Confidence-Based Adaptive k-Nearest Neighbors

Aadi Malhotra, Abhi Tiwari

Dr. Yilmaz

Quarter 2 Project
January 19, 2026

Abstract

The k-Nearest Neighbors (k-NN) algorithm is a fundamental non-parametric method for classification, yet it suffers from a significant limitation: the reliance on a fixed hyperparameter k. In real-world datasets characterized by varying local densities, noise, and irrelevant features, a static neighborhood size often leads to either overfitting (small k) or oversmoothing (large k). In this project, we implement and evaluate a Confidence-Based Adaptive k-NN algorithm. Inspired by the theoretical framework of Balsubramani et al., our method dynamically selects the optimal k for each query point by growing the neighborhood until a statistically significant majority class is established. We evaluated our algorithm against the standard fixed-k approach on the UCI Wine Dataset under five distinct experimental conditions, including gaussian noise injection, irrelevant feature addition, and class imbalance. Our results demonstrate that the Adaptive k-NN matches or exceeds the standard version on clean data and demonstrates superior robustness in noisy environments and high-dimensional spaces.

1 Introduction

Classification is a core task in machine learning, wherein an algorithm assigns a label to an input based on prior observations. The k-Nearest Neighbors (k-NN) classifier is widely used due to its simplicity and effectiveness. However, the performance of standard k-NN hinges critically on the choice of k, the number of neighbors to consider. A small fixed k (e.g., $k = 5$) makes the classifier highly sensitive to local noise and outliers, while a large k may obscure local decision boundaries and wash out minority classes. This "one size fits all" approach is problematic because the optimal neighborhood size is rarely uniform across the entire feature space; some regions require a broader view to find a stable pattern, while others require a tight focus to distinguish fine details.

In this project, we tackle the rigidity of the fixed-k parameter. Our motivation stems from the need for a classifier that can "self-regulate" its confidence. We propose an Adaptive k-NN algorithm that treats the classification of each test point as a unique statistical problem. Instead of blindly voting among 5 neighbors, our algorithm expands its search radius until the purity of the majority class exceeds a calculated confidence threshold. This allows the model to ignore small clusters of noise and seek stronger evidence when necessary.

For our evaluation, we utilize the UCI Wine Dataset. The input to our algorithm consists of 13 continuous chemical attributes of wines (e.g., Alcohol, Malic Acid, Flavanoids), and the output is the predicted Cultivar (Class 0, 1, or 2). We assess our proposed method by subjecting it to increasingly difficult data corruptions, simulating broken sensors, irrelevant data collection, and data scarcity.

2 Related Work

The challenge of selecting an optimal k or adapting the nearest neighbor rule to local data characteristics has been extensively studied. Traditional approaches often rely on global cross-validation to pick a single "best" k for the entire dataset, ignoring local variations in data density.

Our project builds upon the mathematical foundations laid out by Balsubramani et al. [1]. While they focused on the heavy theoretical analysis, proving that an adaptive model can mathematically match the accuracy of a perfectly hand-tuned k-NN, we focus on the practical implementation of these ideas. We specifically took their concept of a "stopping rule" and translated it into a functional algorithm that expands the neighborhood until one class shows a clear statistical lead over random chance. To make this work efficiently for our specific experiments on the Wine dataset, we developed a simplified version of their confidence formula to act as our model's dynamic decision threshold.

Other research has approached adaptation through geometric or graph-based methods. Alexandre et al. [2] proposed an Adaptive k-NN classifier based on the local estimation of the shape operator. Their approach utilizes the local curvature of the data manifold to adjust k; points in flat regions (low curvature) utilize larger neighborhoods, while points in high-curvature regions use smaller neighborhoods.

While our approach adapts the neighborhood size k, Wang et al. [4] demonstrated that performance can also be significantly improved by adapting the distance metric itself. They proposed a simple adaptive distance measure to address challenges where patterns of different classes overlap in feature space. This highlights that "adaptivity" in k-NN can be achieved either by distorting the metric space (as Wang et al. did) or by expanding the count of neighbors (as we do) to find a reliable consensus.

