K-Nearest Neighbor Algorithm Based on the Framework of Ordered Pair of Normalized Real Numbers

Yi Zheng, Xuanbin Ding, Xiang Zhao, Xiaoqin Pan, and Lei Zhou

***Abstract*—The K-nearest neighbors (kNNs) algorithm, a cornerstone of supervised learning, relies on similarity measures constrained by real-number-based distance metrics. A critical limitation of traditional kNN research lies in its confinement to the real-number domain, which inherently restricts its ability to model nonlinear feature interactions in high-dimensional data and amplifies sensitivity to feature redundancy and class imbalance. These limitations arise from the inherent linearity and unidimensional nature of real-number representations, which restrict their ability to model complex feature interdependencies. To transcend these limitations, this article proposes ordered pairs of normalized real numbers (OPNs)-kNN, a novel framework grounded in OPNs. Departing from the conventional real-number paradigm, OPNs-kNN constructs feature pairs as multidimensional OPNs tuples and employs a generalized OPNs-valued metric to explicitly model nonlinear relationships, thereby addressing the inherent shortcomings of real-number-based kNN. Extensive experiments on nine University of California, Irvine (UCI) benchmark datasets (e.g., glass, wines, and seeds) demonstrate that OPNs-kNN achieves statistically significant improvements in classification accuracy, precision, recall, and F1-score compared with traditional kNN and its enhanced variants. This work pioneers a nonreal-number computational framework, proving that moving beyond real-number constraints enables more expressive representations of data relationships, opening new directions for designing robust machine learning models in complex domains.**

***Impact Statement*—Traditional K-nearest neighbor (kNN) algorithms rely on real-number-based distance metrics, which can struggle to capture complex, nonlinear relationships between features in high-dimensional data. This article presents the ordered pairs of normalized real numbers (OPNs)-kNN algorithm, an innovative extension of kNN that incorporates OPNs to model these intricate relationships more effectively. By introducing feature pairing and utilizing a generalized metric based on OPNs, this method improves predictive accuracy and interpretability, especially in fields requiring high precision, such as medical diagnosis and environmental data analysis. The proposed method represents an advancement in machine learning, enhancing the robustness of models when handling complex datasets. This, in turn, opens up a new research direction for both the theory and applications of machine learning.**

***Index Terms*—Feature engineering, generalized metric, machine learning, nearest neighbor algorithm, ordered pair of normalized real numbers (OPNs).**

I. Introduction

THE K-nearest neighbors (kNNs) algorithm, introduced in the 1950s [1], [2], became widely known after Cover and Hart's 1967 formalization [7]. kNN classifies a data point or estimates its value based on its nearest neighbors, following the principle that similar points tend to share common characteristics.

kNN's applications have expanded across fields such as medicine [3], finance [4], and social networks [5]. Key research focuses on improving distance measurements, weighting data, and adaptively selecting k.

kNN often employs a majority vote from its nearest neighbors. Distance-weighted kNN (WkNN) [10] improves accuracy by giving greater weight to closer neighbors, while feature-weighting methods like DCT-kNN [11] enhance classification by assigning weights based on feature importance.

Various similarity measures also refine kNN's performance. While Euclidean distance (ED) is common, alternative measures have proven effective for certain datasets [12], [16]. Additionally, the weighted representation-based k-nearest neighbor (WRKNN) and the weighted local meanrepresentation-based k-nearest neighbor rule (WLMRkNN) methods by Gou et al. [13] enhance classification by considering both the k-neighborhood and representation distance. Locally adaptive k-nearest neighbor algorithm based on discrimination class (DC-LakNN) by Pan et al. [14] further improves adaptability by adjusting k based on class discriminants.

The choice of distance measure is crucial for kNN performance [18], [28], [29]. Euclidean and Manhattan distances (MDs) are widely used but sensitive to outliers [30]. Traditional kNN algorithms rely solely on real-number computations, which may overlook deeper patterns beyond this domain.

Recent advancements in kNN variants, such as fuzzy [19] and soft-set-based approaches, have addressed challenges like class imbalance and uncertainty. For instance, the Bonferronimean

Received 31 December 2024; revised 18 March 2025 and 12 April 2025; accepted 29 April 2025. Date of publication 5 May 2025; date of current version 31 October 2025. This article was recommended for publication by Associate Editor Hamid Bouchachia upon evaluation of the reviewers' comments. *(Corresponding author: Lei Zhou.)*

The authors are with the School of Computer Science and Technology, Southwest University of Science and Technology, Sichuan 621010, China (e-mail: zeanyi@mails.swust.edu.cn; dings@mails.swust.edu.cn; shenzhuoyan@mails.swust.edu.cn; xqpanus@swust.edu.cn; zhoulei137@swust.edu.cn).

Digital Object Identifier 10.1109/TAI.2025.3566925

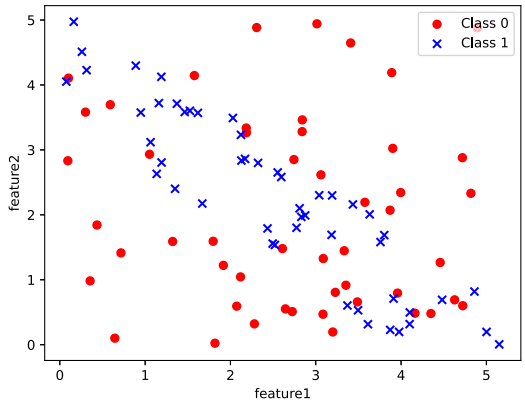

Fig. 1. Feature distribution chart.

based fuzzy k-nearest neighbor (BM-FKNN) classifier leverages the Bonferroni mean to compute local mean vectors, improving robustness in imbalanced datasets [20], while fuzzy parameterized fuzzy soft kNN (FPFS-kNN) and the picture fuzzy soft k-nearest neighbor (PFS-kNN) classifier employ fuzzy soft matrices to handle ambiguity in feature representations [21], [22]. Building on these developments, this article presents the ordered pairs of normalized real numbers (OPNs)-kNN algorithm, an innovative extension of kNN that incorporates OPNs to model intricate feature relationships more effectively. Unlike existing fuzzy or soft-set-based methods, OPNs-kNN introduces a novel mathematical framework for feature pairing and a generalized OPNs-valued metric, enabling the explicit modeling of multidimensional interactions between features.

Fig. 1 illustrates a linear relationship between features, which traditional distance measures may overlook. While feature engineering can introduce new features [15], it struggles with nonlinear relationships and information redundancy, highlighting the need for the OPNs framework to capture complex associations effectively.

The OPNs theory, introduced by Zhou [6] in 2020, provides a novel mathematical framework that has already shown promise in preliminary machine learning applications [8], [9]. The OPNs theory constructs a new computational framework that utilizes pairs of normalized real numbers as fundamental data units. This conversion of real-number data into OPNs enables research and development in machine learning within the OPNs framework. Furthermore, this approach eliminates the necessity of confining original data within the 0–1 range, allowing for unrestricted exploration and development activities within the OPNs framework.

The motivation behind developing the OPNs-kNN algorithm is to address the limitations of traditional kNN algorithms in capturing complex, nonlinear relationships that are not adequately reflected within the real-number domain. Traditional kNN methods, which rely on real-number-based distance measurements, are constrained by their inability to reveal intricate inter-feature relationships or underlying data patterns. In contrast, the OPNs theoretical framework offers a novel mathematical approach with a complete, closed arithmetic system—including rigorous operations and a total ordering—that preserves essential algebraic properties and enables enhanced modeling of multidimensional feature interactions.

By leveraging the OPNs framework, the OPNs-kNN algorithm transforms input data into OPNs, which serve as enhanced data units capable of representing deeper and more complex interdependencies between features. This is achieved through a novel feature pairing strategy that introduces an additional dimension of interaction information. Although the complexity of feature pairings grows with the number of features, our algorithm employs an iterative training process to determine optimal (or near-optimal) pairings, thereby uncovering previously hidden patterns and significantly improving classification performance.

The key contributions of this work are summarized as follows.

1) *Theoretical Advancement:* This study extends OPNs theory by formulating a generalized metric within a fully closed arithmetic framework. This metric not only preserves the properties of classical real-number distance measures (such as symmetry and the triangle inequality) but also effectively captures complex, nonlinear relationships that traditional methods overlook.
2) *Algorithmic Innovation:* We introduce a novel kNN algorithm that incorporates a feature pairing process under the OPNs framework. By transforming input data into OPNs and iteratively identifying optimal feature pairings, the proposed method enriches data representation and models intricate inter-feature dependencies more effectively than conventional kNN.
3) *Practical Validation:* Comprehensive experiments on multiple public datasets (e.g., from the UCI Machine Learning Repository) validate that OPNs-kNN consistently outperforms traditional kNN and its enhanced variants.
4) *Research Impact and Future Directions:* This exploration into the OPNs framework not only provides a robust foundation for constructing machine learning models that operate beyond the limitations of the real-number domain but also opens new avenues for future research in fields that require the modeling of complex data interactions.

In essence, the OPNs-kNN algorithm offers a more flexible and powerful approach to dataset classification by enabling a deeper exploration of data relationships beyond the confines of real numbers. This work addresses the inherent limitations of traditional kNN methods and lays the groundwork for developing other machine learning models within the OPNs framework.

The construction of the kNN algorithm within the OPNs framework is a pioneering endeavor, with only Cui et al. [9] having previously proposed the multisimilarity nearest neighbor classification algorithm using the OPNs pairing concept. The algorithm by Cui et al. [9] utilizes OPNs to combine different

similarity measurement methods, thereby achieving complementarity among various similarity measures. However, this study did not fully leverage the characteristics and computational rules of OPNs, using only the structural form of OPNs and applying OPNs operations in the final result processing. The key difference in our research is that the entire algorithmic process is based on the OPNs theoretical framework, including the OPNs generalized metric and the OPNs pairing theory—elements that were never mentioned in the previous study.

This article primarily focuses on the construction of the kNN algorithm under the OPNs framework, referred to as the OPNs-kNN algorithm. The construction process is divided into three main aspects.

1) Mapping real-number features to OPNs, ensuring that the OPNs-kNN algorithm can process data within the OPNs framework.
2) Introducing an OPNs-valued generalized metric to determine the neighbors of OPNs features.
3) Incorporating a process to train the optimal pairing combination, ensuring the classification effectiveness of the OPNs-kNN algorithm.

II. Preliminary

A. Fundamentals of OPNs

An OPNs, denoted as $\tilde{\alpha} = (\mu_{\tilde{\alpha}}, \nu_{\tilde{\alpha}})$, is defined such that $0 < \mu_{\tilde{\alpha}} < 1$ and $0 < \nu_{\tilde{\alpha}} < 1$. To distinguish OPNs variables, they are marked with a tilde. In terms of form, OPNs may resemble intuitionistic fuzzy numbers (IFNs) [23], [24] and Pythagorean fuzzy numbers (PFNs) [25], [26], but they are fundamentally different. Unlike IFNs, which impose the constraint $\mu_{\tilde{\alpha}} + \nu_{\tilde{\alpha}} \leq 1$, or PFNs, which require $\mu_{\tilde{\alpha}}^2 + \nu_{\tilde{\alpha}}^2 \leq 1$, OPNs impose no such restrictions. This flexibility allows them to handle a broader range of data scenarios, including cases where $\mu_{\tilde{\alpha}} + \nu_{\tilde{\alpha}} > 1$.

Furthermore, while q-rung orthopair fuzzy sets (q-ROFSs) extend IFNs and PFNs by introducing a tunable parameter q [27], they remain constrained by their predefined membership structure. A q-ROFS element is defined as $A(x) = \langle A^+(x), A^-(x) \rangle_q$, where $A^+(x)$ and $A^-(x)$ represent degrees of support and opposition, respectively. However, in machine learning applications, numerical features typically do not possess inherent degrees of support and opposition, which makes q-ROFS less suitable for direct numerical representation. OPNs, by contrast, directly encode two feature components without requiring an interpretation of support or opposition, allowing for greater flexibility in data representation.

Additionally, although increasing q in q-ROFS expands the membership space, it still imposes fixed constraints. Once q is selected, data falling outside the corresponding space cannot be represented. OPNs impose no such restrictions, enabling broader applicability to diverse datasets.

Moreover, subtraction and division are essential in machine learning, particularly for loss functions and distance computations. q-ROFS lacks closed-form support for these operations, as they can violate membership constraints. In contrast, OPNs provide a fully closed arithmetic system, where subtraction and division are well-defined as inverse operations. This closure ensures that all computations remain valid within the OPNs framework, making it a more suitable foundation for constructing machine learning models.

Given these advantages, we adopt OPNs instead of q-ROFS, as they offer a more flexible and mathematically robust approach for learning algorithms.

