Multiple Linear Regression Based on the Framework of Ordered Pair of Normalized Real Numbers

Yi Zheng, Yonglin Huang, Xiaoqin Pan, Hui Zhang, and Lei Zhou

***Abstract*—This study proposes a linear regression model based on the ordered pair of normalized real numbers (OPNs) theory, exploring the potential of developing machine learning models within this novel algebraic framework. While state-of-the-art deep learning models have advanced tabular data regression, they primarily operate within the real-valued domain. This work addresses this by leveraging the OPNs framework—a two-dimensional representation of real values with unique algebraic operations—to construct a new class of linear regression models capable of encoding richer feature interactions. The proposed OPNs-LR model introduces dedicated matrix operations and an optimal feature pairing strategy, which treats permutations of input features as composite features to capture both individual and relational information. A simplified "virtual mapping" mechanism is adopted to reduce preprocessing complexity while preserving theoretical consistency. Comprehensive experimental evaluations across several benchmark datasets demonstrate that OPNs-LR achieves competitive performance against various advanced regression models. Notably, it shows particular strength compared to other models built on alternative algebraic systems. This study validates the OPNs framework as a promising paradigm for enhancing machine learning algorithms. The source code is publicly available at https://github.com/alvinzean/OPNs.**

***Index Terms*—Machine learning, linear regression, ordered pair of normalized real numbers, feature pairing, structured representation learning**

I. Introduction

LINEAR regression remains a cornerstone of statistical learning, originating from Galton's foundational work [1]. While the classical model $y = \mathbf{x}^\top \boldsymbol{\beta} + \varepsilon$ offers interpretability and efficiency, its reliance on fixed real-valued arithmetic and strict linearity often limits its capacity to model complex couplings in modern tabular data. As datasets grow in complexity, the rigid structure of traditional scalar algebra struggles to capture high-order interactions without extensive manual feature engineering. To transcend these limitations, recent research has pivoted toward enriching the underlying algebraic or structural representations of data.

Prominent directions include leveraging hypercomplex-valued neural networks [2–5], exploring neuroalgebraic geometry to understand function spaces [6], and utilizing neural fractional-order differential equations to capture complex dynamics [7,8]. Additionally, spiking neural networks have been adapted for energy-efficient uncertainty estimation in regression [9], and graph neural machines have been proposed to model multi-layer perceptrons as graph structures for tabular data [10,11]. Incorporating uncertainty via fuzzy number-based models [12,13] and utilizing prototype-based frameworks to calibrate representation spaces [14] are also key trends. Recently, this prototype-oriented philosophy has been extended to hypergraph structures to enhance robustness in anomaly detection [15].

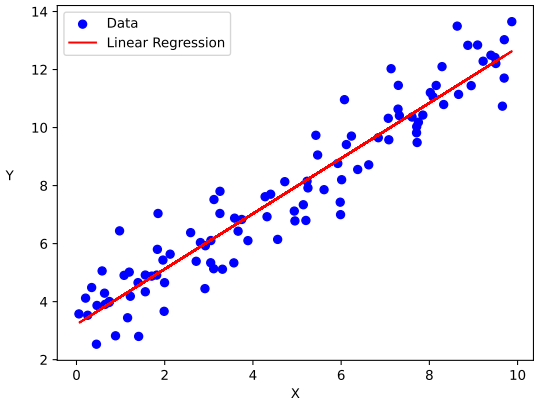

Fig. 1: A simple linear regression model fitting scattered data points

In the domain of deep learning architectures, a comprehensive survey has recently outlined the rapid evolution of representation learning for tabular data [16]. While transformer-based foundation models continue to evolve [17,18], novel self-supervised paradigms like T-JEPA [19] and distributionally robust frameworks [20] have emerged to construct meaningful representations without relying on complex augmentations. Parallel to this, a paradigm shift was marked by Kolmogorov-Arnold Networks (KAN) [21]. Unlike traditional architectures, KAN place learnable activation functions on edges rather than nodes, an approach successfully adapted to tabular modeling (TabKANet) to explicitly capture numerical feature distributions [22]. Performance is further advanced through novel optimization techniques, such as gradient orthogonalization for latent unit specialization [23] or certifiably optimal sparse regression algorithms [24].

To further capture rich inter-feature relationships and exploit structural patterns, Graph Neural Networks (GNN) have been systematically reviewed for tabular learning [25], and

Yi Zheng, Xiaoqin Pan and Lei Zhou is with the School of Computer Science and Technology, Southwest University of Science and Technology, Mianyang, Sichuan 621010, China.

Yonglin Huang is with the School of Mathematics and Physics, Southwest University of Science and Technology, Mianyang, Sichuan 621010, China.

Hui Zhang is with School of Information and Control Engineering, Southwest University of Science and Technology, Mianyang, Sichuan 621010, China.

(Corresponding author: Lei Zhou.)

Source code is available at: https://github.com/alvinzean/OPNs

contrastive hypergraph networks have emerged to model high-order correlations among data instances [26]. Beyond topological structures, identifying explicit feature interactions remains a core challenge. Recent works have proposed automatic feature augmentation frameworks [27] and prototypical neural additive models [28] to enhance interpretability and performance. Adaptive high-order interaction modeling has also been explored in click-through rate prediction [29], while comprehensive pairwise interaction estimation has shown significant promise in disease risk prediction [30]. Furthermore, visual analytics tools now aim to demystify these complex high-order dependencies in black-box models [31]. Collectively, these developments highlight a growing need for frameworks that can model complex dependencies—even within small or weakly structured datasets.

Fuzzy number-based models offer a complementary perspective by focusing on uncertainty and interpretability. Beyond standard approaches like ENNreg [12] and FuzzyGBR [13,32], recent top-tier research has significantly advanced this domain. In terms of privacy and trust, Bárcena et al. [33] proposed federated learning for fuzzy regression trees. For high-dimensional complex data, Yao et al. [34] introduced hierarchical fuzzy topological systems, while others have developed exclusionary neural complex fuzzy systems [35] or integrated radial basis functions [36] to improve robustness against nonlinearities. Furthermore, structural optimization of fuzzy-rule-based models using decision trees has been explored to balance accuracy and complexity [37]. Other approaches explore widely linear complex-valued models [38] or randomized numerical methods [39].

However, despite these sophisticated structural innovations, many such methods either operate as "black boxes" (like deep neural networks) or lack a consistent algebraic foundation akin to real number arithmetic, depending instead on manually defined rules or heuristics (like many fuzzy systems). This gap motivates the exploration of alternative algebraic frameworks that combine the expressiveness of structural learning with the mathematical transparency of deterministic operations.

In this context, we adopt the theory of Ordered Pair of Normalized Real Numbers (OPNs), originally proposed by Zhou in 2020 [40]. This theory models each value as an ordered pair (μ, ν) within the interval $(0, 1)$, governed by a total order and a set of dedicated nonlinear operations. On this foundation, we propose a new regression model—OPNs-LR—that incorporates OPNs matrix operations and optimal feature pairing strategies to construct expressive inputs. The resulting model not only captures raw feature values but also encodes their pairwise relationships, enabling the discovery of latent patterns beyond the reach of conventional regression approaches. Experimental results on benchmark datasets confirm the model's enhanced expressiveness and predictive performance, particularly in nonlinear regression tasks and settings with limited feature structure.

In this context, we adopt the theory of Ordered Pair of Normalized Real Numbers (OPNs), originally proposed by Zhou in 2020 [40]. This theory models each value as an ordered pair (μ, ν) within the interval $(0, 1)$, governed by a total order and a set of dedicated nonlinear operations. On this foundation, we propose a new regression model—OPNs-LR—that incorporates OPNs matrix operations and optimal feature pairing strategies to construct expressive inputs. The resulting model not only captures raw feature values but also encodes their pairwise relationships, enabling the discovery of latent patterns beyond the reach of conventional regression approaches. Experimental results on benchmark datasets confirm the model's enhanced expressiveness and predictive performance, particularly in nonlinear regression tasks and settings with limited feature structure.

This study contributes the following key advancements:

1) **Theoretical Contribution:** We extend the OPNs framework to support linear regression by introducing matrix operations and a structured feature pairing mechanism. This work demonstrates the feasibility of developing machine learning models beyond the traditional real-number domain.
2) **Algorithmic Innovation:** A novel feature pairing strategy is developed to jointly encode feature values and their interactions, enhancing the input representation for regression models.
3) **Empirical Validation:** Experiments across multiple benchmark datasets demonstrate that OPNs-LR outperforms traditional regression models, particularly in polynomial regression scenarios.

II. Preliminary

A. Related Work

Cui et al. [41] proposed an OPNs-based Analytic Hierarchy Process (OFAHP) that encodes expert judgments as the first component of each OPNs pair and data-driven evaluations as the second, fusing these through pairwise preference matrices to enhance decision-making robustness and reduce bias. While OFAHP focuses on decision-making tasks, our work fundamentally differs in several aspects. First, our objective is regression modeling, rather than preference ranking. Second, we employ the full computational rules of OPNs and integrate optimal pairing method directly into the training process, instead of using OPNs solely as a structural representation. Third, rather than relying on fixed preference matrices, our method dynamically constructs and filters feature pairings to capture nonlinear inter-feature dependencies.

In more recent developments, Zheng et al. [42] restructured the classical k-nearest neighbor algorithm under the OPNs framework (OPNs-kNN) by introducing an OPNs-based distance metric. Expanding the theoretical boundary further, Chen et al. [43] proposed a generalized-metric-based pattern recognition method using OPNs. This work demonstrates that OPNs can construct rigorous metric spaces for pattern classification, validating the framework's versatility beyond simple regression tasks. In parallel, Huang et al. [44] developed an OPNs-based stepwise regression algorithm (OPNs-SR). Both studies share the foundational idea of representing real-valued features using OPNs, but differ from our approach in several important ways. OPNs-LR generalizes the regression formulation to include both multiple and polynomial forms, explores a broader space of pairing strategies, and formally analyzes their

impact on performance. Moreover, our work establishes a more comprehensive OPNs matrix theory and introduces the initial application of OPNs-LASSO, enhancing the model's feature selection capability and laying the groundwork for regularized OPNs regression methods.

B. Basic OPNs Calculation

The theory of the Ordered Pair of Normalized Real Numbers (OPNs), proposed by Zhou in 2020 [40], defines an OPNs number as $\tilde{\alpha} = (\mu_{\tilde{\alpha}}, \nu_{\tilde{\alpha}})$, where both components lie in the interval $(0, 1)$. The tilde symbol distinguishes OPNs from real-valued variables. A central concept in this theory is the use of a continuous, strictly monotonic mapping $\varphi : \mathbb{R} \to (0, 1)$, satisfying the condition $\varphi(x) + \varphi(-x) = 1$ for all $x \in \mathbb{R}$. This ensures symmetric encoding of signed real numbers. One example of such a function and its inverse is:

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

and

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

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

Equality in the OPNs domain is defined by component-wise equivalence. Unlike real numbers, which are either positive or negative, OPNs allow for a neutral state, introducing a richer notion of sign. The framework supports basic arithmetic operations (addition, subtraction, multiplication, and division) with closure, enabled by nonlinear mappings through φ and its inverse. These operations are defined between OPNs or between an OPNs and a real number.

For scalar multiplication between a real number c and an OPNs $\tilde{\alpha}$, the operation is defined as:

$$c\tilde{\alpha} = \left(\varphi(c \cdot \varphi^{-1}(\mu_{\tilde{\alpha}})),\ \varphi(c \cdot \varphi^{-1}(\nu_{\tilde{\alpha}}))\right), \tag{3}$$

ensuring compatibility between real-valued scalars and the OPNs domain.

