Confidence-Based Adaptive k-Nearest Neighbors

Aadi, Abhi

The Problem With Standard k-NN

- Static Hyperparameter: Uses one fixed k for all data points.
- Rigidity: Cannot adapt to varying local densities or noise.
- Overfitting: Small k is too sensitive to local outliers.
- Oversmoothing: Large k blurs class boundaries and minority groups.
- "One Size Fits All": Ignores that different regions need different neighborhood sizes.
- Noise Vulnerability: Blindly accepts "fake neighbors" in high dimensions.

Related Works

- Balsubramani et al. [1]: Established mathematical theory for adaptive stopping rules.
- Alexandre et al. [2]: Used local data curvature to adjust neighborhood size.
- Shiliang & Rongqing [3]: Assigned k based on the nearest training neighbor's optimal value.
- Wang et al. [4]: Improved accuracy by adapting the distance metric itself.
- Murrugarra-Llerena et al. [5]: Applied iterative graphs where difficult points get higher k.

Our Solution

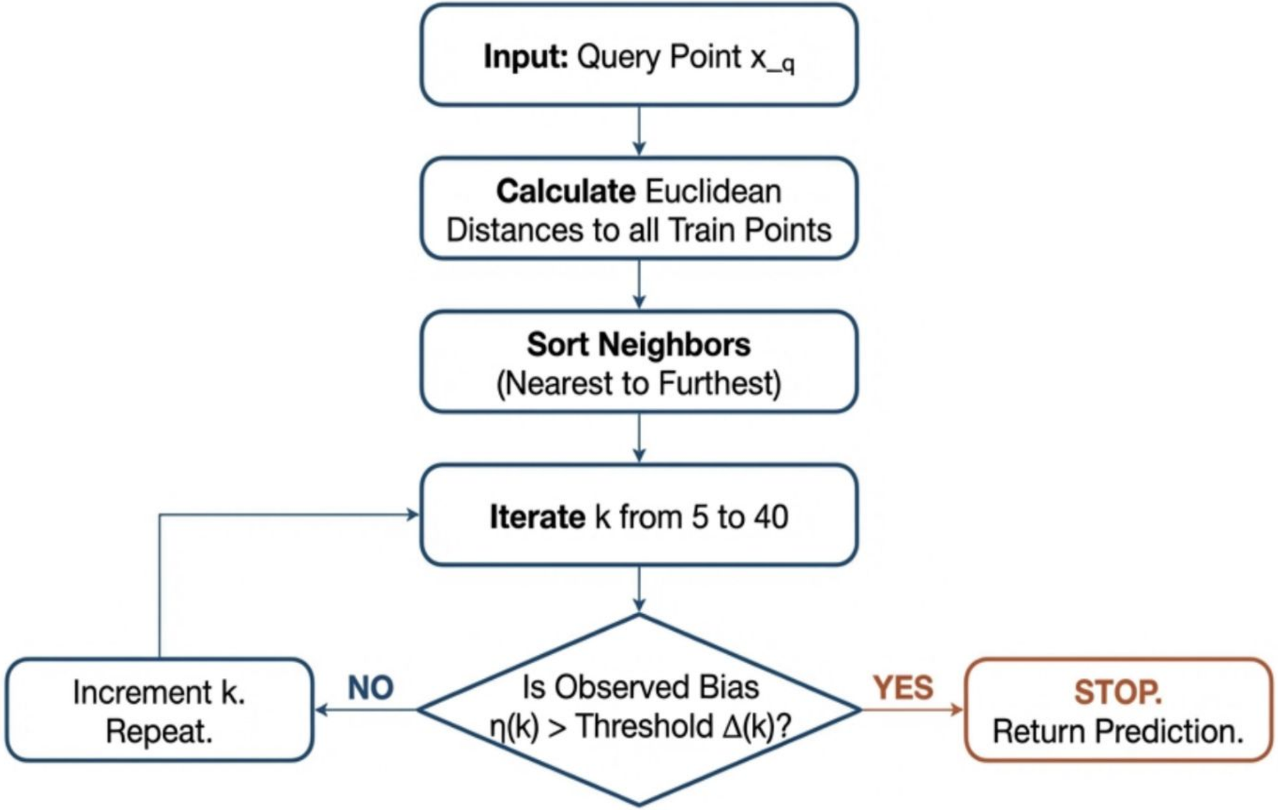
- Core Philosophy
  - Treats the classification of each test point as a unique statistical problem
  - Replaces "Fixed Neighbor Count" with "Fixed Confidence Level".
- Mechanism
  - Dynamically selects the optimal k for each query point during inference
  - Expansion Rule: The algorithm expands the neighborhood radius until the purity of the majority class exceeds a calculated statistical threshold.
- Key Advantages
  - Noise Tolerance: Automatically expands k in noisy regions to average out errors.
  - Precision: Stops at small k for clean, distinct data points to preserve boundaries.