B. Arithmetic and Structural Properties of OPNs

Before defining the arithmetic operations of OPNs, we introduce the following fundamental transformations [6], which ensure consistency with real-number operations

$$c \odot x = \varphi(c\varphi^{-1}(x)) \tag{1}$$

$$x \oplus y = \varphi(\varphi^{-1}(x) + \varphi^{-1}(y)) \tag{2}$$

$$x \otimes y = \varphi(\varphi^{-1}(x)\varphi^{-1}(y)). \tag{3}$$

Here, the function $\varphi : \mathbb{R} \to (0, 1)$ is continuous and strictly monotonically increasing, satisfying $\varphi(x) + \varphi(-x) = 1$. There are infinitely many functions $\varphi(x)$ that satisfy these properties. Below is a practical specific $\varphi(x)$ and $\varphi^{-1}(x)$

$$\varphi(x) = \begin{cases} 10^{-n}x + 1 - \xi_n, & 5 - 9n \leq x < 14 - 9n, \\ & n \geq 2; \\ 10^{-1}x + 0.5, & -4 \leq x < 4; \\ 10^{-n}x + \xi_n, & 9n - 14 \leq x < 9n - 5, \\ & n \geq 2, \end{cases} \tag{4}$$

and

$$\varphi^{-1}(x) = \begin{cases} 10^{n}(x - 1 + \xi_n), & 10^{-n} \leq x < 10^{1-n}, \\ & n \geq 2; \\ 10^{-1}(x - 0.5), & 0.1 \leq x < 0.9; \\ 10^{n}(x - \xi_n), & 1 - 10^{1-n} \leq x < 1 - 10^{-n}, \\ & n \geq 2. \end{cases} \tag{5}$$

where $\xi_1 = 0.5, \xi_{n+1} = 10^{-(n+1)}(81n - 45) + \xi_n, n \geq 1$.

For OPNs $\tilde{\alpha}$ and $\tilde{\beta}$, along with a real number c, the scalar multiplication, addition, and multiplication of OPNs are defined as follows:

$$c\tilde{\alpha} = (c \odot \mu_\alpha, c \odot \nu_\alpha) \tag{6}$$

$$\tilde{\alpha} + \tilde{\beta} = (\mu_\alpha \oplus \mu_\beta, \nu_\alpha \text{oplus} \nu_\beta) \tag{7}$$

$$\tilde{\alpha} \cdot \tilde{\beta} = ((1 - \mu_\alpha) \otimes \nu_\beta \oplus (1 - \nu_\alpha) \otimes \mu_\beta, \\ (1 - \mu_\alpha) \otimes \mu_\beta \oplus (1 - \nu_\alpha) \otimes \nu_\beta). \tag{8}$$

Similar to real numbers, the set of OPNs has a unique additive identity element, denoted as $\tilde{0}$, and a unique multiplicative identity element, denoted as $\tilde{1}$. Specifically, $\tilde{0} = (\varphi(0), \varphi(0))$, and $\tilde{1} = (\varphi(0), \varphi(-1))$. It can easily be shown that $\tilde{\alpha} + \tilde{0} = \tilde{0} + \tilde{\alpha} = \tilde{\alpha}$, and $\tilde{\alpha} \cdot \tilde{1} = \tilde{1} \cdot \tilde{\alpha} = \tilde{\alpha}$.

For any OPNs $\tilde{\beta}$, its additive inverse exists and is unique, denoted as $-\tilde{\beta}$, and it is also an OPNs; it can easily be proven that $-\tilde{\beta} = (-1) \cdot \tilde{\beta}$.

For any nonneutral OPNs $\tilde{\beta}$, satisfying $\mu_\beta \neq \nu_\beta$, its multiplicative inverse exists and is unique, denoted as $\tilde{\beta}^{-1} =$

$(\mu_{\beta^{-1}}, \nu_{\beta^{-1}})$, and this is also an OPNs. The components $\mu_{\beta^{-1}}$ and $\nu_{\beta^{-1}}$ can be calculated using the following formulas:

$$\varphi^{-1}(\mu_{\beta^{-1}}) = \frac{\varphi^{-1}(\mu_\beta)}{(\varphi^{-1}(\mu_\beta))^2 - (\varphi^{-1}(\nu_\beta))^2}$$
$$\varphi^{-1}(\nu_{\beta^{-1}}) = \frac{\varphi^{-1}(\nu_\beta)}{(\varphi^{-1}(\nu_\beta))^2 - (\varphi^{-1}(\mu_\beta))^2}. \quad (9)$$

It can easily be proven that $\tilde{\beta} \cdot \tilde{\beta}^{-1} = \tilde{1}$.

At this point, we can define the subtraction and division of OPNs as follows:

$$\tilde{\alpha} - \tilde{\beta} = \tilde{\alpha} + (-1) \cdot \tilde{\beta}, \quad (10)$$
$$\tilde{\alpha}/\tilde{\beta} = \tilde{\alpha} \cdot \tilde{\beta}^{-1}. \quad (11)$$

Next, we introduce the total order rule for OPNs. To establish a total order, we first classify each OPNs as positive, negative, or neutral based on the sum of its components. Specifically, for an OPNs $\tilde{\alpha}$:

1) If $\mu_{\tilde{\alpha}} + \nu_{\tilde{\alpha}} < 1$, $\tilde{\alpha}$ is classified as positive.
2) If $\mu_{\tilde{\alpha}} + \nu_{\tilde{\alpha}} > 1$, $\tilde{\alpha}$ is classified as negative.
3) If $\mu_{\tilde{\alpha}} + \nu_{\tilde{\alpha}} = 1$, $\tilde{\alpha}$ is classified as neutral.

Two OPNs, $\tilde{\alpha}$ and $\tilde{\beta}$, are considered equal if and only if $\mu_\alpha = \mu_\beta$ and $\nu_\alpha = \nu_\beta$. Any OPNs that is not neutral is referred to as nonneutral. Similar to real numbers, positive and negative OPNs are treated as having opposite signs, while a neutral OPNs is its own opposite.

We define the following comparison rules.

1) If $\tilde{\alpha}$ is positive, or if $\tilde{\alpha}$ is neutral with $\mu_\alpha > \nu_\alpha$, then $\tilde{\alpha} > \tilde{0}$.
2) If $\tilde{\alpha}$ is negative, or if $\tilde{\alpha}$ is neutral with $\mu_\alpha < \nu_\alpha$, then $\tilde{\alpha} < \tilde{0}$.

Finally, the ordering of two OPNs follows naturally.

1) If $\tilde{\alpha} - \tilde{\beta} > \tilde{0}$, then $\tilde{\alpha} > \tilde{\beta}$.
2) If $\tilde{\alpha} - \tilde{\beta} < \tilde{0}$, then $\tilde{\alpha} < \tilde{\beta}$.

C. OPNs-Valued Distance Measure and Its Generalized Metric Properties

In practical applications, while data representations are typically expressed in real numbers, the underlying structures governing data relationships are not necessarily confined to the real-number space. Conventional distance measures, which rely solely on real-number-based calculations, may therefore be insufficient in capturing complex interdependencies within data. To address this, we introduce the concept of a generalized metric, which extends the classical notion of a measure [31]. Unlike traditional metrics that are strictly defined in the real-number domain, a generalized metric allows distances to be derived from any ordered domain, offering greater flexibility in modeling diverse data structures.

Since the OPNs framework provides a well-defined arithmetic structure with a total order, it serves as a natural foundation for constructing a generalized metric. Before formally defining the OPNs-valued distance measure, it is essential to first compare OPNs-based metrics with existing distance formulations, such as those in IFN, PFN, and classical real-number metrics. The following section presents this comparative analysis to highlight the advantages and theoretical distinctions of the OPNs-valued distance measure.

1) Comparison of OPNs Metric With IFN, PFN, and Classical Real-Number Metrics: To understand the fundamental distinctions between the OPNs-valued distance measure and existing metrics in IFN, PFN, and classical real-number domains, we analyze their computational properties, underlying mathematical frameworks, and implications for metric space construction.

a) Classical real-number metrics: In traditional metric spaces, a distance function $d : X \times X \to \mathbb{R}$ satisfies the standard axioms of a metric, including nonnegativity, symmetry, and the triangle inequality. The most commonly used distances, such as the ED and Hamming distance, are defined as follows.

1) ED:

$$d_{\text{Euc}}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \quad (12)$$

2) Hamming distance:

$$d_{\text{Ham}}(x, y) = \sum_{i=1}^{n} |x_i - y_i|. \quad (13)$$

Both distances operate entirely within the real-number space, where all arithmetic computations, including subtraction, squaring, and absolute differences, follow conventional real-number operations.

b) IFN and PFN metrics: real-number-based computation: IFNs and PFNs extend classical fuzzy set theory by introducing additional parameters to represent uncertainty. However, despite their distinct membership constraints, the distance measures used for IFNs and PFNs are ultimately computed in the real-number domain. The commonly used metrics include the following.

1) ED for IFNs and PFNs:

$$d_{\text{IFN/PFN}}(A, B) = \sqrt{\frac{1}{2}\left[(\mu_A - \mu_B)^2 + (\nu_A - \nu_B)^2\right]}. \quad (14)$$

2) Hamming distance for IFNs and PFNs:

$$d_{\text{IFN/PFN}}(A, B) = \frac{1}{2}\left[|\mu_A - \mu_B| + |\nu_A - \nu_B|\right]. \quad (15)$$

Although IFN and PFN frameworks impose constraints on their membership and nonmembership values—$\mu + \nu \leq 1$ for IFNs and $\mu^2 + \nu^2 \leq 1$ for PFNs—their distance computations are fundamentally real-number operations. The constraints are enforced post hoc rather than being inherently embedded in the arithmetic structure.

c) OPNs-valued distance: a fully closed computation system: In contrast, the OPNs-valued distance measure operates entirely within the OPNs framework, ensuring that all intermediate calculations strictly adhere to OPNs arithmetic rules. Unlike IFN and PFN metrics, which borrow real-number operations to compute distance, OPNs define a distance measure that remains closed within the OPNs domain. Every mathematical operation—addition, subtraction, multiplication, and

division—is performed using OPNs arithmetic, preserving the intrinsic properties of OPNs at each computational step.

Furthermore, while the mathematical form of the OPNs-valued distance measure resembles classical real-number metrics, its underlying structure is fundamentally different. The OPNs distance function satisfies the properties of a generalized metric, which extends beyond conventional real-number distances in the following ways.

1) Ordered domain generalization: Unlike classical real-number metrics, a generalized metric can be defined over any ordered domain, and in this case, it is specifically tailored to OPNs arithmetic.
2) Structural consistency: Each operation within the OPNs distance function follows the algebraic rules of OPNs, ensuring that computed distances remain within the OPNs framework.
3) Inverse operations and closure: OPNs support full arithmetic closure, including subtraction and division, which are well-defined within the OPNs system, whereas IFNs and PFNs lack this closure.

These distinctions establish the OPNs-valued distance measure as a mathematically rigorous and internally consistent framework for defining similarity relationships. The following section formally defines the OPNs-valued distance measure and provides a proof that it satisfies the properties of a generalized metric.

2) Definition and Properties of OPNs Generalized Metric: Let $(G, +, \cdot, <)$ be any ordered domain, and N be a nonempty set. If the function $d : N \times N \to G^+ \cup \{0\}$ is a generalized metric, satisfying the following conditions.

1) Identity: Only when $x = y, d(x, y) = 0$.
2) Symmetry: $d(x, y) = d(y, x)$.
3) Triangle inequality: $d(x, y) + d(y, z) \geq d(x, z)$.

Here, the function d is called a generalized metric on the set N.

Next, let us define a generalized metric on the set of OPNs. Analogous to the absolute value in real numbers, for any OPNs $\tilde{\alpha}$, its absolute value $|\tilde{\alpha}|$ is defined as follows:

$$|\tilde{\alpha}| = \begin{cases} \tilde{\alpha}, & \text{if } \tilde{\alpha} \geq \tilde{0} \\ -\tilde{\alpha}, & \text{if } \tilde{\alpha} < \tilde{0}. \end{cases} \tag{16}$$

Although it may seem similar to the corresponding concept in real numbers, it is fundamentally different. The operation of absolute value in OPNs theory is used for OPNs operations, and its result is also an OPNs. For any two OPNs $\tilde{\alpha}$ and $\tilde{\beta}$, the distance between them can be denoted by $d_{\mathrm{OPNs}}(\tilde{\alpha}, \tilde{\beta})$, defined as follows:

$$d_{\mathrm{OPNs}}(\tilde{\alpha}, \tilde{\beta}) = |\tilde{\alpha} - \tilde{\beta}|. \tag{17}$$

This form may appear similar to the corresponding concept in real numbers, but it is fundamentally different. The result of the operation $d_{\mathrm{OPNs}}(a, b)$ is an OPNs. We can define an n-dimensional generalized metric.