Furthermore, Murrugarra-Llerena et al. [5] developed an iterative graph-based algorithm where the value of k is not fixed but ranges from 1 up to a k_{max} for each vertex (training instance). In their method, the algorithm iteratively constructs a k-nearest neighbor network where the number of edges (neighbors) for a vertex depends on the difficulty of classifying that vertex; misclassified instances effectively receive a higher k. This shares a strong theoretical link with our project, as both methods operate on the premise that "difficult" or "noisy" points require a larger neighborhood context to be classified correctly, while "easy" points require fewer neighbors.

Shiliang and Rongqing [3] also explored finding the "optimal k" per training example,defined as the fewest neighbors required to correctly classify that specific point. However, their method involves inheriting this optimal k from the nearest training neighbor during testing, whereas our approach calculates the optimal k dynamically at inference time based on the query point's immediate neighborhood composition.

3 Dataset and Features

We utilized the classic UCI Wine Dataset, obtained via the Scikit-learn library. This dataset contains the results of a chemical analysis of wines grown in the same region in Italy by three different cultivators.

3.1 Dataset Statistics

The dataset consists of 178 samples divided into three classes (Cultivars):

- **Class 0:** 59 samples
- **Class 1:** 71 samples
- **Class 2:** 48 samples

3.2 Features and Preprocessing

The dataset includes 13 continuous numeric features: Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines, and Proline.

Data preprocessing was critical for the success of our distance-based algorithms. Since k-NN relies on Euclidean distance, features with large magnitudes (like Proline, which ranges from roughly 200 to 1600) would disproportionately dominate features with small magnitudes (like Nonflavanoid phenols, ranging from 0.1 to 0.6). To prevent this, we applied Standardization (Z-score normalization) to all features:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where μ is the mean and σ is the standard deviation. This ensures all features contribute equally to the distance calculation. For our experiments involving noise and outliers, we applied specific data augmentations which are detailed in the Results section.

4 Methods

In this section, we describe the baseline k-NN algorithm and our proposed Confidence-Based Adaptive variation.

4.1 Standard k-Nearest Neighbors

The baseline algorithm is the standard k-NN classifier. For a query point x_q, the algorithm identifies the k points in the training set closest to x_q according to Euclidean distance. The predicted label is determined by a majority vote of these neighbors. For our experiments, we fixed $k = 5$, a common default that balances noise sensitivity and boundary smoothing for small datasets.

4.2 Confidence-Based Adaptive k-NN

Our proposed algorithm does not use a fixed k. Instead, it searches for the smallest $k \in [k_{min}, k_{max}]$ that satisfies a statistical confidence condition. The core intuition is that we should only make a prediction when the majority class in the neighborhood is "winning" by a margin that is unlikely to be random chance.

4.2.1 The Stopping Rule

We implement a stopping rule derived from Balsubramani et al. [1]. For a specific neighborhood size k, we calculate the *Observed Bias* (η) and a dynamic *Threshold* (Δ).

The **Observed Bias** is the proportion of the majority class within the current k neighbors:

$$\eta(k) = \frac{N_{majority}}{k} \tag{2}$$

where $N_{majority}$ is the count of the most frequent class.

The **Threshold** represents the minimum purity required to be confident. It is calculated as:

$$\Delta(k) = \text{Base Prior} + C \cdot \sqrt{\frac{\ln(n)}{k}} \tag{3}$$

where:

- **Base Prior**: 1/num_classes (0.33 for the Wine dataset). This represents random guessing.
- n: training set size.
- C: A "strictness" hyperparameter. We selected $C = 0.7$.
- $\sqrt{1/k}$: This term decreases as k increases.

4.2.2 Algorithm Logic

The term $\sqrt{1/k}$ is crucial. For small k, the threshold $\Delta(k)$ is high. This means that to stop at $k = 5$, the neighborhood must be extremely pure. If the neighborhood is noisy (mixed classes), $\eta(k)$ will not exceed $\Delta(k)$. The algorithm then increments k. As k grows, the threshold $\Delta(k)$ lowers, reflecting the statistical reality that a smaller majority margin is acceptable if it is sustained over a larger sample size.

