Supplementary of "Data-Driven KNN Modeling via Optimized Weighted Distance Learning"

Liangliang Zhuang, Gauthier Barbosa, Zhisheng Ye

Contents

1 Weighted Distance Properties **2**

2 Derivation of the Surrogate Prediction **2**

3 Proof of Theorems **3**

3.1 Proof of Theorem 1 3

3.2 Proof of Theorem 2 4

3.3 Proof of Theorem 3 5

4 Implementation Details for Linearized Optimization Problem **6**

5 Additional Materials for Simulation Study **7**

5.1 Results for Continuous Outputs 7

5.2 Results for Categorical Outputs 7

5.3 Detailed Overview of State-of-the-art Methods 8

6 Additional Materials for Case Study **11**

6.1 Dataset Sources and Preprocessing Details 11

6.2 Additional Results for Case Study 15

The supplementary document is organized as follows. Section 1 presents key properties of the proposed weighted distance. Section 2 derives the surrogate prediction expression. Section 3 collects the proofs of theorems 1–3. Section 4 provides implementation details for solving the linearized optimization problem. Finally, Sections 5 and 6 report additional results for the simulation study and the case study, respectively.

S1 Weighted Distance Properties

We verify that the weighted distance function defined in Definition 1 satisfies the standard axioms of a metric and preserves the locality structure required for KNN-based learning. First, $d_{\boldsymbol{\alpha}}$ satisfies the four classical metric properties: (i) Non-negativity: Since each $\alpha_j > 0$ and $d_j(\cdot,\cdot)^2 \geq 0$, we have: $d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \geq 0$. (ii) Identity of indiscernibles: We have $d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = 0 \iff x_i^j = x_{i'}^j$ for all $j \in \{1, \ldots, \vartheta\} \iff \boldsymbol{x}_i = \boldsymbol{x}_{i'}$. (iii) Symmetry: For all i, i', we have: $d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = d_{\boldsymbol{\alpha}}(\boldsymbol{x}_{i'}, \boldsymbol{x}_i)$, since each $d_j(\cdot,\cdot)$ is symmetric. (iv) Triangle inequality: For any $\boldsymbol{x}_i$, $\boldsymbol{x}_{i'}$, and $\boldsymbol{x}_{i''}$, we use Minkowski's inequality: $d_j(x_i^j, x_{i''}^j) \leq d_j(x_i^j, x_{i'}^j) + d_j(x_{i'}^j, x_{i''}^j)$, which yields $d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i''}) \leq d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) + d_{\boldsymbol{\alpha}}(\boldsymbol{x}_{i'}, \boldsymbol{x}_{i''})$. Thus, $d_{\boldsymbol{\alpha}}$ is a valid metric on the feature space.

In addition to satisfying the formal properties of a metric, the proposed distance function $d_{\boldsymbol{\alpha}}$ also preserves locality—that is, it ensures that nearby points in the feature space have small distances and therefore exert greater influence in prediction. Specifically, if a sequence $\{\boldsymbol{x}_i\}$ converges component-wise to $\boldsymbol{x}$, i.e., $x_i^j \to x^j$ for all j, then $d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}) \to 0$. This continuity ensures that nearby points in feature space remain close under the learned metric, which is essential for maintaining the proximity structure that underlies nearest-neighbor learning.

S2 Derivation of the Surrogate Prediction

We present the derivation behind the surrogate objective used in our model. The goal is to reformulate the discrete neighbor selection in standard KNN into a smooth expression amenable to optimization. To begin, note that standard KNN prediction can be written as a combinatorial optimization problem:

$$\min_{\boldsymbol{\delta}} \ \frac{1}{n}\sum_{i=1}^{n}\left|\frac{1}{k}\sum_{\substack{i'=1\\ i'\neq i}}^{n}(\delta_{i,i'}y_{i'} - y_i)\right|_2 \tag{S1}$$

where $\delta_{i,i'} \in \{0,1\}$ indicates whether $x_{i'}$ is one of the k nearest neighbors of x_i:

$$\delta_{i,i'} = \begin{cases} 1, & \text{if } i' \in \arg\min_k\{d(x_i, x_{i'})\}, \\ 0, & \text{otherwise.} \end{cases}$$

This formulation is non-differentiable due to the use of a *hard* neighbor selection function—i.e., only the k closest points receive weight, while all others are ignored. To enable gradient-based optimization, we adopt a *soft* neighbor approximation, which assigns a smoothly decaying weight to each training point based on its distance. Specifically, we convert distances into similarity scores, following conventions in prior work such as Boriah et al. (2008); Abu Alfeilat et al. (2019). The similarity function is defined using the inverse of squared distances:

$$S_{i,i'} = \left(1 + d(x_i, x_{i'})^2\right)^{-1}.$$

This transformation ensures that nearby points yield higher similarity values. To replace the indicator $\delta_{i,i'}$, we use a smooth approximation that emphasizes large similarities:

$$\delta_{i,i'} \approx \frac{S_{i,i'}^{1/k}}{\sum_{i'' \neq i} S_{i,i''}^{1/k}}. \tag{S2}$$

Raising $S_{i,i'}$ to the power $1/k$ provides a smooth distance-decay weighting, where k controls the decay rate of neighbor influence. Substituting Equation (S2) into the objective (S1), we obtain the smooth approximation:

$$\min_{\boldsymbol{\alpha}} \frac{1}{n} \sum_{i=1}^{n} \left| \sum_{\substack{i'=1 \\ i' \neq i}}^{n} \left(\frac{S_{i,i'}^{1/k}}{\sum_{\substack{i''=1 \\ i'' \neq i}}^{n} S_{i,i''}^{1/k}} y_{i'} - y_i \right) \right|. \tag{S3}$$

S3 Proof of Theorems

S3.1 Proof of Theorem 1

We analyze the optimization problem defined in Theorem 1, which is used to learn the feature weights $\boldsymbol{\alpha}$. We show that the problem is both feasible and convex, ensuring theoretical soundness and tractable solution via LP framework. To guarantee feasibility, we construct an explicit weight vector. Let

$$\alpha_j = \frac{1}{2\vartheta \max_{i,i'} d_j(x_i^j, x_{i'}^j)^2},$$

assuming each feature varies across samples. This ensures that for any pair (i, i'),

$$d^2_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \leq \sum_{j=1}^{\vartheta} \frac{1}{2\vartheta} = \frac{1}{2},$$

satisfying the linearization constraint. The prediction values $\hat{y}_i$ can be chosen arbitrarily, and the error terms defined as $e_i = |\hat{y}_i - y_i|$ satisfy the absolute value constraints. Hence, the constraint set admits feasible solutions.

For convexity, we observe that the objective function $\frac{1}{n}\sum_i e_i$ is linear. All constraints are either linear or affine in the decision variables: the absolute error constraints are linear in e_i and $\hat{y}_i$, the predicted output $\hat{y}_i$ is affine in $\boldsymbol{\alpha}$, and the squared distance expressions are linear in $\boldsymbol{\alpha}$ due to fixed data. The feasible set is thus convex. While the objective is not strictly convex, optimal solutions always exist, possibly non-unique up to scaling. This does not impact KNN behavior, which depends only on relative distances.