Methodology

- The Logic:
    - Stop increasing k when Observed Bias > Dynamic Threshold.
- Observed Bias:
    - The percentage of the majority class in the current neighborhood (e.g., 4 out of 5 neighbors = 80%).
- Dynamic Threshold: A calculated bar we must clear to be "confident."
    - The algorithm expands the neighborhood radius until the purity of the majority class exceeds a calculated limit.
    - As k gets larger, the bar gets lower.
- Parameters used:
    - Strictness (C) = 0.7
    - Base Prior = 1/3 (random guessing for 3 classes).

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

Methodology: Algorithm Implementation



Input: Query Point x_q

Calculate Euclidean Distances to all Train Points

Sort Neighbors (Nearest to Furthest)

Iterate k from 5 to 40

Is Observed Bias $\eta(k)$ > Threshold $\Delta(k)$?

NO

Increment k. Repeat.

YES

STOP. Return Prediction.

Fallback: If loop finishes (k=40) without condition met, predict using majority vote at k_max.

Dataset & Preprocessing

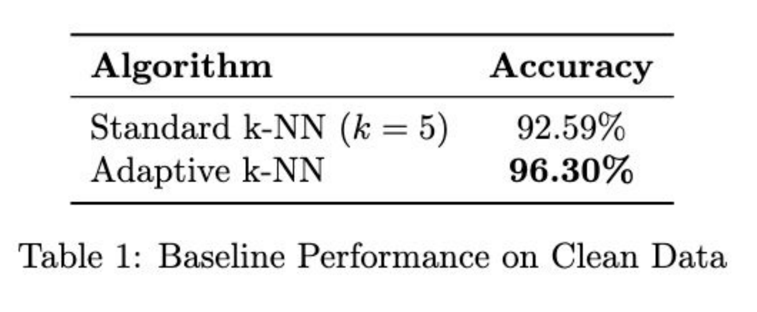
- UCI Wine Dataset: 178 samples across 3 distinct cultivars (classes).
- Features: 13 continuous chemical attributes
  - Examples: total phenols, flavonoids, magnesium
- Z-Score Normalization: Standardized all features to prevent magnitude bias.
- Equal Weighting: Ensures high-range features like Proline do not dominate distance.

Experimental Design

- Validation Protocol: 70/30 train-test split for all trials.
- Reproducibility: Used fixed random seed 97 for direct comparability.
- Five Testing Conditions:
  - 1. Baseline: Clean, standardized data.
  - 2. Gaussian Noise: Simulated broken sensor readings.
  - 3. Irrelevant Features: Added noise columns to test dimensionality.
  - 4. Feature Outliers: Massive geometric distortions in specific features.
  - 5. Class Imbalance: Artificial scarcity in minority classes.

Results: Baseline (Clean Data)

- Condition: Standard Wine dataset.
- Accuracy: 96.30% vs. 92.59% baseline.
- Winner: Adaptive (+3.70%).
- Behavior: Expanded k at boundaries for robust consensus.
- Efficiency: Stopped at small k for clear points.

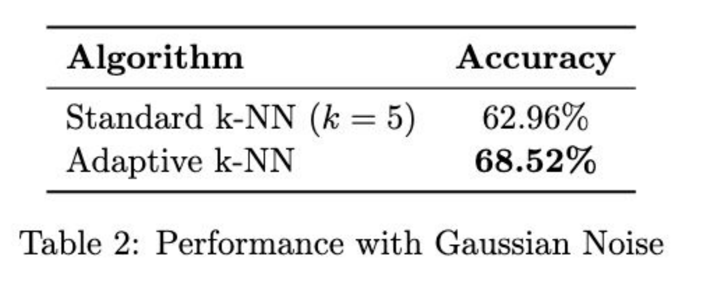


Algorithm	**Accuracy**
Standard k-NN ($k = 5$)	92.59%
Adaptive k-NN	**96.30%**

Table 1: Baseline Performance on Clean Data

Results: Noisy Features

- Gaussian Noise: Injected random noise (sigma=3) to simulate broken sensors.
- Performance Gap: Adaptive k-NN (68.52%) outperformed Standard k-NN (62.96%).
- Key Finding: Achieved a +5.56% improvement by expanding the search radius.
- Mechanism: Increased neighborhood size to average out and cancel high levels of random noise.