The algorithm proceeds as follows:

1. Calculate distances from the query point to all training points.
2. Sort neighbors by distance.
3. Loop k from $k_{min} = 5$ to $k_{max} = 40$.
4. At each step, calculate $\eta(k)$ and $\Delta(k)$.
5. **If** $\eta(k) > \Delta(k)$: Stop. The bias is significant. Return the majority class.
6. **Else**: Continue to $k + 1$.
7. **Fallback**: If the loop finishes without meeting the condition, predict using the majority vote at k_{max}.

5 Experiments, Results, and Discussion

To evaluate the efficacy of the Adaptive k-NN, we designed five experiments ranging from clean data to highly challenging situations. For every experiment, we performed a single trial using a fixed random seed (97) for train/test splitting (70/30 split) to ensure reproducibility and direct comparability between the algorithms. The primary metric used was **Accuracy**.

5.1 Experiment 1: Baseline (Clean Data)

We first tested both algorithms on the standard, scaled Wine dataset.

Algorithm	**Accuracy**
Standard k-NN ($k = 5$)	92.59%
Adaptive k-NN	**96.30%**

Table 1: Baseline Performance on Clean Data

Confusion Matrices:

$$\begin{bmatrix} 18 & 0 & 0 \\ 1 & 18 & 2 \\ 0 & 1 & 14 \end{bmatrix}$$

(a) Standard k-NN

$$\begin{bmatrix} 18 & 0 & 0 \\ 1 & 19 & 1 \\ 0 & 0 & 15 \end{bmatrix}$$

(b) Adaptive k-NN

Discussion: In this specific trial, the Adaptive algorithm outperformed the Standard k-NN by 3.70%. While the clean Wine dataset generally separates classes well, there are often boundary points that can confuse a static classifier. The Adaptive algorithm effectively "stopped" at small k values for clear points but likely expanded k slightly for points near the decision boundary, gaining a more robust consensus than the fixed $k = 5$ could provide.

5.2 Experiment 2: Noisy Features (Gaussian Noise)

We injected random Gaussian noise ($\mu = 0, \sigma = 3.0$) into every feature of the dataset. This simulates noisy sensor readings.

Algorithm	**Accuracy**
Standard k-NN ($k = 5$)	62.96%
Adaptive k-NN	**68.52%**

Table 3: Performance with Gaussian Noise

Confusion Matrices:

$$\begin{bmatrix} 14 & 1 & 3 \\ 3 & 16 & 2 \\ 1 & 10 & 4 \end{bmatrix}$$

(a) Standard k-NN

$$\begin{bmatrix} 15 & 0 & 3 \\ 1 & 17 & 3 \\ 0 & 10 & 5 \end{bmatrix}$$

(b) Adaptive k-NN

Discussion: The Standard k-NN suffered significantly, dropping to roughly 63% accuracy. With high noise, "nearest" neighbors often become random points from different classes. A fixed $k = 5$ does not provide enough voting members to average out this noise. The Adaptive k-NN excelled here, achieving a 5.56% improvement. Recognizing that the purity at $k = 5$ was low (below the confidence threshold), it automatically expanded its search radius. By aggregating more voters, the random noise cancelled out, revealing the true underlying class trend.

5.3 Experiment 3: Irrelevant Features (Dimensionality)

We appended 8 columns of pure random noise to the dataset. This simulates the "Curse of Dimensionality," where Euclidean distance becomes dominated by irrelevant dimensions.