S3.2 Proof of Theorem 2

To begin, we observe that the feasible set for $\boldsymbol{\alpha} \in \mathbb{R}^{\vartheta}_{+}$ is compact. This follows from the constraints $\alpha_j > 0$ and $d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})^2 \leq 1/2$ for all i, i'. Since the data lie in a bounded domain, there exists a constant $D_{\max} > 0$ such that

$$d_j(x^j_i, x^j_{i'})^2 \leq D^2_{\max} \quad \text{for all } j.$$

Then, each weight is upper bounded:

$$\alpha_j \leq \frac{1}{2D^2_{\max}},$$

so the feasible region lies within a closed and bounded subset of $\mathbb{R}^{\vartheta}_{+}$, and is thus compact. As the optimization problem is convex and feasible, the sequence $\hat{\boldsymbol{\alpha}}(n, k)$ admits at least one convergent subsequence.

Next, we show that the limit is unique. Consider a dataset of size n_0 with k_0 neighbors and let $\boldsymbol{\alpha}^*$ denote the corresponding optimizer. Suppose we augment this dataset by duplicating existing points with identical features and labels. The empirical error remains unchanged, since the duplicated points contribute identical prediction errors. Consequently, the objective function is unaffected and $\boldsymbol{\alpha}^*$ remains optimal for the enlarged dataset.

This invariance implies that adding points that do not alter the empirical distribution does not perturb the solution. Since the dataset grows while retaining its local geometry,

the optimization landscape becomes increasingly stable. It follows that the full sequence $\hat{\boldsymbol{\alpha}}(n,k)$ must converge to the same limit:

$$\hat{\boldsymbol{\alpha}}(n,k) \longrightarrow \bar{\boldsymbol{\alpha}} \quad \text{as } n \to \infty,\ k \to \infty,\ \text{and } k/n \to 0.$$

This completes the proof.

S3.3 Proof of Theorem 3

We provide a unified proof that the proposed estimator satisfies Fisher consistency. Specifically, we show that under standard asymptotic conditions, the prediction $\tilde{y}_\ell$ converges to the Bayes-optimal decision for any new input $\boldsymbol{x}_\ell$ (Tasche, 2017). Let the prediction take the general form:

$$\tilde{y}_\ell = \frac{\sum_{i \in \text{neigh}_k(\boldsymbol{x}_\ell)} y_i w_i(\boldsymbol{x}_\ell)}{\sum_{i \in \text{neigh}_k(\boldsymbol{x}_\ell)} w_i(\boldsymbol{x}_\ell)}, \quad \text{where } w_i(\boldsymbol{x}_\ell) = \frac{1}{1 + d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_\ell)^2}.$$

As $n, k \to \infty$ with $k/n \to 0$, the nearest neighbors of $\boldsymbol{x}_\ell$ become arbitrarily close. By the stability of $\boldsymbol{\alpha}$, and assuming the uniform law of large numbers applies, we proceed case by case.

(i) Binary Classification: Let $Y \in \{0, 1\}$, and define the regression function $\eta(\boldsymbol{x}) = \mathbb{P}(Y = 1 \mid X = \boldsymbol{x})$. Under the prediction rule, define:

$$u_\ell = \frac{\sum_{i \in \text{neigh}_k(\boldsymbol{x}_\ell)} y_i w_i(\boldsymbol{x}_\ell)}{\sum_{i \in \text{neigh}_k(\boldsymbol{x}_\ell)} w_i(\boldsymbol{x}_\ell)}, \quad \tilde{y}_\ell = \mathbb{1}\left\{u_\ell \geq \frac{1}{2}\right\}.$$

As $k \to \infty$, the neighbors $\boldsymbol{x}_i \in \text{neigh}_k(\boldsymbol{x}_\ell)$ satisfy $\boldsymbol{x}_i \to \boldsymbol{x}_\ell$. Thus, by continuity of $\eta(\cdot)$, we have $\eta(\boldsymbol{x}_i) \to \eta(\boldsymbol{x}_\ell)$. If $y_i = \eta(\boldsymbol{x}_i) + \iota_i$, where ι_i is zero-mean noise with finite variance, then by the uniform law of large numbers (ULLN), we have:

$$u_\ell \to \eta(\boldsymbol{x}_\ell), \quad \text{and hence} \quad \tilde{y}_\ell \to \mathbb{1}\left\{\eta(\boldsymbol{x}_\ell) \geq \frac{1}{2}\right\}.$$

This proves consistency with the Bayes-optimal classifier.

(ii) Multi-Class Classification: Let $Y \in \{1, \ldots, C\}$, and define $\eta_c(\boldsymbol{x}) = \mathbb{P}(Y = c \mid X = \boldsymbol{x})$. The prediction rule becomes:

$$\tilde{y}_\ell = \arg\max_{c \in \mathcal{Y}} \frac{\sum_{i \in \text{neigh}_k(\boldsymbol{x}_\ell)} \mathbb{1}\{y_i = c\} w_i(\boldsymbol{x}_\ell)}{\sum_{i \in \text{neigh}_k(\boldsymbol{x}_\ell)} w_i(\boldsymbol{x}_\ell)}.$$

As in the binary case, the numerator converges to $\eta_c(\boldsymbol{x}_\ell)$, and the denominator normalizes the weights. Therefore,

$$\tilde{y}_\ell \to \arg\max_{c \in \mathcal{Y}} \eta_c(\boldsymbol{x}_\ell),$$

which is the Bayes-optimal multi-class classifier.

(iii) Regression: Assume $Y \in \mathbb{R}$, and define the conditional expectation $\mu(\boldsymbol{x}) = \mathbb{E}[Y \mid X = \boldsymbol{x}]$. Then the prediction satisfies:

$$\tilde{y}_\ell = \frac{\sum_{i \in \text{neigh}_k(\boldsymbol{x}_\ell)} y_i w_i(\boldsymbol{x}_\ell)}{\sum_{i \in \text{neigh}_k(\boldsymbol{x}_\ell)} w_i(\boldsymbol{x}_\ell)}.$$

Write $y_i = \mu(\boldsymbol{x}_i) + \iota_i$. As before, by continuity of μ and the ULLN, we obtain:

$$\tilde{y}_\ell \to \mu(\boldsymbol{x}_\ell) = \mathbb{E}[Y \mid X = \boldsymbol{x}_\ell].$$

In all cases, the estimator converges to the Bayes-optimal prediction: $\tilde{y}_\ell \longrightarrow h^*(\boldsymbol{x}_\ell)$, with h^* denoting the Bayes rule. This establishes Fisher consistency of the proposed method under the asymptotic settings $n \to \infty$, $k \to \infty$, $k/n \to 0$, and weight stability.

