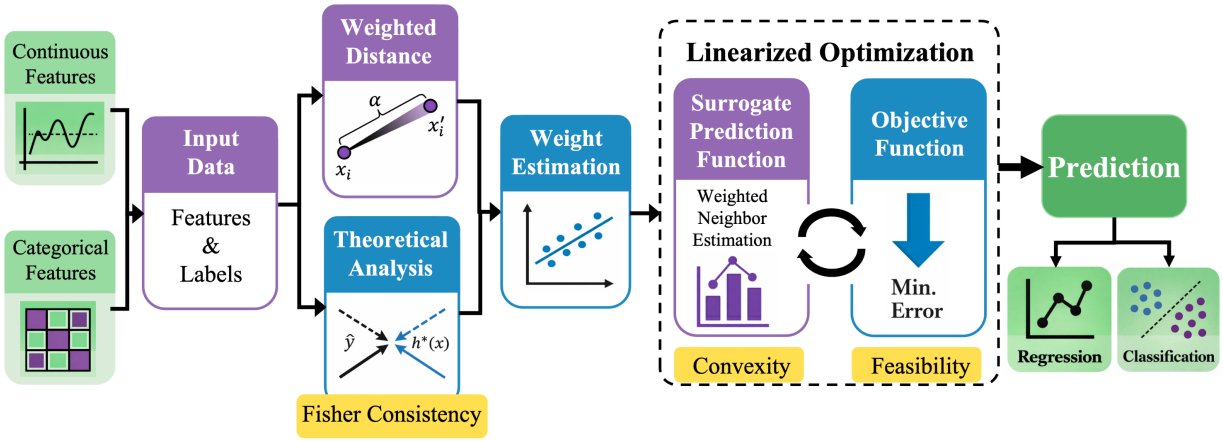
Graphical Abstract

Data-Driven KNN Modeling via Optimized Weighted Distance Learning

Liangliang Zhuang, Gauthier Barbosa, Zhisheng Ye



Data-Driven KNN Modeling via Optimized Weighted Distance Learning

Liangliang Zhuang[a], Gauthier Barbosa[b,*], Zhisheng Ye[b]

[a]*School of Economics and Management, Department of Management Science and Engineering, Nanjing University of Science and Technology, Nanjing, 210094, China*
[b]*Department of Industrial Systems Engineering & Management, National University of Singapore, 117576, Singapore*

Abstract

Many tabular prediction tasks involve predictors with different representations, for which standard KNN may be limited by its fixed distance metric and uniform neighbor weighting. This can lead to poorly represented neighborhood structures and suboptimal predictive performance. To overcome these limitations, we propose a data-driven KNN framework that incorporates an optimized, weighted distance function. Specifically, we extend classical KNN by introducing feature- and neighbor-level weights, enabling the model to better account for mixed feature types and local sample relevance. We then develop a tractable linear approximation to the learning objective, reformulating it as a large-scale linear program that allows efficient estimation of weights. Theoretically, we prove that the proposed method preserves the locality principle fundamental to KNN and satisfies Fisher consistency under standard asymptotic conditions. Experiments on simulated data and several real-world datasets demonstrate that our method achieves competitive and stable performance relative to existing KNN variants. In particular, the simulation results illustrate its empirical robustness under different data-generating mechanisms, feature-importance patterns, and response-noise levels.

Keywords: Data-driven method; K-nearest neighbors; heterogeneous predictors; linear programming; weighted distance measures

*Corresponding author

Email addresses: zhuangll@njust.edu.cn (Liangliang Zhuang), e1439493@u.nus.edu (Gauthier Barbosa), yez@nus.edu.sg (Zhisheng Ye)

1. Introduction

1.1. Background

In the era of big data, the volume and complexity of data are increasing, driving industries such as manufacturing, finance, and healthcare to leverage machine learning methods for data modeling and analysis [1–3]. Among these methods, the k-nearest neighbors (KNN) algorithm stands out as a simple yet effective non-parametric approach for classification and regression [4]. Due to its interpretability and ease of implementation, KNN has been widely applied in fields such as image recognition [5], recommendation systems [6], bioinformatics [7], and text classication [8]. The core idea of KNN is to predict the label or value of a target sample by identifying its k nearest neighbors based on distance metrics. This method performs particularly well on datasets with clear local structures [9]. However, real-world data often include a mix of continuous, categorical, and other data types. For example, in industrial quality control, variables like temperature, pressure, and cycle time are continuous, while equipment model and defect type are categorical. In marketing, customer income is numerical, while purchase preferences and satisfaction ratings are categorical or discrete-valued.

A compelling example comes from medical diagnostics—for instance, in predicting heart disease risk—where datasets often contain predictors with different representations, including numerical measurements (e.g., blood pressure) and categorical variables (e.g., gender, smoking status, and symptom categories). This mix of data types makes it difficult for traditional KNN methods to perform effectively. As shown in Figure 1, even though patient A and patient B have nearly identical blood pressure, they differ in symptom severity—a feature more relevant to disease risk. Nonetheless, traditional KNN tends to select neighbors primarily based on proximity in the numerical dimension, while failing to adequately account for the information carried by discrete or categorical predictors, which may lead to poor predictions. This example highlights a broader challenge in tabular data with predictors of different representations: how to design a distance metric that adaptively balances feature contributions and reflects their relative relevance to the prediction task. In this work, we address this problem by learning feature-specific weights in a unified, data-driven optimization framework.

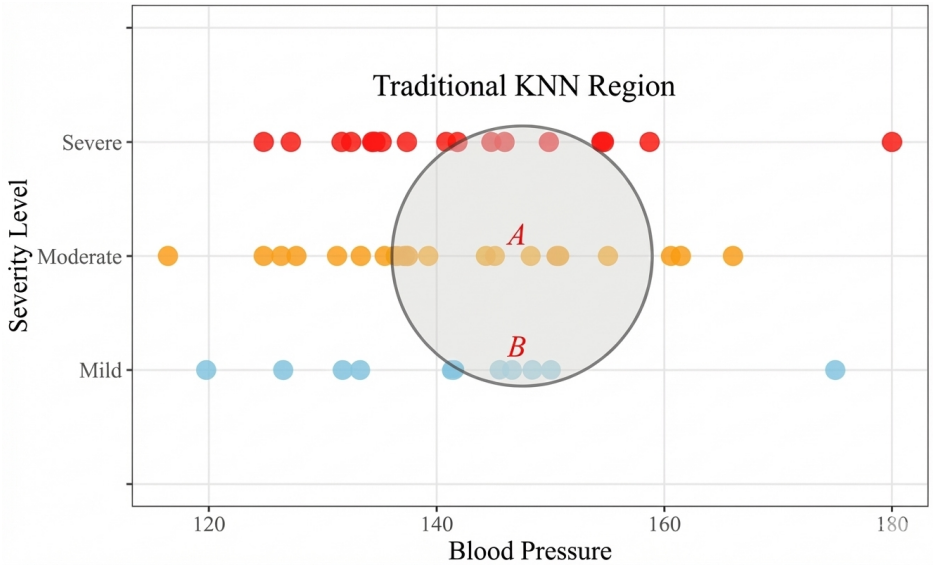


Figure 1: Illustration of misclassification risk in traditional KNN when feature relevance is not adequately reflected in the distance metric, where selected neighbors may be close in one feature but differ in risk-relevant attributes.

1.2. Related Work

To improve the predictive performance of KNN in increasingly complex data settings, numerous KNN variants have been proposed in recent years. These approaches can be broadly categorized into four main directions. One line of research aims to improve similarity measures for KNN by capturing richer structural information than traditional metrics such as Hamming or Jaccard distances. Le and Ho [10] introduced association-based dissimilarity metrics that account for feature co-occurrence patterns, while Ienco et al. [11] proposed context-sensitive distance learning that adapts to local data structure. Boriah et al. [12] showed that incorporating similarity matrices can enhance KNN's discriminative power. A second research direction focuses on feature weighting, where the aim is to assign higher importance to informative attributes during distance computation. This is particularly beneficial for categorical data, where treating all features equally can obscure relevant patterns. Chen and Guo [13] proposed a ranking-based scheme that assigns larger weights to more discriminative features. Nababan et al. [14] used gain-ratio scores to quantify feature relevance, while Karabulut et al. [15] explored weighting strategies specifically tailored for categorical feature spaces. These approaches consistently demonstrate improved predictive performance by embedding feature relevance into the similarity metric.

A third line of research focuses on adaptive and weighted KNN methods, which modify

the construction of neighborhood sets or adjust neighbor contributions during aggregation. Instead of relying on a fixed k, some approaches dynamically determine the neighborhood size based on local data characteristics. For instance, Han et al. [16] proposed a weighted adaptive KNN algorithm, while Ougiaroglou et al. [17] and García-Pedrajas et al. [18] introduced dynamic k-selection strategies that respond to local distributional structures. Other studies refine the aggregation mechanism by assigning different importance to neighbors. Rastin et al. [19] developed a relevance-based weighting scheme based on proximity and label similarity. Amer et al. [20] introduced a local mean-based fuzzy KNN framework that computes class memberships using class-wise local mean representations. Abdalla and Amer [21] proposed a locally informed weighted KNN that assigns and updates instance-level weights during prediction. Amer et al. [22] incorporated neighborhood-consistency scores, such as proximity ratios, as weighting factors, while Amer et al. [23] applied distance- and rank-based weighting schemes within the standard KNN framework. Finally, Abdalla et al. [24] modified the decision mechanism by adopting class-wise neighbor selection and mean-based fuzzy membership aggregation. A fourth research direction focuses on ensemble methods. These approaches enhance predictive performance by combining multiple KNN variants within a unified aggregation framework. For instance, Ali et al. [25] proposed an ExNRule-based ensemble using extended neighborhood rules, while Ali et al. [26] further incorporated random projection and out-of-bag model selection to improve robustness and diversity. Similarly, Abdalla et al. [27] integrated three distinct KNN variants into a single ensemble model, aggregating their class-wise outputs through additive weighting to obtain the final prediction.

Table 1 summarizes recent KNN-based algorithms and their main strengths and limitations. Although these methods have improved KNN from different perspectives, most still face challenges in mixed-type settings: (i) similarity-based methods are typically designed for single-type features; (ii) feature-weighting methods improve discrimination but often lack flexibility across heterogeneous feature types; (iii) adaptive neighbor-weighting methods enhance local adaptability, but they usually operate on a predefined distance structure and therefore do not explicitly learn a task-adaptive metric for heterogeneous data; (iv) ensemble-based methods improve robustness and predictive performance, but usually introduce substantially higher computational complexity.

Table 2 summarizes representative KNN variants across key methodological aspects. Beyond the four main research directions discussed above, Ali et al. [28] proposed a double-

Table 1: Representative KNN variants across four research directions with their main strengths, limitations, and applications.

Category	Method	References	Strengths	Weaknesses	Application
(i)	Similarity measures	[10–12]	Improved distance metrics; Context-aware similarity	Struggles with mixed data; Data heterogeneity	Primarily for homogeneous datasets
(ii)	Feature weighting	[13–15]	Enhances classification; Weights important features	Mixed-type difficulty; Computational overhead	Datasets with clearly important features
(iii)	Neighbor selection and weighting	[16–24]	Dynamic k; Weighted neighbor selection	Higher computational cost; Complex implementation	Varying data density scenarios
(iv)	Ensemble methods	[25–27]	Improves robustness; Enhances diversity	Higher computational cost; Reduced interpretability	Complex classification tasks

weighted KNN framework that combines feature-level and neighbor-level weighting, which can be viewed as a hybrid of these two strategies. In general, many recent KNN extensions improve predictive performance by introducing distance-based or rule-driven reweighting mechanisms for neighbor contributions. For example, in Amer et al. [20] and Abdalla and Amer [21], neighbor weights are computed using predefined scoring schemes such as proximity ratios, distance-based interpolation, or rank-based rules within the standard KNN framework. These approaches enhance discrimination at the neighbor aggregation stage while operating under a fixed distance structure. Although such strategies can effectively refine local decision boundaries, the underlying feature distance is typically specified a priori after encoding and is not learned through a unified optimization objective. As a result, the relative importance of heterogeneous feature types (e.g., continuous, and categorical) is not directly optimized with respect to prediction error. In mixed-type tabular settings, where feature scales and informational contributions differ substantially, explicitly learning a task-adaptive distance metric may lead to a more representative neighborhood structure and improved generalization.