Definition 1: For any two n-dimensional OPNs vectors $\tilde{\mathbf{D}}_1$ and $\tilde{\mathbf{D}}_2$, where $\tilde{\mathbf{D}}_1 = \left(\tilde{d}_1^{(1)}, \tilde{d}_2^{(1)}, \ldots, \tilde{d}_n^{(1)}\right)^{\mathrm{T}}$, and $\tilde{\mathbf{D}}_2 = \left(\tilde{d}_1^{(2)}, \tilde{d}_2^{(2)}, \ldots, \tilde{d}_n^{(2)}\right)^{\mathrm{T}}$. The generalized measure of OPNs values between $\tilde{\mathbf{D}}_1$ and $\tilde{\mathbf{D}}_2$ is defined as follows:

$$d_{\mathrm{OPNs}}\left(\tilde{\mathbf{D}}_1,\ \tilde{\mathbf{D}}_2\right) = \left(\sum_{i=1}^{n}\left(\tilde{d}_i^{(1)} - \tilde{d}_i^{(2)}\right)^2\right)^{\frac{1}{2}}. \tag{18}$$

Theorem 1: d_{OPNs} is a generalized measure on the set of OPNs.

Proof: The following proof process is almost identical to the proof process for real number measures. This is because the operation form and properties of OPNs have a high degree of similarity to those of real numbers, and all the nontrivial parts of this proof have already been demonstrated in OPNs theory [6].

1) Identity: If and only if $\tilde{\mathbf{D}}_1 = \tilde{\mathbf{D}}_2$, then

$$\begin{aligned} d_{\mathrm{OPNs}}\left(\tilde{\mathbf{D}}_1, \tilde{\mathbf{D}}_1\right) &= \left(\sum_{i=1}^{n}\left(\tilde{d}_i^{(1)} - \tilde{d}_i^{(1)}\right)^2\right)^{\frac{1}{2}} \\ &= \left(\sum_{i=1}^{n}\left(\tilde{0}\right)^2\right)^{\frac{1}{2}} \\ &= \tilde{0}. \end{aligned} \tag{19}$$

2) Symmetry: For any two n-dimensional OPNs vectors $\tilde{\mathbf{D}}_1$ and $\tilde{\mathbf{D}}_2$,

$$\begin{aligned} d_{\mathrm{OPNs}}\left(\tilde{\mathbf{D}}_1, \tilde{\mathbf{D}}_2\right) &= \left(\sum_{i=1}^{n}\left(\tilde{d}_i^{(1)} - \tilde{d}_i^{(2)}\right)^2\right)^{\frac{1}{2}} \\ &= \left(\sum_{i=1}^{n}\left(\tilde{d}_i^{(2)} - \tilde{d}_i^{(1)}\right)^2\right)^{\frac{1}{2}} \\ &= d_{\mathrm{OPNs}}\left(\tilde{\mathbf{D}}_2, \tilde{\mathbf{D}}_1\right). \end{aligned} \tag{20}$$

3) Triangle inequality: For any three n-dimensional OPNs vectors $\tilde{\mathbf{D}}_1$, $\tilde{\mathbf{D}}_2$, and $\tilde{\mathbf{D}}_3$,

$$d_{\mathrm{OPNs}}\left(\tilde{\mathbf{D}}_1, \tilde{\mathbf{D}}_2\right) \leq d_{\mathrm{OPNs}}\left(\tilde{\mathbf{D}}_1, \tilde{\mathbf{D}}_3\right) + d_{\mathrm{OPNs}}\left(\tilde{\mathbf{D}}_2, \tilde{\mathbf{D}}_3\right). \tag{21}$$

Based on the Cauchy-Schwarz inequality in OPNs theory, we can demonstrate that the Minkowski inequality also holds within OPNs theory. Specifically, assuming OPNs $\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_k, \tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_k \geq \tilde{0}$, the Minkowski inequality in OPNs theory can be represented as follows:

$$\left(\sum_{k=1}^{n}\left(\tilde{a}_k + \tilde{b}_k\right)^2\right)^{\frac{1}{2}} \leq \left(\sum_{k=1}^{n}\tilde{a}_k^2\right)^{\frac{1}{2}} + \left(\sum_{k=1}^{n}\tilde{b}_k^2\right)^{\frac{1}{2}}. \tag{22}$$

The specific proof is as follows:

$$\begin{aligned} \sum_{k=1}^{n}\left(\tilde{a}_k + \tilde{b}_k\right)^2 &= \sum_{k=1}^{n}\left(\tilde{a}_k^2 + 2\tilde{a}_k\tilde{b}_k + \tilde{b}_k^2\right) \\ &\leq \sum_{k=1}^{n}\tilde{a}_k^2 + 2\left(\sum_{k=1}^{n}\tilde{a}_k^2\right)^{\frac{1}{2}}\left(\sum_{k=1}^{n}\tilde{b}_k^2\right)^{\frac{1}{2}} + \sum_{k=1}^{n}\tilde{b}_k^2 \end{aligned}$$

$$= \left(\left(\sum_{k=1}^{n} \tilde{a}_k^2 \right)^{\frac{1}{2}} + \left(\sum_{k=1}^{n} \tilde{b}_k^2 \right)^{\frac{1}{2}} \right)^2. \quad (23)$$

Therefore, the Minkowski inequality holds in OPNs theory. Next, we utilize the Minkowski inequality to prove the triangle inequality:

$$\begin{aligned} & d_{\mathrm{OPNs}}\left(\tilde{\mathbf{D}}_1, \tilde{\mathbf{D}}_2\right) + d_{\mathrm{OPNs}}\left(\tilde{\mathbf{D}}_2, \tilde{\mathbf{D}}_3\right) \\ &= \left(\sum_{i=1}^{n} \left(\tilde{\alpha}_i - \tilde{\beta}_i\right)^2 \right)^{\frac{1}{2}} + \left(\sum_{i=1}^{n} \left(\tilde{\beta}_i - \tilde{\gamma}_i\right)^2 \right)^{\frac{1}{2}} \\ &\geq \left(\sum_{i=1}^{n} \left(\tilde{\alpha}_i - \tilde{\gamma}_i\right)^2 \right)^{\frac{1}{2}} \\ &= d_{\mathrm{OPNs}}\left(\tilde{\mathbf{D}}_1, \tilde{\mathbf{D}}_3\right). \end{aligned} \quad (24)$$

■

III. Construction of the OPNs-kNN Algorithm

A. Feature Space Mapping

First, we define two key concepts: feature pairing and pairing combinations. Feature pairing within the OPNs framework involves combining features to form OPNs features, where each feature is virtually mapped by the function φ. Consider a dataset $\mathcal{D}$ with m features. Randomly pairing two features forms a pair $(\mathbf{x}_p, \mathbf{x}_k)$, where $1 \leq p, k \leq m$ and $p \neq k$. This process, known as feature pairing, enables flexible construction of OPNs features by allowing any two distinct features to be paired.

For a general dataset $\mathcal{D}$ with m samples and n features, constructing the OPNs feature sequence (also referred to as OPNs feature combinations) $\tilde{\mathbf{x}}^{(i)}$ for the i-th sample requires generating a sequence of OPNs features. Structurally, this is represented as: $\tilde{\mathbf{x}}^{(i)} = \left(\tilde{x}_1^{(i)}, \tilde{x}_2^{(i)}, \ldots, \tilde{x}_n^{(i)}\right)$, where $\tilde{x}_j^{(i)}$ represents the j-th OPNs feature of the i-th sample. This specific sequence, $\tilde{\mathbf{x}}^{(i)}$, is referred to as an OPNs feature combination, or simply a combination. By applying the same pairing method to each sample in the original dataset, an OPNs space mapping operation is completed.

Due to the ordered nature of OPNs, the mapping process increases the feature dimensionality from n to $P(n, 2)$, where $P(n, 2)$ represents the number of permutations of n. This transformation enriches the feature information, enabling the model to better capture relationships between features. However, since feature pairing may introduce redundancy, feature selection is essential to retain only the combinations that enhance classification performance. We will illustrate this with a concrete example.

Assume a dataset $\mathcal{D}$ contains four features: $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, and $\mathbf{x}_4$. The i-th sample is represented as $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)}]$. Let us first consider the case in the conventional real number domain. For a sample $\mathbf{x}'^{(j)}$ in the test set, the ED between this test sample and $\mathbf{x}^{(i)}$ is calculated as follows:

$$\sqrt{\sum_{k=1}^{4} \left(x_k^{(i)} - x_k'^{(j)}\right)^2}. \quad (25)$$

Now, let us consider the same scenario in the OPNs environment, where adjacent features are paired. The i-th sample becomes $\tilde{\mathbf{x}}^{(i)} = \left[\tilde{x}_1^{(i)}, \tilde{x}_2^{(i)}\right] = \left[(x_1^{(i)}, x_2^{(i)}), (x_3^{(i)}, x_4^{(i)})\right]$. Using (18), the distance between the test sample $\tilde{\mathbf{x}}'^{(j)}$ and the i-th sample is as follows:

$$\sqrt{\sum_{k=1}^{2} \left(\tilde{x}_k^{(i)} - \tilde{x}_k'^{(j)}\right)^2}. \quad (26)$$

After applying OPNs calculation rules, we get the OPNs distance $d_{\mathrm{OPNs}}(\tilde{\mathbf{x}}^{(i)}, \tilde{\mathbf{x}}'^{(j)})$ as in (27), shown at the bottom of the page.

It can be observed that the second term of this OPNs distance is identical to the ED in the real number domain, while the first term introduces additional information.

The second term $d_{\mathrm{OPNs}}\left(\tilde{\mathbf{x}}^{(i)}, \tilde{\mathbf{x}}'^{(j)}\right)$ depends only on the number of times a feature appears in a paired combination and is independent of how the features are paired. For example, the pairing combinations $[(\mathbf{x}_1, \mathbf{x}_2), (\mathbf{x}_3, \mathbf{x}_4)]$ and $[(\mathbf{x}_1, \mathbf{x}_3), (\mathbf{x}_2, \mathbf{x}_4)]$ yield the same second term in the OPNs distance. The difference between these pairings lies in the first term, where the cross-products of different feature pairs form different combinations.

In practice, using more OPNs features in training does not necessarily lead to better classification results. Selecting too many OPNs features can degrade performance by introducing redundant information. For instance, when pairing combinations are $[(\mathbf{x}_1, \mathbf{x}_2), (\mathbf{x}_3, \mathbf{x}_4), (\mathbf{x}_1, \mathbf{x}_3)]$, features $\mathbf{x}_1$ and $\mathbf{x}_3$ are repeated. As a result, the cross-product terms in the first term will include an additional term $\left(x_1^{(i)} - x_1'^{(j)}\right)\left(x_3^{(i)} - x_3'^{(j)}\right)$, and the coefficient of $\left(x_k^{(i)} - x_k'^{(j)}\right)^2$ in the second term will be 2, where $k = 1, 3$. The new information mainly affects the first term, while the changes in coefficients in the second term have minimal impact on the final calculation.

When introducing new information, it is important to maintain a balance between feature pairings. Some pairings may not be relevant, and utilizing irrelevant information can degrade model performance. Therefore, the goal is to add as much useful information as possible while minimizing redundant or irrelevant information.

$$\sqrt{\left(-2\left[\sum_{m=1}^{2}\left(x_{2m-1}^{(i)} - x_{2m-1}'^{(j)}\right)\left(x_{2m}^{(i)} - x_{2m}'^{(j)}\right)\right], -\sum_{k=1}^{4}\left(x_k^{(i)} - x_k'^{(j)}\right)^2\right)}. \quad (27)$$

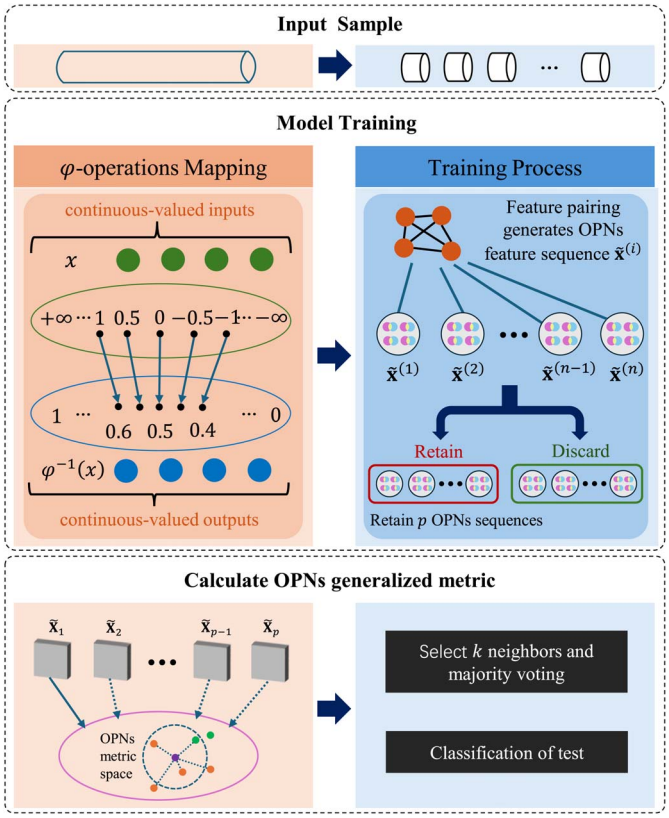

Fig. 2. OPNs-kNN algorithm structure.