S4 Implementation Details for Linearized Optimization Problem

We implement the linearized version of our optimization problem using the Gurobi solver in Python. Gurobi is well-suited for solving large-scale linear programs with structured constraints (Anand et al., 2017), and its integration with Python—alongside libraries such as NumPy and Pandas—facilitates efficient modeling and data preprocessing. The optimization procedure consists of the following steps:

(i) Define the decision variables α_j, which represent the feature-level weights used in the distance metric. the objective function (empirical prediction error). Set up the objective function and the associated linear constraints.

(ii) Solve the resulting linear program using Gurobi' s Simplex or Barrier method to obtain a feasible solution.

(iii) Refine the solution using primal-dual or dual simplex iterations to improve optimality.

(iv) Monitor convergence and feasibility to ensure that the computed weights respect the constraints imposed by the distance metric.

To ensure positivity of the weights α_j, we impose a lower bound constraint: $\alpha_j \geq 10^{-5}$. This avoids degeneracy and ensures that the learned metric remains well-defined throughout optimization.

S5 Additional Materials for Simulation Study

S5.1 Results for Continuous Outputs

Table S1 reports the predictive performance of our method across varying combinations of total sample size $n+m$, feature dimension ϑ, and data-generating mechanisms. The results highlight how linear, quadratic, and interaction effects influence model accuracy, providing a comprehensive assessment of robustness under different structural assumptions.

S5.2 Results for Categorical Outputs

We evaluate predictive accuracy under three generative mechanisms adapted for categorical outputs: linear, quadratic, and interaction-based models. Each data instance $\boldsymbol{X}_i$ consists of ϑ features, with n training and n test points. The response variable y_i is generated according to the following logistic models (Gogoshin et al., 2021):

(i) Linear: $y_i = \frac{e^{\sum_{j=1}^{p} a_i^j x_i^j + \sum_{j=p+1}^{q+p} b_i^j x_i^j}}{1+e^{\sum_{j=1}^{p} a_i^j x_i^j + \sum_{j=p+1}^{q+p} b_i^j x_i^j}} + \epsilon.$

(ii) Quadratic: $y_i = \frac{e^{\sum_{j=1}^{p} a_i^j (x_i^j)^2 + \sum_{j=p+1}^{q+p} b_i^j (x_i^j)^2}}{1+e^{\sum_{j=1}^{p} a_i^j (x_i^j)^2 + \sum_{j=p+1}^{q+p} b_i^j (x_i^j)^2}} + \epsilon.$

(iii) Interaction: $y_i = \frac{e^{\sum_{j=1}^{p} a_i^j x_i^j + \sum_{j=p+1}^{q+p} b_i^j x_i^j + \sum_{j=1}^{\vartheta} c_i^j (x_i^j)^2 + \sum_{j=1}^{\vartheta} \sum_{j'=1}^{\vartheta} s_i^{j,j'} x_i^j x_i^{j'}}}{1+e^{\sum_{j=1}^{p} a_i^j x_i^j + \sum_{j=p+1}^{q+p} b_i^j x_i^j + \sum_{j=1}^{\vartheta} c_i^j (x_i^j)^2 + \sum_{j=1}^{\vartheta} \sum_{j'=1}^{\vartheta} s_i^{j,j'} x_i^j x_i^{j'}}} + \epsilon.$

The categorical response is then obtained by thresholding the noisy score at 0.5. Here, ϵ perturbs the latent response score rather than directly flipping observed class labels. All other simulation parameters follow the setup described in Section 6. Table S2 summarizes performance across varying combinations of $n+m$, ϑ, and data-generating mechanisms. As with the continuous output case, predictive accuracy improves with larger sample size and fewer features. The proposed method remains robust across all structural settings.

Table S1: The predictive performance of the proposed method under different combinations of $n+m$, ϑ, and data-generating mechanisms (with y as a continuous variable).

Type	$n+m$	Matrix	$\vartheta=4$			$\vartheta=6$			$\vartheta=8$		
			(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
Linear	100	MSE	201.18	71.76	22.30	285.73	194.28	81.29	498.66	303.07	154.75
		R^2	0.56	0.73	0.90	0.51	0.61	0.78	0.41	0.53	0.69
	200	MSE	150.24	91.78	11.25	282.03	166.01	104.28	443.39	280.34	101.17
		R^2	0.70	0.71	0.93	0.52	0.66	0.84	0.42	0.52	0.75
	500	MSE	124.61	74.03	3.52	236.05	184.14	41.35	348.27	232.78	115.50
		R^2	0.72	0.72	0.98	0.58	0.64	0.90	0.47	0.63	0.84
Quadratic	100	MSE	484.36	327.20	104.06	799.31	647.05	359.53	1179.96	1191.91	769.77
		R^2	0.36	0.60	0.90	0.21	0.52	0.78	0.11	0.38	0.66
	200	MSE	471.50	309.17	48.21	717.21	638.78	229.04	1149.71	999.98	651.27
		R^2	0.50	0.65	0.96	0.39	0.47	0.85	0.28	0.42	0.74
	500	MSE	340.34	172.16	8.36	727.58	620.21	137.97	856.64	1106.28	325.12
		R^2	0.63	0.75	0.99	0.48	0.61	0.92	0.42	0.44	0.84
Interaction	100	MSE	220.57	75.35	27.16	334.95	195.60	83.05	441.58	402.19	262.88
		R^2	0.56	0.66	0.86	0.36	0.49	0.75	0.40	0.39	0.58
	200	MSE	182.73	70.37	13.92	256.37	184.53	67.40	440.06	224.65	141.30
		R^2	0.60	0.81	0.92	0.53	0.62	0.83	0.44	0.59	0.72
	500	MSE	56.15	29.95	2.56	145.28	96.25	32.49	319.37	241.20	82.89
		R^2	0.85	0.89	0.98	0.74	0.77	0.90	0.61	0.64	0.83

S5.3 Detailed Overview of State-of-the-art Methods

Three categories of state-of-the-art methods are considered, with details and implementation instructions provided below:

(i) **Traditional KNN Method**: In traditional KNN, neighbors are selected based on their proximity to the query instance within the feature space, typically using the Euclidean distance metric. The similarity S between instances $\boldsymbol{x_i}$ and $\boldsymbol{x_{i'}}$ is inversely related to their distance $d(\boldsymbol{x_i}, \boldsymbol{x_{i'}})$, calculated as $d(\boldsymbol{x_i}, \boldsymbol{x_{i'}}) = \sqrt{\sum_{j=1}^{\vartheta}(x_i^j - x_{i'}^j)^2}$. Neigh-

Table S2: The predictive performance of the proposed method under different combinations of $n+m$, ϑ, and data-generating mechanisms (with y as a categorical variable).