For two OPNs $\tilde{\alpha} = (\mu_{\tilde{\alpha}}, \nu_{\tilde{\alpha}})$ and $\tilde{\beta} = (\mu_{\tilde{\beta}}, \nu_{\tilde{\beta}})$, the addition is defined as:

$$\begin{aligned} \tilde{\alpha} + \tilde{\beta} = \Big(&\varphi\left(\varphi^{-1}(\mu_{\tilde{\alpha}}) + \varphi^{-1}(\mu_{\tilde{\beta}})\right), \\ &\varphi\left(\varphi^{-1}(\nu_{\tilde{\alpha}}) + \varphi^{-1}(\nu_{\tilde{\beta}})\right)\Big). \end{aligned} \tag{4}$$

The multiplication operation is more involved, yet central to the algebraic structure of OPNs. It is defined as:

$$\begin{aligned} \tilde{\alpha} \cdot \tilde{\beta} = \Big(&1 - \varphi\left(\varphi^{-1}(\mu_{\tilde{\alpha}})\varphi^{-1}(\nu_{\tilde{\beta}}) + \varphi^{-1}(\nu_{\tilde{\alpha}})\varphi^{-1}(\mu_{\tilde{\beta}})\right), \\ &1 - \varphi\left(\varphi^{-1}(\mu_{\tilde{\alpha}})\varphi^{-1}(\mu_{\tilde{\beta}}) + \varphi^{-1}(\nu_{\tilde{\alpha}})\varphi^{-1}(\nu_{\tilde{\beta}})\right)\Big). \end{aligned} \tag{5}$$

Next are some computational formulas that will be utilized in polynomial regression under the OPNs framework, providing the model with additional non-linear operations. These include exponentiation operations:

$$\begin{aligned} \tilde{\alpha}^n = \Bigg(&\varphi\Bigg(\frac{(-1)^{n+1}}{2}\left(\varphi^{-1}(\mu_{\tilde{\alpha}}) + \varphi^{-1}(\nu_{\tilde{\alpha}})\right)^n \\ &+ \frac{1}{2}\left(\varphi^{-1}(\mu_{\tilde{\alpha}}) - \varphi^{-1}(\nu_{\tilde{\alpha}})\right)^n\Bigg), \\ &\varphi\Bigg(\frac{(-1)^{n+1}}{2}\left(\varphi^{-1}(\mu_{\tilde{\alpha}}) + \varphi^{-1}(\nu_{\tilde{\alpha}})\right)^n \\ &- \frac{1}{2}\left(\varphi^{-1}(\mu_{\tilde{\alpha}}) - \varphi^{-1}(\nu_{\tilde{\alpha}})\right)^n\Bigg)\Bigg). \end{aligned} \tag{6}$$

Where $n \ge 1$. Additionally, there are sine and cosine trigonometric function operations

$$\begin{aligned} \sin\tilde{\alpha} = \Big(&\varphi\left(\sin\varphi^{-1}(\mu_{\tilde{\alpha}})\cos\varphi^{-1}(\nu_{\tilde{\alpha}})\right), \\ &\varphi\left(\cos\varphi^{-1}(\mu_{\tilde{\alpha}})\sin\varphi^{-1}(\nu_{\tilde{\alpha}})\right)\Big), \\ \cos\tilde{\alpha} = \Big(&\varphi\left(\sin\varphi^{-1}(\mu_{\tilde{\alpha}})\sin\varphi^{-1}(\nu_{\tilde{\alpha}})\right), \\ &\varphi\left(\cos\varphi^{-1}(\mu_{\tilde{\alpha}})\cos\varphi^{-1}(\nu_{\tilde{\alpha}})\right)\Big). \end{aligned} \tag{7}$$

The introduction of these fundamental operations furnishes us with the numerical tools required to handle OPNs. The OPNs theory parallels the basic operations of real numbers while introducing new structures (such as neutrality and a total order relation) that enrich data representation. This solid foundation enables the gradual development of novel algorithmic frameworks derived from conventional linear regression methods.

C. OPNs Total Order Relation

The OPNs theory not only supports various basic operations and matrix calculations but also introduces a total order relation, providing a natural way to arrange real number pairs. For any two OPNs, $\tilde{\alpha}$ and $\tilde{\beta}$, their total order relation can be determined by the following definition:

$$\begin{aligned} &1.\ \tilde{\alpha} < \tilde{\beta},\ \text{if}\ \tilde{\alpha} - \tilde{\beta} < \tilde{0} \\ &2.\ \tilde{\alpha} > \tilde{\beta},\ \text{if}\ \tilde{\beta} < \tilde{\alpha}; \\ &3.\ \tilde{\alpha} \le \tilde{\beta},\ \text{if}\ \tilde{\alpha} < \tilde{\beta}\ \text{or}\ \tilde{\alpha} = \tilde{\beta}; \\ &4.\ \tilde{\alpha} \ge \tilde{\beta},\ \text{if}\ \tilde{\beta} \le \tilde{\alpha}. \end{aligned} \tag{8}$$

Building upon the definitions of addition and subtraction, we can further elucidate the total order relation on OPNs. When $\varphi^{-1}(\mu_{\tilde{\alpha}}) + \varphi^{-1}(\nu_{\tilde{\alpha}}) < 0$, or $\varphi^{-1}(\mu_{\tilde{\alpha}}) < 0$ if $\varphi^{-1}(\mu_{\tilde{\alpha}}) + \varphi^{-1}(\nu_{\tilde{\alpha}}) = 0$, then $\tilde{\alpha} < \tilde{0}$. When $\varphi^{-1}(\mu_{\tilde{\alpha}}) + \varphi^{-1}(\nu_{\tilde{\alpha}}) > \varphi^{-1}(\mu_{\tilde{\beta}}) + \varphi^{-1}(\nu_{\tilde{\beta}})$, or $\varphi^{-1}(\mu_{\tilde{\alpha}}) < \varphi^{-1}(\mu_{\tilde{\beta}})$ if $\varphi^{-1}(\mu_{\tilde{\alpha}}) + \varphi^{-1}(\nu_{\tilde{\alpha}}) = \varphi^{-1}(\mu_{\tilde{\beta}}) + \varphi^{-1}(\nu_{\tilde{\beta}})$, then $\tilde{\alpha} < \tilde{\beta}$. This establishes the strict conditions under which the ordering relation $\tilde{\alpha} < \tilde{0}$ or $\tilde{\alpha} < \tilde{\beta}$ holds based on the inverse functions φ^{-1}.

III. Construction of Multiple Linear Regression Algorithm Based on OPNs Framework

The previous sections reviewed related work in linear regression and summarized the relevant foundations of OPNs

theory. This section extends the OPNs framework to vector and matrix operations, and constructs univariate and multivariate regression models, followed by the introduction of the OPNs pairing method. Section III and all subsequent sections focus entirely on the proposed methodological developments.

A. OPNs Matrix Operation

In general linear regression, the model is expressed as $f(\mathbf{x}) = \boldsymbol{\beta}^T\mathbf{x} + b$. We extend this to the OPNs domain by defining vectors and matrices with OPNs elements. An OPNs vector $\tilde{\mathbf{v}}$ and an $m \times n$ OPNs matrix $\tilde{\mathbf{A}}$ are defined as:

$$\tilde{\mathbf{v}} = (\tilde{v}_1, \ldots, \tilde{v}_n), \quad \tilde{\mathbf{A}} = [\tilde{a}_{ij}]_{m\times n}, \tag{9}$$

where components are OPNs numbers. Element-wise operations (addition, subtraction, multiplication, division) on OPNs matrices follow the algebraic rules defined in Section II, enabling standard computations such as $\tilde{\mathbf{C}} = \tilde{\mathbf{A}} \circ \tilde{\mathbf{B}}$. Similarly, scalar multiplication by a real number c is defined element-wise: $\tilde{b}_{ij} = c\tilde{a}_{ij}$.

The formula (9) reveals some techniques where by replacing the field of elements in a vector, the concept of vectors can be transformed into the realm of OPNs theory. Similarly, we can also establish matrices in the form of OPNs. An OPNs matrix, denoted as $\tilde{\mathbf{A}}$, consists of OPNs arranged in an $m \times n$ rectangular array. Each element, $\tilde{a}_{ij}$, located at the i-th row and j-th column, represents an OPNs, with m indicating the number of rows and n the number of columns. This structure allows for the representation of matrices in the OPNs form, treating each OPNs as a single, indivisible entity within the matrix framework, paving the way for defining operations specific to OPNs matrices.

Element-wise operations on OPNs matrices are fundamental for computational processes such as matrix inverses through block methods. For two OPNs matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ of identical dimensions, their element-wise operation, represented as $\tilde{\mathbf{C}} = \tilde{\mathbf{A}} \circ \tilde{\mathbf{B}}$, results in a matrix where each element $\tilde{c}_{ij}$ is derived from the corresponding elements of $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$. These operations encompass addition, subtraction, multiplication, and division, maintaining the integrity of OPNs properties. For example, addition is performed as $\tilde{c}_{ij} = \tilde{a}_{ij} + \tilde{b}_{ij}$ for corresponding elements. This principle similarly applies to multiplication, subtraction, and division. Such element-wise operations are crucial for extending OPNs matrices' utility in various mathematical computations, preparing the ground for further operations like scalar multiplication and matrix transpose within the OPNs framework.

Scalar multiplication in OPNs matrices entails multiplying each element of an OPNs matrix $\tilde{\mathbf{A}}$ by a real number c, resulting in a matrix $\tilde{\mathbf{B}}$ of the same dimensions as $\tilde{\mathbf{A}}$. The computation of each element $\tilde{b}_{ij}$ in $\tilde{\mathbf{B}}$ is as follows:

$$\tilde{b}_{ij} = c\tilde{a}_{ij},\ 1 \le i \le m,\ 1 \le j \le n. \tag{10}$$

This process highlights the adaptation of scalar multiplication to the OPNs framework, distinguishing it from traditional real number multiplication. The paper proceeds under the assumption that readers can distinguish between operations specific to OPNs and those pertaining to real numbers without further specification.

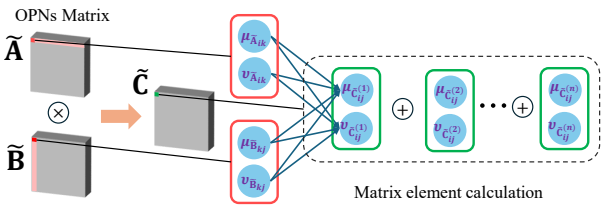

Fig. 2: OPNs Matrix Multiplication

Matrix multiplication in the OPNs framework involves combining two OPNs matrices, $\tilde{\mathbf{A}}$ with dimensions $m \times p$ and $\tilde{\mathbf{B}}$ with dimensions $p \times n$, to produce a new OPNs matrix $\tilde{\mathbf{C}}$ of dimensions $m \times n$, as illustrated in Fig. 2. The element $\tilde{c}_{ij}$ in $\tilde{\mathbf{C}}$ can be expressed as follows:

$$\tilde{c}_{ij} = \sum_{k=1}^{p} \tilde{a}_{ik} \cdot \tilde{b}_{kj}. \tag{11}$$

The resulting matrix $\tilde{\mathbf{C}}$ also maintains the OPNs form, and the summation symbol "$\sum$" in the equation represents the cumulative operation for OPNs.

The concept of an OPNs inverse matrix is grounded in the OPNs identity matrix, similar to its real number counterpart's reliance on the identity matrix. OPNs feature a unique additive identity, denoted by $\tilde{0}$, and a unique multiplicative identity, denoted by $\tilde{1}$, satisfying $\tilde{\alpha} + \tilde{0} = \tilde{\alpha}$ and $\tilde{\alpha} \cdot \tilde{1} = \tilde{\alpha}$ for any OPNs $\tilde{\alpha}$. Here, $\tilde{0} = (\varphi(0), \varphi(0))$ and $\tilde{1} = (\varphi(0), \varphi(-1))$. An OPNs identity matrix $\tilde{\mathbf{I}}$ is defined such that its diagonal elements are $\tilde{1}$, and off-diagonal elements are $\tilde{0}$, facilitating the establishment of inverse matrices within the OPNs domain.

Within the OPNs framework, the inverse matrix of an $n \times n$ OPNs matrix $\tilde{\mathbf{A}}$ is an OPNs matrix $\tilde{\mathbf{B}}$ that, when multiplied by $\tilde{\mathbf{A}}$, results in the OPNs identity matrix $\tilde{\mathbf{I}}$ ($\tilde{\mathbf{A}}\tilde{\mathbf{B}} = \tilde{\mathbf{B}}\tilde{\mathbf{A}} = \tilde{\mathbf{I}}$), and is denoted as $\tilde{\mathbf{A}}^{-1}$. This concept enables operations like matrix division and the solving of linear equations in the OPNs domain. However, similar to real-number matrices, finding an inverse matrix directly via properties like the adjoint matrix approach or elementary transformations can be computationally intensive and less precise for larger matrices. Thus, in the OPNs context, it's recommended to cautiously approach matrix inversion, considering the complexity and precision issues inherent in such calculations.

The transpose of an OPNs matrix $\tilde{\mathbf{A}}$, with original dimensions $m \times n$, is denoted as $\tilde{\mathbf{A}}^{\mathrm{T}}$ and has dimensions $n \times m$. In this transposed matrix $\tilde{\mathbf{A}}^{\mathrm{T}}$, the element at the i-th row and j-th column, $\tilde{a}_{ij}^{\mathrm{T}}$, originates from the element at the j-th row and i-th column of the original matrix, $\tilde{a}_{ji}$. Transposing an OPNs matrix effectively swaps its rows and columns, ensuring that each element's position in $\tilde{\mathbf{A}}^{\mathrm{T}}$ mirrors its original position in $\tilde{\mathbf{A}}$ with reversed row and column indices.