B. Model Training Process

The implementation and method of the OPNs-kNN algorithm are detailed below, with a focus on explaining the algorithm's training process.

In the proposed algorithm (as illustrated in Fig. 2), the raw data are divided into multiple training and testing instances. The φ-operation function is applied to map real-valued data into the interval $(0, 1)$. Next, feature pairing is performed to construct multiple OPNs feature sequences, and a filtering method is used to reduce the search space of OPNs feature sequences. A subset of retained feature pairings is then iteratively selected to construct OPNs-based training and testing samples. Subsequently, the distance between each test sample and all training samples is computed within the OPNs-valued generalized metric space. The algorithm identifies the k nearest neighbors of the test sample and assigns it to the class with the highest number of votes among its neighbors. This process is repeated while recording the average classification performance in each iteration to determine the optimal pairing combination and its corresponding classification accuracy.

Although the φ mapping is crucial for preserving the integrity of OPNs theory, practical machine learning applications suggest that strictly following these mapping steps may not always be necessary. Precision loss and the fact that machine learning results do not need to be confined to the interval $(0, 1)$ make this clear. Therefore, preprocessing can be simplified by assuming the data have undergone a virtual φ mapping, avoiding the need for direct mapping and inverse operations. This simplifies the algorithm, reducing complexity while still leveraging the full advantages of OPNs for model development. In the following sections, we will discuss model construction and training within this streamlined framework.

The overall training process of the OPNs-kNN algorithm is shown in Fig. 3. The first step involves pairing features and selecting the most effective combinations. Since the correlations between features vary, it is important to retain only the strongly correlated pairs, which can be done using a filter. The filter ranks all OPNs pair combinations based on their correlations and removes weaker ones. A threshold value δ is set in the filter, and if the proportion of feature pairs with relatively strong correlation in a combination exceeds δ, the combination is kept; otherwise, it is discarded.

Next, the original dataset is mapped into the OPNs space according to the selected feature pairing method. In this space, the distance between a test sample and each known sample is calculated using the OPNs generalized metric. The known samples are vectorized, and each component of the OPNs vector is observed as a 2-D point in a Cartesian coordinate system. Distances are compared using the OPNs order relation.

When the feature pairing method changes, the OPNs features and the distribution of sample points in the OPNs space also change. There are two major ways in which this variation can occur, depending on the influence of the changes on the sample points. For instance, consider a sample with four features x_1, x_2, x_3, x_4, with an initial OPNs coordinate distribution of $\tilde{A}_0 = ((x_1, x_3), (x_2, x_4))$.

In the first case, after one feature pairing change, the point shifts to $\tilde{A}_1 = ((x_1, x_4), (x_2, x_3))$, which corresponds to a rotation by an angle θ relative to the OPNs coordinate system origin. In the second case, the point changes to $\tilde{A}_2 = ((x_2, x_1), (x_4, x_3))$, representing a more irregular transformation that cannot be explained by simple rotation. This transformation is analogous to randomly shuffling the components of a point in a high-dimensional real space, causing its position to change unpredictably.

The goal of changing the feature pairing method is to find a distribution in the OPNs space that maximally separates the categories. This process forms the iterative training of OPNs feature pairing.

C. OPNs Feature Pairing Theory

The number of OPNs feature pairings grows factorially as the original dataset increases. At the same time, classification performance must be calculated for different values of k with each pairing combination, leading to a significant increase in time complexity. A key aspect of OPNs pairing theory is addressing this issue by identifying the optimal or near-optimal solutions among the many pairing combinations. The following section introduces a method to reduce the search space for pairings in the OPNs-kNN algorithm.

1) Dimensionality Reduction of OPNs Features: After mapping, we obtained $P(n, 2)$ OPNs features. Since $P(n, 2)$ grows quadratically with n at a rate of $O(n^2)$, the explosion

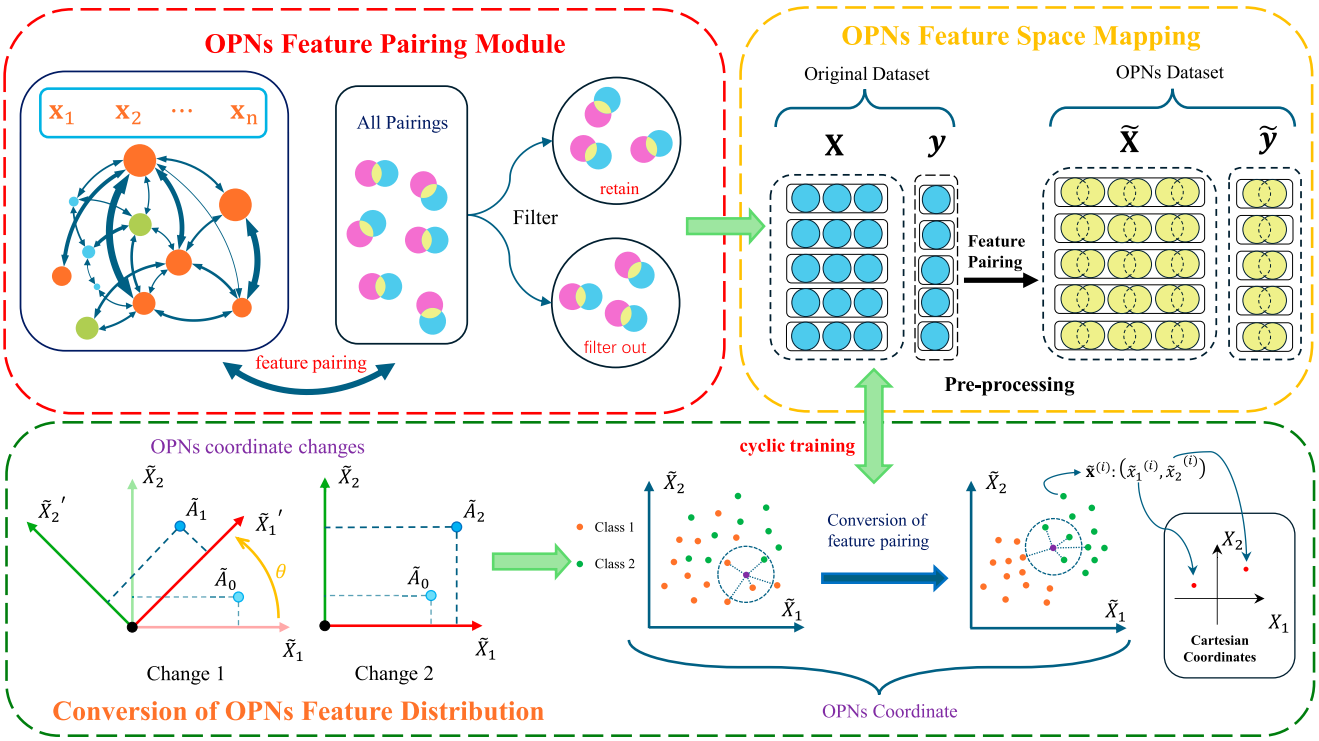

Fig. 3. OPNs-kNN training process.

in the number of OPNs features makes exhaustive enumeration experimentally unfeasible. To ensure the feasibility of the experiment, we currently use a heuristic strategy combined with the analysis from Section A to determine the number of OPNs features to select. That is, for n features, $n/2$ OPNs features are selected, ensuring that the selected OPNs features do not repeat real features, and these $n/2$ OPNs feature combinations are input into the training process each time. This is a common heuristic method, typically used to reduce the search space in cases of combinatorial explosion. This strategy [17], a form of incomplete enumeration, can save a significant amount of computational resources in practice.

In this strategy, there are unordered and nonrepetitive combination methods. Specifically, if D_i has n features, where n is even, then two features are selected to form one OPNs feature, and then two more are chosen from the remaining features until no more features are left. The total number of combinations for this process can be expressed as $\prod_{k=1}^{n/2}(n-2k+1)$. For example, $[(d_1,\ d_2),(d_3,\ d_4),\ \ldots,\ (d_{n-1},\ d_j)]$ is one of the unordered and nonrepetitive combinations.

Additionally, considering some datasets have an odd number of features, according to the above strategy, the combination methods are varied, and the following are the complete combination steps.

Step 1. Determine the number of features n and check if n is odd:

If n is odd, add a column of feature d_0 to the dataset, where the feature value of d_0 is assigned to 0 for all.

$$d_0 = \{0,\ 0,\ \ldots,\ 0\},$$

$$D_i = (d_1,\ d_2,\ \ldots,\ d_n, d_0),$$
$$n = n+1.$$

Step 2. When n is even, combine features in pairs:

Suppose S is a set containing n elements, and we define O as the set of all possible pairings. The recursive formula is as follows: select two elements d_k and d_l from the set S as a pair, denoted as (d_k, d_l). Recursively generate all pairings from the remaining set $S\backslash\{d_k, d_l\}$. Mathematically, this can be expressed by the following recursive formula:

$$O = \bigcup_{d_k, d_l \in S} \{(d_k, d_l)\} \times Pair\,(S\backslash\{a, b\}) \tag{28}$$

where $\times$ denotes the Cartesian product of the pair combinations.

Step 3. Store all combination methods, saved in set O:

The set $O = \{O_1,\ O_2,\ \ldots,\ O_N\}$. $N = \prod_{k=1}^{n/2}(n-2k+1)$, represents a total of N combination methods.

2) Linear Filtering of OPNs Combinations: To further reduce the search space, linear filtering is applied to the set O based on the total order property of OPNs. Using linear fitting's correlation information and a threshold, OPNs combinations with significant correlations between sample categories are selected to reduce the search space, making the selected OPNs more likely to have a significant impact on the classification problem. Combining statistical linear fitting and threshold setting, irrelevant feature combinations are filtered to improve the efficiency and performance of the model.

Step 1. Calculate correlation coefficients and linear fitting parameters:

Iterate through each combination method O_i with $n/2$ OPNs features in set O. For each pairing in O_i, calculate the correlation coefficient ρ or the linear fitting slope m for each pair of features by sample category. In the following discussion, we mainly use ρ as an example. For a dataset with c classes, c values of ρ need to be calculated for each pair of features.

Step 2. Threshold setting and marking:

If at least one of the c correlation coefficients for a pair of features exceeds the threshold η, the OPNs feature is marked. Alternatively, slope m can be used, and η needs to be reset.

Step 3. Count marked numbers and filter:

Count the number of marked OPNs q in each O_i. Filter based on the proportion ω of q to the number of OPNs $n/2$. If ω exceeds the set threshold δ, keep O_i; otherwise, filter O_i out.

It is important to note that the feature pairing method is dataset-dependent. The approach mentioned here is mainly applicable to OPNs-kNN, while different OPNs-based algorithms may require different methods to find the optimal or near-optimal pairings. As research progresses, OPNs feature pairing theory will continue to evolve and improve.

3) Training Optimal OPNs Combinations: After filtering the set O, preprocess each O_i and input it into the algorithm training process, eventually obtaining combinations with superior classification effects and results.

Step 1. Calculate OPNs-valued generalized metric:

Calculate the OPNs-valued generalized metric between the test sample $\tilde{\mathbf{x}}$ and each sample $\tilde{\mathbf{x}}_i$ in the training set.

Step 2. Select neighbors:

Sort the generalized metric of OPNs-valued using the total order rule of OPNs, and select the top k closest training samples.

Step 3. Majority voting:

Count the occurrence of each category among these k samples, and assign the most frequent category to the test sample as the predicted result y.

D. OPNs kNNs Algorithm

Algorithm 1 represents a typical kNN algorithm based on the OPNs framework, incorporating all the key steps mentioned earlier and presenting them in a simplified manner. As seen, OPNs-kNN introduces two additional threshold parameters, η and δ, compared with traditional kNN. These thresholds mainly pertain to Phase 2: selecting and filtering OPNs feature combinations. This phase is not strictly mandatory. Its primary purpose is to reduce the search space generated by feature pairings and combinations, thereby saving computational time. If the input dataset has few features or sufficient computational resources are available, this phase can even be omitted, and the threshold parameters need not be set. The proposed algorithm provides an effective way to reduce the search space, but adjustments can be made in practice. For instance, instead of calculating the correlation coefficient for each pairing by class, it can be computed directly for all pairings without classification. Additionally, the correlation coefficient can be alternated with the slope as needed.