Type	$n+m$	Matrix	$\vartheta=4$			$\vartheta=6$			$\vartheta=8$		
			(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
Linear	50	Accuracy	0.86	0.87	0.92	0.75	0.82	0.80	0.70	0.83	0.85
		Precision	0.87	0.87	0.92	0.82	0.84	0.80	0.75	0.81	0.88
	100	Accuracy	0.81	0.84	0.95	0.75	0.85	0.89	0.66	0.82	0.86
		Precision	0.87	0.85	0.96	0.78	0.85	0.88	0.72	0.81	0.86
	500	Accuracy	0.80	0.90	0.97	0.76	0.84	0.90	0.69	0.82	0.90
		Precision	0.80	0.90	0.97	0.77	0.84	0.90	0.70	0.81	0.90
Quadratic	50	Accuracy	0.81	0.84	0.89	0.71	0.77	0.82	0.71	0.72	0.78
		Precision	0.81	0.85	0.91	0.70	0.77	0.81	0.71	0.72	0.77
	100	Accuracy	0.79	0.85	0.94	0.69	0.77	0.86	0.64	0.73	0.80
		Precision	0.80	0.85	0.94	0.68	0.77	0.87	0.64	0.72	0.80
	500	Accuracy	0.76	0.87	0.97	0.67	0.78	0.88	0.66	0.73	0.82
		Precision	0.75	0.87	0.97	0.66	0.77	0.88	0.64	0.72	0.82
Interaction	50	Accuracy	0.79	0.82	0.81	0.69	0.77	0.71	0.64	0.73	0.71
		Precision	0.83	0.80	0.81	0.71	0.78	0.70	0.66	0.73	0.71
	100	Accuracy	0.76	0.84	0.79	0.66	0.81	0.69	0.64	0.72	0.64
		Precision	0.81	0.84	0.80	0.67	0.81	0.69	0.68	0.72	0.64
	500	Accuracy	0.77	0.88	0.76	0.64	0.79	0.67	0.63	0.74	0.64
		Precision	0.77	0.88	0.75	0.67	0.79	0.66	0.63	0.73	0.66

bors with the smallest distances are chosen as the nearest neighbors.

(ii) **Similarity Measure Methods**: In Boriah et al. (2008), the authors propose a range of similarity-based approaches for nearest neighbor selection, wherein the neighbors in the KNN algorithm are determined based on the highest similarity scores. Adopting the notation from that work, the similarity between two instances $\boldsymbol{x_i}$ and $\boldsymbol{x_{i'}}$ is defined

as

$$S(\boldsymbol{x_i}, \boldsymbol{x_{i'}}) = \sum_{j=1}^{\vartheta} w_j S_j(\boldsymbol{x_i}, \boldsymbol{x_{i'}}),$$

where $S_j(\boldsymbol{x_i}, \boldsymbol{x_{i'}})$ denotes the similarity measure specific to feature i. This formulation serves as the foundation for the similarity measures explored in Boriah et al. (2008), with different approaches emerging from the definitions of w_i and S_j. In our study, we focus on three specific similarity measures—*Overlap*, *Occur-Frequency (OF)*, and *Eskin*—chosen for their conceptual clarity and practical effectiveness in handling categorical data. For these methods, we adopt uniform weighting, setting $w_i = 1/\vartheta$ for all features. The similarity score $S_j(\boldsymbol{x_i}, \boldsymbol{x_{i'}})$ is defined as 1 when the categorical values of feature i are identical in both instances. When the values differ, the similarity is computed as follows:

(a) **Overlap:** $S_j = 0$.

(b) **OF:**

$$S_j(j_{tN}, j_{tL})^{-1} = \left(1 + \log f_{ij=j_{tN}}(\boldsymbol{X_N})\right) \left(\log f_{j=j_{tL}}(\boldsymbol{X_L})\right),$$

where $f_{j=j_{tN}}(\boldsymbol{X_N})$ and $f_{j=j_{tL}}(\boldsymbol{X_L})$ denote the empirical frequencies of values j_{tN} and j_{tL} in the training and prediction datasets, respectively.

(c) **Eskin:**

$$S_j = \frac{n_j^2}{n_j^2 + 1},$$

where n_j is the number of unique categories (i.e., the cardinality) of feature j.

(iii) **Generalized Weighted Distance KNN**: In the study presented by Rastin et al. (2021), the authors introduce a sophisticated method characterized by a complex training phase. The methodology is based on weighting each neighbor; to adapt it to our notations, this is expressed as

$$d(\boldsymbol{x_i}, \boldsymbol{x_{i'}}) = \frac{||\boldsymbol{x_i} - \boldsymbol{x_{i'}}||}{w_i},$$

where the weight w_i is associated with each neighbor. Subsequently, a loss $\mathcal{L}$ is computed as the average of the loss of neighbors with the same label and those with different labels. A gradient is then computed to update the weights as

$$G = \sum_{neigh} \sigma'(\mathcal{L}) \frac{1}{w_i},$$

where $\sigma'(\mathcal{L}) = \beta\sigma(\mathcal{L})(1-\sigma(\mathcal{L}))$ and σ is the sigmoid function. Finally, the weights are updated using a gradient descent technique:

$$w_i \leftarrow w_i - \eta \cdot G,$$

where η is the learning rate. In this framework, there are two tuning parameters: η and β. To implement this method, we attempted to obtain optimal hyperparameters, similar to the approach used for k in other methods, based on the suggested best values: $\beta = 8$ and $\eta = 0.1$. The prediction is then made using the computed weight for each neighbor.

S6 Additional Materials for Case Study

This section provides supplementary details for the real-world case study. Section 6.1 gives a detailed overview of the benchmark datasets, and Section 6.2 reports additional experimental results, including full performance tables, significance tests, and runtime summaries.

S6.1 Dataset Sources and Preprocessing Details

Here, we provide dataset-level descriptions, source information, preprocessing details, and the processed sample sizes actually used in the experiments. It should be noted that some datasets required only limited data representation or missing-value handling before model fitting; therefore, the training and test sample sizes reported below correspond to the processed datasets used in the experiments rather than necessarily the raw repository totals.

For datasets with categorical responses:

(1) **Houses votes**: This dataset is from the UCI Machine Learning Repository[1] and contains voting records of U.S. House of Representatives members on key issues, with the goal of predicting party affiliation. The original dataset contains 435 observations and 16 categorical predictors. In preprocessing, the voting entries were encoded as numerical values: "y" $\rightarrow$ 1, "n" $\rightarrow$ 0, and "?" $\rightarrow$ 0.5. The response variable `party` was encoded as a binary label. The processed dataset used in the experiments was split into 348 training samples and 87 test samples.

(2) **Ionosphere**: This dataset is from the UCI repository[2] and consists of radar signal

[1] https://archive.ics.uci.edu/ml/datasets/congressional+voting+records

[2] https://archive.ics.uci.edu/dataset/52/ionosphere

measurements used to classify ionospheric returns as either "good" or "bad". The dataset contains 351 observations and 34 numerical predictors. No feature preprocessing was applied, and only the response labels were encoded as binary values. The processed dataset used in the experiments was split into 280 training samples and 71 test samples.