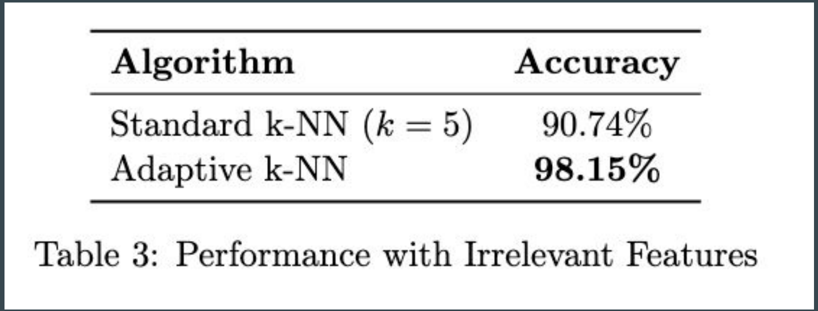


Algorithm	**Accuracy**
Standard k-NN ($k = 5$)	62.96%
Adaptive k-NN	**68.52%**

Table 2: Performance with Gaussian Noise

Results: Irrelevant Features (Dimensionality)

- Irrelevant Features (Exp 3): Appended 8 columns of pure random noise.
- Performance Gap: Adaptive k-NN (98.15%) vs. Standard k-NN (90.74%).
- Key Finding: Largest performance gain of the study at +7.41% improvement.
- Mechanism: Confidence-based statistical check rejected "fake" neighbors caused by noise dimensions.
- Reliability: Only made predictions when a local majority was statistically significant.

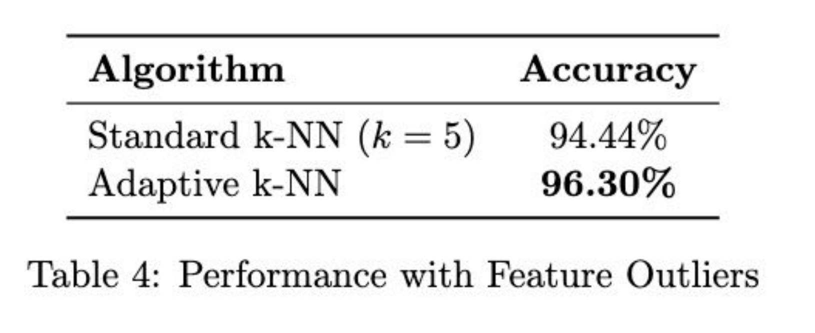


Algorithm	Accuracy
Standard k-NN ($k = 5$)	90.74%
Adaptive k-NN	**98.15%**

Table 3: Performance with Irrelevant Features

Results: Feature Outliers

- Accuracy: Adaptive (96.30%) beat Standard (94.44%).
- Winner: Adaptive (+1.85%).
- Mechanism: Expanded k to dilute outlier influence.
- Purity Check: Monitored class purity to trigger neighborhood growth.

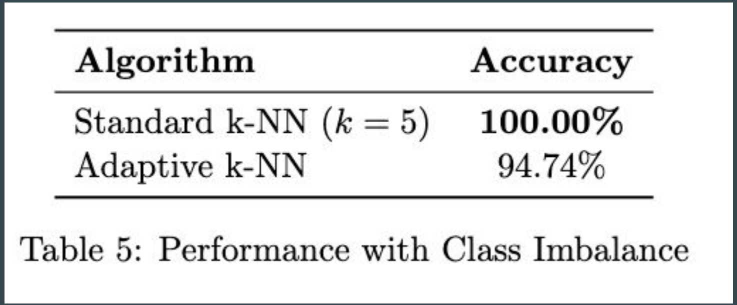


Algorithm	Accuracy
Standard k-NN ($k = 5$)	94.44%
Adaptive k-NN	**96.30%**

Table 4: Performance with Feature Outliers

Results: High Class Imbalance

- Condition: Removed 75% of Class 1 samples.
- Accuracy: Standard (100%) beat Adaptive (94.74%).
- Winner: Standard k-NN (+5.26%).
- Failure Point: Adaptive was overly cautious in sparse regions.
- Trade-off: Seeking high confidence pulled in incorrect neighbors.



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

Table 5: Performance with Class Imbalance

Conclusion

- Conclusion:
  - We successfully implemented a dynamic confidence rule.
  - The Adaptive algorithm is superior in "messy" environments (noise, outliers, extra dimensions).
  - It avoids overfitting local noise while preventing over smoothing in clean regions.
- Future Work:
  - Optimize the strictness parameter (C) dynamically during training.
  - Test on different distances (e.g., Manhattan & Minkowski distance)
  - Apply to larger datasets like MNIST or other real-world datasets to test trade-offs and performance.