Algorithm 1: OPNs K-Nearest Neighbors Algorithm.

Input: Training set $D = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ with m samples; Query $\mathbf{x}$; Maximum neighbors k_{max}; Correlation coefficient threshold η; Proportion threshold δ.
Output: Predicted label y for query $\mathbf{x}$; Optimal combination O^*.
Phase 1: Feature Space Mapping
1.1 Generate OPNs feature pairs:

$$OPNsSets = \{(\mathbf{x}_k, \mathbf{x}_l) | 1 \le k < l \le n\},$$

where n is the number of features in the original dataset.
1.2 Compute the correlation coefficients ρ between the two features of OPNs pairs for each category in $OPNsSets$.
Phase 2: Select and Filter OPNs Feature Combinations
2.1 If n is odd, append a zero-valued feature d_0 to make n even.
2.2 Obtain a set of all pairs where each feature appears exactly once. The i-th combination $O_i = \left(\tilde{d}_1^{(i)}, \tilde{d}_2^{(i)}, \ldots, \tilde{d}_{n/2}^{(i)}\right)$.
2.3 Store all selection methods in set $O = \{O_1, O_2, \ldots, O_N\}$ and N represents the total number of combinations.
2.4 Filter combinations using correlation coefficients:
for each O_i in O **do**
  for each ρ of every category of $\tilde{d}^{(i)}$ **do**
    if At least one category of ρ exceeds η **then**
      Mark $\tilde{d}_j^{(i)}, 1 \le j \le n/2$
    end if
    $\omega = \frac{q}{n/2}$, where q is the number of marked items
  end for
  if ρ exceeds threshold δ **then**
    Retain O_i
  end if
end for
Phase 3: Train and Evaluate OPNs Combinations
for O_i in retained $O^{'}$ **do**
  for $k = 1$ to $k_{\max}$ **do**
    3.1 Apply pairing methods to the dataset: $\tilde{D} \overset{O_i}{\leftarrow} D$, $\tilde{\mathbf{x}} \overset{O_i}{\leftarrow} \mathbf{x}$
    3.2 Compute the OPNs generalized metric d_{OPNs}:
    for $\tilde{\mathbf{x}}_i$ in $\tilde{D}$ **do**
      Calculate d_{OPNs} between the test sample $\tilde{\mathbf{x}}$ and each training sample $\tilde{\mathbf{x}}_i$:

$$d_{\mathrm{OPNs}}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}_i) = \left(\sum_{j=1}^{n/2} \left(\tilde{d}_j^{(\tilde{\mathbf{x}})} - \tilde{d}_j^{(\tilde{\mathbf{x}}_i)}\right)^2\right)^{\frac{1}{2}},$$

$$S = \{d_{\mathrm{OPNs}}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}_i) \mid i = 1, 2, \ldots, m\}$$

    end for
    3.3 Sort and select the top k closest training samples:

$$(y_1, y_2, \ldots, y_k) \leftarrow \mathrm{sort}(S)$$

    3.4 Perform majority voting among the k selected samples and assign the predicted label y_{O_i} for the combination O_i:

$$y_{O_i} \leftarrow \text{Majority Vote}(y_1, y_2, \ldots, y_k)$$

  end for
  Compare classification accuracy for each O_i
end for
Return the optimal combination O^* and final prediction y

IV. Experiments and Results

This section mainly verifies the performance of OPNs-kNN through experiments and explains the impact of parameter selection on the algorithm. It compares OPNs-kNN with kNN

TABLE I
DATASETS IN UCI

Dataset	Size	Features	Classes
Iris(iris)	150	4	3
Wine(wine)	178	13	3
Balance Scale(balance)	625	4	3
Appendicites(appendicites)	107	7	2
Liver Disorders(bupa)	345	6	2
Heart Disease(heart)	270	13	2
Glass Identification(glass)	214	9	6
Seeds(seeds)	210	7	3
Hayes-Roth(hayes)	160	4	3
MONK's Problems(monk)	432	6	2
Rice(rice)	3810	7	2

using various traditional metrics and various improved kNN algorithms, showing that OPNs-kNN, built under the OPNs theoretical framework, outperforms other algorithms in performance on experimental datasets.

A. Evaluation Methods and Dataset Description

For the evaluation of classification performance, we adopt accuracy, precision, recall, and F1-score as the primary metrics. Higher values of these metrics indicate better classification performance. The definitions of these metrics are as follows.

Accuracy: The proportion of correctly classified instances among all instances

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{29}$$

Precision: The proportion of correctly predicted positive instances among all predicted positive instances

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{30}$$

Recall: The proportion of correctly predicted positive instances among all actual positive instances

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{31}$$

F1-score: The harmonic mean of precision and recall

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{32}$$

In the experiments, evaluation metrics were computed using functions from the scikit-learn library. Since the class distribution in each dataset was relatively balanced, the "average" parameter was set to "macro" when calculating precision, recall, and F1-score. Multiple public datasets were used in the experiments, and their detailed information is presented in Table I.

To ensure a fair and robust evaluation of the proposed OPNs-kNN algorithm and its variants, we adopt a repeated ten-fold cross-validation protocol. The dataset is randomly partitioned into ten equal-sized subsets. In each iteration, one subset serves as the test set, while the remaining nine subsets constitute the training set. This process is repeated for ten independent runs, generating 100 total evaluations (10 folds $\times$ 10 runs). Final performance metrics are computed by averaging all results to ensure robustness against random partitioning effects. This design ensures that the reported metrics remain stable and reliable, minimizing the influence of random splits and avoiding biases introduced by specific partitions.

The ten-fold cross-validation strategy is justified by three key principles.

1) Stability enhancement: Mitigates variance in accuracy/F1-score metrics through multi-fold averaging, particularly critical for distance-sensitive kNN variants.
2) Partition agnosticism: Ensures all data instances participate in both training and testing phases, addressing kNN's sensitivity to local data distributions.
3) Computational pragmatism: Aligns with machine learning conventions where ten-fold partitioning optimally balances statistical reliability ($\sim$90% training data utilization) and computational overhead.

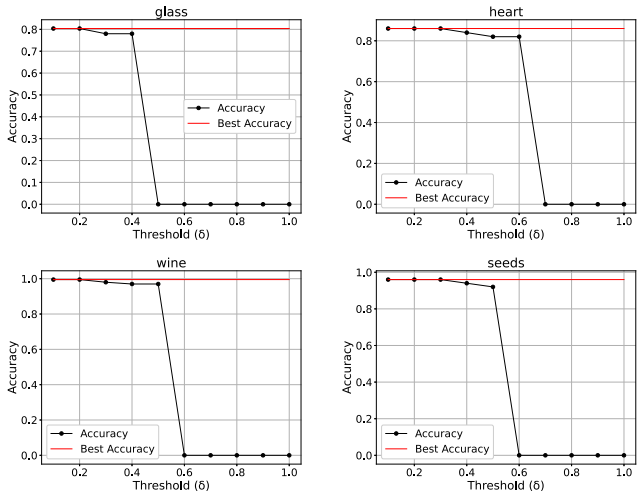

Fig. 4. Impact of threshold δ on OPNs-kNN.

B. Impact of Parameter δ on Model Performance

For the OPNs-kNN algorithm, the primary parameters that need to be set are k, η, and δ. All of these parameters have a significant impact on the model's performance. In this experiment, η and δ represent the correlation coefficient threshold and the ratio threshold, respectively. Both parameters influence the search space in a similar way: the larger their values, the smaller the search space becomes. Therefore, in this experiment, we focus primarily on observing and discussing the effect of δ on the model's performance. The following section will illustrate how this parameter affects the results.

δ is a proportion threshold that primarily impacts the algorithm's time complexity. An appropriate threshold ensures that the algorithm identifies the optimal OPNs feature pairing combination within a reduced search space, thereby improving the efficiency of the OPNs-kNN algorithm. In practice, δ is typically set between 0.2 and 0.4 and is used to filter OPNs feature sequences based on the proportion of linearly correlated feature pairs. Specifically, δ is compared with a ratio ω, defined as the number of correlated feature pairs divided by the total number of feature pairs in an OPNs feature sequence. If $\omega > \delta$,

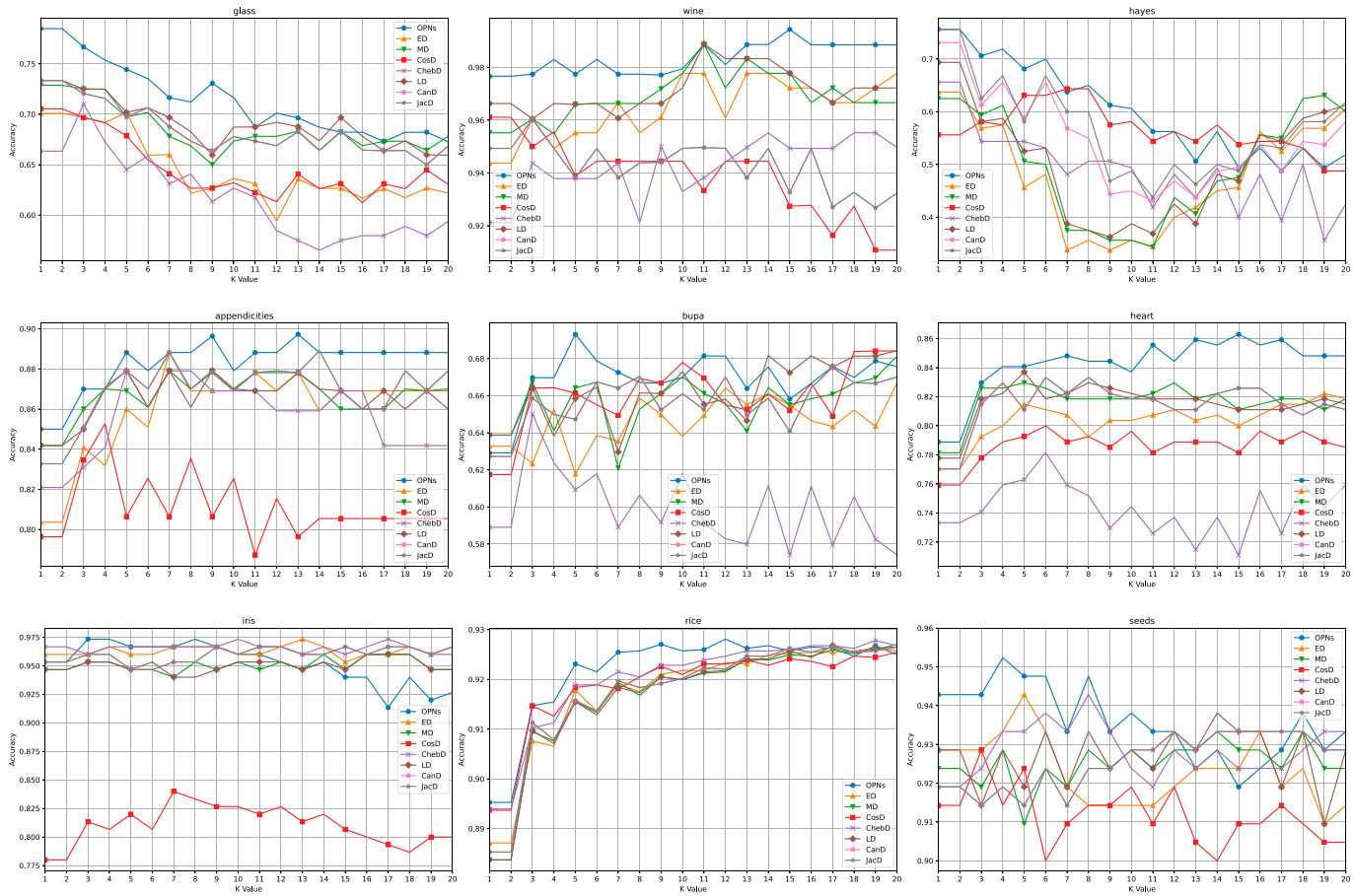

Fig. 5. Comparison of classification accuracy between OPNs-kNN with various k values and seven kNN algorithms using traditional distance measures on different datasets.

the feature sequence O_i is retained for training and testing; otherwise, it is discarded.

A lower δ value leads to more retained feature sequences, which increases the number of training iterations and computational cost. However, it also enhances the likelihood of identifying an optimal OPNs feature pairing combination. Conversely, a higher δ value decreases the number of retained feature sequences, reducing computational cost but potentially restricting the search space for optimal pairings.