(3) **Steel Industry:** The source data were obtained from the Steel Industry Energy Consumption dataset[3]. In our experiments, this dataset was used to construct a classification task with `Load_Type` as the categorical response. The original `date` variable was parsed into datetime format, and only observations recorded at hours 00, 08, and 16 with minute equal to 00 were retained. From the date field, three derived variables (`day`, `hour`, and `minute`) were extracted, and the original `date` column was removed. The variables `WeekStatus` and `Day_of_week` were numerically encoded, and `Load_Type` was label-encoded as the target. After preprocessing, the dataset used in the experiments contained 1095 observations with 6 categorical/discrete predictors and 6 numerical predictors, and was split into 876 training samples and 219 test samples.

(4) **Pima Indians Diabetes**: This dataset was obtained from the Kaggle mirror of the UCI benchmark[4] and contains medical measurements used to predict diabetes onset. The dataset contains 768 observations and 8 numerical predictors. It was directly loaded and used without additional preprocessing. The response variable `Outcome` is binary. The processed dataset used in the experiments was split into 614 training samples and 154 test samples.

(5) **Liver Disorders**: The dataset used in our experiments corresponds to the Indian Liver Patient Dataset (ILPD)[5]. It contains 583 observations and 10 predictors, including one categorical variable (`Gender`) and nine numerical variables. In preprocessing, `Gender` was encoded numerically, and missing numerical values were imputed using column means. The response variable `Selector` was treated as a binary label. The processed dataset used in the experiments was split into 466 training samples and 117 test samples.

[3] https://www.kaggle.com/datasets/joebeachcapital/steel-industry-energy-consumption

[4] https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

[5] https://archive.ics.uci.edu/ml/datasets/ILPD%2B%28Indian%2BLiver%2BPatient%2BDataset%29

(6) **Iris:** This classic dataset is from the UCI repository[6] and contains 150 observations of iris flowers, each described by 4 numerical predictors (sepal length, sepal width, petal length, and petal width). The task is three-class classification. No feature preprocessing was applied, and the class labels were encoded as integer class indices. The processed dataset used in the experiments was split into 120 training samples and 30 test samples.

(7) **Heart Disease:** This dataset is based on the UCI Heart Disease repository[7], and in our experiments we used the processed Cleveland subset. The predictors include both categorical/discrete and numerical variables, containing both categorical/discrete and numerical variables. In preprocessing, entries represented by "?" were treated as missing values and removed. The original response variable was then converted into a binary label by grouping all nonzero disease levels into the positive class. After removing incomplete observations, the processed dataset contained 297 samples and was split into 237 training samples and 60 test samples.

(8) **Australian Credit:** This dataset is from the UCI repository[8] and contains 690 observations with 14 predictors. In the file used in our experiments, the predictors are stored numerically, but they include both discrete/categorical-like and numerical variables. The response variable was converted into a binary decision label. No additional feature preprocessing was applied. The processed dataset used in the experiments was split into 552 training samples and 138 test samples.

(9) **Sonar:** This dataset is from the UCI repository[9] and consists of 208 observations described by 60 numerical sonar signal predictors. The task is binary classification (mine versus rock). No feature preprocessing was applied, and only the response labels were encoded as binary values. The processed dataset used in the experiments was split into 166 training samples and 42 test samples.

(10) **Balance Scale:** This dataset is from the UCI repository[10] and contains 625 observations. It includes 4 discrete predictors describing left/right weight and distance config-

[6] https://archive.ics.uci.edu/dataset/53/iris

[7] https://archive.ics.uci.edu/dataset/45/heart+disease

[8] https://archive.ics.uci.edu/dataset/143/statlog+australian+credit+approval

[9] https://archive.ics.uci.edu/dataset/151/connectionist+bench+sonar+mines+vs+rocks

[10] https://archive.ics.uci.edu/dataset/12/balance+scale

urations, and a three-class categorical response indicating left, balanced, or right. In our implementation, these input variables were grouped into the categorical/discrete side. The response labels "L", "B", and "R" were encoded as integer class labels. The processed dataset used in the experiments was split into 500 training samples and 125 test samples.

For datasets with continuous responses:

(1) **Insurance Charges**: This dataset was obtained from Kaggle[11] and contains 1338 observations with 6 predictors. In our experiments, `sex`, `smoker`, and `region` were encoded numerically, while the remaining predictors were already numerical. The response variable `charges` was treated as continuous. The processed dataset used in the experiments was split into 1070 training samples and 268 test samples.

(2) **Servo**: This dataset is from the UCI repository[12] and contains 167 observations with 4 predictors. The variables `motor` and `screw` are categorical, while `pgain` and `vgain` are numerical. In preprocessing, `motor` and `screw` were label-encoded. The response variable `class` was treated as continuous. The processed dataset used in the experiments was split into 134 training samples and 33 test samples.

(3) **Concrete Strength:** This dataset is from the UCI repository[13] and contains 1030 observations with 8 numerical predictors. The response variable is concrete compressive strength and is continuous. The dataset was directly loaded and used without additional preprocessing. The processed dataset used in the experiments was split into 824 training samples and 206 test samples.

(4) **Airfoil Self-Noise:** This dataset is from the UCI repository[14] and contains 1503 observations with 5 numerical predictors. The response variable is the scaled sound pressure level and is continuous. The processed dataset used in the experiments was split into 1202 training samples and 301 test samples.

[11] https://www.kaggle.com/datasets/teertha/ushealthinsurancedataset

[12] https://archive.ics.uci.edu/dataset/87/servo

[13] https://archive.ics.uci.edu/dataset/165/concrete+compressive+strength

[14] https://archive.ics.uci.edu/dataset/291/airfoil+self-noise

S6.2 Additional Results for Case Study

Figure S1 provides a visual comparison of representative KNN variants across the benchmark datasets, complementing the complete numerical results reported in the main text.

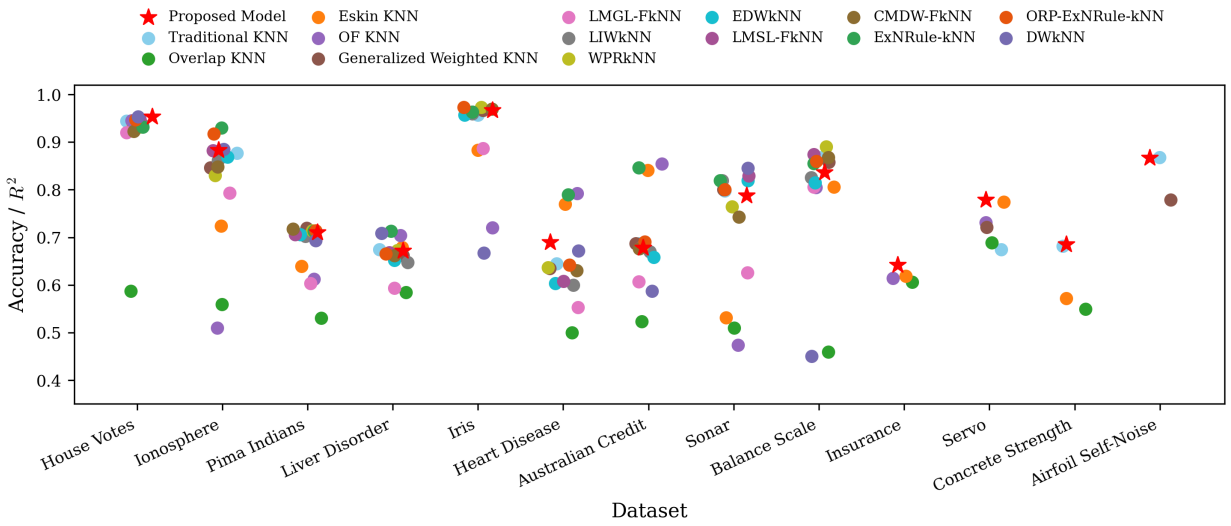

Figure S1: Comparison of method performance across several datasets, highlighting cross-dataset differences among competing KNN variants.