To address these limitations, this study introduces a data-driven KNN framework based on an optimized weighted distance metric. Unlike rule-based reweighting approaches, the proposed method formulates weight learning as a unified optimization problem, directly minimizing a prediction-error-driven objective under explicit constraints. The framework is further motivated by correlation constructions in Gaussian process models, which provide

a principled way to quantify similarity among predictors with different representations [29–32]. By adaptively emphasizing informative features and down-weighting irrelevant ones, the proposed approach constructs a more appropriate distance structure and thereby improves predictive accuracy and stability [33, 34].

Table 2: Summary of representative KNN research directions and the proposed method across several methodological aspects.

Method	Feature weights	Neighbor weights	Unified optimization	References
Similarity measures	✗	✗	✗	[10–12]
Feature weighting	✓	✗	✗	[13–15]
Neighbor selection and weighting	✗	✓	✗	[16–24]
Ensemble methods	✗	✓	✗	[25–27]
Double-weighted KNN	✓	✓	✗	[28]
LP-based weighted KNN (Proposed)	✓	✓	✓	This paper

Building on this idea, we introduce a general weighted distance formulation into the KNN framework. Unlike prior approaches that focus solely on feature weighting, our method incorporates both feature-level and neighbor-level weights. This combined weighting scheme addresses imbalances in feature contributions and refines distance measurement by jointly considering feature relevance and neighbor influence. To efficiently estimate the feature weights, a linearized approximation of the prediction error is employed, enabling the optimization problem to be cast as a large-scale linear programming (LP) problem. A key theoretical contribution lies in establishing that the proposed model preserves the core KNN principle of "similarity by proximity". Building on this foundation, we further prove that the model satisfies Fisher consistency under asymptotic conditions, ensuring convergence to the Bayes optimal rule as the sample size increases. The main contributions of this paper are as follows:

- A data-driven KNN framework is proposed that integrates both feature-level and neighbor-level weights to construct a task-adaptive weighted distance.
- A linearized approximation method is introduced to construct a tractable LP formulation for estimating feature weights adaptively, thereby avoiding exhaustive search.
- Fisher consistency is proved, ensuring asymptotic convergence to the Bayes-optimal predictor.

The rest of the paper is organized as follows. Section 2 introduces the model framework. Sections 3 and 4 give the data-driven weighted KNN model and its implementation. Theoretical analysis is provided in Section 5. Sections 6 and 7 present simulation studies and real-world applications, respectively. Conclusions and discussions are presented in Section 8, and additional results and technical proofs are provided in the supplementary material.

2. Framework Overview and Weighted Distance

Figure 2 illustrates the overall structure of the proposed data-driven weighted KNN framework. The method comprises four core components: feature-weighted distance computation, linearized weight estimation, prediction, and theoretical consistency analysis. Specifically, the model takes a tabular feature matrix as input and improves similarity measurement by learning optimized feature weights, thereby enhancing predictive accuracy.

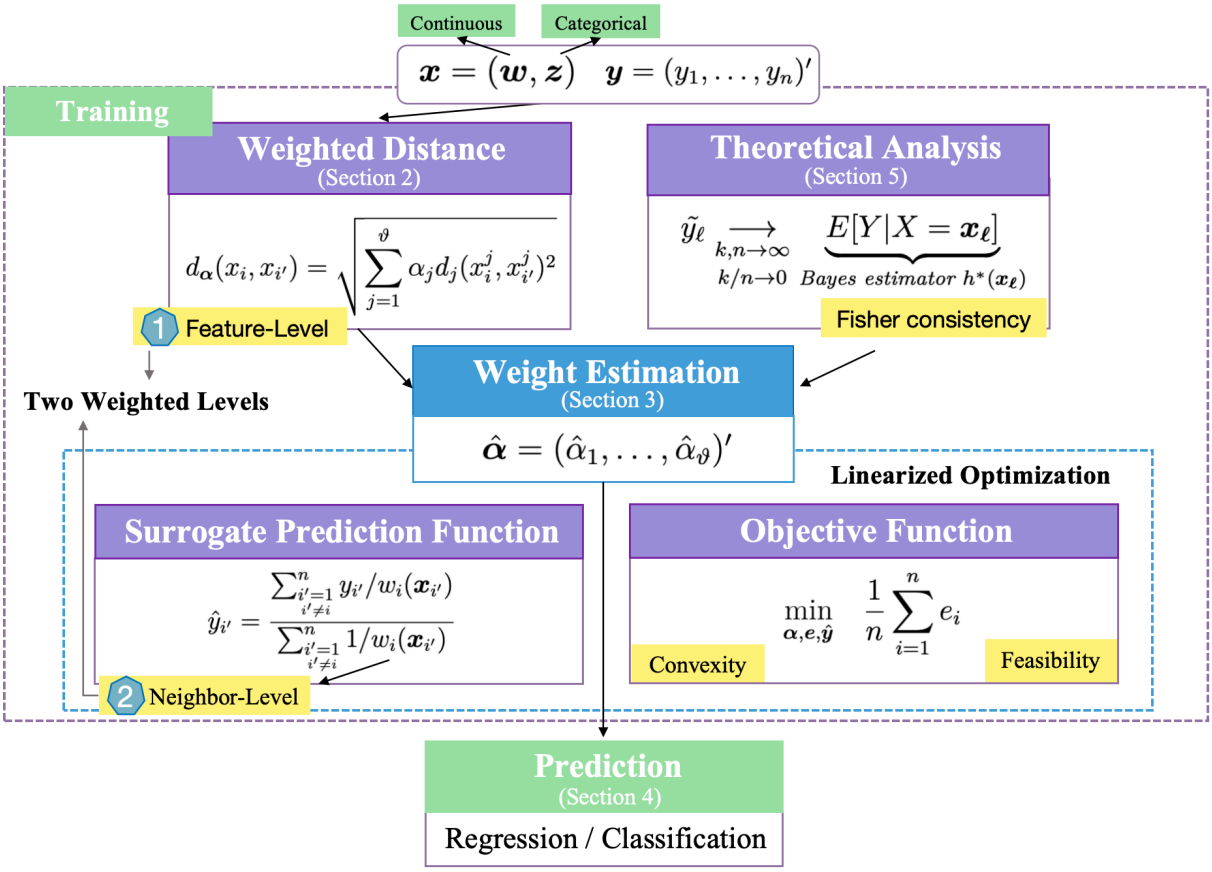


Figure 2: Overview of the proposed data-driven weighted KNN framework: feature-weighted distance learning, LP-based weight estimation via linearized optimization, prediction, and Fisher-consistency analysis.

We begin by formalizing the input representation. Consider a subject with a response y and a feature vector $\boldsymbol{x} = (\boldsymbol{w}, \boldsymbol{z})$ comprising both continuous factors $\boldsymbol{w} = (w^1, \ldots, w^p)$ and categorical factors $\boldsymbol{z} = (z^1, \ldots, z^q)$. Suppose we observe n independent observations of $(\boldsymbol{x}, y)$. The full dataset can be written as an input matrix $= (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)' \in \mathbb{R}^{n \times \vartheta}$ and an output vector $\boldsymbol{y} = (y_1, \ldots, y_n)'$, where we denote $\vartheta = p + q$ as the total number of features.

To improve the predictive performance of KNN, we propose a distance function that incorporates feature-specific weights. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_\vartheta)'$ denote the vector of positive weights associated with the features. These weights are used to compute a weighted distance between data points, as outlined in Definition 1.

Definition 1 (Weighted Distance)**.** *Consider two data points $\boldsymbol{x_i}$ and $\boldsymbol{x_{i'}}$ from the dataset. The distance between these two vectors is defined as:*

$$d_{\boldsymbol{\alpha}}(x_i, x_{i'}) = \sqrt{\sum_{j=1}^{\vartheta} \alpha_j d_j(x_i^j, x_{i'}^j)^2}, \tag{1}$$

where $d_r(x_i^j, x_{i'}^j)$ represents the distance between the j-th features of elements i and i'; this distance can be Euclidean, Hamming, Manhattan, or any other suitable metric [35]. The weights $\alpha_1, \ldots, \alpha_\vartheta$ determine the relative contribution of each feature in the overall distance.

To provide further intuition, consider again the medical diagnostic scenario introduced earlier, where the dataset includes blood pressure (numerical) together with categorical or discrete clinical descriptors such as gender and symptom severity levels. Suppose that symptom severity is more relevant to the prediction task than gender. In Equation (1), assigning a larger weight α_j to symptom severity increases its contribution to the overall distance. Consequently, two patients with similar blood pressure but substantially different symptom severity will have a larger weighted distance and are therefore less likely to be selected as neighbors. In contrast, less informative features with smaller weights contribute less to the distance computation. Through this adaptive weighting mechanism, the distance metric reflects task-specific feature relevance rather than treating all features equally. The weighted distance defined above plays a central role in the proposed framework, as it allows more informative features to exert greater influence in both neighbor selection and prediction. When the feature weights are strictly positive, this distance function satisfies the four fundamental properties of a metric—non-negativity, identity of indiscernibles, symmetry, and the triangle inequality. A formal proof is provided in Supplementary Section S1.

Building on the weighted distance formulation, the proposed framework incorporates an additional neighbor-level weighting scheme to enhance prediction accuracy. This is implemented via a surrogate prediction function that accounts for neighbor influence. To enable efficient training, we linearize the surrogate function and objective, leading to a tractable LP for learning feature weights (Section 3). Once trained, predictions are made using a standard KNN structure augmented with learned weights (Section 4).

3. Weight Estimation via Linearized Optimization

We propose a tractable approach to estimate the feature-level weights $\boldsymbol{\alpha}$ used in the distance function. To avoid the intractability of direct nonlinear optimization, we introduce a differentiable surrogate prediction function and linearize it via first-order Taylor expansion (see Section 3.1). Based on this linearization, we formulate a convex objective for estimating the feature weights $\hat{\boldsymbol{\alpha}}$, enabling efficient training and adaptive feature reweighting of the model (see Section 3.2).

3.1. Surrogate Prediction Function

A common KNN predictor uses hard voting over the k nearest neighbors through an indicator rule. However, this discrete mechanism does not readily accommodate feature-specific weighting or enable smooth optimization. Following the soft similarity idea of Tan [36], we replace hard voting with a distance-based weighting scheme: each training sample $\boldsymbol{x}_{i'}$ contributes to the prediction at $\boldsymbol{x}_i$ with weight $w_i(\boldsymbol{x}_{i'}) = (1 + d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})^2)^{-1/k}$. This function ensures that closer neighbors receive more influence, while the parameter k controls the rate of decay: smaller values emphasize locality, whereas larger values allow broader neighborhoods to contribute. Figure 3 visualizes this effect, and further interpretation is provided in supplementary Section S2.