As shown in Fig. 4, when the threshold δ is less than 0.2, that is, in all OPNs feature pairing combinations where the OPNs with linear relationships account for less than 0.2 of the total number of OPNs, it is possible to find an optimal or suboptimal OPNs feature pairing combination. When the threshold δ is greater than 0.6, the search space for filtered OPNs feature pairing combinations is empty.

C. Comparison Across Multiple Performance Metrics

Fig. 5 compares the classification accuracy of the OPNs-kNN algorithm with seven kNN algorithms using traditional distance measures across various datasets at different k values. These traditional distances include ED, MD, Cosine distance (CosD), Chebyshev distance (ChebD), natural logarithm of absolute difference (LD), Canberra distance (CanD), and Jaccard dissimilarity (JacD) [16]. In some cases, CanD-kNN and JacD-kNN performed identically, resulting in overlapping visualizations. Overall, OPNs-kNN consistently outperforms the other seven kNN variants in both peak and average classification accuracy across most datasets. Like all traditional kNN algorithms, OPNs-kNN is influenced by the k value. Although the average classification accuracy of OPNs-kNN at certain k values may not always be the highest, the algorithm consistently achieves superior classification outcomes compared to others, benefiting from its feature pairing advantage, as observed in the *iris* and *seeds* datasets.

Fig. 6 presents a performance comparison of various models across different classification evaluation metrics, including precision, recall, and F1-score. The results demonstrate that the OPNs-kNN model outperforms other algorithms on most datasets across multiple metrics, showcasing strong classification capabilities. This demonstrates that OPNs-kNN not only achieves superior classification accuracy but also maintains a robust balance between precision and recall, thereby enhancing overall classification quality. These findings further validate the effectiveness and robustness of the OPNs-kNN model in addressing this classification task.

However, on two datasets, the recall and F1-scores of OPNs-kNN were slightly outperformed by other methods. On the wine

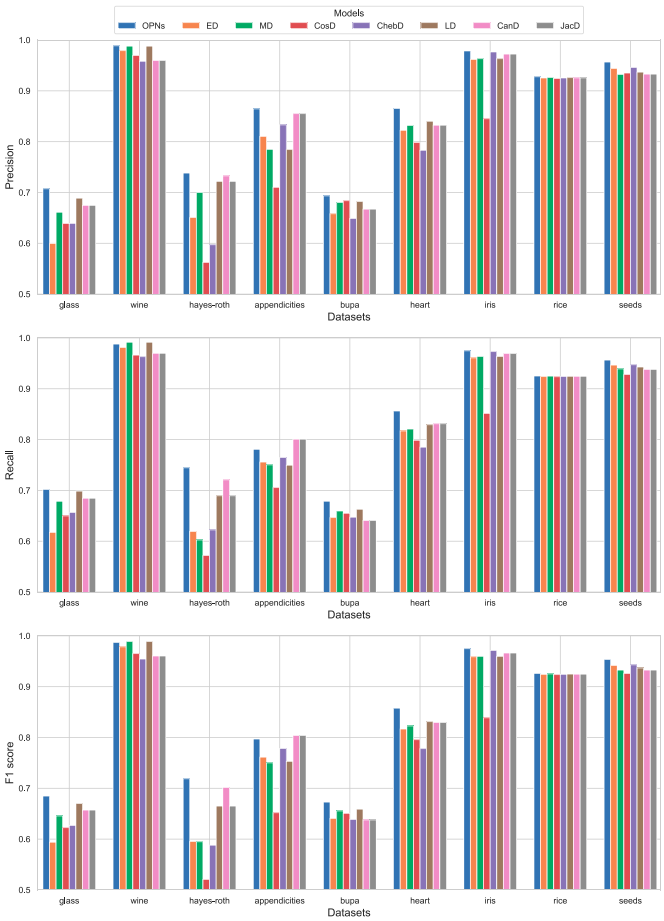

Fig. 6. Comparison of various classification metrics between OPNs-kNN and seven conventional kNNs on different datasets under their respective optimal k values.

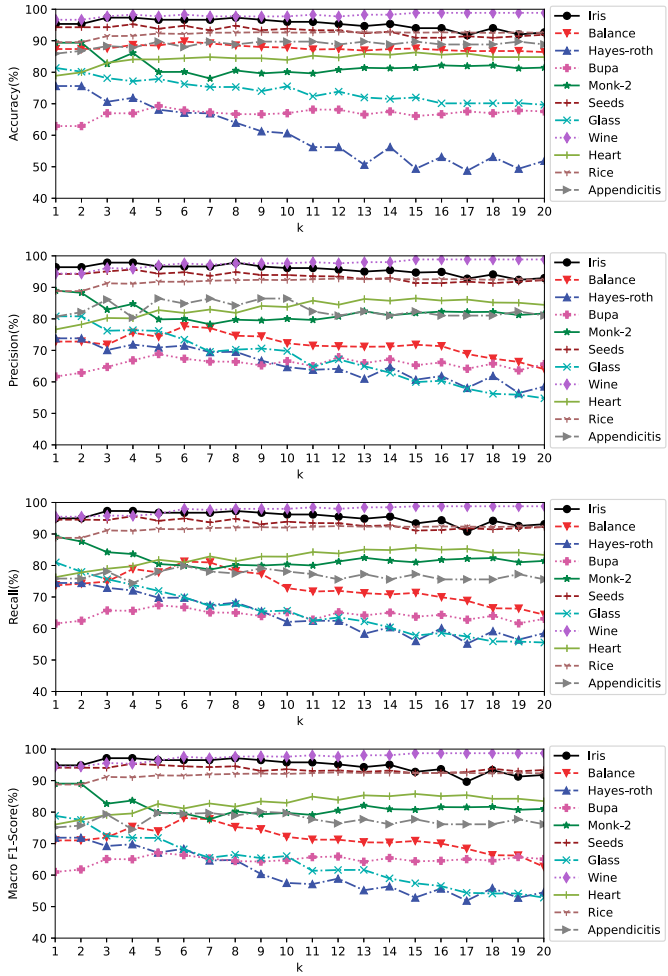

Fig. 7. Performance of the OPNs-kNN algorithm in terms of accuracy, precision, recall, and F1-score as the number of nearest neighbors (k) varies from 1 to 20.

dataset, OPNs-kNN, along with kNN based on MD and LD (both tied for first place), achieved exceptional performance, making further improvements in classification accuracy challenging. On the appendicitis dataset, OPNs-kNN was slightly inferior to kNN based on CanD and JacD (both tied for first place).

D. Graphical Results of Different kNN Values on OPNs-kNN

Fig. 7 presents the performance percentages across various datasets as k varies. The accuracy metric tends to be more stable than other performance indicators. For example, in the *Hayes-roth* and *Glass* datasets, the performance of OPNs-kNN noticeably declines with increasing k, a trend that is evident across all evaluated metrics. In contrast, for the *Wine*, *Iris*, *Seeds*, and *Rice* datasets, the performance remains relatively close for different k values.

Furthermore, aside from the stability observed in the accuracy curves, the results are particularly concentrated when k is in the range of 5–9. However, as k increases beyond 10, the performance curves tend to diverge, indicating a higher sensitivity of the algorithm to larger k values. This sensitivity analysis demonstrates that the optimal k value may vary with the dataset characteristics, and it reinforces the importance of performing a parameter sweep to select an appropriate k for each application.

E. Comparison of OPNs-kNN With Improved kNN Algorithms

To further highlight the advantages of OPNs-KNN, we compare it with several state-of-the-art improved kNN algorithms. The comparison includes eight advanced kNN variants: WkNN [10], DC-LAkNN [14], a two-layer nearest neighbor (kTLNN) classifier [32], fuzzy k-nearest neighbor (FkNN) [19], BM-FkNN [20], FPFS-kNN [21], the picture fuzzy soft k-nearest neighbor (PFSkNN) classifier [22], and ensemble centroid displacement-based k-NN [33]. Their parameter configurations and evaluation metrics are provided in Table II.

The results in Table II indicate that OPNs-kNN outperforms other models on 6 out of 11 datasets when considering all performance metrics. When focusing only on accuracy, OPNs-kNN achieves the highest performance on 7 out of 11 datasets. Similarly, when considering only the F1-score,

TABLE II
COMPARISON OF CLASSIFICATION ACCURACY AT FIXED k VALUE BETWEEN OPNS-KNN AND IMPROVED KNN ON VARIOUS DATASETS (%)