Tables S3 and S4 present the results of the Wilcoxon signed-rank test based on the same 10 repeated splits. Each entry reports the numbers of wins, ties, and losses of the proposed method against a given baseline, together with the corresponding p-value. Overall, the results show that the proposed method achieves competitive performance compared with most baseline methods across datasets. On several datasets, such as *Ionosphere*, *Pima Indians*, *Insurance*, and *Servo*, the proposed method records more wins than losses against many baselines and in some cases achieves statistically significant improvements. However, the advantages are not uniform across all comparisons. In particular, strong rule-based ensemble variants such as ExNRule-kNN and ORP-ExNRule-kNN achieve comparable or better results on some datasets. These results indicate that while the proposed method provides robust and competitive performance across diverse datasets, certain specialized ensemble approaches may still achieve stronger results in specific cases.

We further examine performance sensitivity to the neighborhood size k. Table S5 summarizes the average performance ranks across the classification datasets for $k \in \{3, 5, 7, 9\}$ (smaller is better), where the number in parentheses denotes the overall rank among all compared methods. We focus on classification because several advanced kNN variants are not

Table S3: Wilcoxon signed-rank test summary (10 repeated splits) comparing each baseline against the proposed method. Each entry reports Wins/Ties/Losses, with the corresponding p-value shown in parentheses.

Baseline	House Votes	Ionosphere	Steel Industry	Pima Indians	Liver Disorder	Iris	Heart Disease	Australian Credit	Sonar	Balance Scale
Overlap	10/0/0	10/0/0	10/0/0	9/0/1	5/0/5	10/0/0	9/0/1	10/0/0	10/0/0	10/0/0
	(0.002)	(0.002)	(0.002)	(0.004)	(0.814)	(0.002)	(0.004)	(0.002)	(0.002)	(0.002)
Eskin	7/1/2	9/1/0	10/0/0	10/0/0	6/0/4	9/1/0	1/2/7	0/0/10	10/0/0	7/2/1
	(0.070)	(0.004)	(0.002)	(0.002)	(0.748)	(0.004)	(0.023)	(0.002)	(0.002)	(0.031)
OF	5/3/2	10/0/0	10/0/0	10/0/0	2/0/8	10/0/0	1/0/9	0/0/10	10/0/0	8/0/2
	(0.281)	(0.002)	(0.002)	(0.002)	(0.066)	(0.002)	(0.004)	(0.002)	(0.002)	(0.029)
GwKNN	7/0/3	7/0/3	5/3/2	3/0/7	6/2/2	2/6/2	7/0/3	5/1/4	5/1/4	3/1/6
	(0.160)	(0.049)	(0.609)	(0.533)	(0.266)	(1.000)	(0.062)	(0.766)	(0.953)	(0.223)
LMGL-FkNN	8/1/1	9/1/0	10/0/0	9/0/1	8/1/1	9/1/0	9/0/1	10/0/0	10/0/0	8/0/2
	(0.020)	(0.004)	(0.002)	(0.004)	(0.008)	(0.004)	(0.006)	(0.002)	(0.002)	(0.021)
LIWkNN	8/2/0	5/2/3	9/1/0	6/0/4	6/1/3	2/8/0	8/1/1	5/1/4	2/1/7	7/0/3
	(0.008)	(0.336)	(0.004)	(0.611)	(0.113)	(0.500)	(0.012)	(0.801)	(0.164)	(0.342)
WPRkNN	8/2/0	8/0/2	6/1/3	4/1/5	3/3/4	0/8/2	6/1/3	3/3/4	6/1/3	0/0/10
	(0.008)	(0.012)	(0.465)	(0.703)	(0.828)	(0.500)	(0.078)	(0.641)	(0.457)	(0.002)
EDWkNN	6/3/1	5/1/4	9/0/1	5/0/5	7/0/3	3/7/0	9/0/1	5/3/2	2/1/7	6/2/2
	(0.062)	(0.371)	(0.006)	(0.977)	(0.164)	(0.250)	(0.008)	(0.156)	(0.168)	(0.289)
LMSL-FkNN	5/0/5	5/1/4	8/0/2	5/0/5	7/0/3	1/9/0	9/0/1	4/1/5	1/1/8	0/2/8
	(0.807)	(1.000)	(0.082)	(0.842)	(0.477)	(1.000)	(0.014)	(1.000)	(0.031)	(0.008)
CMDW-FkNN	9/1/0	7/0/3	4/1/5	3/1/6	8/0/2	0/9/1	7/1/2	4/2/4	6/1/3	1/0/9
	(0.004)	(0.033)	(0.883)	(0.371)	(0.311)	(1.000)	(0.035)	(0.812)	(0.168)	(0.008)
ExNRule-kNN	9/0/1	0/2/8	9/0/1	5/1/4	2/0/8	3/5/2	0/0/10	0/0/10	3/1/6	3/1/6
	(0.010)	(0.008)	(0.012)	(0.977)	(0.014)	(1.000)	(0.002)	(0.002)	(0.324)	(0.203)
ORP-ExNRule-kNN	5/3/2	1/1/8	7/2/1	3/0/7	5/2/3	1/7/2	6/1/3	3/1/6	4/1/5	1/1/8
	(0.281)	(0.016)	(0.039)	(0.566)	(0.516)	(0.750)	(0.172)	(0.148)	(0.582)	(0.031)
DWkNN	2/3/5	4/3/3	8/0/2	6/0/4	2/0/8	10/0/0	5/2/3	10/0/0	2/2/6	10/0/0
	(0.453)	(0.938)	(0.088)	(0.359)	(0.008)	(0.002)	(0.484)	(0.002)	(0.062)	(0.002)

Table S4: Wilcoxon signed-rank test summary (10 repeated splits) on regression datasets. Each entry reports Wins/Ties/Losses with the corresponding p-values in parentheses.