Using this distance-based weighting, the surrogate predicted output for each data point $\boldsymbol{x}_i$ is defined as:

$$\hat{y}_{i'} = \frac{\sum_{\substack{i'=1 \\ i' \neq i}}^{n} y_{i'} w_i(\boldsymbol{x}_{i'})}{\sum_{\substack{i'=1 \\ i' \neq i}}^{n} w_i(\boldsymbol{x}_{i'})}. \tag{2}$$

Note that this formulation is used exclusively during the training phase to estimate $\boldsymbol{\alpha}$. It serves as a differentiable and analytically tractable surrogate for KNN-based prediction, enabling us to define a continuous loss function over the entire training set.

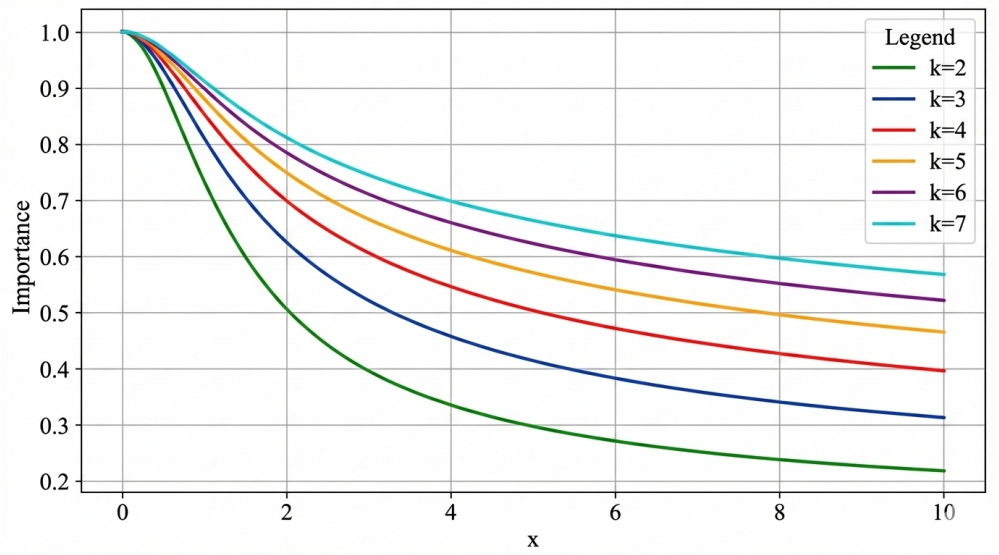


Figure 3: Decay of the importance function $(1+x^2)^{-1/k}$ with distance x for different k; smaller k emphasizes locality via faster decay.

However, despite its smoothness, the surrogate still involves nonlinear distance computations that complicate direct optimization. To enable a tractable optimization procedure, we linearize the surrogate prediction function defined in Equation (2). Specifically, we apply a first-order Taylor expansion to the kernel function $(1+x)^{-a}$ as $x \to 0$, which yields:

$$\frac{1}{(1+x)^a} = 1 - ax + o(x).$$

This approximation is valid when the weighted distances between data points are sufficiently small. Fortunately, this condition can be enforced without affecting neighbor selection: scaling the weight vector $\boldsymbol{\alpha}$ by a positive constant λ does not alter the ordering of distances, i.e., $d_{\boldsymbol{\alpha}}$ and $d_{\lambda\boldsymbol{\alpha}}$ induce the same k-nearest neighbors.

This scaling invariance allows us to normalize the weights so that all pairwise distances satisfy $d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \ll 1$, ensuring the validity of the linear approximation. In practice, this constraint also simplifies implementation—since only the relative values of $\boldsymbol{\alpha}$ matter—and enables easy verification by checking that $\max_{i,i'} d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})^2 < 1$.

This allows each neighbor weight to be approximated via a first-order Taylor expansion, retaining only the leading quadratic term in distance [37]. For any pair of points $(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$, we have:

$$\frac{1}{(1+d_{\boldsymbol{\alpha}}(\boldsymbol{x_i}, \boldsymbol{x_{i'}})^2)^{1/k}} \approx 1 - \frac{1}{k} d_{\boldsymbol{\alpha}}(\boldsymbol{x_i}, \boldsymbol{x_{i'}})^2 + o(d_{\boldsymbol{\alpha}}(\boldsymbol{x_i}, \boldsymbol{x_{i'}})^2).$$

Applying this approximation, the numerator becomes $\sum_{\substack{i'=1 \\ i' \neq i}}^{n} y_{i'} \left[1 - d^2_{\boldsymbol{\alpha}}(\boldsymbol{x_i}, \boldsymbol{x_{i'}})/k\right]$, and

the denominator becomes $n - \sum_{\substack{i'=1 \\ i' \neq i}}^{n} d_{\boldsymbol{\alpha}}^2(\boldsymbol{x_i}, \boldsymbol{x_{i'}})/k$. Substituting into the prediction rule and simplifying gives the linearized approximation for the predicted output, denoted as $\hat{y}_i^{\text{app}}$, which represents the first-order approximation of $\hat{y}_i$:

$$\hat{y}_i \approx \hat{y}_i^{app} = \frac{1}{n} \sum_{\substack{i'=1 \\ i' \neq i}}^{n} y_{i'} \left[1 - \frac{1}{k} d_{\boldsymbol{\alpha}}(\boldsymbol{x_i}, \boldsymbol{x_{i'}})^2 + \frac{1}{nk} \sum_{i''=1}^{n} d_{\boldsymbol{\alpha}}(\boldsymbol{x_i}, \boldsymbol{x_{i''}})^2 \right]. \tag{3}$$

This expression is obtained by omitting higher-order terms, assuming that distances raised to the fourth power and beyond are negligible. Intuitively, this approximation linearizes the original nonlinear weighting function via a first-order Taylor expansion in a local neighborhood where pairwise distances are small, yielding a prediction expression that is approximately linear in $\boldsymbol{\alpha}$. Since scaling $\boldsymbol{\alpha}$ does not change the distance ordering, the approximation does not affect neighbor selection, and the neglected higher-order terms introduce only minor errors. As a result, $\hat{y}_i^{app}$ becomes an explicit and approximately linear function of $\boldsymbol{\alpha}$, which is used in the next subsection to construct the LP-based training objective.

Remark. The use of linear programming in the proposed framework is not merely a choice of optimization solver, but a direct consequence of the model design. By linearizing the prediction error with respect to the feature weights, the original nonlinear weighting problem is transformed into a structured linear program with well-defined constraints and a globally optimal solution. In contrast to approaches that rely on grid search or local update rules for weight selection, the LP formulation provides a stable and interpretable mechanism for learning the weights and can be solved efficiently using standard LP solvers. This formulation also naturally supports the joint learning of feature-level and neighbor-level weights within a unified framework, which is difficult to achieve in existing weighted or metric-learning KNN methods.

3.2. Objective Function

Having obtained the linear approximation of the predicted output (Equation (3)), we now formulate an optimization problem to learn the feature weights $\boldsymbol{\alpha}$. The goal is to minimize the discrepancy between the linearized prediction and the true response over the training data. For regression tasks, we adopt the absolute error as the loss criterion, leading to the following objective:

$$\mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{n} \sum_{i=1}^{n} \left| \hat{y}_i - y_i \right|,$$

where $\hat{y}_i$ denotes the linearized prediction for observation i, as defined in Equation (3). For notational simplicity, we drop the superscript "app".

To enable efficient optimization, we express the absolute value using auxiliary slack variables $e_i \geq 0$, satisfying: $e_i \geq \hat{y}_i - y_i$, and $e_i \geq y_i - \hat{y}_i$. This reformulation converts the non-smooth objective into a linear program, allowing the entire estimation procedure to be solved using standard LP solvers. It also avoids potential instability associated with direct optimization over non-differentiable loss functions.

Based on the linearized prediction function and absolute error loss, the feature weights $\boldsymbol{\alpha}$ can be estimated by solving the following linear program:

$$
\begin{aligned}
\min_{\boldsymbol{\alpha}, \boldsymbol{e}, \hat{\boldsymbol{y}}} \quad & \frac{1}{n} \sum_{i=1}^{n} e_i \\
\text{s.t.} \quad & e_i \geq \hat{y}_i - y_i, \\
& e_i \geq y_i - \hat{y}_i, \\
& \hat{y}_i = \frac{1}{n} \sum_{\substack{i'=1 \\ i' \neq i}}^{n} y_{i'} \left[1 - \frac{1}{k} d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})^2 + \frac{1}{nk} \sum_{i''=1}^{n} d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i''})^2 \right], \\
& d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})^2 \leq \frac{1}{2}, \\
& \alpha_j > 0.
\end{aligned}
\tag{4}
$$

To guarantee that this LP is well-defined for any dataset, we next establish Theorem 1, which shows that the feasible set is nonempty under a mild scaling of the feature weights and that the resulting problem is convex. The proof is provided in supplementary Section S3.1.

Theorem 1 (Feasibility and Convexity)**.** *Under the distance definition in Definition 1, the LP described above is convex and is feasible. In particular, feasibility can be ensured by selecting the weights α_j such that $\alpha_j^{-1} = 2\vartheta \cdot \max_{i,i'} d_j(x_i^j, x_{i'}^j)^2$.*

Convexity follows directly since the objective function and all constraints are linear in the decision variables $(\boldsymbol{\alpha}, \hat{\boldsymbol{y}}, \boldsymbol{e})$. As a result, the optimization falls within the class of LPs, for which global optima can be found efficiently. In terms of scalability, the total number of decision variables is $2n + \vartheta$, which grows linearly with the sample size n and the number of features ϑ. This ensures that the problem remains computationally tractable for a wide range of practical datasets.

4. Prediction Procedure and Algorithmic Implementation

Having formulated a complete linear optimization problem for learning feature-specific distances, we now present the implementation of the proposed method. This section includes the procedure for making predictions on new data (see Section 4.1), algorithmic workflow, and computational complexity analysis (see Section 4.2).

4.1. Prediction Using the Learned Distance Metric

Once the feature weights $\hat{\boldsymbol{\alpha}}$ are learned—by solving the LP defined in Theorem 1 using a standard solver (see supplementary Section S4)—predictions for new data points are made using a k-nearest neighbors procedure. Let $\text{neigh}_k(\boldsymbol{x}_\ell)$ denote the indices of the k training points closest to a new input $\boldsymbol{x}_\ell$, as measured by the learned distance metric $d_{\hat{\boldsymbol{\alpha}}}$. The choice of k follows standard practice and is determined via cross-validation on the training set. Each neighbor $\boldsymbol{x}_i \in \text{neigh}_k(\boldsymbol{x}_\ell)$ contributes a weight given by: $w_i(\boldsymbol{x}_\ell) = (1 + d_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}_i, \boldsymbol{x}_\ell)^2)^{-1/k}$. The final predicted output $\tilde{y}_\ell$ is computed as:

Definition 2 (Prediction Rule)**.** *Given a new input $\boldsymbol{x}_\ell$, the predicted output $\tilde{y}_\ell$ is:*

$$\tilde{y}_\ell = \begin{cases} \mathbb{1}\left\{ \dfrac{\sum_{i \in neigh_k(\boldsymbol{x}_\ell)} y_i w_i(\boldsymbol{x}_\ell)}{\sum_{i \in neigh_k(\boldsymbol{x}_\ell)} w_i(\boldsymbol{x}_\ell)} \geq \frac{1}{2} \right\}, & \textit{(Binary classification)} \\ \arg\max\limits_{c \in \mathcal{Y}} \dfrac{\sum_{i \in neigh_k(\boldsymbol{x}_\ell)} \mathbb{1}\{y_i = c\} w_i(\boldsymbol{x}_\ell)}{\sum_{i \in neigh_k(\boldsymbol{x}_\ell)} w_i(\boldsymbol{x}_\ell)}, & \textit{(Multi-class classification)} \\ \dfrac{\sum_{i \in neigh_k(\boldsymbol{x}_\ell)} y_i w_i(\boldsymbol{x}_\ell)}{\sum_{i \in neigh_k(\boldsymbol{x}_\ell)} w_i(\boldsymbol{x}_\ell)}. & \textit{(Regression)} \end{cases}$$

4.2. Algorithm Description and Complexity Analysis

The complete procedure is presented in Algorithm 1, which integrates the three major components of our framework: weight estimation via LP, selection of the optimal number of neighbors, and prediction for new data points based on the learned metric. In particular, the feature weights $\boldsymbol{\alpha}$ are learned during the training stage. For each candidate neighborhood size k, the linear programming model defined in Theorem 1 is constructed on the internal training subset and solved accordingly. The weight-learning process is considered complete once the solver reaches an optimal solution under the specified constraints. The learned weights are then fixed and used for prediction on the test data. This unified algorithm therefore provides a complete and coherent workflow for implementing the proposed distance-weighted KNN framework, from training to inference.