B. Research Methods and Algorithm Processes

The algorithm developed in this study, while mirroring traditional methods in structure, incorporates distinct steps tailored to the OPNs framework:

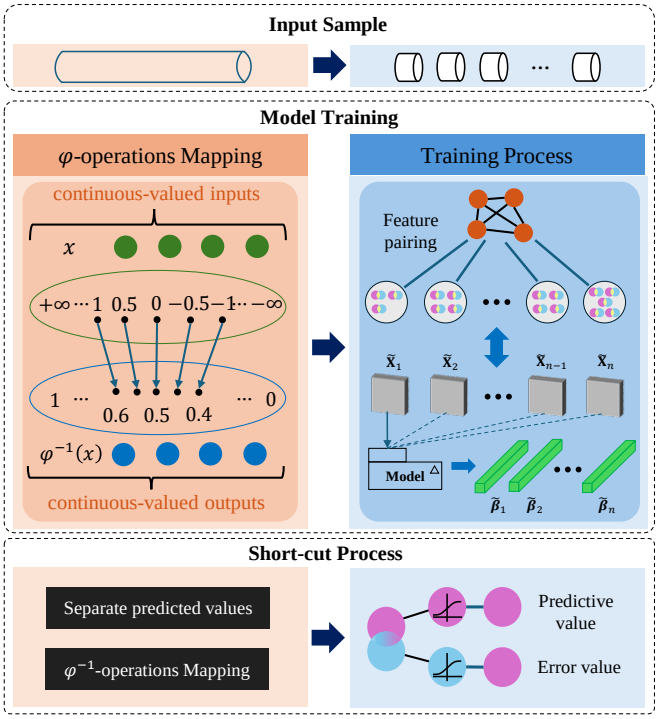

Fig. 3: Algorithm Flow

data is first mapped from real numbers to OPNs (creating 2D pairs); a training step then identifies the optimal feature pairing strategy; finally, OPNs outputs are converted back to real numbers for evaluation. To streamline complexity, we adopt a "virtual mapping" strategy, conceptually applying the φ transformation without explicit computational overhead during intermediate steps.

As illustrated in Fig. 3, the raw data undergoes preprocessing to generate multiple training and testing sets. During training, the φ-operation function is applied to map real-valued data into the interval $(0, 1)$, as specified by the OPNs theory. This is followed by a dynamic feature pairing process, designed as an iterative training step to identify the optimal pairing strategy. After n iterations, this process ultimately determines the optimal feature pairing strategy and the corresponding weight parameters $\tilde{\beta}_i$. Each pairing strategy generates a distinct feature matrix $\tilde{\mathbf{X}}_i$ with different feature representations. The testing phase then applies the same pairing strategy and the obtained weight parameters. Both the training and testing data are constrained within the interval $(0, 1)$, and the results are mapped back to the real number range using the inverse function φ^{-1}, allowing the separation of the actual predicted values.

Although φ mapping plays a crucial role in preserving the integrity of OPNs theory, its strict enforcement in machine learning applications may be unnecessary. This is due to the potential for precision loss and the realization that machine learning outcomes do not need to be constrained within $(0, 1)$. Thus, preprocessing can be simplified by assuming data has undergone a virtual φ mapping, bypassing direct mapping and inverse operations. This approach streamlines the algorithm, reducing complexity while fully utilizing OPNs advantages for model development. Next, we'll explore model construction and training in this simplified framework.

C. OPNs Univariate Linear Regression

Exploring linear regression within the OPNs framework involves adapting traditional algorithmic structures. A basic linear regression formula, $y = \beta_0 + \beta_1 x_1 + \varepsilon$, serves as a reference for constructing a univariate linear regression model using OPNs, transforming into $\tilde{y} = \tilde{\beta}\tilde{x} + \tilde{\varepsilon}$, where variables and coefficients are represented as OPNs. This approach simplifies by integrating the intercept and error term into $\tilde{\varepsilon}$.

In practical applications, this model can be expressed in matrix form as follows:

$$\tilde{\mathbf{y}} = \tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}} + \tilde{\boldsymbol{\varepsilon}}, \tag{12}$$

with the OPNs adaptation requiring samples as two-dimensional number pairs. This necessitates a novel preprocessing step for real datasets: appending a column of dummy zeros to meet OPNs model input requirements, thereby ensuring compatibility without introducing new information. This modification allows for the direct application of traditional sample structures to the linear regression framework based on OPNs.

In the OPNs-based univariate linear regression, introducing dummy labels to real-number data forms OPNs sample features, $\tilde{x}_j^{(i)}$, as either $\left(\varphi(x_j^{(i)}), \varphi(0)\right)$ or $\left(\varphi(0), \varphi(x_j^{(i)})\right)$, maintaining consistency in the position of the dummy label across samples. This approach simplifies to using direct numerical pairs, $(x_j^{(i)}, 0)$, bypassing the need for virtual mapping with φ. For the entire input feature matrix $\tilde{\mathbf{X}}$, it should take the form $\tilde{\mathbf{X}} = \left[\tilde{\mathbf{x}}^{(1)} \quad \tilde{\mathbf{x}}^{(2)} \quad \cdots \quad \tilde{\mathbf{x}}^{(m)}\right]^{\mathrm{T}}$, where $\tilde{\mathbf{x}}^{(i)}$ is the OPNs feature of the i-th sample, and the integer index i satisfies $1 \leq i \leq m$. This method also applies to the label value vector $\tilde{\mathbf{y}}$. We refer to this process of converting real-number data into OPNs as the OPNization of data.

In the OPNs-based univariate linear regression model, the combination of parameters $\tilde{\boldsymbol{\beta}}$ and error terms $\tilde{\boldsymbol{\varepsilon}}$ into a single vector $\tilde{\boldsymbol{\beta}}^*$ simplifies the model to $F(\tilde{\mathbf{X}}) = \tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}}^*$. Model performance is assessed via mean square error, leading to the loss function:

$$L\left(\tilde{\boldsymbol{\beta}}^*\right) = \left(\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}}^*\right)^{\mathrm{T}} \left(\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}}^*\right), \tag{13}$$

with the partial derivative given by:

$$\partial L\left(\tilde{\boldsymbol{\beta}}^*\right) / \partial \tilde{\boldsymbol{\beta}}^* = 2\tilde{\mathbf{X}}^{\mathrm{T}}\left(\tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}}^* - \tilde{\mathbf{y}}\right). \tag{14}$$

Setting this derivative to zero results in the optimal parameter vector:

$$\hat{\tilde{\boldsymbol{\beta}}}^* = \left(\tilde{\mathbf{X}}^{\mathrm{T}}\tilde{\mathbf{X}}\right)^{-1}\tilde{\mathbf{X}}^{\mathrm{T}}\tilde{\mathbf{y}}. \tag{15}$$

The derivation from equation (13) to equation (15) follows limit-based reasoning similar to that in the real domain. As these technical details are beyond the scope of this work, they are omitted here. This process illustrates the adaptability of the OPNs framework and forms the basis for extending it to multivariate regression.

D. OPNs Multiple Linear Regression

To extend linear regression into a multivariate setting within the OPNs framework, we reformulate the classical model as

$$\tilde{y} = \tilde{\beta}_1\tilde{x}_1 + \tilde{\beta}_2\tilde{x}_2 + \cdots + \tilde{\beta}_n\tilde{x}_n + \tilde{\varepsilon}, \tag{16}$$

where $\tilde{y}$, $\tilde{\beta}_j$, and $\tilde{x}_j$ are expressed in the OPNs domain. This formulation is consistent with the matrix-vector representation introduced earlier in (11), that is, $\tilde{\mathbf{y}} = \tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}} + \tilde{\boldsymbol{\varepsilon}}$. Here, $\tilde{\mathbf{X}} \in \mathbb{R}^{m\times d}$ denotes the OPNs feature matrix derived from the original dataset, where m is the number of samples and d is the number of OPNs features obtained through systematic pairings of the original real-valued features. Constructing this matrix involves transforming real-valued features into OPNs units through systematic pairings, as defined below.

Definition 1 (Feature Pairing). Given two real-valued features $\mathbf{x}_i$ and $\mathbf{x}_j$, the ordered pair $(\mathbf{x}_i, \mathbf{x}_j)$ is treated as an OPNs feature. This symbolic combination is called a feature pairing and serves as a structural unit in OPNs-based input construction.

Building on Definition 1, a feature pairing combination selects multiple pairings to construct the full input matrix.

Definition 2 (Feature Pairing Combination). A feature pairing combination refers to a sequence of d ordered feature pairings selected from all possible pairings of the original input features. Each such combination defines a specific set of OPNs feature columns, which together form the structure of the input matrix $\tilde{\mathbf{X}}$ used across all samples.

For example, given a sample with features $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, a valid feature pairing combination might include $(\mathbf{x}_1, \mathbf{x}_2)$ and $(\mathbf{x}_3, \mathbf{x}_1)$. The complete input matrix $\tilde{\mathbf{X}}$ is constructed by stacking such combinations column-wise: $\tilde{\mathbf{X}} = \begin{bmatrix}\tilde{\mathbf{x}}_1 & \tilde{\mathbf{x}}_2 & \dots & \tilde{\mathbf{x}}_d\end{bmatrix}$.

Figure 5 provides an illustration of the pairing structure. The specific choice of combinations greatly influences model performance. Further discussion of pairing strategy design, structural properties, and computational implications is provided in Section IV-A, with empirical evaluation in Section V-C.

E. OPNs Polynomial Regression

Polynomial regression extends linear models by incorporating higher-order terms to capture nonlinear relationships. In the OPNs framework, this is expressed as:

$$\tilde{y}_i = \tilde{\boldsymbol{\beta}}_0 + \tilde{\boldsymbol{\beta}}_1\tilde{\mathbf{x}}_i + \tilde{\boldsymbol{\beta}}_2\tilde{\mathbf{x}}_i^2 + \cdots + \tilde{\boldsymbol{\beta}}_n\tilde{\mathbf{x}}_i^n + \tilde{\varepsilon}_i, \tag{17}$$

where $\tilde{\mathbf{x}}_i$ is the OPNs input vector, and $\tilde{\boldsymbol{\beta}}$ and $\tilde{\varepsilon}_i$ denote the coefficient and error terms, respectively.

In matrix form, the model becomes:

$$\begin{bmatrix}\tilde{y}_1\\ \tilde{y}_2\\ \vdots\\ \tilde{y}_m\end{bmatrix} = \begin{bmatrix}\tilde{1} & \tilde{\mathbf{x}}_1 & \tilde{\mathbf{x}}_1^2 & \cdots & \tilde{\mathbf{x}}_1^n\\ \tilde{1} & \tilde{\mathbf{x}}_2 & \tilde{\mathbf{x}}_2^2 & \cdots & \tilde{\mathbf{x}}_2^n\\ \vdots & \vdots & \ddots & \vdots & \\ \tilde{1} & \tilde{\mathbf{x}}_m & \tilde{\mathbf{x}}_m^2 & \cdots & \tilde{\mathbf{x}}_m^n\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{\beta}}_0\\ \tilde{\boldsymbol{\beta}}_1\\ \vdots\\ \tilde{\boldsymbol{\beta}}_n\end{bmatrix} + \begin{bmatrix}\tilde{\varepsilon}_1\\ \tilde{\varepsilon}_2\\ \vdots\\ \tilde{\varepsilon}_m\end{bmatrix}. \tag{18}$$

Although the regression includes nonlinear terms, it remains linear in the parameters $\tilde{\boldsymbol{\beta}}$, allowing estimation via the least squares solution in Equation (15). In practice, polynomial terms are treated as additional input variables within a multivariate regression model.

The OPNs framework also supports a generalized formulation using basis functions:

$$\tilde{y}_i = \tilde{\beta}_0 + \tilde{\beta}_1\psi_1(\tilde{x}_i) + \tilde{\beta}_2\psi_2(\tilde{x}_i) + \cdots + \tilde{\beta}_m\psi_m(\tilde{x}_i) + \tilde{\varepsilon}_i, \tag{19}$$

where $\psi_j(\cdot)$ denotes an arbitrary basis function such as polynomial, exponential, or trigonometric functions. This generalization enhances the model's capacity to fit diverse nonlinear relationships.

Despite the added complexity, the estimation process remains consistent with real-valued models. In OPNs polynomial regression, nonlinear behavior arises from both the polynomial structure and the OPNs-specific algebraic operations. The model parameters reflect the strength of influence across the OPNs feature space, capturing deeper interactions than conventional real-number models. Care must be taken to avoid overfitting by limiting the number of basis functions.

F. Model Training Process

Figure 4 illustrates the overall training workflow of the proposed OPNs-LR model. The process begins with the OPNs-Matrices Generation Module, where the original real-valued dataset is transformed into an OPNs-compatible format through feature pairing. Each OPNs feature is formed by pairing two scalar features, aligning the data structure with the ordered pair representation required by the OPNs framework.

For datasets with a small number of features or sufficient computational resources, it is feasible to exhaustively search all valid feature pairing combinations to identify the optimal configuration. When exhaustive search becomes impractical, the number of combinations can be reduced using heuristic strategies or the OPNs-LASSO algorithm, aiming for a near-optimal solution. This pairing process is conceptually encapsulated in the Pairing and Combination Module, which supports both pattern-based pairing (detailed in Section IV-B) and LASSO-driven feature selection (discussed in Section IV-C).

