down from 21 to 10 features (52.38% reduction) and marginally increasing k-NN accuracy from 0.859 to 0.863. This suggests that PCA can enhance classifier performance by preserving critical information or eliminating noise (See: Table 3).

Table 3. Comparison between CNN and PCA Methods

CRITERIA	CNN	PCA
ORIGINAL DATA	4000 EXAMPLES	21 FEATURES
REDUCED DATA	1826 EXAMPLES	10 FEATURES
PERCENTAGE REDUCTION	54.35%	52.38%
ORIGINAL ACCURACY	0.79	0.859
REDUCED ACCURACY	0.763	0.863

3. Speeding Up 1-Nearest Neighbor Calculations

To optimize the slow brute force 1NN on a large, high-dimensional waveform dataset (5000 examples, 21 features), we employed speed-enhancing methods. Brute force took 0.5705 seconds to label 1000 test points with an accuracy of 0.79. Data reduction then lowered the training set from 4000 to 1826 examples, quickening the prediction time to 0.1508 seconds but reducing accuracy to 0.763, suggesting some information loss. Conversely, KD-trees, which organize data into a binary tree for efficient searching without altering the dataset, improved prediction time to 0.1751 seconds while maintaining an accuracy of 0.79. These methods demonstrate varied impacts on performance and dataset integrity, with KD-trees providing a balanced approach to maintaining accuracy and enhancing speed (See: Figure 2, Table 4).

Figure 2. Time Comparison of Brute Force and KD-tree with and without data reduction

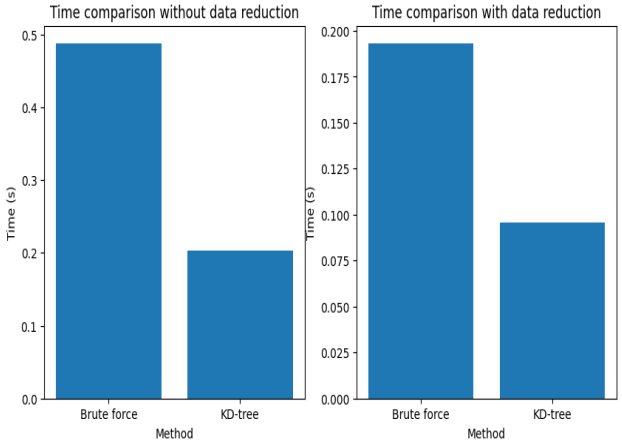

Table 4. Comparison of Methods to Speed Up 1NN Calculations

Method	Performance Impact	Time (seconds)	Accuracy
Brute Force 1NN	Very slow and inefficient	0.5705	0.79
Data Reduction	Faster than brute force but slower than KD-trees	0.1508	0.763
KD-trees	Faster than brute force and data reduction	0.1751	0.79

4. Impact of Artificial Imbalance on 1-Nearest Neigh- bor Classifier

To introduce imbalance, we resampled classes (600 for class 0, 1000 for class 1, 1383 for class 2), leading to an imbalance ratio of 2.305. This imbalance decreased classifier accuracy and F-measure on the test set from 0.79 to 0.786 and 0.783, respectively, indicating deteriorated performance due to potentially less information from the minority class and more noise from the majority. We tuned the k parameter using cross-validation, finding the optimal k at 10 with a validation set F-measure of 0.8338. Post-tuning, the classifier's accuracy and F-measure on the test set slightly improved to 0.837 and 0.8338, surpassing pre-tuning results (0.786 and 0.783) but still underperforming compared to the original balanced data (both 0.79). This suggests that the classifier suffers from the class imbalance problem, as it may not be able to capture the characteristics of the minority class or distinguish it from the majority class (See: Table 5).

Table 5. Impact of Class Imbalance on kNN Classifier

Metric	Before Imbalance	After Imbalance
Number of Samples (Class 0, 1, 2)	1324, 1293, 1383	600, 1000, 1383
Accuracy on Test Set	0.79	0.786
F-measure on Test Set	0.79	0.783
Best k (Cross-Validation)	58	10
Accuracy with Best k	0.79	0.837
F-measure with Best k	0.79	0.8338

5. Conclusion

This study helps us optimize the k-Nearest Neighbors (k-NN) classifier. The k-NN classifier can find patterns in data, but it also has some challenges. We explored four challenges and solutions in this study: 1. How to choose the best value of k for the classifier. We found that $k = 58$ gives the highest accuracy and stability. 2. How to reduce the data size and complexity without losing accuracy. We compared two methods: Condensed Nearest Neighbor (CNN) and Principal Component Analysis (PCA). We found that CNN reduces data size but lowers accuracy, while PCA improves accuracy but changes the data. 3. How to speed up the classifier without affecting accuracy. We used KD-trees to make the classifier faster and simpler, without changing the data or the accuracy. 4. How to handle data with different sizes of groups. We found that the classifier does not work well when the data is imbalanced.