The complete procedure is summarized in Algorithm 1. The algorithm consists of three main stages: LP-based feature-weight learning, validation-based selection of the neighborhood size, and prediction for new data points using the learned metric. Specifically, the training data are first split into an internal training subset and an internal validation subset. Let $\mathcal{K} = \{2, \ldots, 9\}$ denote the candidate set of neighborhood sizes. For each $k \in \mathcal{K}$, the linear programming model in Theorem 1 is solved on the internal training subset to obtain the corresponding learned weight vector $\hat{\boldsymbol{\alpha}}$. The pair $(k^*, \hat{\boldsymbol{\alpha}}^*)$ is then selected as the validation-optimal neighborhood size and weight vector, namely the one achieving the smallest validation error on the internal validation subset, where the superscript $*$ denotes the validation-optimal choice. Once selected, the learned weights are fixed and used for prediction on new test points through the weighted KNN rule. This algorithm therefore provides a complete workflow from training to inference for the proposed distance-weighted KNN framework.

We analyze the computational complexity of Algorithm 1 by separating the training and prediction phases. In the training phase, for each candidate neighborhood size $k \in \mathcal{K}$, the method solves one LP to estimate the feature weights $\hat{\boldsymbol{\alpha}}$ and then evaluates the corresponding validation error. The LP involves $2n + \vartheta$ decision variables, together with $2n$ inequality constraints for the absolute-error bounds, n equality constraints from the linearized prediction function, n^2 inequality constraints for distance control, and ϑ positivity constraints on the feature weights. Using a standard LP solver, the worst-case cost of one solve is on the order of $\mathcal{O}(n^2\vartheta^2)$ [38]. Therefore, if $|\mathcal{K}|$ candidate values are considered, the total training cost is on the order of $\mathcal{O}(|\mathcal{K}|\, n^2\vartheta^2)$, with the LP step dominating the validation cost.

At the prediction stage, once $(k^*, \hat{\boldsymbol{\alpha}}^*)$ has been selected, the learned weights are fixed and no further optimization is required. For each query point, computing the weighted distances to all n training samples requires $\mathcal{O}(n\vartheta)$ operations, and the subsequent neighbor selection and weighted aggregation remain of the same order as standard KNN. Hence, the additional computational burden of the proposed framework is concentrated in the training phase, while the test-time complexity remains comparable to classical KNN.

5. Theoretical Analysis

We analyze the proposed method under asymptotic conditions to verify its theoretical validity. Specifically, we aim to show that the model preserves the locality principle of

Algorithm 1: Weighted-KNN Algorithm for Prediction

Require: Training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, candidate neighborhood set $\mathcal{K}$.

1: Split the training data once into internal training and validation subsets, with 20% of the data used for validation; stratified splitting is used for classification tasks, whereas a random split is used for regression tasks.
2: Define weighted distance function $d_{\boldsymbol{\alpha}}$ as in Equation (1);
3: **for** each k in $\mathcal{K}$ **do**
4: Solve the LP in Equation (4) to obtain $\hat{\boldsymbol{\alpha}}$ by minimizing the linearized prediction error subject to the constraints; feasibility is guaranteed by Theorem 1.
5: **end for**
6: Select the validation-optimal pair $(k^*, \hat{\alpha}^*)$ using the task-specific validation criterion over $k \in \mathcal{K}$ (e.g., accuracy for classification tasks and R^2 for regression tasks).
7: **for** each test point $\boldsymbol{x}_\ell$ **do**
8: Identify the k^* nearest training samples of $\boldsymbol{x}_\ell$ using $d_{\hat{\boldsymbol{\alpha}}^*}$;
9: Compute neighbor weights via the distance-decay rule $w_i(\boldsymbol{x}_\ell) = (1 + d_{\hat{\boldsymbol{\alpha}}^*}(\boldsymbol{x}_i, \boldsymbol{x}_\ell)^2)^{-1/k^*}$; see Figure 3. Predict $\tilde{y}_\ell$ using the learned neighbor weights: weighted majority vote for classification and weighted average for regression.
10: **end for**
11: **return** Final predictions $\{\tilde{y}_\ell\}$, learned weights $\hat{\boldsymbol{\alpha}}^*$, and selected k^*.

KNN (see Section 5.1) and achieves Fisher consistency as the dataset size increases (see Section 5.2).

5.1. Locality Preservation

A foundational principle of KNN-based methods is the locality assumption: points that are closer in feature space should exert greater influence on predictions. In this section, we show that the proposed weighted distance metric $d_{\boldsymbol{\alpha}}$ preserves this principle, thereby ensuring that the model remains consistent with the core intuition of KNN.

First, we note that $d_{\boldsymbol{\alpha}}$ is continuous and satisfies: If $\boldsymbol{x}_i \to \boldsymbol{x}$, then $d_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}) \to 0$. This continuity ensures that sufficiently similar inputs are assigned small distances, and thus high influence in both training and prediction. We now show how this principle is explicitly preserved during both the training and prediction stages of our model.

(i) Training phase: the linearized surrogate prediction function (Equation (3)) is constructed as a weighted average over all training responses, where weights decrease quadratically with distance. This ensures that the optimization process emphasizes nearby points and suppresses those that are far, consistent with the locality principle.

(ii) Prediction phase: once the weights $\hat{\boldsymbol{\alpha}}$ are learned, inference is carried out using a soft KNN rule (Equation (2)). Each neighbor's contribution is scaled by a decreasing function of distance $(1 + d^2_{\hat{\boldsymbol{\alpha}}})^{-1/k}$, ensuring that prediction remains localized.

Together, these mechanisms guarantee that the model preserves KNN's locality structure throughout, laying the theoretical foundation for the Fisher consistency result proved in the next section.

5.2. Fisher Consistency

We establish that the proposed model is Fisher consistent—i.e., its predictions converge to the Bayes optimal estimator as the sample size increases [39]. Following the classical KNN asymptotic setting [40, 41], we assume $n \to \infty$, $k \to \infty$, and $k/n \to 0$, and analyze whether the learned predictor approaches the Bayes rule.

To begin, we examine the asymptotic behavior of the learned feature weights. Since the weights $\boldsymbol{\alpha}$ are obtained via a data-dependent optimization process, their values may fluctuate as new samples $(\boldsymbol{x}_i, y_i)$ are introduced. Ensuring Fisher consistency therefore requires that the learned distance metric remains stable as the sample size increases. Theorem 2 shows that the estimated weights converge to a limiting vector as $n \to \infty$ (see supplementary Section S3.2 for details).

Theorem 2 (Stability of the Weights)**.** *Let $\hat{\boldsymbol{\alpha}}(n, k)$ denote the solution of the LP in Theorem 1 computed from n training samples and k nearest neighbors. Under the assumptions of Theorem 1, and as $n \to \infty$, $k \to \infty$, and $k/n \to 0$, the estimated weights converge to a limiting vector $\bar{\boldsymbol{\alpha}}$, that is,*

$$\hat{\boldsymbol{\alpha}}(n, k) \longrightarrow \bar{\boldsymbol{\alpha}}.$$

Given this stability result, we now analyze the consistency of the resulting prediction rule. Theorem 3 shows that, under the corresponding prediction rule in Definition 2, the predictor $\tilde{y}_\ell$ converges to the Bayes-optimal rule under standard asymptotic conditions (see supplementary Section S3.3 for details).

Theorem 3 (Fisher Consistency)**.** *Under the assumptions of Theorem 1, and using the prediction rule from Definition 2, the model satisfies Fisher consistency for any new input* $\boldsymbol{x}_\ell$*. That is,* $\tilde{y}_\ell \longrightarrow h^*(\boldsymbol{x}_\ell)$ *as* $n \to \infty$*,* $k \to \infty$*, and* $k/n \to 0$*, where* $h^*(\boldsymbol{x}_\ell)$ *denotes the Bayes-optimal prediction defined as follows:*

$$h^*(\boldsymbol{x}_\ell) = \begin{cases} \mathbb{1}\left\{\mathbb{P}(Y = 1 \mid X = \boldsymbol{x}_\ell) \geq \frac{1}{2}\right\}, & \textit{(Binary classification)} \\ \arg\max_{c \in \mathcal{Y}} \eta_c(\boldsymbol{x}_\ell), & \textit{(Multi-class classification)} \\ \mathbb{E}[Y \mid X = \boldsymbol{x}_\ell]. & \textit{(Regression)} \end{cases}$$

Importantly, the consistency result does not depend on the specific form of the learned weights, but only requires that the resulting distance metric satisfies basic locality and positivity conditions. Although the theoretical guarantee holds under idealized asymptotic assumptions—such as large sample sizes and vanishing k/n ratios—these conditions may not always be satisfied in practical applications. Nevertheless, the proposed method remains effective in finite-sample settings by minimizing empirical prediction error and adaptively learning the weight vector $\boldsymbol{\alpha}$ from data. This data-driven adaptation promotes better local structure and neighbor selection, enhancing predictive performance even beyond the asymptotic setting.

6. Simulation Study

This section evaluates the predictive performance of the proposed KNN algorithm using the Gurobi optimizer to solve the linearized objective. Euclidean distance is used to calculate the feature distances, but alternative metrics can be adopted depending on the application context. For each dataset, we conduct repeated random splits into training and test sets. Within each training split, an internal validation subset is further generated to select the neighborhood size k. For each candidate k, feature weights $\boldsymbol{\alpha}$ are learned on the corresponding internal training data via the proposed optimization model, and validation performance is evaluated accordingly. The selected k and the associated learned weights are then used to produce predictions on the outer test set. All simulations are conducted on a MacBook Pro equipped with an Apple M4 processor, 16GB of RAM, running macOS Tahoe 26.3.

6.1. Data Generation

We consider three data generation methods: linear, quadratic, and interaction scenarios. Let there be n training points (the m validation points for performance computation will be generated the same way), and ϑ features for each vector $\boldsymbol{x}_i$. The data generation formulas are as follows, (i) Linear: $y_i = \sum_{j=1}^{p} a_i^j x_i^j + \sum_{j=p+1}^{q+p} b_i^j x_i^j + \epsilon$, (ii) Quadratic: $y_i = \sum_{j=1}^{p} a_i^j (x_i^j)^2 + \sum_{j=p+1}^{q+p} b_i^j (x_i^j)^2 + \epsilon$, (iii) Interaction: $y_i = \sum_{j=1}^{p} a_i^j x_i^j + \sum_{j=p+1}^{q} b_i^j x_i^j + \sum_{j=1}^{\vartheta} \sum_{j'=1}^{\vartheta} s_n^{j,j'} x_i^j x_i^{j'} + \epsilon$, where $i \in \{1, \ldots, n+m\}$, and ϵ is an additive noise term on the response variable. For the dataset generation, a_i^j, b_i^j and $s_i^{j,j'}$ are drawn from a normal distribution $\mathcal{N}(0, 10)$. If x_i^j is continuous, it is drawn from a normal distribution $\mathcal{N}(0, 1)$. If x_i^j is discrete, it is drawn from a uniform distribution with possible outcomes $\{0, 1, 2\}$. The noise term ϵ follows $\mathcal{N}(0, 1)$.