Once the OPNs matrix $\tilde{\mathbf{X}}$ is constructed, it is passed into the OPNs Scale Operation Module, which performs regression fitting in the OPNs coordinate space. The model parameters $\tilde{\boldsymbol{\beta}}$ are computed using the optimization formula described in (15). The model is evaluated after each pairing combination to assess performance. If the termination condition is not met—such as a fixed iteration budget or performance threshold—a new pairing combination is applied, and the training loop repeats. Ultimately, the pairing combination yielding the best or a satisfactory result is retained for prediction.

Each prediction output is an OPNs value $\tilde{\hat{y}}_k = (\mu_{y_k}, \nu_{y_k})$. If the dependent variable consists of a single real-valued target, a real-valued prediction is extracted from $\tilde{\hat{y}}_k$ according to the encoding used during training. Specifically, if the true label $\tilde{y}_k$ was encoded as $(\mu_{y_k}, 0)$, then the predicted value is taken as μ_{y_k}. ν_{y_k} in this case can be interpreted as residual information.

With the model structure defined, the core challenge becomes identifying an effective pairing strategy. The selection process must account for feature reuse, pairing diversity, and compatibility between different basis function families and the

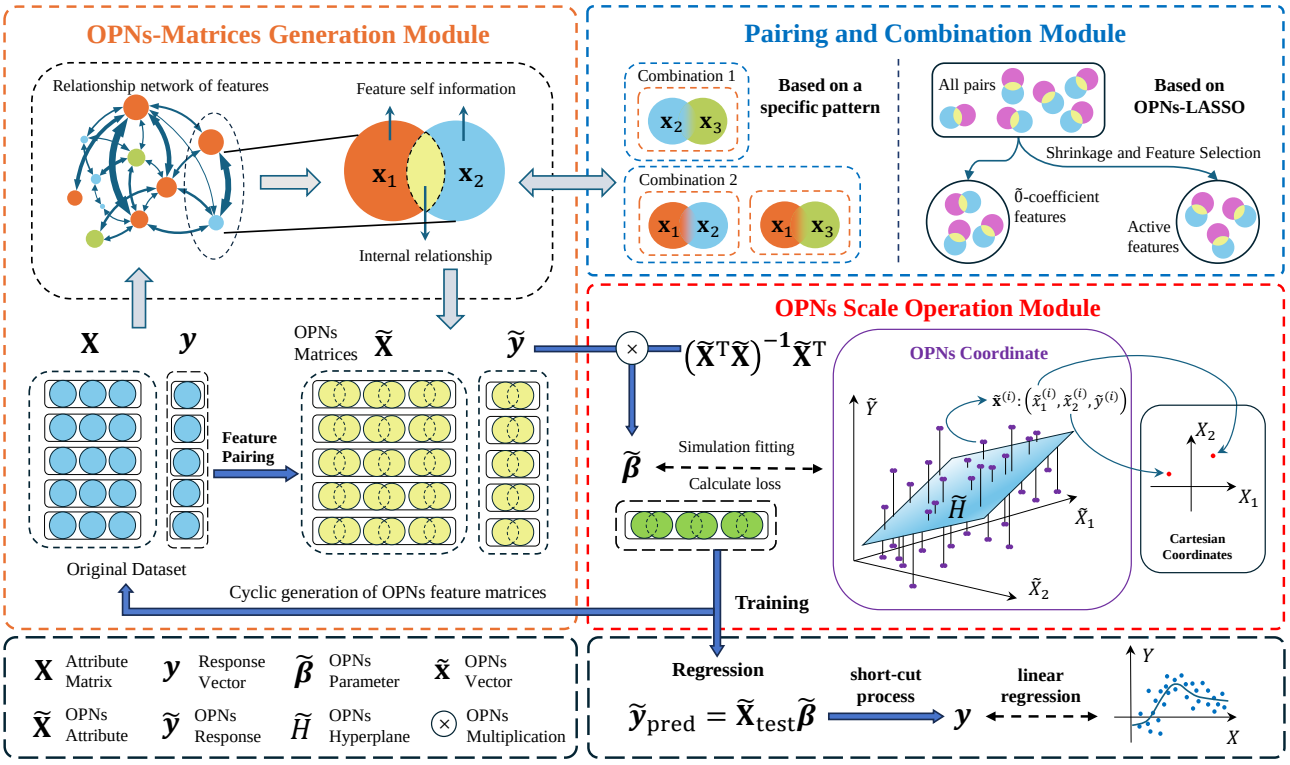

Fig. 4: Training Process

resulting OPNs matrix $\tilde{\mathbf{X}}$. These considerations are further detailed in Section IV.

G. OPNs Linear Regression Algorithm

Algorithm 1 is a typical linear regression algorithm based on the OPNs framework. It considers a training set $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test}$, each containing m_1 and m_2 samples, respectively. Here, $\mathbf{X}_{tr}$ and $\mathbf{X}_{te}$ represent the traditional training and test sets, while $\tilde{\mathbf{X}}_{tr}$, $\tilde{\mathbf{y}}_{tr}$, and $\tilde{\mathbf{X}}_{te}$ denote the feature matrix, label vector, and feature matrix of the OPNs test set, respectively. The training set $\mathbf{X}_{tr}$ comprises a set of n features, and $\mathbf{y}_{tr}$ contains a single label value. After preprocessing, additional steps are introduced, notably the OPNsization of data, with a critical feature combination step. For lower-dimensional features, exhaustive traversal is feasible, while random combinations break local loops in larger sets. During training, feature pairing combinations form loops, aiming for optimal strategies. Termination occurs upon reaching pre-set evaluation metrics or iteration limits. Evaluation metrics computation mirrors prediction, multiplying the OPNs feature matrix $\tilde{\mathbf{X}}_{tr}^{*}$ by the weight vector $\tilde{\beta}$ to derive the OPNs label vector, followed by extracting relevant values to form the predicted label vector.

Post-processing converts OPNs results into real-number prediction data. When the dataset's label values $\mathbf{y}$ comprise a single column, from the two-dimensional number pairs in $\tilde{\mathbf{y}}$, only one item represents the prediction data. Its position depends on the combination order of $\mathbf{y}$ and the dummy label $\mathbf{0}$ during processing. For instance, if the combination order is $(\mathbf{0}, \mathbf{y})$, the model outputs $(\varepsilon', \mathbf{y}')$, with the second item as the actual prediction value. Post-processing involves extracting this item and mapping it to real numbers, ensuring accurate regression prediction results.

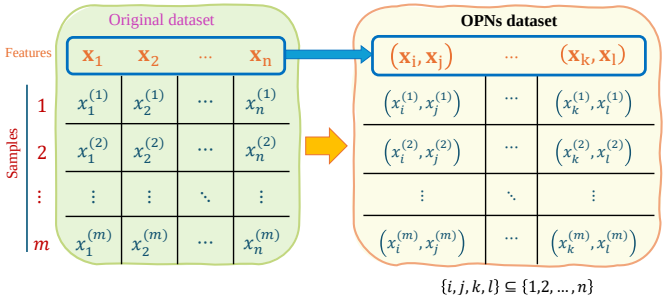

Fig. 5: Feature Pairing

IV. OPNs Pairing Method

A. Challenges and Structural Properties of OPNs Pairing

Transforming n features into OPNs units involves pairing, leading to $n(n-1)$ possible ordered pairs. Selecting k pairs results in a combinatorial input space of size $C(n(n-1), k)$, which grows exponentially with n.

The pairing module, highlighted in Fig. 5, introduces not only the individual feature values but also inter-feature relational information—both linear and nonlinear—into the input representation. This distinguishes OPNs-based models from conventional real-valued regression models, which only use raw feature values and ignore structural dependencies among features. Feature pairing captures this additional layer of information, enabling the model to learn from deeper patterns.

Algorithm 1 OPNs Linear Regression Algorithm

Require: A training dataset with m_1 samples, $\mathcal{D}_{train} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m_1}$: input Matrix $\mathbf{X}_{tr}$ and corresponding labels $\mathbf{y}_{tr}$. A Testing dataset $\mathcal{D}_{test} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m_2}$: input Matrix $\mathbf{X}_{te}$

Ensure: Output Predicted value vector $\mathbf{y}_{\text{pred}}$

TRAIN($\mathbf{X}_{tr}$)

$\mathbf{X}_{tr} \leftarrow \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}, \mathbf{y}_{tr} \leftarrow \{\mathbf{y}_1\}$

Construct an OPNs feature $\tilde{\mathbf{x}}_i$ by combining the numerical values of two features $\mathbf{x}_j$ and $\mathbf{x}_k$ in the dataset into a pair of real number.

$\tilde{\mathbf{x}}_i \leftarrow (\mathbf{x}_j, \mathbf{x}_k)$ **where** $j \neq k$ **and** $1 \leq j, k \leq n$

$\tilde{\mathbf{y}} \leftarrow (\mathbf{y}_1, \mathbf{0})$

The OPNs feature set $\tilde{\mathbf{X}}$ is composed of all OPNs feature vectors $\tilde{\mathbf{x}}_i$

$\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_i \mid$ for all possible $\tilde{\mathbf{x}}_i$ constructed using the above methods $\}$, that is, $\tilde{\mathbf{X}} \leftarrow \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots\}$

repeat

OPNs variable set $\tilde{\mathbf{X}}_{tr}$ is formed by randomly selecting elements from $\tilde{\mathbf{X}}$

$$\tilde{\mathbf{X}}^* = \left\{\tilde{\mathbf{x}} \mid \tilde{\mathbf{x}} \in \tilde{\mathbf{X}}\right\}$$

The final training feature set $\tilde{\mathbf{X}}_{tr}$ is calculated by matrix $\tilde{\mathbf{X}}^*$ through δ basis functions ψ and combined in columns

$$\tilde{\mathbf{X}}_{tr} \leftarrow \left[\tilde{\mathbf{X}}^* \quad \psi_1\left(\tilde{\mathbf{X}}^*\right) \quad \psi_2\left(\tilde{\mathbf{X}}^*\right) \quad \cdots \quad \psi_\delta\left(\tilde{\mathbf{X}}^*\right)\right],$$

where $\psi\left(\tilde{\mathbf{X}}^*\right) \in \mathbb{R}^{\mathbf{d}_\psi}$

Compute the output weight voctor $\tilde{\boldsymbol{\beta}}$

$\tilde{\boldsymbol{\beta}} \leftarrow \left(\tilde{\mathbf{X}}_{tr}^T \tilde{\mathbf{X}}_{tr}\right)^{-1} \tilde{\mathbf{X}}_{tr}^T \tilde{\mathbf{y}}$

$\tilde{\mathbf{y}}_{tr\_pred} = (\mathbf{y}_1, \mathbf{y}_2) \leftarrow \tilde{\mathbf{X}}_{tr} \tilde{\boldsymbol{\beta}}$

$\mathbf{y}_{tr\_pred} \leftarrow \mathbf{y}_\theta$ **for** $\theta = 1$ **or** $\theta = 2$

until The evaluation indicators exceed the threshold or the number of cycles is exhausted

return Optimal parameter $\tilde{\boldsymbol{\beta}}^*$ **and** the optimal feature pairing combination for $\tilde{\mathbf{X}}^*$

PREDICT($\mathbf{X}_{te}$)

The OPNs feature matrix $\tilde{\mathbf{X}}_{te}$ is composed of training set $\mathbf{X}_{te}$ that matches features according to the local optimal feature pairing combination of $\tilde{\mathbf{X}}^*$, and is calculated by δ basis functions ψ and merged into columns

$\tilde{\mathbf{y}} = (\mathbf{y}_1, \mathbf{y}_2) \leftarrow \tilde{\mathbf{X}}_{te} \tilde{\boldsymbol{\beta}}^*$

$\mathbf{y}_{pred} \leftarrow \mathbf{y}_i, i = \theta$

return $\mathbf{y}_{pred}$

Notably, OPNs pairing is asymmetric: $(\mathbf{x}_i, \mathbf{x}_j) \neq (\mathbf{x}_j, \mathbf{x}_i)$. However, in practice, many OPNs expressions exhibit symmetry with respect to feature order. For polynomial expansions and certain types of models, this symmetry can be exploited to reduce redundant combinations, effectively shrinking the search space.

Another challenge lies in ensuring sufficient feature coverage. Ideally, each original feature should appear at least once across the input pairs, to maximize information utilization. A minimal yet effective strategy is to generate $n(n-1)/2$ unique OPNs features by pairing each feature with others exactly once. This offers a good balance between expressiveness and complexity, mitigating overfitting while retaining interaction potential.