Baseline	Insurance	Servo	Concrete Strength	Airfoil Self-Noise
Traditional KNN	8/0/2 (0.010)	9/0/1 (0.010)	8/0/2 (0.432)	3/1/2 (1.000)
Overlap KNN	7/0/3 (1.000)	7/0/3 (0.049)	10/0/0 (0.002)	5/0/0 (0.062)
Eskin KNN	7/0/3 (1.000)	6/0/4 (0.770)	10/0/0 (0.002)	5/0/0 (0.062)
OF KNN	7/0/3 (1.000)	8/0/2 (0.105)	10/0/0 (0.002)	5/0/0 (0.062)
GwKNN	4/0/1 (0.125)	6/0/4 (0.131)	9/0/0 (0.004)	4/0/1 (0.312)

applicable to continuous-output regression; on the regression datasets, the proposed method also shows competitive average ranking performance across the tested k values (see the supplementary material). Overall, the proposed method maintains stable mid-to-top rankings across all tested k, indicating robust performance with respect to k. The ensemble-based variants (ExNRule-kNN and ORP-ExNRule-kNN) are also frequently ranked among the top methods, albeit with higher prediction-time cost, as reflected in the timing results.

Table S5: Average performance ranks of KNN-based methods across classification datasets for different values of k. Smaller ranks indicate better overall performance. Numbers in parentheses denote the corresponding rank positions among the compared methods.

Method	k=3	k=5	k=7	k=9
Proposed Model	4.250 (2)	**2.875 (1)**	**3.500 (1)**	**5.125 (1)**
Overlap KNN	15.000 (15)	14.750 (15)	14.750 (15)	14.750 (15)
Eskin KNN	11.250 (13)	9.750 (12)	9.750 (12)	9.625 (13)
OF KNN	8.875 (11)	8.875 (11)	8.500 (9)	9.375 (12)
GwKNN	8.875 (10)	8.125 (10)	6.375 (6)	6.000 (5)
LMGL-FkNN	13.500 (14)	13.500 (14)	12.500 (14)	12.500 (14)
LIWkNN	9.375 (12)	10.000 (13)	8.125 (8)	6.500 (6)
WPRkNN	7.000 (6)	8.000 (7)	10.000 (13)	8.500 (10)
EDWkNN	7.750 (7)	8.125 (9)	8.000 (7)	7.125 (7)
LMSL-FkNN	4.875 (5)	5.125 (4)	5.125 (4)	5.250 (2)
CMDW-FkNN	8.500 (9)	8.125 (8)	8.750 (10)	8.875 (11)
DWkNN	4.500 (3)	4.750 (3)	5.000 (2)	5.375 (3)
ExNRule-kNN	4.500 (4)	4.500 (2)	5.125 (3)	7.125 (8)
ORP-ExNRule-kNN	**3.375 (1)**	5.750 (5)	8.875 (11)	8.500 (9)

In addition to predictive accuracy, computational efficiency is also examined. Tables S6 and S7 report the average training and prediction times, respectively. As expected, the proposed method incurs higher training cost due to the unified optimization procedure for learning feature and neighbor weights. However, once training is completed, the prediction stage only involves weighted distance computations, resulting in competitive and stable test-time efficiency across all classification datasets. This behavior supports the suitability of the proposed method for practical scenarios where models are trained offline but predictions are required frequently.

Table S6: Average training time (mean ± sd, in seconds) over 10 repeated splits on selected classification datasets.

Method	House Votes	Ionosphere	Liver Disorder	Pima Indians	Steel Industry
Proposed Model	11.563 ± 1.634	13.657 ± 0.622	26.835 ± 0.595	56.553 ± 4.291	127.132 ± 3.891
Traditional KNN	0.012 ± 0.003	0.010 ± 0.003	0.010 ± 0.001	0.013 ± 0.004	0.017 ± 0.002
Overlap KNN	0.358 ± 0.038	0.428 ± 0.006	0.409 ± 0.010	0.563 ± 0.012	1.564 ± 0.034
Eskin KNN	0.386 ± 0.042	0.556 ± 0.015	0.510 ± 0.017	0.699 ± 0.020	1.837 ± 0.045
OF KNN	2.115 ± 0.241	4.389 ± 0.055	3.657 ± 0.041	5.077 ± 0.180	10.732 ± 0.271
GwKNN	14.916 ± 3.102	14.536 ± 4.288	8.688 ± 0.083	13.319 ± 0.468	42.820 ± 13.821
LMGL-FkNN	0.069 ± 0.019	0.095 ± 0.032	0.145 ± 0.031	0.262 ± 0.121	0.284 ± 0.067
LIWkNN	0.044 ± 0.014	0.042 ± 0.018	0.145 ± 0.085	0.289 ± 0.233	0.107 ± 0.034
WPRkNN	0.091 ± 0.063	0.080 ± 0.039	0.238 ± 0.119	0.583 ± 0.461	0.291 ± 0.070
EDWkNN	0.035 ± 0.029	0.020 ± 0.006	0.073 ± 0.025	0.174 ± 0.163	0.078 ± 0.031
LMSL-FkNN	0.042 ± 0.016	0.137 ± 0.067	0.071 ± 0.031	0.077 ± 0.020	0.295 ± 0.061
CMDW-FkNN	0.101 ± 0.052	0.068 ± 0.009	0.217 ± 0.065	0.409 ± 0.324	0.252 ± 0.048
DWkNN	0.018 ± 0.001	0.067 ± 0.028	0.019 ± 0.003	0.031 ± 0.007	0.093 ± 0.041
ExNRule-kNN	201.422 ± 10.562	126.790 ± 0.465	354.505 ± 3.609	238.133 ± 6.130	470.652 ± 11.871
ORP-ExNRule-kNN	341.318 ± 1.253	218.694 ± 0.626	607.501 ± 6.389	413.531 ± 11.203	808.392 ± 20.988

Table S7: Average prediction time (mean ± sd, in seconds) over 10 repeated splits on selected classification datasets.

Method	House Votes	Ionosphere	Liver Disorder	Pima Indians	Steel Industry
Proposed Model	0.016 ± 0.009	0.021 ± 0.013	0.035 ± 0.027	0.022 ± 0.027	0.037 ± 0.032
Traditional KNN	0.001 ± 0.001	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
Overlap KNN	0.070 ± 0.008	0.086 ± 0.003	0.080 ± 0.002	0.109 ± 0.003	0.323 ± 0.039
Eskin KNN	0.076 ± 0.008	0.108 ± 0.005	0.097 ± 0.003	0.136 ± 0.004	0.364 ± 0.030
OF KNN	0.412 ± 0.050	0.874 ± 0.032	0.717 ± 0.022	0.976 ± 0.026	2.089 ± 0.042
GwKNN	2.562 ± 0.925	3.191 ± 1.001	7.055 ± 2.599	3.694 ± 0.729	8.301 ± 1.988
LMGL-FkNN	0.011 ± 0.004	0.021 ± 0.010	0.021 ± 0.007	0.035 ± 0.017	0.042 ± 0.019
LIWkNN	0.005 ± 0.002	0.004 ± 0.002	0.018 ± 0.013	0.049 ± 0.042	0.016 ± 0.008
WPRkNN	0.022 ± 0.019	0.011 ± 0.004	0.044 ± 0.031	0.071 ± 0.057	0.045 ± 0.016
EDWkNN	0.007 ± 0.007	0.002 ± 0.001	0.019 ± 0.027	0.025 ± 0.019	0.009 ± 0.005
LMSL-FkNN	0.006 ± 0.003	0.016 ± 0.009	0.009 ± 0.008	0.011 ± 0.006	0.031 ± 0.019
CMDW-FkNN	0.019 ± 0.016	0.008 ± 0.002	0.033 ± 0.013	0.047 ± 0.025	0.044 ± 0.013
DWkNN	0.007 ± 0.005	0.013 ± 0.010	0.011 ± 0.005	0.011 ± 0.007	0.012 ± 0.010
ExNRule-kNN	15.548 ± 4.355	9.226 ± 0.083	51.692 ± 29.150	36.101 ± 23.642	73.698 ± 28.494
ORP-ExNRule-kNN	39.986 ± 22.698	18.047 ± 5.284	119.207 ± 45.478	58.564 ± 24.784	146.439 ± 62.141