To assess the predictive performance of the proposed models under different scenarios, we consider various combinations of the number of samples and feature dimensions, including $n = 50, 100, 500$ and $\vartheta = p + q = 4, 6, 8$, where ϑ represents the total number of variables. Additionally, the scenarios are divided into three types based on the distribution of discrete and continuous variables: (a) $(p, q) = (\vartheta, 0)$, (b) $(p, q) = (\vartheta/2, \vartheta/2)$, and (c) $(p, q) = (0, \vartheta)$. For each setup, we perform 500 repetitions of data generation and use the proposed model for fitting and prediction.

6.2. Evaluation Criteria

Depending on whether y is continuous or categorical, different evaluation metrics are used. For regression tasks, we evaluate the model using MSE and R^2:

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} (y_i - \tilde{y}_i)^2, \quad \text{and} \quad R^2 = 1 - \frac{\sum_{i=1}^{m} (y_i - \tilde{y}_i)^2}{\sum_{i=1}^{m} (y_i - \bar{y})^2},$$

where y_i is the true value, $\tilde{y}_i$ is the predicted value, $\bar{y}$ is the mean of the target variable, and m is the number of test samples. MSE measures the discrepancy between predicted and actual values, with smaller values indicating better performance. R^2 indicates the proportion of variance in the target variable explained by the model, with a value closer to 1 representing a better fit. For classification tasks, we use accuracy and precision as the evaluation metric, with higher values indicating better predictive performance. Note that this section focuses on predictive performance for continuous target variables only; results for categorical targets are reported in supplementary Section S5.2.

6.3. Performance under Varying Sample Size and Feature Dimension

First, we present the predictive performance of the proposed model under different scenarios and parameter settings. As an example, we showcase the results for scenario (b) across various values of n and ϑ, as shown in Figure 4. Detailed results for other scenarios are provided in supplementary Section S5.1. Our conclusions are as follows: (1) As n increases, the MSE almost always decreases, while the R^2 almost always increases. This trend occurs because a larger sample size allows the model to better fit the data, thereby improving predictive accuracy; (2) As ϑ increases, the MSE increases, while R^2 declines. This happens because a higher ϑ adds complexity to the model. However, if the sample size n grows faster than ϑ, MSE decreases, and R^2 improves.

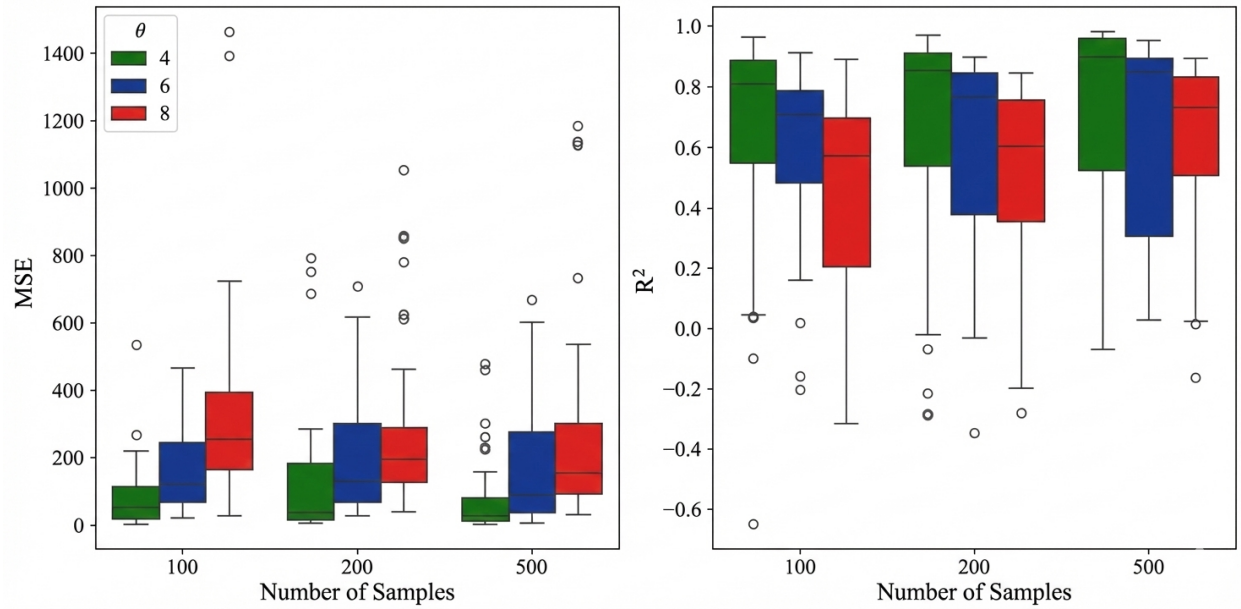


Figure 4: Predictive performance of the proposed method under linear-data scenario (b), measured by MSE (left) and R^2 (right), across different sample sizes and feature dimensions θ.

6.4. Comparison with Other Methods

To illustrate the performance of the data-driven weighted-KNN method, we compare it with three state-of-the-art methods. These methods were selected for their relevance and practicality: they are widely cited in the literature, making them established benchmarks; they are straightforward to implement, ensuring reproducibility; and they have demonstrated strong performance on datasets similar to those used in our study, providing a robust basis for comparison:

(i) Traditional KNN Method [40]: Neighbors are selected purely based on Euclidean distance in the feature space, with equal weight assigned to each neighbor.

(ii) Similarity Measures Method [12]: this approach uses tailored similarity scores for categorical features, such as Overlap, Occur-Frequency (OF), and Eskin measures. These similarity scores replace Euclidean distance in the neighbor selection step.

(iii) Generalized Weighted Distance KNN (GwKNN) [19]: this method introduces learnable weights for neighbors and features. The weights are optimized using a gradient-based method to minimize prediction loss, adapting both neighbor contributions and feature relevance.

Specific details and implementation procedures for these methods are provided in supplementary Section S5.3. Next, we compare the predictive performance of these methods across different scenarios in ϑ (Section 6.4.1), feature importance (Section 6.4.2), and response-noise levels (Section 6.4.3).

6.4.1. Performance under Different Predictor Compositions

This section evaluates different methods under three predictor-composition scenarios: continuous-only, continuous–discrete, and discrete-only predictors. Figure 5 presents a comparative analysis of these methods. Note that the R^2 score is reported only for the proposed method and traditional KNN, as MSE measurements showed suboptimal performance for the other methods. From this figure, we can draw the following conclusions:

- In Scenario (a), which involves only continuous data, the proposed method experiences a slight decline in performance, reflected by marginally higher MSE and more variable R^2 values. Although the proposed method shows slightly worse MSE compared to traditional KNN and GwKNN, it achieves better R^2 values. Furthermore, it significantly outperforms similarity-based methods.

- In Scenario (b), which includes both continuous and discrete predictors, the proposed method performs comparably to methods like KNN and GwKNN, particularly in terms of consistency and reduced variability. It is important to note that we set a_i^j, b_i^j, c_i^j to be drawn from $\mathcal{N}(0, 10)$, meaning that the weights for different data types are similar, which aligns with the traditional KNN approach. This explains why the MSE results are similar to ours. However, the proposed method has an advantage in terms of R^2,

as it adjusts the distance weights in a data-driven manner, allowing for better fitting to the data.

- In Scenario (c), which involves only discrete data, the proposed method demonstrates superior performance with lower MSE and higher R^2 values, even when compared to methods that specialize in handling discrete data, such as similarity measures (Overlap, Eskin, OF).

In summary, under the assumption of similar weights in the data generation mechanism, the proposed method shows robust performance across all scenarios. It achieves nearly optimal R^2 values, with MSE results comparable to the state-of-the-art methods.

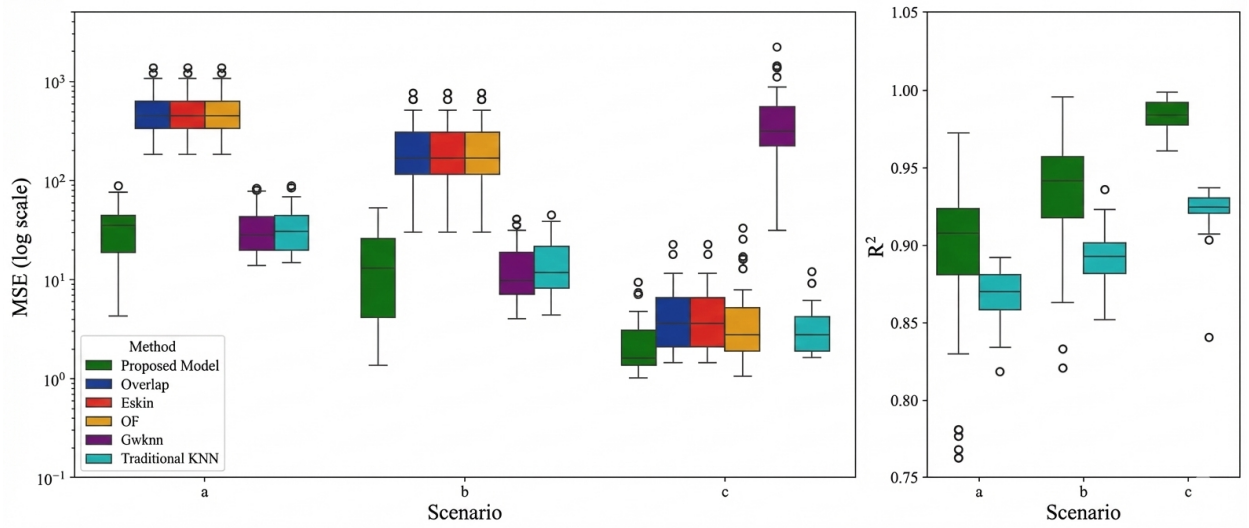
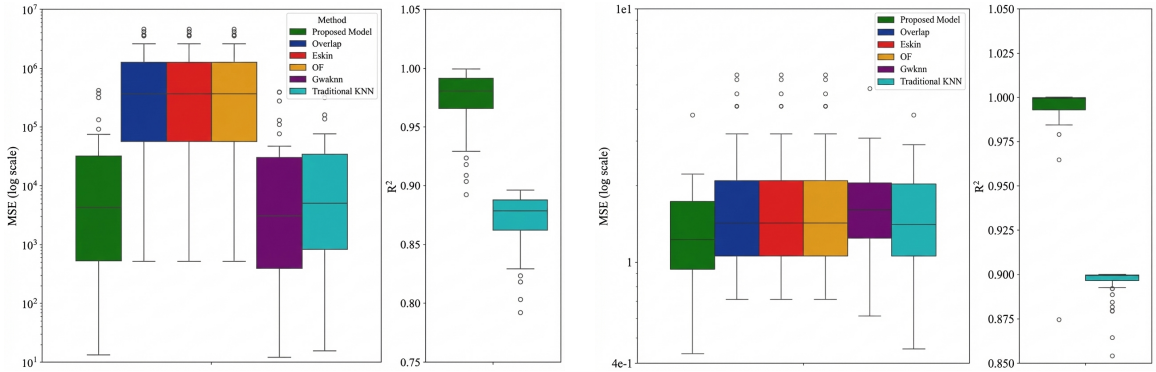


Figure 5: Comparison across methods under interaction-based data generation ($\vartheta = 4$, $n + m = 500$), reporting MSE on a log scale (left) and R^2 (right) across scenarios (a–c).