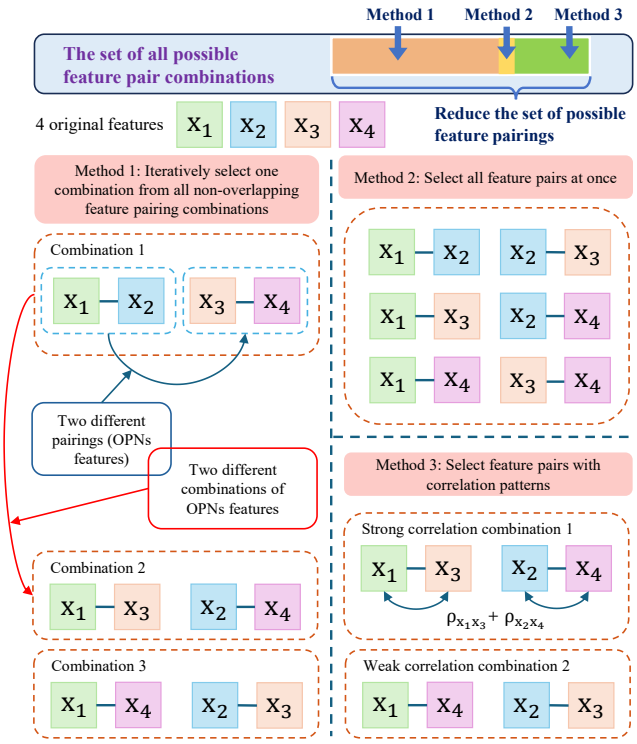

Fig. 6: The feature pair combinations produced by each method constitute a subset of the full combination space.

Unlike conventional regression, where repeated features in multiple dimensions are often redundant, OPNs pairing allows the same feature to appear in multiple pairings, potentially contributing new relational insights. For example, $(\mathbf{x}_3, \mathbf{x}_5)$ and $(\mathbf{x}_3, \mathbf{x}_6)$ both involve $\mathbf{x}_3$ but encode distinct relationships.

In polynomial OPNs models, pairing patterns can vary across degrees. For example, one configuration might use $(\mathbf{x}_1, \mathbf{x}_2)$ and $(\mathbf{x}_3, \mathbf{x}_4)$ for the linear term $\tilde{\mathbf{X}}$, and $(\mathbf{x}_1, \mathbf{x}_3)$, $(\mathbf{x}_1, \mathbf{x}_4)$, etc., for $\tilde{\mathbf{X}}^2$. The number of pairings per term is also flexible, unlike real-valued regression, where input dimension is typically fixed.

Due to the vast number of possible feature pairings, exhaustive search is often infeasible. Therefore, heuristics based on prior knowledge or statistical correlation are typically used to guide the selection of promising pairs. In summary, OPNs feature pairing introduces additional degrees of freedom that enhance modeling flexibility and expressive power, but also bring structural and computational challenges that require principled design and empirical validation.

B. Default Pairing Strategies

To summarize: There are various ways to select feature pairings. As illustrated in Fig. 6, in an effort to minimize the complexity of feature pairing combinations, the experimental section of this paper will primarily explore the following three pairing selection methods:

Method 1: Each feature in the original dataset is used exactly once. Suppose S is a set containing n elements, and we define $Pair(S)$ as the set of all possible pairings. The recursive formula is as follows: select two elements $\mathbf{x}_i$ and $\mathbf{x}_j$ from the set S as a pair, denoted as $(\mathbf{x}_i, \mathbf{x}_j)$. Recursively generate all pairings from the remaining set $S \setminus \{\mathbf{x}_i, \mathbf{x}_j\}$.Mathematically, this can be expressed by the following recursive formula:

$$Pair(S) = \bigcup_{\mathbf{x}_i, \mathbf{x}_j \in S} \{(\mathbf{x}_i, \mathbf{x}_j)\} \times Pair\left(S \setminus \{\mathbf{x}_i, \mathbf{x}_j\}\right), \quad (20)$$

where $\times$ denotes the Cartesian product of the pair combinations. The pairings generated by this method form a subset of the combinations produced by Method 2. This approach fully utilizes the information contained in the original features themselves, but it does not make full use of the potential pairing information. When the original features are numerous, a significant amount of pairing information may be overlooked.

Method 2: All features are considered simultaneously along with all possible pairing combinations. Let S be a set containing n elements, denoted as $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. The set of all possible pairings can be expressed as:

$$\{(\mathbf{x}_i, \mathbf{x}_j) \mid 1 \leq i < j \leq n\}. \quad (21)$$

Although these pairings are fed into the model all at once without an iterative process, the results still vary across different datasets. If, in the original dataset, most or all of the feature-to-feature relationships are strongly correlated, the model's performance will be significantly better, far surpassing traditional methods. However, if some pairings are uncorrelated or weakly correlated, the training process might amplify these weak correlations, increase their weights, and ultimately lead to model overfitting.

Method 3: Each feature in the original dataset is used exactly once. Correlation-based pairing is conducted by grouping features according to their strong or weak correlations. The pairings generated by this method represent a special case within the set $Pair(S)$ produced by Method 1. The correlation between features is calculated using the Pearson correlation coefficient $\rho_{\mathbf{x}_1,\mathbf{x}_2} = cov(\mathbf{x}_1, \mathbf{x}_2)/\sigma_{\mathbf{x}_1}\sigma_{\mathbf{x}_2}$, where $cov(\mathbf{x}_1, \mathbf{x}_2)$ is the covariance of $\mathbf{x}_1$ and $\mathbf{x}_2$, and $\sigma_{\mathbf{x}_1}$ and $\sigma_{\mathbf{x}_2}$ are the standard deviations of $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. The Pearson coefficient measures the strength of the linear relationship between features. When features are paired based on strong correlations, the goal is to maximize the use of the linear relationship information. However, since most of the operations in the OPNs framework involve nonlinear computations, pairing features based on weak correlations aims to minimize linear relationships and instead leverage the information derived from the nonlinear interactions between features.

C. OPNs-LASSO for Feature Selection

In addition to the aforementioned heuristic and correlation-based pairing strategies, we introduce a more automated and scalable method for feature selection based on OPNs-LASSO regression. Efficient feature selection remains a critical challenge for large-scale tabular data, with recent approaches optimizing cutoff points [45] or employing class-level weighting strategies [46]. Analogous to standard LASSO [47] and its recent robust variants [48,49], which perform both regression and feature selection by shrinking some coefficients to zero, the proposed OPNs-LASSO operates within the OPNs framework and is capable of selecting informative OPNs features while discarding redundant or unimportant ones.

Formally, suppose the set of all candidate OPNs features (i.e., pairwise combinations of original features) is denoted as $\widetilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_p]$, and their corresponding OPNs-LASSO parameters are $\widetilde{\boldsymbol{\beta}} = [\tilde{\beta}_1, \tilde{\beta}_2, ..., \tilde{\beta}_p]^T$. The OPNs-LASSO objective is defined as:

$$\min_{\widetilde{\boldsymbol{\beta}}} \left\| \widetilde{\mathbf{y}} - \widetilde{\mathbf{X}}\widetilde{\boldsymbol{\beta}} \right\|^2 + \tilde{\lambda} \sum_{j=1}^{p} \left\| \tilde{\beta}_j \right\|, \quad (22)$$

where $|\cdot|$ denotes the appropriate norm under the OPNs definition, and $\tilde{\lambda} > \tilde{0}$ is a regularization parameter. The process begins with pairing based on Pairing Method 2, as introduced in the previous subsection. All resulting feature pairs, including higher-order polynomial terms, are then fed into the model. Similar to its real-valued counterpart, this formulation encourages sparsity in $\widetilde{\boldsymbol{\beta}}$, effectively assigning $\tilde{0}$ weights to less informative OPNs features.

Through this mechanism, OPNs-LASSO efficiently narrows the candidate feature space without requiring exhaustive enumeration. It serves as a pre-processing module before training OPNs-LR, significantly reducing computational complexity. In our experimental pipeline (see revised Fig. 4), OPNs features are first generated from the original data, and then OPNs-LASSO is applied to identify a compact yet effective subset of features, which are subsequently used in OPNs-LR for regression modeling.

Certainly, these methods are just a starting point, and other systematic or unsystematic approaches exist. Each method may have different applicability to various datasets and linear regression models, and research on this theory is still ongoing. Further exploration of these relationships can deepen our understanding of practical application issues.

V. Experimental results and analysis

A. Evaluation Methods and Experimental Data

This section presents experiments to evaluate the proposed OPNs-based linear regression model and validate its pairing strategy. Performance is assessed using four standard metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2), defined as follows:

$$\begin{aligned} \mathrm{MSE} &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \qquad \mathrm{R}^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \\ \mathrm{MAE} &= \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \qquad \mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \end{aligned} \quad (23)$$

Here, n is the number of samples, y_i is the observed value, and $\hat{y}_i$ is the predicted value. The value range of MSE, MAE, and RMSE is $[0, \infty)$, with smaller values indicating better model performance. In contrast, a larger R^2 is better, with the ideal value being 1.

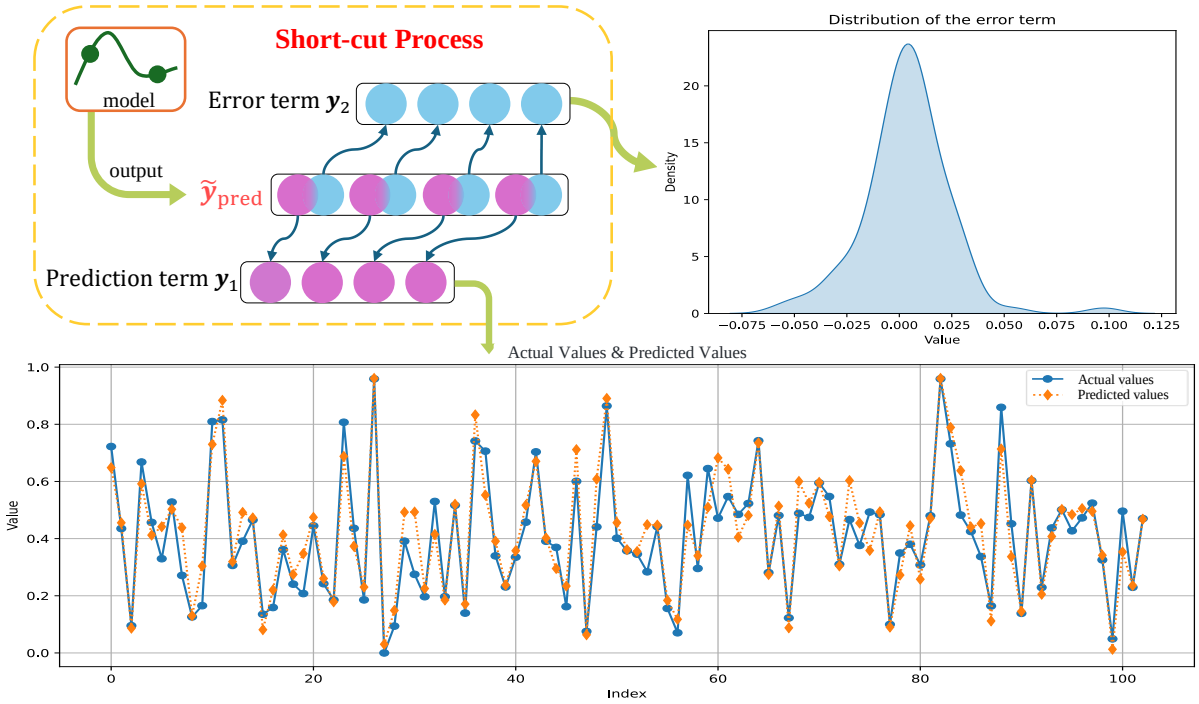

Fig. 7: The actual test data (represented by the orange dashed line) is extracted from one-fifth of the Boston dataset, while the predicted results (depicted by the blue solid line) are generated by the trained OPNs-LR model.

TABLE I: Number of publicly available dataset samples and features

Datasets	Samples	Features
Abalone	4177	8
Boston House	506	13
California House	20640	9
Concrete	1030	8
Diabetes	768	8
Folds5x2	9568	4
Wine Quality	6497	12
Energy Efficiency	766	8
Bike Sharing	731	9
Yacht	308	6

TABLE II: Parameter Settings for Compared Regression Algorithms

Algorithm	Parameter Configuration
LASSO	`alpha=0.01`
OKRidge	`k=10, lambda=10`
ModernNCA	`dim=128, dropout=0.1, d_block=512, n_blocks=0`
TabPFN v2	N/A
PTaRL	`d_layers=[256, 256], dropout=0.2, n_clusters=20, d_embedding=64`
TANGOS	`d_layers=[256,256,256], dropout=0.1, lambda1=1, lambda2=0.01, subsample=50`
kHNN	`epoch=200, lr=0.015`
ENNreg	`K=30, nstart=100, c=1, lambda=0.9`
FGBR	`xi=0, rho=0, defuz_method='WABL', opt_index=0.5, distance='D4', learning_rate=0.1, tree_depth=1`
AMADNN	`operation='add', activation='swish'`
OPNs-SR	`backward=0.9, forward=0.001`

The datasets listed in Table I, including California Housing, Boston Housing, and Wine Quality, were obtained from Kaggle and used for regression analysis. Standard preprocessing steps such as cleaning, encoding, and normalization were applied. To ensure comparability of MSE, MAE, and RMSE across datasets, target values were scaled to the $[0, 1]$ range. The experiments evaluate model performance and examine the effect of OPNs-based feature pairing.