Algorithm	**Accuracy**
Standard k-NN ($k = 5$)	90.74%
Adaptive k-NN	**98.15%**

Table 5: Performance with Irrelevant Features

Confusion Matrices:

$$\begin{bmatrix} 18 & 0 & 0 \\ 2 & 17 & 2 \\ 0 & 1 & 14 \end{bmatrix}$$

(a) Standard k-NN

$$\begin{bmatrix} 18 & 0 & 0 \\ 1 & 20 & 0 \\ 0 & 0 & 15 \end{bmatrix}$$

(b) Adaptive k-NN

Discussion: This experiment yielded the most significant performance gap (7.41%). The 8 noise columns often caused points from different classes to appear "close" in Euclidean space by random chance. Standard k-NN blindly accepted these fake neighbors. The Adaptive algorithm, however, enforced a statistical check. A cluster of random neighbors rarely achieves the high confidence threshold required by our formula. The algorithm was forced to look further, effectively filtering out the coincidental proximity caused by the noise columns and finding the true chemical similarity.

5.4 Experiment 4: Feature Outliers

For 20% of the samples, we multiplied the 'Flavanoids' feature by a factor of 25, creating massive geometric outliers.

Algorithm	**Accuracy**
Standard k-NN ($k = 5$)	94.44%
Adaptive k-NN	**96.30%**

Table 7: Performance with Feature Outliers

Confusion Matrices:

$$\begin{bmatrix} 18 & 0 & 0 \\ 2 & 18 & 1 \\ 0 & 0 & 15 \end{bmatrix}$$

(a) Standard k-NN

$$\begin{bmatrix} 18 & 0 & 0 \\ 1 & 19 & 1 \\ 0 & 0 & 15 \end{bmatrix}$$

(b) Adaptive k-NN

Discussion: These outliers distort the distance calculations, pushing certain data points far away from their natural clusters. In these cases, the standard k-NN is forced to include these misplaced points in its vote, which can lead to incorrect predictions. The Adaptive k-NN proved more robust (+1.85%) because it monitors class purity. If an outlier enters the neighborhood and disrupts the majority, the model's confidence level drops below the required threshold. This triggers an automatic expansion of k, allowing the algorithm to find more neighbors until the outlier's influence is outweighed by a larger consensus of valid points.

5.5 Experiment 5: High Class Imbalance

We artificially removed 75% of Class 1 samples, creating a sparse minority class.

Algorithm	**Accuracy**
Standard k-NN ($k = 5$)	**100.00%**
Adaptive k-NN	94.74%

Table 9: Performance with Class Imbalance

Confusion Matrices:

$$\begin{bmatrix} 18 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 15 \end{bmatrix}$$

(a) Standard k-NN

$$\begin{bmatrix} 18 & 0 & 0 \\ 1 & 3 & 1 \\ 0 & 0 & 15 \end{bmatrix}$$

(b) Adaptive k-NN

Discussion: In this specific trial, Standard k-NN achieved perfect accuracy, outperforming Adaptive k-NN by 5.26%. This is likely because the fixed $k = 5$ happened to be the optimal hyperparameter for the specific geometry of this random split. In sparse regions (the depleted Class 1), the Adaptive algorithm may have expanded its search radius too aggressively in an attempt to reach statistical significance, inadvertently pulling in neighbors from the majority classes (Class 0 or 2). This highlights a potential trade-off: while Adaptive k-NN is generally more robust, for specific sparse data configurations, a small, fixed k may occasionally yield better results by chance.