6.4.2. Feature Importance Variation

In the previous section, we assume similar weights for different features. However, in practice, some features may be more important, as shown by the examples in Section 1. Therefore, in this section, we adjust the weights of each feature (a_n^i, etc.) and regenerate the simulated data to examine how varying feature importance affects model fitting and prediction performance. Figure 6 presents the predictive performance results in the scenario with interaction, where we set two special cases: one with larger weights for continuous

variables and smaller weights for discrete variables, and the other with the opposite configuration. The results show that when feature weights are highly imbalanced, the performance of other methods decreases, mainly because the less important features affect model fitting and predictions. In contrast, our method automatically adjusts weights in a data-driven manner, ensuring prediction accuracy and reliability, further confirming its robustness in handling varying feature importance distributions.



(a) $a_i^j \sim \mathcal{N}(0, 1000), b_i^j \sim \mathcal{N}(0, 1)$ (b) $a_i^j \sim \mathcal{N}(0, 1), b_i^j \sim \mathcal{N}(0, 100)$

Figure 6: Predictive performance in Scenario (b) with an interaction-based model ($n + m = 100$, $\vartheta = 4$) under two feature-importance settings (a)–(b). MSE (log scale, left) and R^2 (right) are reported to illustrate how varying feature importance affects different methods.

6.4.3. Noise Sensitivity Analysis

This section focuses on the impact of varying response-noise levels on the predictive performance of different methods. Using Scenario (c) as an example, Figure 7 shows the predictive performance of different methods under varying noise levels with $\vartheta = 4$, $n + m = 200$, and interaction-based data generation. Note that the GwKNN method is excluded from the simulation due to difficulties in determining appropriate tuning parameters β and η (a limitation of this method), which leads to consistently low R^2 values across all scenarios. The results show that the proposed method exhibits robust performance across all noise variances, with lower MSE and higher R^2 values. As the response-noise variance increases from 1 to 10, the predictive performance of all methods, including the proposed one, decreases. However, our method maintains the best MSE and R^2 values, indicating its relative resilience to noise. This indicates that while all methods are affected by noise, the proposed method is less sensitive, underscoring its robustness.

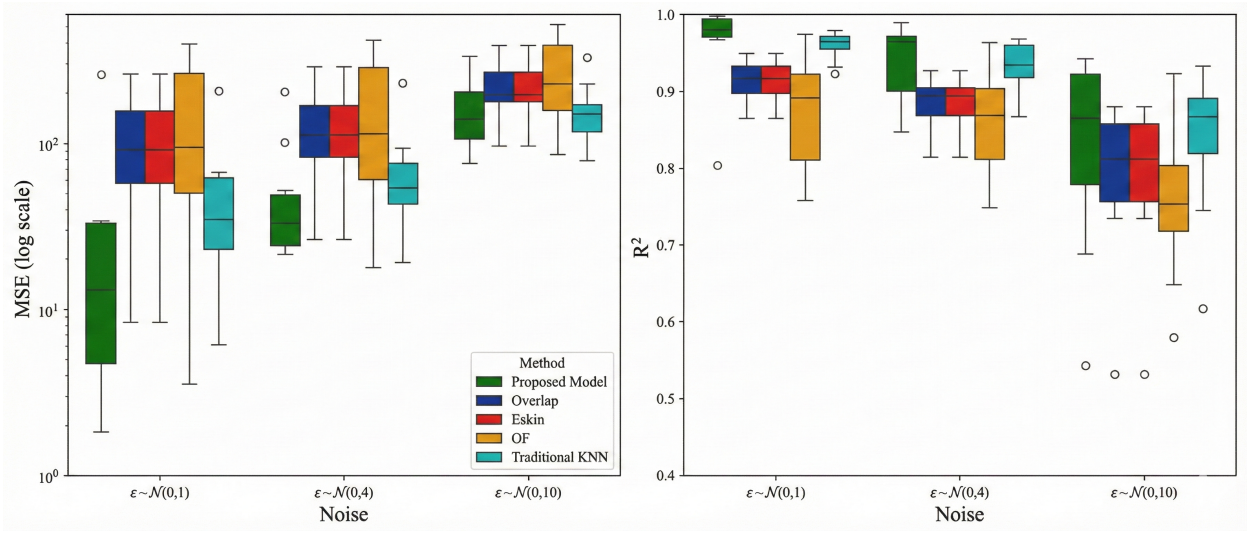


Figure 7: Predictive performance across different methods under varying noise levels in Scenario (c) with $\vartheta = 4$ and $n + m = 200$, based on interaction-based data generation. MSE (log scale, left) and R^2 (right) are reported for different noise settings.

7. Case Study

In this section, we evaluate the proposed data-driven weighted KNN method on real-world datasets from diverse domains (e.g., medical, political, and industrial applications) and compare it with a set of representative KNN-based methods. We further include several recent state-of-the-art KNN variants for classification, namely LMGL-FkNN [20], LIWkNN [21], WPRkNN [22], EDWkNN [23]. In addition, we consider LMSL-FkNN and CMDW-FkNN, both proposed in [24], the double-weighted kNN (DWkNN) [28], as well as two ensemble-based extensions, ExNRule-kNN [25] and ORP-ExNRule-kNN [26]. Unless otherwise specified, we use the default parameter settings suggested in the corresponding papers. It should be noted that several of these baselines are specifically designed for classification tasks. In contrast, the proposed method focuses on learning feature weights through a unified optimization framework and constructing a data-adaptive distance metric. The comparison therefore serves to evaluate the effectiveness of the learned distance in different prediction settings.

7.1. Multiple Datasets Analysis

Table 3 summarizes these datasets, including the number of training and test samples, the number of categorical and numerical predictors, the response type, and the application

domain. Sample sizes correspond to the processed datasets actually used in the experiments. Among the benchmark datasets, ten have categorical responses and four have continuous responses, allowing us to evaluate the proposed framework under different prediction settings. It should be noted that no additional preprocessing mechanism is introduced as part of the proposed method. We only use the numerical representations already available in the datasets where applicable, so that all distance-based methods can be applied in the same input space. The same data representation is used for all competing methods to ensure a fair comparison. More background information on these datasets can be found in Supplementary Section S6.1. For reproducibility, we provide the raw data and corresponding analysis code in the GitHub repository[1].

The experimental protocol described in the previous section is consistently applied to all benchmark datasets. All competing methods are evaluated under the same repeated train/test split protocol and the same data representation, and test-label information is not used during model updating or prediction. Results are reported as the mean and standard deviation over 10 random train/test splits. In each repetition, the number of neighbors k is varied from 2 to 9. Depending on the type of y, appropriate evaluation metrics are adopted: for regression tasks, MSE and R^2 are reported; for classification tasks, accuracy and precision are used.

Table 4 and Table 5 report the regression and classification results, respectively, across the benchmark datasets. From these results, the proposed method can be seen to remain competitive and relatively stable across the benchmark datasets, whereas some baseline methods show greater variability, especially on regression datasets with continuous responses. To assess the statistical significance of performance differences between the proposed method and representative baseline methods, we employed the Wilcoxon signed-rank test. The statistical testing procedure is summarized as follows: (i) For each dataset, 10 repeated random train–test splits were generated. (ii) For each split, the best-performing k was selected for each method. (iii) Performance differences between the proposed method and each baseline were evaluated across the 10 splits using the Wilcoxon signed-rank test. Table 6 summarizes these test results. A filled circle (•) indicates that the proposed method significantly outperforms the corresponding baseline at the 5% level, while an open circle (∘) indicates no significant difference. "–" denotes cases where the corresponding comparison is

[1] https://github.com/Liangliangzhuang/WeightedKNN

Table 3: Summary of the benchmark datasets used in the experiments, including sample splits, feature types, output type, and application domain. Sample sizes correspond to the processed datasets used in the experiments. Detailed dataset descriptions and source links are provided in supplementary material Section S6.1.

Dataset	Training	Test	Categorical Variables	Numerical Variables	y	Domain
House Votes	348	87	16	0	Categorical	Politics
Ionosphere	280	71	0	34	Categorical	Physics
Steel Industry	876	219	6	6	Categorical	Industry
Pima Indians	614	154	0	8	Categorical	Medical
Liver Disorder	466	117	1	9	Categorical	Medical
Iris	120	30	0	4	Categorical	Biology
Heart Disease	237	60	6	7	Categorical	Medical
Australian Credit	552	138	8	6	Categorical	Finance
Sonar	166	42	0	60	Categorical	Defense
Balance Scale	500	125	4	0	Categorical	Education
Insurance	1070	268	3	3	Continuous	Finance
Servo	134	33	2	2	Continuous	Industry
Concrete Strength	824	206	0	8	Continuous	Civil Engineering
Airfoil Self-Noise	1202	301	0	5	Continuous	Aerospace

Table 4: Regression results (mean ± std) of all methods across benchmark datasets.

Method	Insurance	Servo	Concrete Strength	Airfoil Self-Noise
Proposed Model	0.642 ± 0.334	0.779 ± 0.108	0.685 ± 0.047	0.867 ± 0.055
Traditional KNN	0.135 ± 0.047	0.674 ± 0.100	0.682 ± 0.049	0.868 ± 0.047
Overlap KNN	0.606 ± 0.052	0.689 ± 0.114	0.549 ± 0.061	0.167 ± 0.053
Eskin KNN	0.619 ± 0.061	0.774 ± 0.089	0.572 ± 0.058	0.254 ± 0.064
OF KNN	0.614 ± 0.036	0.731 ± 0.115	0.157 ± 0.092	0.132 ± 0.055

Table 5: Classification results (mean ± std) of all methods across benchmark datasets.