The proposed method was compared against a diverse set of eleven state-of-the-art and baseline regression algorithms, implemented using official code releases or standard Python libraries. This comprehensive benchmark includes advanced deep learning models such as Modern Neighborhood Component Analysis (ModernNCA) [50], Tabular Prior-data Fitted Network v2 (TabPFN v2) [17], Prototype-based Tabular Representation Learning (PTaRL) [14], and Tabular Neural Gradient Orthogonalization and Specialization (TANGOS) [23]. The implementations for these four models are detailed in the TALENT study [51–53]. Additional models in the comparison include Hypercomplex-valued Neural Networks (KHNN) [5] and an Adaptive Multiscale Attention Deep Neural Network (AMADNN) [54]. The comparison also features recent specialized techniques including Optimal k-Sparse Ridge Regression (OKRidge) [24], an evidential neural network for regression (ENNreg) [12], Fuzzy Gradient Boosting Regression (FGBR) [13], and OPNs-based Stepwise Regression (OPNs-SR) [44], alongside the well-established LASSO regression baseline [47]. Parameter settings for these models followed the recommendations from their original publications or widely adopted defaults to ensure a fair and rigorous comparison, as summarized in Table II.

Deep learning models were trained on an NVIDIA RTX 3090 GPU (24GB), while non-deep learning models ran on an AMD R7-5800H CPU (16GB RAM). All experiments used consistent Python 3.9 and PyTorch 2.8 environments to ensure fairness.

B. *Short-cut Process*

As shown in Fig. 7, after the model generates the output $\tilde{\mathbf{y}}_{\text{pred}}$, it is necessary to isolate the value used for prediction. Suppose that when constructing the OPNs response $\tilde{y}$ before training, the original dataset's y is used as the first component.

TABLE III: Top 5 Best-Performing Pairings (4 OPNs Features)

Pairing Combinations	R^2	MSE	MAE	RMSE
(1, 2) (3, 8) (4, 5) (6, 7)	0.594±0.005	0.018±0.002	0.105±0.007	0.133±0.008
(1, 3) (2, 5) (4, 6) (7, 8)	0.574±0.007	0.018±0.002	0.105±0.007	0.134±0.009
(1, 3) (2, 8) (4, 5) (6, 7)	0.587±0.007	0.018±0.003	0.105±0.008	0.133±0.010
(1, 5) (2, 3) (4, 6) (7, 8)	0.577±0.003	0.018±0.002	0.105±0.008	0.134±0.009
(1, 8) (2, 3) (4, 5) (6, 7)	0.587±0.003	0.018±0.002	0.105±0.008	0.132±0.009

TABLE IV: Bottom 5 Worst-Performing Pairings (4 OPNs Features)

Pairing Combinations	R^2	MSE	MAE	RMSE
(1, 6) (2, 7) (3, 4) (5, 8)	0.344±0.005	0.029±0.004	0.139±0.010	0.169±0.011
(1, 6) (2, 7) (3, 5) (4, 8)	0.323±0.007	0.029±0.003	0.139±0.008	0.169±0.009
(1, 6) (2, 7) (3, 8) (4, 5)	0.354±0.007	0.029±0.003	0.139±0.009	0.170±0.010
(1, 7) (2, 6) (3, 5) (4, 8)	0.352±0.003	0.028±0.004	0.139±0.009	0.168±0.010
(1, 7) (2, 6) (3, 8) (4, 5)	0.335±0.003	0.028±0.004	0.139±0.009	0.168±0.011

In this case, the first component of each $\tilde{y}_{\text{pred}}^{(i)}$ in the model output corresponds to the actual value predicted during fitting. The second component can be considered an error term for the first component, and its overall distribution may be viewed as normally distributed. In practice, the fitting performance obtained by summing both components as the prediction value is nearly identical to that obtained by using only the first component as the prediction value.

C. The impact of different pairings on the model

To investigate how different OPNs feature pairing combinations affect model performance, we conduct two groups of experiments using the "Concrete" dataset, which contains only eight features. For convenience, these features are indexed as 1 through 8. Since OPNs features are constructed from ordered pairs, there are $7 \times 5 \times 3 = 105$ distinct 4-pair combinations using disjoint feature pairings, allowing comprehensive comparison without repetition.

We first fix the number of OPNs features to 4 and evaluate model performance under all valid pairing combinations. The model is a simple multivariate linear regression constructed under the OPNs framework. Tables III and IV show the top 5 and bottom 5 combinations in terms of R^2 and error metrics.

Even with a fixed number of features, the choice of pairing combinations significantly affects model performance. On average, the best-performing combinations yield an R^2 that is 0.242 higher and an RMSE that is 0.036 lower than the worst-performing ones. Certain low-performing patterns, such as the use of pairs like (1,6), (1,7), (2,6), and (2,7), consistently appear in the bottom set, suggesting some pairings encode less relevant or redundant information.

The impact of the number of OPNs features is also investigated. As shown in Fig. 8, the model starts with 4 OPNs features, incrementally increasing up to the full set of 28. Performance improves consistently and exceeds that of conventional polynomial regression once 17 OPNs features

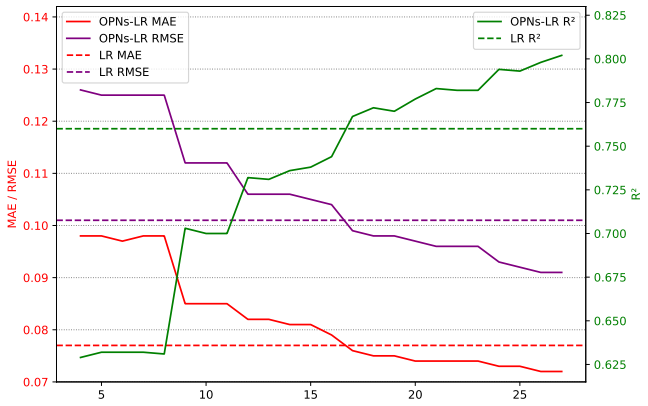

Fig. 8: The influence of different OPNs feature numbers on the model

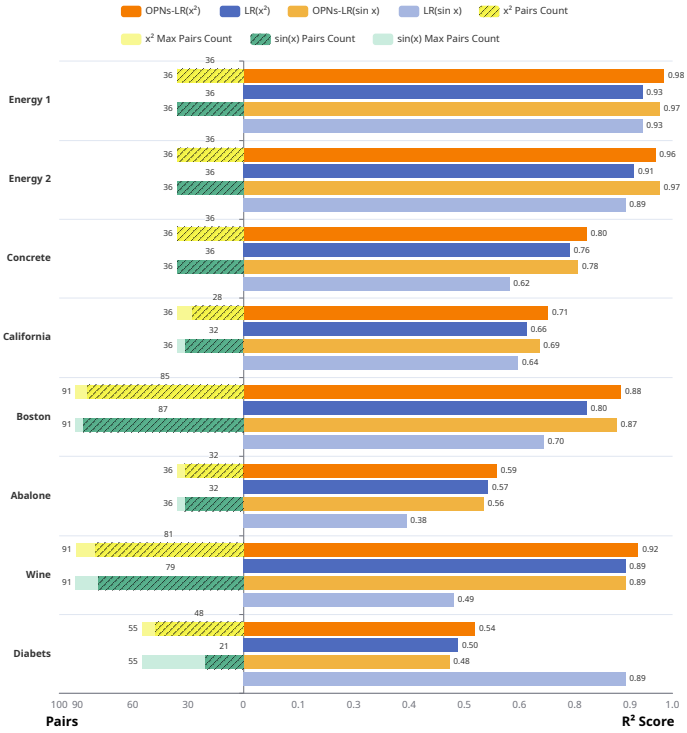

Fig. 9: The right half shows the results obtained by OPNs-LR and LR, computed using the expressions $x + x^2$ and $x + \sin x$, respectively. The left half of the bar represents the number of feature pairs used by OPNs-LR to achieve the corresponding result, with the dark portion indicating the actual number of pairs used, and the light portion representing the maximum number of pairs achievable.

are included. However, since no regularization is applied, the risk of overfitting increases beyond this point.

These results confirm that the structure of pairing combinations plays a central role in OPNs-based modeling. They not only determine how input information is encoded but also influence how effectively the model captures inter-feature interactions.

D. Comparison of the OPNs model with conventional linear regression.

In both univariate and multivariate linear regression, the computations are relatively straightforward. In the OPNs model, even the most complex operations involve only multiplication, thereby preserving linearity. However, the true

TABLE V: Multiple models are compared across different datasets. In each metric, the top-performing results with minor differences are highlighted in bold.