Datasets	Classifiers	Settings	Accuracy ± Std	Precision ± Std	Recall ± Std	F1 ± Std
	WkNN	$k=7$	95.90 ± 5.50	96.10 ± 5.60	96.00 ± 5.40	95.70 ± 5.90
	DC-LAkNN	$k=7$	94.22 ± 1.23	94.25 ± 1.24	94.21 ± 1.25	94.21 ± 1.25
	kTLNN	$k=7$	95.33 ± 5.85	95.53 ± 6.16	95.56 ± 5.59	95.03 ± 6.59
	FkNN	$k=7$	96.47 ± 3.09	96.39 ± 2.45	96.47 ± 3.09	96.23 ± 3.16
Iris	BM-FkNN	$k=7, p=0.5, q=3, m=1.5$	95.80 ± 5.05	96.43 ± 4.49	95.80 ± 5.05	95.74 ± 5.13
	FPFS-kNN	$k=7$, corrname=Pearson	95.73 ± 5.03	96.30 ± 4.61	95.73 ± 5.03	95.68 ± 5.10
	PFS-kNN	$k=7, L=0.5, p=5$	95.93 ± 5.49	96.35 ± 5.23	95.93 ± 5.49	95.86 ± 5.60
	ECDNN	$k=7$	91.88 ± 7.00	89.93 ± 9.92	90.25 ± 9.21	89.13 ± 9.88
	OPNs-kNN	$k=7$	$\mathbf{96.73 \pm 4.37}$	$\mathbf{96.58 \pm 4.82}$	$\mathbf{96.76 \pm 4.72}$	$\mathbf{96.35 \pm 5.23}$
	WkNN	$k=7$	87.90 ± 3.90	60.80 ± 6.70	63.90 ± 2.70	61.90 ± 3.60
	DC-LAkNN	$k=7$	83.68 ± 0.84	60.30 ± 2.17	61.53 ± 1.02	60.50 ± 1.36
	kTLNN	$k=7$	86.72 ± 4.22	60.02 ± 4.49	63.11 ± 2.17	61.25 ± 2.81
	FkNN	$k=7$	89.21 ± 2.15	59.74 ± 1.37	64.54 ± 1.51	61.99 ± 1.42
Balance	BM-FkNN	$k=7, p=0.5, q=3, m=1.5$	87.12 ± 2.75	58.34 ± 1.77	63.02 ± 1.94	60.46 ± 1.89
	FPFS-kNN	$k=7$, corrname=Pearson	89.54 ± 1.30	59.94 ± 1.09	64.92 ± 1.13	62.29 ± 1.09
	PFS-kNN	$k=7, L=0.5, p=5$	84.90 ± 3.06	59.20 ± 6.29	61.88 ± 3.15	60.15 ± 3.70
	ECDNN	$k=7$	86.14 ± 4.17	58.45 ± 5.55	60.97 ± 5.04	59.50 ± 4.97
	OPNs-kNN	$k=7$	$\mathbf{89.75 \pm 2.94}$	$\mathbf{77.84 \pm 7.03}$	$\mathbf{81.62 \pm 9.08}$	$\mathbf{78.23 \pm 7.40}$
	WkNN	$k=7$	$\mathbf{69.60 \pm 11.50}$	69.50 ± 17.40	65.40 ± 13.10	63.90 ± 15.00
	DC-LAkNN	$k=7$	67.70 ± 3.27	75.25 ± 2.99	64.69 ± 2.73	67.30 ± 2.70
	kTLNN	$k=7$	52.12 ± 13.03	58.20 ± 15.41	55.54 ± 14.18	52.62 ± 13.99
	FkNN	$k=7$	68.06 ± 10.00	73.67 ± 14.72	64.28 ± 11.44	64.68 ± 12.76
Hayes-roth	BM-FkNN	$k=7, p=0.5, q=3, m=1.5$	36.38 ± 10.36	47.31 ± 10.15	42.23 ± 11.14	42.96 ± 10.16
	FPFS-kNN	$k=7$, corrname=Pearson	72.88 ± 9.57	$\mathbf{79.82 \pm 10.62}$	$\mathbf{70.58 \pm 11.16}$	$\mathbf{71.59 \pm 11.63}$
	PFS-kNN	$k=7, L=0.5, p=5$	36.06 ± 11.07	46.07 ± 15.58	36.86 ± 12.26	38.01 ± 12.95
	ECDNN	$k=7$	67.32 ± 10.33	63.02 ± 16.99	60.09 ± 12.34	57.93 ± 13.48
	OPNs-kNN	$k=7$	68.44 ± 9.85	69.97 ± 12.24	65.23 ± 11.90	64.76 ± 13.00
	WkNN	$k=7$	63.30 ± 7.50	62.50 ± 8.30	62.10 ± 8.30	61.40 ± 8.10
	DC-LAkNN	$k=7$	63.47 ± 1.25	62.21 ± 1.20	61.60 ± 1.16	61.69 ± 1.20
	kTLNN	$k=7$	59.74 ± 8.66	59.18 ± 9.11	59.22 ± 9.27	58.56 ± 9.03
	FkNN	$k=7$	46.19 ± 7.07	23.10 ± 3.53	50.00 ± 0.00	31.45 ± 3.14
Bupa	BM-FkNN	$k=7, p=0.5, q=3, m=1.5$	63.40 ± 7.40	62.54 ± 8.05	62.00 ± 7.74	61.72 ± 7.96
	FPFS-kNN	$k=7$, corrname=Pearson	61.28 ± 7.16	60.09 ± 8.87	58.43 ± 7.19	57.90 ± 7.71
	PFS-kNN	$k=7, L=0.5, p=5$	60.13 ± 8.03	58.92 ± 8.82	58.24 ± 8.21	57.92 ± 8.44
	ECDNN	$k=7$	59.84 ± 8.80	56.67 ± 11.65	56.21 ± 10.94	55.29 ± 11.11
	OPNs-kNN	$k=7$	$\mathbf{67.83 \pm 7.93}$	$\mathbf{67.20 \pm 8.19}$	$\mathbf{66.27 \pm 8.08}$	$\mathbf{65.77 \pm 8.51}$
	WkNN	$k=7$	88.10 ± 5.10	88.40 ± 5.10	88.00 ± 5.20	87.80 ± 5.20
	DC-LAkNN	$k=7$	94.49 ± 0.99	94.76 ± 0.99	94.33 ± 1.00	94.45 ± 1.00
	kTLNN	$k=7$	87.09 ± 5.41	88.14 ± 5.12	87.56 ± 4.92	86.83 ± 5.51
	FkNN	$k=7$	98.36 ± 2.12	98.41 ± 2.00	98.43 ± 2.04	$\mathbf{98.36 \pm 2.13}$
Monk-2	BM-FkNN	$k=7, p=0.5, q=3, m=1.5$	73.88 ± 3.12	73.12 ± 3.10	73.12 ± 3.10	73.21 ± 3.18
	FPFS-kNN	$k=7$, corrname=Pearson	$\mathbf{99.88 \pm 1.16}$	$\mathbf{99.90 \pm 0.99}$	$\mathbf{99.89 \pm 1.08}$	99.88 ± 1.16
	PFS-kNN	$k=7, L=0.5, p=5$	89.20 ± 4.78	90.17 ± 4.54	88.91 ± 4.89	89.02 ± 4.90
	ECDNN	$k=7$	95.68 ± 3.02	92.51 ± 10.46	92.78 ± 9.95	92.53 ± 10.21
	OPNs-kNN	$k=7$	78.91 ± 6.73	79.00 ± 6.91	78.98 ± 6.80	78.54 ± 6.83
	WkNN	$k=7$	92.90 ± 5.70	93.10 ± 5.60	92.80 ± 5.90	92.20 ± 6.20
	DC-LAkNN	$k=7$	83.96 ± 1.11	43.64 ± 3.12	44.48 ± 3.08	44.02 ± 3.11
	kTLNN	$k=7$	92.67 ± 5.41	92.51 ± 5.71	92.57 ± 5.68	91.79 ± 5.85
	FkNN	$k=7$	89.43 ± 6.08	90.61 ± 5.52	89.43 ± 6.08	89.34 ± 6.08
Seeds	BM-FkNN	$k=7, p=0.5, q=3, m=1.5$	93.24 ± 5.52	93.98 ± 5.12	93.24 ± 5.52	93.16 ± 5.61
	FPFS-kNN	$k=7$, corrname=Pearson	90.10 ± 6.86	91.02 ± 6.42	90.10 ± 6.86	90.03 ± 6.87
	PFS-kNN	$k=7, L=0.5, p=5$	92.67 ± 6.04	93.47 ± 5.58	92.67 ± 6.04	92.61 ± 6.10
	ECDNN	$k=7$	94.21 ± 3.67	92.01 ± 10.74	92.11 ± 9.87	91.62 ± 10.51
	OPNs-kNN	$k=7$	$\mathbf{95.02 \pm 5.74}$	$\mathbf{94.95 \pm 6.29}$	$\mathbf{94.62 \pm 5.98}$	$\mathbf{94.16 \pm 5.61}$
	WkNN	$k=7$	69.50 ± 9.70	58.90 ± 15.20	59.20 ± 14.40	57.20 ± 14.40
	DC-LAkNN	$k=7$	47.30 ± 1.52	31.42 ± 1.51	31.66 ± 1.16	31.33 ± 1.16
	kTLNN	$k=7$	68.58 ± 9.97	55.87 ± 14.34	57.00 ± 13.21	54.68 ± 13.48
	FkNN	$k=7$	67.13 ± 9.83	57.61 ± 13.50	59.21 ± 12.82	56.26 ± 12.68
Glass	BM-FkNN	$k=7, p=0.5, q=3, m=1.5$	69.92 ± 8.74	63.17 ± 13.51	65.30 ± 12.24	62.49 ± 12.36
	FPFS-kNN	$k=7$, corrname=Pearson	66.15 ± 8.08	43.09 ± 10.43	45.90 ± 9.28	43.35 ± 9.39
	PFS-kNN	$k=7, L=0.5, p=5$	63.05 ± 8.33	45.77 ± 12.04	47.83 ± 10.56	45.38 ± 10.80
	ECDNN	$k=7$	76.39 ± 6.39	59.54 ± 13.40	62.43 ± 12.63	59.67 ± 12.79
	OPNs-kNN	$k=7$	$\mathbf{77.25 \pm 5.84}$	$\mathbf{68.30 \pm 11.36}$	$\mathbf{66.30 \pm 10.06}$	$\mathbf{64.76 \pm 8.84}$

(Continued)

TABLE II
(Continued.) COMPARISON OF CLASSIFICATION ACCURACY AT FIXED k VALUE BETWEEN OPNs-kNN AND IMPROVED kNN ON VARIOUS DATASETS (%)

Datasets	Classifiers	Settings	Accuracy ± Std	Precision ± Std	Recall ± Std	F1 ± Std
	WkNN	$k=7$	96.20 ± 5.00	96.10 ± 5.30	96.70 ± 4.30	95.90 ± 5.30
	DC-LAkNN	$k=7$	66.29 ± 2.19	64.56 ± 2.21	64.49 ± 2.12	64.36 ± 2.23
	kTLNN	$k=7$	96.51 ± 5.08	96.40 ± 5.12	97.08 ± 4.27	96.27 ± 5.33
	FkNN	$k=7$	75.14 ± 9.91	76.02 ± 10.75	74.77 ± 10.24	73.85 ± 10.61
Wine	BM-FkNN	$k=7, p=0.5, q=3, m=1.5$	97.36 ± 3.51	97.56 ± 3.23	$\mathbf{97.64 \pm 3.14}$	97.35 ± 3.51
	FPFS-kNN	$k=7$, corrname=Pearson	96.28 ± 4.23	96.60 ± 3.80	96.90 ± 3.51	96.38 ± 4.16
	PFS-kNN	$k=7, L=0.5, p=5$	95.11 ± 4.81	95.61 ± 4.35	95.72 ± 4.20	95.16 ± 4.79
	ECDNN	$k=7$	90.60 ± 6.79	88.25 ± 11.97	87.71 ± 10.71	86.89 ± 11.62
	OPNs-kNN	$k=7$	$\mathbf{97.76 \pm 1.29}$	$\mathbf{97.82 \pm 1.46}$	97.24 ± 0.99	$\mathbf{97.69 \pm 2.02}$
	WkNN	$k=7$	81.10 ± 7.40	81.00 ± 7.30	80.90 ± 7.70	80.30 ± 7.50
	DC-LAkNN	$k=7$	66.70 ± 1.83	66.30 ± 1.92	65.60 ± 1.81	65.69 ± 1.85
	kTLNN	$k=7$	76.96 ± 8.71	76.67 ± 8.67	76.71 ± 9.14	75.94 ± 8.84
	FkNN	$k=7$	65.56 ± 8.79	65.90 ± 10.17	65.90 ± 10.17	63.82 ± 9.36
Heart	BM-FkNN	$k=7, p=0.5, q=3, m=1.5$	75.11 ± 7.07	75.47 ± 7.52	74.48 ± 7.30	74.40 ± 7.53
	FPFS-kNN	$k=7$, corrname=Pearson	81.07 ± 7.61	81.07 ± 7.61	80.40 ± 7.83	80.50 ± 7.96
	PFS-kNN	$k=7, L=0.5, p=5$	79.30 ± 7.37	79.69 ± 7.58	78.82 ± 7.62	78.79 ± 7.67
	ECDNN	$k=7$	78.66 ± 7.22	76.78 ± 10.85	75.16 ± 10.50	74.84 ± 10.73
	OPNs-kNN	$k=7$	$\mathbf{84.30 \pm 4.51}$	$\mathbf{83.52 \pm 3.28}$	$\mathbf{83.67 \pm 3.78}$	$\mathbf{82.77 \pm 4.78}$
	WkNN	$k=7$	92.00 ± 1.50	91.80 ± 1.60	91.80 ± 1.50	91.70 ± 1.50
	DC-LAkNN	$k=7$	87.03 ± 0.26	86.95 ± 0.26	86.46 ± 0.31	86.67 ± 0.29
	kTLNN	$k=7$	91.99 ± 1.43	91.76 ± 1.50	91.91 ± 1.45	91.80 ± 1.47
	FkNN	$k=7$	88.46 ± 1.29	88.60 ± 1.37	87.80 ± 1.32	88.10 ± 1.33
Rice	BM-FkNN	$k=7, p=0.5, q=3, m=1.5$	81.21 ± 1.13	81.34 ± 1.26	81.64 ± 1.26	81.01 ± 1.35
	FPFS-kNN	$k=7$, corrname=Pearson	92.33 ± 1.13	92.21 ± 1.15	92.13 ± 1.19	92.15 ± 1.16
	PFS-kNN	$k=7, L=0.5, p=5$	92.11 ± 1.25	91.99 ± 1.28	91.91 ± 1.30	91.93 ± 1.27
	ECDNN	$k=7$	$\mathbf{99.68 \pm 0.28}$	$\mathbf{93.69 \pm 12.82}$	$\mathbf{93.74 \pm 12.81}$	$\mathbf{93.72 \pm 12.82}$
	OPNs-kNN	$k=7$	92.51 ± 1.01	92.56 ± 1.03	92.12 ± 1.15	92.31 ± 1.23
	WkNN	$k=7$	87.30 ± 9.20	77.80 ± 21.00	74.20 ± 19.50	73.60 ± 19.20
	DC-LAkNN	$k=7$	84.70 ± 1.77	76.75 ± 2.92	70.92 ± 2.20	73.11 ± 2.47
	kTLNN	$k=7$	86.80 ± 9.86	77.92 ± 21.09	73.01 ± 19.49	72.95 ± 19.37
	FkNN	$k=7$	87.65 ± 7.98	81.55 ± 20.08	75.75 ± 15.82	76.43 ± 16.62
Appendicitis	BM-FkNN	$k=7, p=0.5, q=3, m=1.5$	80.32 ± 5.68	73.47 ± 18.21	70.38 ± 16.10	70.33 ± 15.17
	FPFS-kNN	$k=7$, corrname=Pearson	85.54 ± 8.29	75.73 ± 21.89	71.43 ± 15.56	71.70 ± 17.30
	PFS-kNN	$k=7, L=0.5, p=5$	87.41 ± 7.42	79.46 ± 21.34	73.74 ± 15.48	74.59 ± 17.04
	ECDNN	$k=7$	$\mathbf{96.48 \pm 5.33}$	$\mathbf{90.01 \pm 17.78}$	$\mathbf{89.26 \pm 17.81}$	$\mathbf{89.03 \pm 17.96}$
	OPNs-kNN	$k=7$	89.16 ± 5.46	82.25 ± 4.61	78.27 ± 4.31	78.13 ± 4.03

Note: When the η and δ parameters of OPNs-kNN are not set, it indicates that all feature pair combinations are explored, which is suitable for datasets with fewer features. The parameters for each comparison model were configured according to their original publications or widely accepted default values. The best performance is shown in bold.