Method	House Votes	Ionosphere	Pima Indians	Liver Disorder	Iris	Heart Disease	Australian Credit	Sonar	Balance Scale
Proposed Model	0.953 ± 0.016	0.883 ± 0.047	0.710 ± 0.048	0.672 ± 0.041	0.967 ± 0.027	0.690 ± 0.085	0.678 ± 0.025	0.788 ± 0.057	0.836 ± 0.026
Traditional KNN	0.944 ± 0.013	0.877 ± 0.031	0.710 ± 0.049	0.674 ± 0.034	0.957 ± 0.039	0.645 ± 0.066	0.687 ± 0.036	0.798 ± 0.064	0.868 ± 0.014
Overlap KNN	0.587 ± 0.069	0.559 ± 0.143	0.530 ± 0.154	0.584 ± 0.202	0.333 ± 0.000	0.500 ± 0.035	0.523 ± 0.056	0.510 ± 0.023	0.459 ± 0.004
Eskin KNN	0.939 ± 0.014	0.724 ± 0.157	0.639 ± 0.024	0.679 ± 0.024	0.883 ± 0.059	0.770 ± 0.054	0.841 ± 0.022	0.531 ± 0.070	0.806 ± 0.036
OF KNN	0.945 ± 0.020	0.510 ± 0.191	0.612 ± 0.040	0.704 ± 0.006	0.720 ± 0.107	0.792 ± 0.065	0.854 ± 0.021	0.474 ± 0.065	0.805 ± 0.026
Generalized Weighted KNN	0.944 ± 0.013	0.846 ± 0.039	0.719 ± 0.040	0.662 ± 0.040	0.967 ± 0.035	0.635 ± 0.067	0.687 ± 0.046	0.800 ± 0.080	0.858 ± 0.020
LMGL-FkNN	0.920 ± 0.031	0.793 ± 0.028	0.603 ± 0.042	0.593 ± 0.059	0.887 ± 0.071	0.553 ± 0.082	0.607 ± 0.040	0.626 ± 0.087	0.806 ± 0.042
LIWkNN	0.937 ± 0.019	0.862 ± 0.034	0.702 ± 0.031	0.647 ± 0.040	0.960 ± 0.031	0.600 ± 0.056	0.670 ± 0.034	0.819 ± 0.066	0.826 ± 0.029
WPRkNN	0.932 ± 0.022	0.830 ± 0.038	0.714 ± 0.048	0.673 ± 0.030	0.973 ± 0.026	0.637 ± 0.078	0.681 ± 0.029	0.764 ± 0.087	0.890 ± 0.011
EDWkNN	0.941 ± 0.019	0.869 ± 0.030	0.706 ± 0.033	0.652 ± 0.051	0.957 ± 0.032	0.603 ± 0.034	0.658 ± 0.034	0.819 ± 0.060	0.815 ± 0.049
LMSL-FkNN	0.949 ± 0.024	0.882 ± 0.039	0.706 ± 0.031	0.668 ± 0.051	0.963 ± 0.033	0.608 ± 0.032	0.677 ± 0.034	0.829 ± 0.052	0.874 ± 0.021
CMDW-FkNN	0.923 ± 0.025	0.848 ± 0.042	0.718 ± 0.045	0.662 ± 0.042	0.970 ± 0.029	0.630 ± 0.051	0.676 ± 0.032	0.743 ± 0.076	0.868 ± 0.010
ExNRule-kNN	0.932 ± 0.018	0.930 ± 0.028	0.713 ± 0.019	0.713 ± 0.016	0.963 ± 0.037	0.790 ± 0.068	0.846 ± 0.023	0.819 ± 0.053	0.855 ± 0.021
ORP-ExNRule-kNN	0.947 ± 0.016	0.917 ± 0.030	0.714 ± 0.036	0.665 ± 0.028	0.973 ± 0.026	0.642 ± 0.069	0.691 ± 0.028	0.800 ± 0.057	0.860 ± 0.015
DWkNN	0.953 ± 0.021	0.885 ± 0.036	0.693 ± 0.044	0.709 ± 0.000	0.667 ± 0.000	0.672 ± 0.033	0.587 ± 0.025	0.845 ± 0.045	0.450 ± 0.016

not available. The complete results are reported in the supplementary material.

Overall, the proposed method demonstrates competitive and relatively stable performance across the benchmark datasets. Clear advantages are observed on datasets such as *Ionosphere*, *Steel*, and *Heart-Cleveland*, where the proposed method significantly outperforms several baseline methods. By contrast, on datasets such as *Pima* and *Iris*, the number of statistically significant improvements is more limited, suggesting that performance differences among competing KNN variants are comparatively small on these datasets. For regression datasets such as *Servo* and *Concrete*, the proposed method also remains competitive against several available baselines.

Table 6: Statistical significance comparison (representative datasets) between the proposed method and baseline methods based on the Wilcoxon signed-rank test. • indicates that the proposed method significantly outperforms the corresponding baseline at the 5% level, ∘ indicates no significant difference, and "–" denotes unavailable comparisons.

Dataset	Traditional	Overlap	Eskin	OF	GwKNN	LMGL-FkNN	LIWkNN	WPRkNN	EDWkNN	LMSL-FkNN	CMDW-FkNN	ExNRule-kNN	ORP-ExNRule-kNN	DWkNN
House Votes	∘	•	∘	∘	∘	•	•	•	∘	∘	•	•	∘	∘
Ionosphere	∘	•	•	•	•	•	∘	•	∘	∘	•	∘	∘	∘
Steel Industry	∘	•	•	•	∘	•	•	∘	•	∘	∘	•	•	∘
Pima	∘	•	•	•	∘	•	∘	∘	∘	∘	∘	∘	∘	∘
Iris	∘	•	•	•	∘	•	∘	∘	∘	∘	∘	∘	∘	•
Heart Disease	∘	•	∘	∘	∘	•	•	∘	•	•	•	∘	∘	∘
Servo	•	•	∘	∘	∘	–	–	–	–	–	–	–	–	–
Concrete	∘	•	•	•	•	–	–	–	–	–	–	–	–	–

Besides predictive performance, we evaluate computational efficiency by reporting the average training and prediction times of all methods on the real-world datasets (full results

are provided in the supplementary material). From a theoretical complexity perspective, let n denote the number of training samples, ν the number of features, k the number of nearest neighbors, and C the number of classes. For traditional KNN and most distance-weighted KNN variants, the dominant cost in the prediction stage comes from computing distances between the query and all training samples, resulting in a per-query complexity of $\mathcal{O}(n\nu)$. For EDWkNN, this order remains unchanged, with only a small additional cost for weight computation. For LMGL-FkNN and LMSL-FkNN, although extra local-mean or membership computations are involved, the prediction complexity can be approximately written as $\mathcal{O}(n\nu + Ck\nu)$, which is still dominated by $\mathcal{O}(n\nu)$ in the same asymptotic sense. In contrast, methods such as LIWkNN, WPRkNN, and CMDW-FkNN introduce additional weighting or membership computations during training or preprocessing, while ExNRule-kNN and ORP-ExNRule-kNN further increase both training and prediction costs through ensemble construction, random projection, and model selection.

In the main text, we focus on prediction time, since test-time efficiency is particularly important in practical applications. Figure 8 shows the average prediction time of different methods across the real-world datasets, while the average training time is reported in the supplementary material. The proposed method achieves competitive and stable prediction time across datasets. This is because inference only requires weighted distance computations using the learned feature weights and does not introduce additional test-time optimization, making the method suitable for scenarios with offline training and frequent or latency-sensitive prediction.

7.2. Detailed Analysis of Insurance Dataset

Next, we take the *Insurance* dataset as an example to explain why the proposed method achieves superior predictive performance. Table 7 shows the feature weights obtained through the proposed data-driven methods (rounded), where the `Smoker` feature has a significant impact on the prediction. To further validate this, we examine the performance of traditional KNN with only one feature retained (see Figure 9(a)) and only one feature excluded (see Figure 9(b)). The results confirm that the `Smoker` feature is crucial for the output: when considered alone, the R^2 value exceeds that of traditional KNN, while removing it drops the R^2 below 0.1. These findings highlight the significant imbalance in feature importance within this dataset. Traditional KNN performs poorly because it assumes equal importance for all features, and redundant weights negatively affect the results. As for the

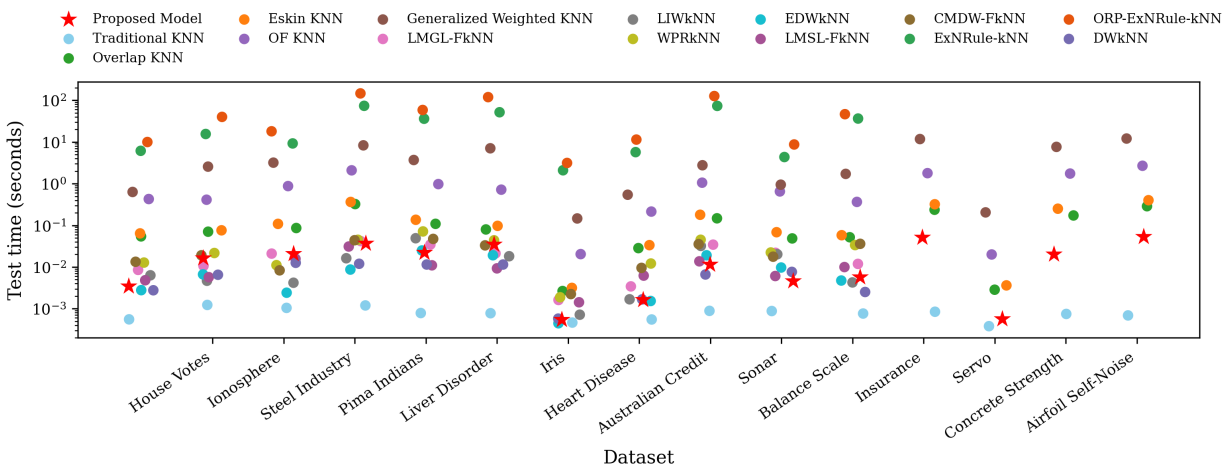


Figure 8: Average prediction time across multiple real-world datasets. Each marker corresponds to the mean test time of a method aggregated over repeated train–test splits (log-scale on the y-axis). The proposed method is highlighted with a red star for visual emphasis.

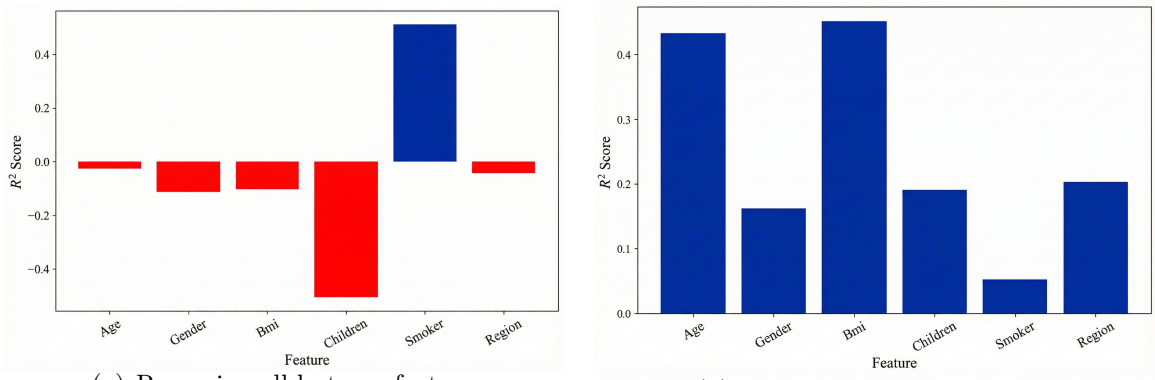
GwKNN method, its strength lies in parameter tuning to achieve optimal performance, but its adjustment capability is limited for this type of dataset and it may not reach the best outcome. In contrast, similarity-based methods improve the performance of traditional KNN, particularly in handling discrete variables. However, compared to these methods, the proposed method assigns a weight to each feature and dynamically adjusts the weights through a linear optimization approach, better fitting the data and delivering superior predictive performance despite the imbalance in feature importance.

Table 7: Estimated feature weights for the Insurance dataset using the proposed method with $k = 5$ (values scaled by 10^{-5} for readability).

	Age	Gender	Bmi	Children	Smoker	Region
Weight	$\hat{\alpha}_1 = 1.00$	$\hat{\alpha}_2 = 1.00$	$\hat{\alpha}_3 = 1.00$	$\hat{\alpha}_4 = 1.00$	$\hat{\alpha}_5 = 5.00 \times 10^4$	$\hat{\alpha}_6 = 1.00$

8. Conclusion

This study presents a data-driven KNN model that uses an optimized weighted distance to learn feature relevance from data. By applying weights to both distance calculations and neighbor contributions, it corrects imbalanced feature importance and improves similarity measurement across continuous and categorical variables. A novel optimization reformulates



(a) Removing all but one feature

(b) Removing only one feature

Figure 9: R^2 of traditional KNN on the Insurance dataset under two feature-ablation settings: keeping only one feature (a) and removing one feature (b).

the learning objective in linear form and solves it via an LP framework, allowing feature weights to adjust dynamically to the data's distribution. Theoretical analysis further verifies the core KNN assumption of "similarity by proximity" and demonstrates Fisher consistency. These empirical results should be interpreted from the perspective of distance learning: the proposed method aims to construct a task-adaptive weighted distance that can be used with standard KNN prediction rules for different response types. Key findings include: (a) Predictive performance steadily improves as sample size grows, maintaining low error. (b) Across different predictor compositions, varying feature importance, and noisy conditions, the proposed method shows competitive performance in terms of stability and accuracy relative to benchmark KNN-based methods (c) Across several real-world datasets, the proposed method demonstrates competitive predictive performance, particularly on datasets with heterogeneous feature importance.