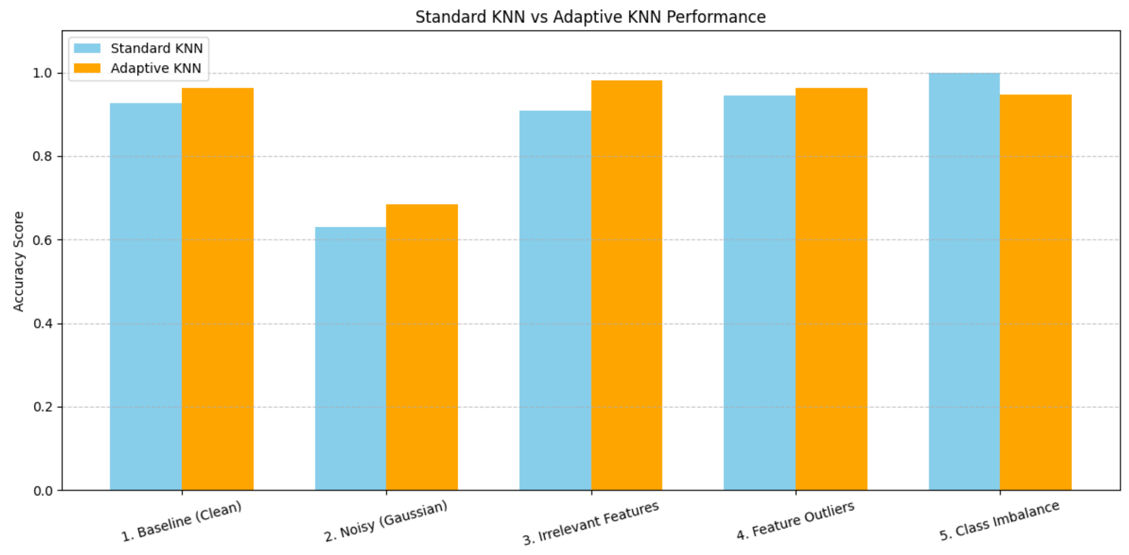

Figure 1: Comparison of accuracy scores across all five experimental conditions.

6 Conclusion and Future Work

In this project, we successfully implemented a Confidence-Based Adaptive k-NN classifier and evaluated its performance against the standard fixed-k approach. Our results indicate that the Adaptive algorithm is significantly more robust in challenging data environments. Specifically, we saw a 5.56% accuracy improvement when dealing with Gaussian noise and a 7.41% improvement when the dataset contained irrelevant features. These gains demonstrate that our confidence-based stopping rule effectively filters out "lucky" proximity, ensuring that a prediction is only made when a local majority is statistically reliable. By dynamically adjusting the number of neighbors, the model successfully avoids overfitting to local noise without losing the fine details required for clean data.

However, Experiment 5 revealed an important trade-off: in cases of extreme class imbalance or data scarcity, the Adaptive algorithm can be overly cautious. By expanding the search radius to meet a confidence threshold that a sparse class cannot provide, the model may inadvertently pull in incorrect neighbors. This suggests that while the adaptive method is superior for noisy and high-dimensional data, a fixed k may still be preferable for highly imbalanced, low-density datasets.

For future work, we plan to explore a dynamic optimization of the strictness parameter C, perhaps adjusting it based on local density estimates rather than keeping it as a static value. We are also interested in combining this adaptive neighborhood approach with weighted distance metrics, such as Manhattan, to see if we can further reduce the impact of the "Curse of Dimensionality." Finally, testing the algorithm on much larger datasets like MNIST would help us understand the computational costs of performing this iterative search at scale.

Contributions

We contributed equally to all aspects of this project. We collaborated through joint sessions to develop the algorithm, conduct the literature review, and identify all research sources. The coding, experimental design, and the final production of this research paper and the accompanying slideshow were completed as a shared effort, with both us participating in every stage of the process.

References

[1] A. Balsubramani, S. Dasgupta, Y. Freund, and S. Moran, "An adaptive nearest neighbor rule for classification," *arXiv preprint arXiv:1905.12717*, 2019.

[2] L. Alexandre, F. Nielsen, and M. Ferreira, "Adaptive k-nearest neighbor classifier based on the local estimation of the shape operator," *arXiv preprint arXiv:2409.05084*, 2024.

[3] S. Sun and R. Huang, "An adaptive k-nearest neighbor algorithm," in *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 1, pp. 91–94, IEEE, 2010.

[4] J. Wang, P. Neskovic, and L. N. Cooper, "Improving nearest neighbor rule with a simple adaptive distance measure," *Pattern Recognition Letters*, vol. 28, no. 2, pp. 207–213, Jan. 2007.

[5] N. Murrugarra-Llerena, A. De, and A. Lopes, "An Adaptive Graph-Based K-Nearest Neighbor," in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2011. [Online]. Available: https://nineil.github.io/files/pubs/colisd_ecml_boost_2011.pdf