TABLE III
COMPREHENSIVE RANKING OF CLASSIFIERS

Classifiers	Accuracy	Precision	Recall	F1
OPNs-kNN	2	2.09	2	2
WkNN	4	4.36	3.82	4.55
FPFS-kNN	4.09	4.27	4.36	4.27
ECDNN	4.55	5.27	5.45	5.45
FkNN	5.55	5.73	5.36	5.45
PFS-kNN	5.68	5.36	5.68	5.36
kTLNN	5.95	5.64	5.5	5.64
BM-FkNN	6	5.91	5.64	5.82
DC-LAkNN	7.18	6.36	7.18	6.45

it also outperforms other methods on 7 out of 11 datasets. Moreover, in the *heart*, *glass*, and *bupa* datasets, the proposed method demonstrates significantly better performance than most other approaches. Additionally, accuracy, precision, recall, and F1-score results are expressed as percentages, with the best performance in each category highlighted in bold.

To compare the performance of OPNs-kNN with other improved kNN variants, we conducted a nonparametric statistical analysis following Demšar and Demsar [33], using the Friedman test and the Nemenyi post-hoc test. The Friedman test evaluates whether significant differences exist among classifiers, while the Nemenyi test determines pairwise differences when the null hypothesis is rejected.

The Friedman test ($\alpha = 0.05$) yielded statistically significant results across all evaluation metrics: accuracy ($p = 0.0007$), precision ($p = 0.0129$), recall ($p = 0.0016$), and F1-score ($p = 0.0113$), rejecting the null hypothesis that all classifiers perform equally. The critical difference (CD) for the Nemenyi post-hoc test was computed as 3.62. Table III summarizes the average rank of each classifier, where lower ranks indicate superior performance.

To illustrate the statistical significance of performance differences, we present the Nemenyi critical difference (CD) diagrams in Figs. 8 and 9. Fig. 8 depicts the ranking analysis based on accuracy, while Fig. 9 shows the ranking analysis based on

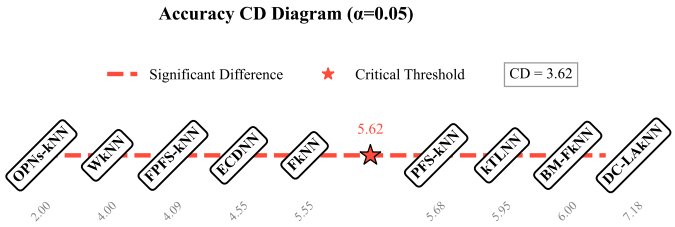

Fig. 8. Nemenyi critical difference (CD) diagram based on accuracy rankings.

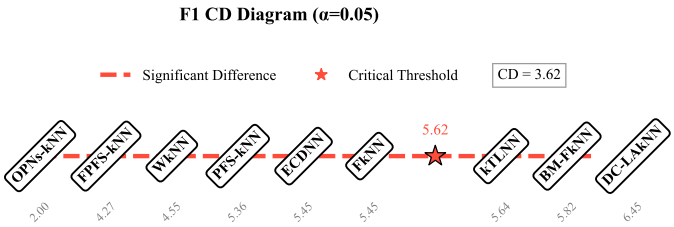

Fig. 9. Nemenyi CD diagram based on F1-score rankings.

the F1-score. In both figures, classifiers connected by the same horizontal bar do not exhibit statistically significant differences.

Results indicate that OPNs-kNN consistently exhibits statistically significant advantages over kTLNN, BM-FkNN, and DC-LAkNN (i.e., $\Delta > \mathrm{CD}$), while no significant differences were observed when compared with WkNN, FPFS-kNN, ECDNN, and FkNN (i.e., $\Delta < \mathrm{CD}$). Here, Δ represents the difference in rank scores between OPNs-kNN and the other models. Notably, kTLNN (2022), ECDNN (2023), FPFS-kNN (2022), and PFS-kNN (2023) are among the most recently proposed state-of-the-art methods.

The observed ranking hierarchy—where OPNs-kNN obtains an accuracy rank of 2 and an F1-score rank of 2, compared to the next best methods having an accuracy rank of 4 and an F1-score rank of 4.27—suggests that, despite the performance gap being reduced by modern variants, OPNs-kNN maintains a consistent advantage within the framework of statistical significance.

The results indicate that OPNs-kNN achieves superior performance compared with other improved kNN methods across multiple evaluation metrics, further validating its effectiveness.

In certain cases, OPNs-kNN may underperform compared to some traditional or enhanced algorithms. For example, its performance on the *Hayes-roth* and *Monk-2* datasets is inferior to that of some other models. This can be attributed to two main factors. First, OPNs-kNN is built upon the traditional kNN algorithm, with its primary distinctions lying in the computational units, rules, and distance calculation methods. However, similar to standard kNN, OPNs-kNN lacks an adaptive mechanism for selecting the optimal k value. Second, OPNs-kNN incorporates a unique feature pairing step designed to identify optimal or near-optimal feature pairings. Since feature pairings must be determined separately for each k value, recalculating the optimal pairings for every instance can be computationally expensive. To mitigate this cost, a suboptimal feature pairing combination is often used, which may not always yield the best classification performance.

Despite these challenges, the algorithm benefits from the nonlinear computation framework of OPNs, the effectiveness of the extended OPNs generalized metric, and the use of complex relationships between features in the OPNs feature pairing theory. As a result, OPNs-kNN outperforms other algorithms in most cases. Additionally, this study demonstrates the strong compatibility of the OPNs framework, suggesting that traditional or enhanced kNN algorithms can be incorporated into the OPNs framework to achieve better results and address more complex problems.

V. Conclusion

This study, by constructing the OPNs-kNN algorithm within the OPNs theoretical framework, combines features, filters combinations, and uses the OPNs-valued generalized metric to train the combination with the best classification effect. It aims to uncover the latent connections between real-number features, thereby enhancing classification performance. By comparing the OPNs-kNN algorithm with seven kNN algorithms based on traditional distance measures and eight enhanced kNN methods across 9 to 11 UCI datasets, results indicate that OPNs-kNN consistently outperforms the other algorithms in terms of classification accuracy and robustness. Similar to traditional kNN algorithms, the accuracy of this algorithm remains influenced by the choice of k. Additionally, there is room for further optimization in the process of filtering and training the optimal OPNs pairing combinations. Therefore, future work will focus on adaptive selection of the optimal k value and improving the efficiency of training optimal OPNs pairing combinations.

In this article, we study the kNN algorithm for the first time under the OPNs framework, proposing a set of rules for filtering OPNs features and a general method for constructing the kNN algorithm within the OPNs framework. Moreover, in addition to the kNN algorithm, many tasks in machine learning, including classification, clustering, and dimensionality reduction, are influenced by the chosen similarity or distance measurement methods. Therefore, the idea of constructing machine learning algorithms under the OPNs theoretical framework proposed in this article has the potential for generalization. It is possible to construct other algorithms under the OPNs theoretical framework to improve algorithm performance. This also represents a direction for further research by the authors in the future.

References

[1] I. Pollack, "The information of elementary auditory displays," *J. Acoust. Soc. Am.*, vol. 24, no. 6, pp. 745–749, 1952.
[2] I. Pollack, "The information of elementary auditory displays. II," *J. Acoust. Soc. Am.*, vol. 25, no. 4, pp. 765–769, 1953.
[3] I. K. A. Enriko, M. Suryanegara, and D. Gunawan, "Heart disease prediction system using k-nearest neighbor algorithm with simplified patient's health parameters," *J. Telecommun., Electron. Comput. Eng. (JTEC)*, vol. 8, no. 12, pp. 59–65, 2016.
[4] J. Galindo and P. Tamayo, "Credit risk assessment using statistical and machine learning: Basic methodology and risk modeling applications," *Comput. Econ.*, vol. 15, no. 1–2, pp. 107–143, Apr. 2000.
[5] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artif. Intell.*, vol. 2009, no. 1, 2009, Art. no. 421425.

[6] L. Zhou, "Ordered pair of normalized real numbers," *Inf. Sci.*, vol. 538, pp. 290–313, Oct. 2020.
[7] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.
[8] H. Cui et al., "Fuzzy analytic hierarchy process with ordered pair of normalized real numbers," *Soft Comput.*, vol. 27, no. 17, pp. 12267–12288, 2023.
[9] H. Cui, H. Zhang, L. Zhou, C. Yang, B. Li, and X. Zhao, "Multi-similarity K-nearest neighbor classification algorithm with ordered pairs of normalized real numbers," *J. Comput. Appl.*, vol. 43, no. 9, p. 2673, 2023.
[10] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, no. 4, pp. 325–327, Apr. 1976.
[11] J. Huang, Y. Wei, J. Yi, and M. Liu, "An improved kNN based on class contribution and feature weighting," in *Proc. 10th Int. Conf. Measuring Technol. Mechatron. Automat. (ICMTMA)*, 2018, pp. 313–316.
[12] H. T. Kahraman, "A novel and powerful hybrid classifier method: Development and testing of heuristic k-NN algorithm with fuzzy distance metric," *Data Knowl. Eng.*, vol. 103, pp. 44–59, May 2016.
[13] J. Gou, W. Qiu, Z. Yi, X. Shen, Y. Zhan, and W. Ou, "Locality constrained representation-based K-nearest neighbor classification," *Knowl. Based Syst.*, vol. 167, pp. 38–52, Mar. 2019.
[14] Z. Pan, Y. Wang, and Y. Pan, "A new locally adaptive k-nearest neighbor algorithm based on discrimination class," *Knowl. Based Syst.*, vol. 204, pp. 106185, Sep. 2020.
[15] R. Schlüter, A. Zolnay, and H. Ney, "Feature combination using linear discriminant analysis and its pitfalls," in *Proc. Interspeech*, 2006, pp. 345–348.
[16] H. A. Abu Alfeilat et al., "Effects of distance measure choice on k-nearest neighbor classifier performance: A review," *Big Data*, vol. 7, no. 4, pp. 221–248, 2019.
[17] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, 2014.
[18] M. Hagedoorn, "Pattern matching using similarity measures," Ph.D. dissertation, Utrecht University, Utrecht, the Netherlands, 2000.
[19] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy K-nearest neighbor algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 4, pp. 580–585, Jul./Aug. 1985.
[20] M. M. Kumbure, P. Luukka, and M. Collan, "A new fuzzy k-nearest neighbor classifier based on the Bonferroni mean," *Pattern Recognit. Lett.*, vol. 140, pp. 172–178, Dec. 2020.
[21] S. Memiş, S. Enginoğlu, and U. Erkan, "Fuzzy parameterized fuzzy soft k-nearest neighbor classifier," *Neurocomputing*, vol. 500, pp. 351–378, Aug. 2022.
[22] S. Memiş, "Picture fuzzy soft matrices and application of their distance measures to supervised learning: Picture fuzzy soft k-nearest neighbor (PFS-kNN)," *Electron.*, vol. 12, no. 19, Oct. 2023, Art. no. 4129.
[23] K. T. Atanassov, "Intuitionistic fuzzy sets," *Fuzzy Sets Syst.*, vol. 20, no. 1, pp. 87–96, 1986.
[24] K. T. Atanassov, "More on intuitionistic fuzzy sets," *Fuzzy Sets Syst.*, vol. 33, no. 1, pp. 37–45, Oct. 1989.
[25] R. R. Yager, "Pythagorean membership grades in multicriteria decision making," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 4, pp. 958–965, Aug. 2014.
[26] R. R. Yager and A. M. Abbasov, "Pythagorean membership grades, complex numbers, and decision making," *Int. J. Intell. Syst.*, vol. 28, no. 5, pp. 436–452, 2013.
[27] R. R. Yager, "Generalized orthopair fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 5, pp. 1222–1230, Oct. 2017.
[28] Y. He, W. Chen, Y. Chen, and Y. Mao, "Kernel density metric learning," in *Proc. IEEE 13th Int. Conf. Data Mining*, 2013, pp. 271–280.
[29] J. Wessel and N. J. Schork, "Generalized genomic distance-based regression methodology for multilocus association analysis," *Amer. J. Human Genetics*, vol. 79, no. 5, pp. 792–806, 2006.
[30] S. D. Bharkad and M. Kokare, "Performance evaluation of distance metrics: Application to fingerprint recognition," *Int. J. Pattern Recognit Artif Intell.*, vol. 25, no. 6, pp. 777–806, 2011.
[31] G. Gruenhage, "Generalized metric spaces," in *Handbook of Set-Theoretic Topology*, North-Holland: Elsevier, 1984, pp. 423–501.
[32] Y. Wang, Z. Pan, and J. Dong, "A new two-layer nearest neighbor selection method for kNN classifier," *Knowl. Based Syst.*, vol. 235, Jan. 2022, Art. no. 107604.
[33] A. X. Wang, S. S. Chukova, and B. P. Nguyen, "Ensemble k-nearest neighbors based on centroid displacement," *Inf. Sci.*, vol. 629, pp. 313–323, Jun. 2023.
[34] J. Demšar and J. Demsar, "Statistical comparisons of Classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.