Although the proposed framework demonstrates improved predictive performance and interpretability, several limitations merit further investigation. From a computational perspective, the proposed framework learns weights through a unified optimization model, where the original problem is approximated and reformulated as a linear program to improve tractability. This provides a principled and theoretically grounded way to jointly learn the weighting structure, but still makes training more expensive than classical KNN, particularly for datasets with larger sample sizes. Future work may therefore explore more scalable variants, such as subsampling, prototype selection, parallel LP solvers, or decomposition-based strategies. In addition, while the model reduces manual parameter tuning by learning feature

weights in a data-driven manner, the choice of neighborhood size k may still influence performance in practice; adaptive or data-driven strategies for selecting k (e.g., validation-based rules or locally adaptive neighborhoods) are therefore worth exploring. Finally, when the number of categorical features is extremely large, when the associated categories are highly diverse, or when the observed responses are affected by label noise, additional strategies such as feature screening, embedding/representation learning, robust label-noise handling, or hybrid preprocessing pipelines may be needed to further improve efficiency and robustness. Recent advances in anti-noisy-label learning for fuzzy classification on imbalanced data [42] may provide useful directions for such extensions. For applications where categorical responses are of primary interest, an interesting future direction is to integrate the proposed distance-learning framework with classification-specific aggregation or calibration strategies.

Acknowledgment

The work was supported by Singapore MOE AcRF Tier 2 grant (A-8001052-00-00, A-8002472-00-00) and the Future Resilient Systems project supported by the National Research Foundation Singapore under its CREATE programme.

References

[1] A. Xu, R. Wang, X. Weng, Q. Wu, L. Zhuang, Strategic integration of adaptive sampling and ensemble techniques in federated learning for aircraft engine remaining useful life prediction, Applied Soft Computing 175 (2025) 113067. doi:10.1016/j.asoc.2025.113067.

[2] J. Shen, T. Ma, D. Song, F. Xu, An embedded physical information network for blade crack detection considering dynamic multi-level credibility, Mechanical Systems and Signal Processing 224 (2025) 111948. doi:10.1016/j.ymssp.2024.111948.

[3] D. Zhu, A. Xu, Z. Chen, S. Ding, G. Fang, An online Bayesian framework for identifying latent system degradation states, IEEE Transactions on Reliability 75 (2026) 542–554. doi:10.1109/TR.2025.3647489.

[4] P. Cunningham, S. J. Delany, K-nearest neighbour classifiers-a tutorial, ACM computing surveys 54 (2021) 1–25. doi:10.1145/3459665.

[5] S.-B. Chen, Y.-L. Xu, C. H. Ding, B. Luo, A nonnegative locally linear KNN model for image recognition, Pattern Recognition 83 (2018) 78–90. doi:10.1016/j.patcog.2018.05.024.

[6] D. A. Adeniyi, Z. Wei, Y. Yongquan, Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method, Applied Computing and Informatics 12 (2016) 90–108. doi:10.1016/j.aci.2014.10.001.

[7] T. Prasartvit, A. Banharnsakun, B. Kaewkamnerdpong, T. Achalakul, Reducing bioinformatics data dimension with ABC-kNN, Neurocomputing 116 (2013) 367–381. doi:10.1016/j.neucom.2012.01.045.

[8] J. Zhang, Y. Li, F. Shen, Y. He, H. Tan, Y. He, Hierarchical text classification with multi-label contrastive learning and KNN, Neurocomputing 577 (2024) 127323. doi:10.1016/j.neucom.2024.127323.

[9] R. K. Halder, M. N. Uddin, M. A. Uddin, S. Aryal, A. Khraisat, Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications, Journal of Big Data 11 (2024) 113. doi:10.1186/s40537-024-00973-y.

[10] S. Q. Le, T. B. Ho, An association-based dissimilarity measure for categorical data, Pattern Recognition Letters 26 (2005) 2549–2557. doi:10.1016/j.patrec.2005.06.002.

[11] D. Ienco, R. G. Pensa, R. Meo, Context-based distance learning for categorical data clustering, in: Advances in Intelligent Data Analysis VIII, 2009, pp. 83–94. doi:0.1007/978-3-642-03915-7_8.

[12] 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.

[13] L. Chen, G. Guo, Nearest neighbor classification of categorical data by attributes weighting, Expert Systems with Applications 42 (2015) 3142–3149. doi:10.1016/j.eswa.2014.12.002.

[14] A. Nababan, O. Sitompul, et al., Attribute weighting based K-nearest neighbor using gain ratio, in: Journal of Physics: Conference Series, volume 1007, 2018, p. 012007. doi:10.1088/1742-6596/1007/1/012007.

[15] B. Karabulut, G. Arslan, H. M. Ünver, A weighted similarity measure for k-nearest neighbors algorithm, Celal Bayar Üniversitesi Fen Bilimleri Dergisi 15 (2019) 393–400. doi:10.18466/cbayarfbe.618964.

[16] E.-H. Han, G. Karypis, V. Kumar, Text categorization using weight adjusted k-nearest neighbor classification, in: Advances in Knowledge Discovery and Data Mining, 2001, pp. 53–65. doi:10.1007/3-540-45357-1_9.

[17] S. Ougiaroglou, A. Nanopoulos, A. N. Papadopoulos, Y. Manolopoulos, T. Welzer-Druzovec, Adaptive k-nearest-neighbor classification using a dynamic number of nearest neighbors, in: Advances in Databases and Information Systems, 2007, pp. 66–82. doi:10.1007/978-3-540-75185-4_7.

[18] N. García-Pedrajas, J. A. R. del Castillo, G. Cerruela-García, A proposal for local k values for k-nearest neighbor rule, IEEE Transactions on Neural Networks and Learning Systems 28 (2017) 470–475. doi:10.1109/TNNLS.2015.2506821.

[19] 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.

[20] A. A. Amer, S. D. Ravana, R. A. A. Habeeb, On enhancing data classification using local mean-based fuzzy K-nearest neighbor algorithms, Advances in Data Analysis and Classification (2025). doi:10.1007/s11634-025-00653-6.

[21] H. I. Abdalla, A. A. Amer, Enhancing data classification using locally informed weighted k-nearest neighbor algorithm, Expert Systems with Applications 276 (2025) 126942. doi:10.1016/j.eswa.2025.126942.

[22] A. A. Amer, S. D. Ravana, R. A. A. Habeeb, Effective k-nearest neighbor models for data classification enhancement, Journal of Big Data 12 (2025) 86. doi:10.1186/s40537-025-01137-2.

[23] A. A. Amer, S. D. Ravana, R. A. Ariyaluran Habeeb, Enhanced distance-based weighted k-nearest neighbor algorithm for data classification, KSII Transactions on Internet & Information Systems 19 (2025).

[24] H. I. Abdalla, A. A. Amer, M. Nassef, New fuzzy k-nearest neighbor algorithms for classification performance improvement, Future Generation Computer Systems (2025) 108139. doi:10.1016/j.future.2025.108139.

[25] A. Ali, M. Hamraz, N. Gul, D. M. Khan, S. Aldahmani, Z. Khan, A k nearest neighbour ensemble via extended neighbourhood rule and feature subsets, Pattern Recognition 142 (2023) 109641. doi:10.1016/j.patcog.2023.109641.

[26] A. Ali, Z. Khan, D. M. Khan, S. Aldahmani, An optimal random projection k nearest neighbors ensemble via extended neighborhood rule for binary classification, IEEE Access 12 (2024) 61401–61409. doi:10.1109/ACCESS.2024.3392729.

[27] H. I. Abdalla, A. Altaf, A. A. Hamzah, A threefold-ensemble k-nearest neighbor algorithm, International Journal of Computers and Applications 47 (2025) 70–83. doi:10.1080/1206212X.2024.2446896.

[28] A. Ali, Z. Khan, H. Du, S. Aldahmani, Double weighted k nearest neighbours for binary classification of high dimensional genomic data, Scientific reports 15 (2025) 12681. doi:10.1038/s41598-025-97505-2.

[29] P. Z. G. Qian, H. Wu, C. J. Wu, Gaussian process models for computer experiments with qualitative and quantitative factors, Technometrics 50 (2008) 383–396. doi:10.1198/004017008000000262.

[30] Y. Zhang, S. Tao, W. Chen, D. W. Apley, A latent variable approach to Gaussian process modeling with qualitative and quantitative factors, Technometrics 62 (2020) 291–302. doi:10.1080/00401706.2019.1638834.

[31] Q. Zhang, P. Chien, Q. Liu, L. Xu, Y. Hong, Mixed-input Gaussian process emulators for computer experiments with a large number of categorical levels, Journal of Quality Technology 53 (2021) 410–420. doi:10.1080/00224065.2020.1778431.

[32] X. Cai, L. Xu, C. D. Lin, Y. Hong, X. Deng, Adaptive-region sequential design with quantitative and qualitative factors in application to HPC configuration, Journal of Quality Technology 56 (2024) 5–19. doi:10.1080/00224065.2023.2241680.

[33] W.-A. Lin, C.-L. Sung, R.-B. Chen, Category tree gaussian process for computer experiments with many-category qualitative factors and application to cooling system design, Journal of Quality Technology 56 (2024) 391–408. doi:10.1080/00224065.2024.2359431.

[34] Q. Xiao, Y. Wang, A. Mandal, X. Deng, Modeling and active learning for experiments with quantitative-sequence factors, Journal of the American Statistical Association 119 (2024) 407–421. doi:10.1080/01621459.2022.2123335.

[35] G. Chirici, M. Mura, D. McInerney, N. Py, E. O. Tomppo, L. T. Waser, D. Travaglini, R. E. McRoberts, A meta-analysis and review of the literature on the k-nearest neighbors technique for forestry applications that use remotely sensed data, Remote Sensing of Environment 176 (2016) 282–294. doi:10.1016/j.rse.2016.02.001.

[36] S. Tan, Neighbor-weighted k-nearest neighbor for unbalanced text corpus, Expert Systems with

Applications 28 (2005) 667–671. doi:https://doi.org/10.1016/j.eswa.2004.12.023.

[37] E. F. Vonesh, H. Wang, D. M. and, Generalized least squares, taylor series linearization and fisher's scoring in multivariate nonlinear regression, Journal of the American Statistical Association 96 (2001) 282–291. doi:10.1198/016214501750332857.

[38] R. Shamir, The efficiency of the simplex method: A survey, Management Science 33 (1987) 301–334. doi:10.1287/mnsc.33.3.301.

[39] 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.

[40] E. Fix, J. L. Hodges, Discriminatory analysis-nonparametric discrimination: consistency properties, Project Rand Res Memo 17 (1951) 1–12. doi:10.2307/1403797.

[41] C. J. Stone, Consistent nonparametric regression, Annal of Statistics 5 (1977) 595–620. doi:10.1214/aos/1176343886.

[42] Y. Li, F.-l. Chung, S. Wang, Anti-noisy-labeling AUC maximization learning for fuzzy classification on imbalanced data, IEEE Transactions on Fuzzy Systems 34 (2026) 623–637. doi:10.1109/TFUZZ.2025.3642192.