Datasets	Regressor	Avg. R^2 ± Std	Avg. RMSE ± Std	Avg. MAE ± Std	Avg. Time (s) ± Std
Energy_Cooling	LASSO	0.9009 ± 0.0007	0.0806 ± 0.0003	0.0605 ± 0.0002	$\mathbf{0.0231 \pm 0.0296}$
	OKRidge	0.8935 ± 0.0142	0.0832 ± 0.0081	0.0611 ± 0.0044	$\mathbf{0.1400 \pm 0.0100}$
	ModernNCA	$\mathbf{0.9899 \pm 0.0038}$	$\mathbf{0.0253 \pm 0.0047}$	$\mathbf{0.0164 \pm 0.0026}$	$\mathbf{0.8614 \pm 0.2791}$
	TabPFN v2	0.9760 ± 0.0039	0.0395 ± 0.0033	0.0233 ± 0.0019	$\mathbf{0.1649 \pm 0.0262}$
	PTaRL	0.9452 ± 0.0068	0.0597 ± 0.0036	0.0435 ± 0.0018	26.0014 ± 1.7715
	TANGOS	$\mathbf{0.9844 \pm 0.0022}$	$\mathbf{0.0318 \pm 0.0023}$	$\mathbf{0.0208 \pm 0.0018}$	91.9494 ± 15.8697
	kHNN	$\mathbf{0.9868 \pm 0.0088}$	$\mathbf{0.0282 \pm 0.0082}$	$\mathbf{0.0202 \pm 0.0055}$	13.6590 ± 1.7592
	ENNreg	0.9045 ± 0.0158	0.0785 ± 0.0074	0.0544 ± 0.0051	4.4853 ± 0.1247
	FGBR	0.9310 ± 0.0153	0.0670 ± 0.0087	0.0441 ± 0.0062	2.5663 ± 0.2768
	AMADNN	0.9615 ± 0.0100	0.0518 ± 0.0054	0.0311 ± 0.0032	43.5400 ± 1.4900
	OPNs-SR	0.9012 ± 0.0294	0.0793 ± 0.0183	0.0553 ± 0.0150	103.131 ± 9.4000
	OPNs-LR	0.9682 ± 0.0023	0.0454 ± 0.0024	0.0299 ± 0.0020	7.4831 ± 0.1270
Energy_Heating	LASSO	0.9284 ± 0.0004	0.0727 ± 0.0002	0.0551 ± 0.0002	$\mathbf{0.0228 \pm 0.0360}$
	OKRidge	0.9085 ± 0.0117	0.0816 ± 0.0061	0.0575 ± 0.0048	$\mathbf{0.1400 \pm 0.0100}$
	ModernNCA	$\mathbf{0.9979 \pm 0.0001}$	$\mathbf{0.0122 \pm 0.0003}$	$\mathbf{0.0082 \pm 0.0001}$	1.1556 ± 0.6347
	TabPFN v2	$\mathbf{0.9984 \pm 0.0001}$	$\mathbf{0.0107 \pm 0.0003}$	$\mathbf{0.0073 \pm 0.0003}$	$\mathbf{0.1548 \pm 0.0277}$
	PTaRL	0.9734 ± 0.0028	0.0434 ± 0.0023	0.0352 ± 0.0017	25.6681 ± 1.3570
	TANGOS	$\mathbf{0.9916 \pm 0.0187}$	$\mathbf{0.0194 \pm 0.0148}$	$\mathbf{0.0148 \pm 0.0095}$	50.4929 ± 13.3609
	kHNN	$\mathbf{0.9902 \pm 0.0045}$	0.0231 ± 0.0054	0.0235 ± 0.0045	14.5794 ± 1.8196
	ENNreg	0.9411 ± 0.0133	0.0654 ± 0.0095	0.0452 ± 0.0065	4.6325 ± 0.2458
	FGBR	0.9549 ± 0.0266	0.0567 ± 0.0107	0.0413 ± 0.0106	2.2663 ± 0.0805
	AMADNN	0.9712 ± 0.0023	0.0467 ± 0.0019	0.0280 ± 0.0012	43.2500 ± 1.7000
	OPNs-SR	0.9251 ± 0.0239	0.0763 ± 0.0118	0.0571 ± 0.0103	99.4320 ± 5.6443
	OPNs-LR	$\mathbf{0.9967 \pm 0.0011}$	0.0154 ± 0.0024	$\mathbf{0.0115 \pm 0.0020}$	8.1881 ± 0.1775
Folds	LASSO	0.9367 ± 0.0000	0.0569 ± 0.0000	0.0451 ± 0.0000	$\mathbf{0.0417 \pm 0.0080}$
	OKRidge	0.9288 ± 0.0045	0.0610 ± 0.0023	0.0486 ± 0.0015	$\mathbf{0.1300 \pm 0.0100}$
	ModernNCA	$\mathbf{0.9681 \pm 0.0008}$	$\mathbf{0.0400 \pm 0.0005}$	$\mathbf{0.0256 \pm 0.0002}$	5.9721 ± 1.9405
	TabPFN v2	$\mathbf{0.9663 \pm 0.0012}$	0.0411 ± 0.0007	$\mathbf{0.0272 \pm 0.0006}$	$\mathbf{0.2199 \pm 0.0277}$
	PTaRL	0.9356 ± 0.0010	0.0569 ± 0.0004	0.0434 ± 0.0005	414.0346 ± 548.972
	TANGOS	0.9395 ± 0.0006	0.0551 ± 0.0003	0.0406 ± 0.0003	266.8615 ± 47.9511
	kHNN	0.9339 ± 0.0096	0.0579 ± 0.0039	0.0449 ± 0.0031	128.1222 ± 15.9567
	ENNreg	0.9383 ± 0.0041	0.0561 ± 0.0019	0.0440 ± 0.0008	20.5239 ± 0.4428
	FGBR	0.9361 ± 0.0054	0.0573 ± 0.0025	0.0446 ± 0.0012	23.1291 ± 4.4684
	AMADNN	0.9377 ± 0.0018	0.0559 ± 0.0008	0.0439 ± 0.0007	198.3300 ± 5.1200
	OPNs-SR	0.9281 ± 0.0011	0.0606 ± 0.0007	0.0488 ± 0.0007	85.7612 ± 9.1415
	OPNs-LR	0.9447 ± 0.0025	0.0572 ± 0.0010	0.0460 ± 0.0013	3.4775 ± 0.0900
Concrete	LASSO	0.7643 ± 0.0007	0.1010 ± 0.0001	0.0770 ± 0.0001	$\mathbf{0.0467 \pm 0.0184}$
	OKRidge	0.6190 ± 0.4555	$\mathbf{0.0139 \pm 0.0284}$	$\mathbf{0.0010 \pm 0.0020}$	1.1700 ± 0.0300
	ModernNCA	0.8873 ± 0.0130	0.0661 ± 0.0038	0.0448 ± 0.0021	$\mathbf{0.6442 \pm 0.2179}$
	TabPFN v2	$\mathbf{0.9330 \pm 0.0030}$	0.0510 ± 0.0012	0.0290 ± 0.0005	$\mathbf{0.1601 \pm 0.0267}$
	PTaRL	0.8172 ± 0.0105	0.0843 ± 0.0024	0.0707 ± 0.0022	18.9482 ± 1.4575
	TANGOS	0.8649 ± 0.0081	0.0714 ± 0.0022	0.0573 ± 0.0030	63.7877 ± 13.5700
	kHNN	0.8832 ± 0.0327	0.0697 ± 0.0086	0.0508 ± 0.0053	17.0313 ± 2.4290
	ENNreg	0.8433 ± 0.0305	0.0813 ± 0.0077	0.0618 ± 0.0045	5.5021 ± 0.1839
	FGBR	0.8047 ± 0.0254	0.0918 ± 0.0076	0.0715 ± 0.0055	3.8527 ± 1.4745
	AMADNN	0.8829 ± 0.0560	0.0694 ± 0.0137	0.0530 ± 0.0094	121.0500 ± 77.360
	OPNs-SR	0.5964 ± 0.0351	0.1376 ± 0.0096	0.1074 ± 0.0073	89.1323 ± 34.1274
	OPNs-LR	0.8686 ± 0.0121	0.0705 ± 0.0054	0.0545 ± 0.0042	14.4936 ± 0.1268
Diabetes	LASSO	0.4495 ± 0.0635	0.1707 ± 0.0090	0.1383 ± 0.0084	$\mathbf{0.0009 \pm 0.0001}$
	OKRidge	0.4831 ± 0.0578	0.1724 ± 0.0105	0.1399 ± 0.0097	$\mathbf{0.1500 \pm 0.0100}$
	ModernNCA	0.4868 ± 0.0146	$\mathbf{0.1687 \pm 0.00248}$	0.1395 ± 0.0032	$\mathbf{0.2012 \pm 0.1288}$
	TabPFN v2	$\mathbf{0.5121 \pm 0.0035}$	0.1752 ± 0.0022	0.1453 ± 0.0017	10.7730 ± 1.3423
	PTaRL	0.4670 ± 0.0202	0.1719 ± 0.0033	0.1412 ± 0.0030	10.0430 ± 1.3512
	TANGOS	0.4461 ± 0.0141	0.1752 ± 0.0022	0.1453 ± 0.0017	10.7730 ± 1.3423
	kHNN	0.1406 ± 0.1315	0.4375 ± 0.0353	0.3325 ± 0.0276	4.9471 ± 0.3101
	ENNreg	0.3379 ± 0.1170	0.1913 ± 0.0200	0.1461 ± 0.0158	4.7420 ± 0.3105
	FGBR	0.4223 ± 0.1202	0.1815 ± 0.0158	0.1515 ± 0.0153	2.1071 ± 0.2145
	AMADNN	0.4265 ± 0.0287	0.1717 ± 0.0043	0.1319 ± 0.0040	38.1100 ± 1.2900
	OPNs-SR	0.4309 ± 0.0011	$\mathbf{0.1688 \pm 0.0001}$	0.1338 ± 0.0001	63.5283 ± 10.9732
	OPNs-LR	$\mathbf{0.5262 \pm 0.0141}$	$\mathbf{0.1605 \pm 0.0037}$	$\mathbf{0.1282 \pm 0.0035}$	5.6458 ± 0.1675
Wine	LASSO	0.9006 ± 0.0029	0.1218 ± 0.0018	0.0953 ± 0.0012	$\mathbf{0.0510 \pm 0.0104}$
	OKRidge	0.8694 ± 0.0315	0.1360 ± 0.0152	0.1075 ± 0.0115	2.4100 ± 0.6200
	ModernNCA	$\mathbf{0.9882 \pm 0.0064}$	$\mathbf{0.0088 \pm 0.0115}$	$\mathbf{0.0019 \pm 0.0028}$	1.2573 ± 1.4060
	TabPFN v2	$\mathbf{0.9915 \pm 0.0007}$	$\mathbf{0.0102 \pm 0.0037}$	$\mathbf{0.0040 \pm 0.0009}$	$\mathbf{0.1626 \pm 0.0361}$
	PTaRL	0.9447 ± 0.0207	0.0667 ± 0.0126	0.0481 ± 0.0103	9.8096 ± 2.1731
	TANGOS	0.9565 ± 0.0402	$\mathbf{0.0565 \pm 0.0209}$	0.0401 ± 0.0169	13.3015 ± 5.3376
	kHNN	0.9462 ± 0.0408	0.0810 ± 0.0271	0.0557 ± 0.0183	1.6215 ± 0.1592
	ENNreg	0.9049 ± 0.0542	0.1109 ± 0.0233	0.0822 ± 0.0202	5.5839 ± 0.2253
	FGBR	0.8400 ± 0.0883	0.1484 ± 0.0349	0.1168 ± 0.0311	1.7239 ± 1.3610
	AMADNN	$\mathbf{0.9745 \pm 0.0137}$	0.0588 ± 0.0159	$\mathbf{0.0297 \pm 0.0068}$	38.0400 ± 2.8500
	OPNs-SR	0.8059 ± 0.0231	0.1752 ± 0.0204	0.1532 ± 0.0193	40.6800 ± 0.4900
	OPNs-LR	0.9263 ± 0.0128	0.1092 ± 0.0060	0.0737 ± 0.0058	3.2608 ± 0.0287

TABLE V *(continue)*

Datasets	Regressor	Avg. R^2 ± Std	Avg. RMSE ± Std	Avg. MAE ± Std	Avg. Time (s) ± Std
Boston	LASSO	0.8026 ± 0.0045	0.0907 ± 0.0010	0.0616 ± 0.0005	$\mathbf{0.0223 \pm 0.0144}$
	OKRidge	0.7117 ± 0.0502	0.1090 ± 0.0144	0.0778 ± 0.0083	$\mathbf{0.1600 \pm 0.0000}$
	ModernNCA	$\mathbf{0.9133 \pm 0.0236}$	$\mathbf{0.0591 \pm 0.0079}$	$\mathbf{0.0441 \pm 0.0059}$	$\mathbf{0}.3996 \pm 0.1713$
	TabPFN v2	$\mathbf{0.9392 \pm 0.0018}$	$\mathbf{0.0556 \pm 0.0008}$	$\mathbf{0.0387 \pm 0.0006}$	$\mathbf{0.1729 \pm 0.0261}$
	PTaRL	0.6641 ± 0.4133	0.0982 ± 0.0641	0.0704 ± 0.0431	12.6607 ± 4.3997
	TANGOS	$\mathbf{0.9035 \pm 0.0153}$	$\mathbf{0.0627 \pm 0.0048}$	$\mathbf{0.0437 \pm 0.0050}$	17.3980 ± 5.6691
	kHNN	0.8423 ± 0.1391	0.0743 ± 0.0186	0.0516 ± 0.0090	9.6059 ± 0.9479
	ENNreg	0.8722 ± 0.0522	0.0710 ± 0.0140	$\mathbf{0.0489 \pm 0.0056}$	5.6487 ± 0.2794
	FGBR	-1.4683 ± 1.6210	0.2883 ± 0.0980	0.2733 ± 0.1092	2.0440 ± 0.1547
	AMADNN	0.7429 ± 0.2258	0.1034 ± 0.0488	0.0699 ± 0.0434	40.4600 ± 3.4100
	OPNs-SR	0.7412 ± 0.3084	0.1065 ± 0.0503	0.0710 ± 0.0472	70.0600 ± 0.0142
	OPNs-LR	0.8768 ± 0.0165	0.0760 ± 0.0089	0.0520 ± 0.0035	15.2521 ± 1.0400
Yacht	LASSO	0.9205 ± 0.0009	0.0684 ± 0.0004	0.0565 ± 0.0003	$\mathbf{0.0095 \pm 0.0188}$
	OKRidge	0.6263 ± 0.0413	0.1461 ± 0.0156	0.1181 ± 0.0102	$\mathbf{0.4600 \pm 0.4800}$
	ModernNCA	$\mathbf{0.9961 \pm 0.0031}$	$\mathbf{0.0117 \pm 0.0057}$	$\mathbf{0.0067 \pm 0.0028}$	$\mathbf{0.3766 \pm 0.2551}$
	TabPFN v2	$\mathbf{0.9994 \pm 0.0002}$	$\mathbf{0.0050 \pm 0.0008}$	$\mathbf{0.0024 \pm 0.0003}$	$\mathbf{0.1511 \pm 0.0256}$
	PTaRL	0.8916 ± 0.2577	0.0521 ± 0.0440	0.0305 ± 0.0354	42.7154 ± 10.6659
	TANGOS	0.9729 ± 0.0070	$\mathbf{0.0295 \pm 0.0050}$	$\mathbf{0.0188 \pm 0.0029}$	9.2972 ± 2.8502
	kHNN	0.9888 ± 0.0127	$\mathbf{0.0223 \pm 0.0122}$	$\mathbf{0.0139 \pm 0.0073}$	5.7274 ± 0.8086
	ENNreg	0.9525 ± 0.0276	0.0483 ± 0.0135	0.0345 ± 0.0106	5.1711 ± 0.9736
	FGBR	0.9892 ± 0.0060	$\mathbf{0.0244 \pm 0.0089}$	$\mathbf{0.0149 \pm 0.0036}$	1.8487 ± 0.4516
	AMADNN	0.8031 ± 0.2216	0.0837 ± 0.0544	0.0561 ± 0.0447	39.5600 ± 3.6100
	OPNs-SR	0.8158 ± 0.0045	0.1259 ± 0.0027	0.1005 ± 0.0022	211.030 ± 119.330
	OPNs-LR	$\mathbf{0.9916 \pm 0.0026}$	$\mathbf{0.0212 \pm 0.0037}$	$\mathbf{0.0139 \pm 0.0018}$	4.3841 ± 0.1423
Bike	LASSO	0.8478 ± 0.0011	0.0869 ± 0.0003	0.0642 ± 0.0003	$\mathbf{0.0231 \pm 0.0124}$
	OKRidge	0.7626 ± 0.0435	0.1098 ± 0.0101	0.0826 ± 0.0070	$\mathbf{0.1400 \pm 0.0100}$
	ModernNCA	$\mathbf{0.9201 \pm 0.0057}$	$\mathbf{0.0602 \pm 0.0022}$	$\mathbf{0.0451 \pm 0.0011}$	$\mathbf{0.6784 \pm 0.2146}$
	TabPFN v2	$\mathbf{0.9391 \pm 0.0019}$	$\mathbf{0.0526 \pm 0.0008}$	$\mathbf{0.0411 \pm 0.0006}$	$\mathbf{0.1615 \pm 0.0265}$
	PTaRL	0.7213 ± 0.3624	0.0959 ± 0.0588	0.0782 ± 0.0507	19.2140 ± 18.4108
	TANGOS	$\mathbf{0.9133 \pm 0.0048}$	$\mathbf{0.0627 \pm 0.0017}$	$\mathbf{0.0487 \pm 0.0018}$	35.8853 ± 6.0040
	kHNN	0.8326 ± 0.0450	0.0895 ± 0.0125	0.0638 ± 0.0084	6.6238 ± 0.3212
	ENNreg	0.8714 ± 0.0485	0.0766 ± 0.0085	0.0541 ± 0.0043	7.9027 ± 0.4139
	FGBR	0.8412 ± 0.0314	0.0889 ± 0.0104	0.0671 ± 0.0065	2.9242 ± 0.4250
	AMADNN	0.8113 ± 0.1990	0.0861 ± 0.0489	0.0586 ± 0.0403	40.5100 ± 3.9800
	OPNs-SR	0.8479 ± 0.0366	0.0746 ± 0.0082	0.0599 ± 0.0075	8284.065 ± 265.348
	OPNs-LR	0.8727 ± 0.0135	0.0807 ± 0.0040	0.0577 ± 0.0025	8.8426 ± 0.1400
Abalone	LASSO	0.5547 ± 0.0028	$\mathbf{0.0768 \pm 0.0002}$	0.0547 ± 0.0000	$\mathbf{0.0746 \pm 0.0500}$
	OKRidge	0.5168 ± 0.0299	0.0800 ± 0.0048	0.0578 ± 0.0032	$\mathbf{0.1300 \pm 0.0100}$
	ModernNCA	$\mathbf{0.5867 \pm 0.0102}$	0.0801 ± 0.0010	0.0549 ± 0.0007	1.7332 ± 0.5576
	TabPFN v2	$\mathbf{0.6065 \pm 0.0020}$	0.0782 ± 0.0002	$\mathbf{0.0522 \pm 0.0001}$	$\mathbf{0.1967 \pm 0.0327}$
	PTaRL	$\mathbf{0.5844 \pm 0.0054}$	0.0803 ± 0.0005	0.0550 ± 0.0004	38.2941 ± 5.1876
	TANGOS	0.5787 ± 0.0127	0.0809 ± 0.0012	$\mathbf{0.0539 \pm 0.0007}$	112.3719 ± 27.9908
	kHNN	0.5225 ± 0.0724	0.0792 ± 0.0061	0.0562 ± 0.0040	62.2243 ± 8.1138
	ENNreg	0.5294 ± 0.0304	0.0785 ± 0.0051	0.0565 ± 0.0030	10.1336 ± 0.4529
	FGBR	0.4500 ± 0.0272	0.0865 ± 0.0055	0.0625 ± 0.0037	11.9349 ± 2.7836
	AMADNN	0.5661 ± 0.1259	0.1164 ± 0.0402	0.0864 ± 0.0332	67.9700 ± 28.6100
	OPNs-SR	0.5320 ± 0.1702	$\mathbf{0.0778 \pm 0.0735}$	0.0596 ± 0.0673	1035.76 ± 204.3654
	OPNs-LR	0.5612 ± 0.0026	$\mathbf{0.0756 \pm 0.0002}$	$\mathbf{0.0546 \pm 0.0001}$	31.6780 ± 0.6777