Beyond predictive performance and computational efficiency, we further examine the robustness of the proposed framework under variations in key experimental settings. The weight-learning procedure is formulated as a constrained mathematical programming problem and solved by an optimization solver to optimality under fixed constraints. As such, the method does not rely on learning-rate-based iterative parameter updates or gradient-step tuning. Consequently, sensitivity primarily arises from the choice of the neighborhood size k and the internal validation ratio used during model selection.

We first examine the sensitivity with respect to the neighborhood size k. Specifically, k is systematically varied within the range $\{2, \ldots, 9\}$, and for each dataset the mean performance and corresponding standard deviation over 10 repeated splits are reported (see Table S8). To further quantify performance variability, the rightmost row of the table reports the range of the mean performance across different k values (max–min). Overall, the results indicate that performance changes smoothly as k varies, and the observed ranges remain moderate across datasets. This suggests that within a reasonable neighborhood size interval, the proposed method exhibits stable behavior and does not suffer from excessive sensitivity to the choice of k.

Table S8: Sensitivity analysis of the proposed method with respect to the neighborhood size k (mean ± std over splits). The last row reports the range of the mean performance across k (max–min).

k	House Votes	Insurance	Ionosphere	Liver Disorder	Pima Indians	Servo	Steel Industry
2	0.947 ± 0.017	0.404 ± 0.420	0.894 ± 0.031	0.656 ± 0.039	0.656 ± 0.036	0.815 ± 0.153	0.747 ± 0.024
3	0.954 ± 0.020	0.449 ± 0.399	0.894 ± 0.039	0.668 ± 0.046	0.705 ± 0.032	0.806 ± 0.131	0.789 ± 0.017
4	0.954 ± 0.020	0.465 ± 0.395	0.887 ± 0.039	0.671 ± 0.041	0.703 ± 0.038	0.797 ± 0.085	0.814 ± 0.021
5	0.954 ± 0.024	0.478 ± 0.387	0.861 ± 0.054	0.672 ± 0.041	0.719 ± 0.037	0.774 ± 0.080	0.823 ± 0.025
6	0.953 ± 0.022	0.485 ± 0.383	0.866 ± 0.048	0.668 ± 0.034	0.718 ± 0.039	0.752 ± 0.101	0.830 ± 0.028
7	0.951 ± 0.028	0.492 ± 0.379	0.858 ± 0.055	0.679 ± 0.048	0.723 ± 0.047	0.728 ± 0.104	0.833 ± 0.024
8	0.947 ± 0.028	0.495 ± 0.376	0.866 ± 0.052	0.676 ± 0.037	0.721 ± 0.043	0.715 ± 0.108	0.832 ± 0.023
9	0.949 ± 0.034	0.501 ± 0.371	0.856 ± 0.058	0.683 ± 0.033	0.709 ± 0.043	0.709 ± 0.103	0.834 ± 0.023
max–min	0.007	0.097	0.038	0.026	0.067	0.106	0.087

To further evaluate robustness, we analyze the effect of varying the internal validation ratio used during neighborhood selection. The ratio is set to 0.10, 0.15, 0.20, 0.25, and 0.30 within the outer training set, and the results are reported in Table S9. Across most datasets (House-Votes, Insurance, and Pima), the variation in mean performance remains limited (within approximately 0.01–0.02), and no systematic trend is observed as the validation ratio changes. For the Servo dataset, larger fluctuations are observed, which can be attributed to its small sample size and the inherent variability of regression tasks. These results suggest that no single universally optimal validation ratio is observed across datasets; instead, the proposed method remains relatively stable within a reasonable range of ratios. Therefore, in the case study, we use 0.20 as a practical default setting that provides a good balance between validation reliability and training-sample retention.

Table S9: Sensitivity analysis of the proposed method with respect to the internal validation ratio used for neighborhood selection. Results are reported as mean ± standard deviation over repeated splits.

Dataset	Metric	Internal Validation Ratio				
		0.10	0.15	0.20	0.25	0.30
House Votes	Accuracy	0.826 ± 0.102	0.842 ± 0.075	0.834 ± 0.092	0.828 ± 0.087	0.832 ± 0.087
Pima	Accuracy	0.706 ± 0.038	0.724 ± 0.008	0.701 ± 0.048	0.708 ± 0.025	0.719 ± 0.012
Insurance	R^2	0.833 ± 0.036	0.837 ± 0.031	0.841 ± 0.026	0.841 ± 0.026	0.841 ± 0.026
Servo	R^2	0.886 ± 0.040	0.844 ± 0.025	0.768 ± 0.069	0.886 ± 0.040	0.798 ± 0.116

References

S. Boriah, V. Chandola, V. Kumar, Similarity measures for categorical data: A comparative evaluation, in: Proceedings of the 2008 SIAM international conference on data mining, 2008, pp. 243–254. doi:10.1137/1.9781611972788.22.

H. A. Abu Alfeilat, A. B. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, H. S. Eyal Salman, V. S. Prasath, Effects of distance measure choice on k-nearest neighbor classifier performance: A review, Big Data 7 (2019) 221–248. doi:10.1089/big.2018.0175.

D. Tasche, Fisher consistency for prior probability shift, Journal of Machine Learning Research 18 (2017) 1–32. doi:https://doi.org/10.48550/arXiv.1701.05512.

R. Anand, D. Aggarwal, V. Kumar, A comparative analysis of optimization solvers, Journal of Statistics and Management Systems 20 (2017) 623–635. doi:10.1080/09720510.2017.1395182.

G. Gogoshin, S. Branciamore, A. Rodin, Synthetic data generation with probabilistic bayesian networks, Math Biosci Eng 18 (2021) 8603–8621. doi:10.3934/mbe.2021426.

N. Rastin, M. Z. Jahromi, M. Taheri, A generalized weighted distance k-nearest neighbor for multi-label problems, Pattern Recognition 114 (2021) 107526. doi:10.1016/j.patcog.2020.107526.