advantage of OPNs computations emerges in polynomial regression.

Figure 9 illustrates the superior performance of OPNs polynomial regression compared to traditional polynomial regression across different datasets. Each method is evaluated using two polynomial forms, $x + x^2$ and $x + \sin x$, with 10-fold cross-validation performed over 10 iterations per dataset, and the average results are recorded. Additionally, the figure provides information on the number of OPNs feature pairings. It is evident that OPNs polynomial regression consistently outperforms traditional polynomial regression. These results clearly highlight the advantages of algorithms built within the OPNs framework.

Formula (6) reveals that selecting α^n with $n = 2$ yields $\tilde{x}^2 = (-2\mu_x\nu_x, -\mu_x^2 - \nu_x^2)$. This demonstrates that within the OPNs framework, considering only the "x^2" term introduces more information. Feature square terms may be repeated in pairing combinations, forming different linear combinations within $\tilde{x}_i^2$. However, cross-multiplication terms will not be repeated, enriching the information introduced by OPNs polynomial operations. For the "$\sin x$" term, the computation in the OPNs model is $\sin \tilde{x} = (\sin \mu_x \cos \nu_x, \cos \mu_x \sin \nu_x)$. Compared to conventional methods, the OPNs model introduces mixed multiplication terms of $\cos \alpha$ and $\sin \alpha$, lacking an independent $\sin x$ term.

E. Comparison with multiple regression models

Table V summarizes the performance of twelve regression models across nine public benchmark datasets. To rigorously evaluate the OPNs-LR model, higher-order polynomial terms

were incorporated, enabling it to compete with advanced deep learning architectures.

Experimental results, detailed in Table V, demonstrate the competitive overall performance of OPNs-LR. The Friedman test rankings (Table VI) place OPNs-LR third among the twelve models based on predictive accuracy metrics (R^2, RMSE, MAE). Dataset-level analysis (Table V) indicates that while OPNs-LR performs strongly, it is often surpassed by the top two deep learning models, TabPFN v2 and ModernNCA. In such cases, the difference in the coefficient of determination (R^2) is typically between 0.02 and 0.07. Performance parity varies, with OPNs-LR achieving the best score on the Diabetes dataset among all competitors.

A noteworthy observation arises when comparing OPNs-LR to models employing alternative algebraic structures. Table V shows that OPNs-LR frequently exhibits superior predictive accuracy compared to models based on fuzzy numbers (ENNreg, FGBR) or hypercomplex numbers (kHNN) on most tested datasets. Regarding computational cost, OPNs-LR ranks seventh in average runtime according to Table VI. While slower than simpler baselines (LASSO) or highly optimized models (OKRidge, TabPFN v2), its runtime is often comparable to or less than several other complex deep learning methods (e.g., TANGOS, PTaRL, AMADNN), remaining practical for offline training scenarios.

TABLE VI: Comprehensive Ranking of Regression Models. The values represent the average rank of each model across all datasets based on the Friedman test. Lower values indicate better overall performance.

Regressor	R^2	RMSE	MAE	Runtime
TabPFN v2	**1.5**	**2.0**	**1.8**	2.8
ModernNCA	1.9	2.5	2.6	3.7
OPNs-LR	4.1	4.6	4.95	7
TANGOS	4.4	5.1	4.3	10.3
KHNN	6.7	6.5	6.05	7.4
AMADNN	6.9	6.9	6.2	10.6
ENNreg	7.5	7.1	7.4	6.3
PTaRL	7.6	7.65	7.3	9.3
LASSO	8.2	7.85	8.5	**1**
FGBR	9.0	9.3	9.3	5.3
OPNs-SR	9.8	8.8	9.9	11.5
OKRidge	10.4	9.7	9.7	2.8

VI. Conclusion

This study presents a foundational effort to integrate linear regression into the OPNs framework, demonstrating the potential of developing machine learning models beyond the real-number domain. By leveraging the nonlinear computational characteristics of OPNs and a novel feature pairing mechanism, the proposed OPNs-LR model achieves richer feature interactions and structural expressiveness.

Experimental results confirm its competitive performance against a range of state-of-the-art tabular learning models. While not always outperforming the top deep learning architectures, OPNs-LR shows notable efficacy, particularly when compared to other models built on alternative algebraic systems like fuzzy or hypercomplex numbers. This suggests that the OPNs framework provides a valid and promising paradigm for machine learning.

While this work establishes the feasibility of constructing learning models under the OPNs framework, the complexity introduced by the pairing mechanism warrants further investigation. Future research will focus on refining pairing strategies, exploring automatic selection methods, and extending the approach to more complex nonlinear and deep learning models, contributing to the broader development of OPNs-based machine learning theory.

References

[1] F. Galton, "Regression Towards Mediocrity in Hereditary Stature." *The Journal of the Anthropological Institute of Great Britain and Ireland*, vol. 15, pp. 246–263, 1886.

[2] X. Zhu, Y. Xu, H. Xu, and C. Chen, "Quaternion convolutional neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 631–647.

[3] W. Jinhui and J. Yuan, "Universal approximation and approximation advantages of quaternion-valued neural networks," *Journal of Computer Research and Development*, vol. 62, pp. 1–11, 2024.

[4] E. Lopez, E. Grassucci, D. Capriotti, and D. Comminiello, "Towards Explaining Hypercomplex Neural Networks," in *2024 International Joint Conference on Neural Networks (IJCNN)*, Jun. 2024, pp. 1–8.

[5] A. Niemczynowicz and R. A. Kycia, "KHNN: Hypercomplex-valued neural networks computations via Keras using TensorFlow and PyTorch," *SoftwareX*, vol. 30, p. 102163, May 2025.

[6] G. L. Marchetti, V. Shahverdi, S. Mereta, M. Trager, and K. Kohn, "Algebra unveils deep learning – an invitation to neuroalgebraic geometry," *arXiv preprint arXiv:2501.18915*, 2025.

[7] W. Cui, Q. Kang, X. Li, K. Zhao, W. P. Tay, W. Deng, and Y. Li, "Neural variable-order fractional differential equation networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 15, pp. 16 109–16 117, 2025.

[8] Q. Kang, X. Li, K. Zhao, W. Cui, Y. Zhao, W. Deng, and W. P. Tay, "Efficient training of neural fractional-order differential equation via adjoint backpropagation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 17, pp. 17 750–17 759, 2025.

[9] T. Sun and S. M. Bohté, "Average-over-time spiking neural networks for uncertainty estimation in regression," *arXiv preprint arXiv:2412.00278*, 2024.

[10] G. Nikolentzos, S. Wang, J. Lutzeyer, and M. Vazirgiannis, "Graph neural machine: A new model for learning with tabular data," *arXiv preprint arXiv:2402.02862*, 2024.

[11] A. Alkhatib and H. Boström, "Interpretable graph neural networks for heterogeneous tabular data," in *Discovery Science*. Springer, 2025, pp. 310–324.

[12] T. Denœux, "Quantifying prediction uncertainty in regression using random fuzzy sets: the ennreg model," *IEEE Transactions on Fuzzy Systems*, vol. 31, no. 10, pp. 3690–3699, 2023.

[13] R. Nasiboglu and E. Nasibov, "WABL method as a universal defuzzifier in the fuzzy gradient boosting regression model," *Expert Systems with Applications*, vol. 212, p. 118771, Feb. 2023.

[14] H. Ye, W. Fan, X. Song, S. Zheng, H. Zhao, D. dan Guo, and Y. Chang, "PTaRL: Prototype-based Tabular Representation Learning via Space Calibration," in *The Twelfth International Conference on Learning Representations*, Jan. 2024.

[15] S. Li, Y. Lu, S. Jiu, H. Huang, G. Yang, and J. Yu, "Prototype-oriented hypergraph representation learning for anomaly detection in tabular data," *Information Processing & Management*, vol. 62, no. 1, p. 103877, 2025.

[16] J.-P. Jiang, S.-Y. Liu, H.-R. Cai, Q. Zhou, and H.-J. Ye, "Representation learning for tabular data: A comprehensive survey," *arXiv preprint arXiv:2504.16109*, 2025.

[17] N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, S. B. Hoo, R. T. Schirrmeister, and F. Hutter, "Accurate predictions on small data with a tabular foundation model," *Nature*, vol. 637, no. 8045, pp. 319–326, Jan. 2025.

[18] H.-J. Ye, S.-Y. Liu, and W.-L. Chao, "A closer look at tabPFN v2: Understanding its strengths and extending its capabilities," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[19] H. Thimonier, J. L. D. M. Costa, F. Popineau, A. Rimmel, and B.-L. DOAN, "T-JEPA: Augmentation-free self-supervised learning for tabular data," in *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

[20] S. Ghosh, T. Xie, and M. Kuznetsov, "Distributionally robust self-supervised learning for tabular data," in *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024.

[21] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljacic, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov–arnold networks," in *The Thirteenth International Conference on Learning Representations*, 2025.

[22] W. Gao, Z. Gong, Z. Deng, and L. Ma, "Revisiting the numerical feature embeddings structure in neural network-based tabular modelling," *Knowledge-Based Systems*, vol. 330, p. 114697, 2025.

[23] A. Jeffares, T. Liu, J. Crabbé, F. Imrie, and M. van der Schaar, "TANGOS: Regularizing Tabular Neural Networks through Gradient Orthogonalization and Specialization," in *The Eleventh International Conference on Learning Representations*, Feb. 2023.

[24] J. Liu, S. Rosen, C. Zhong, and C. Rudin, "OKRidge: Scalable Optimal k-Sparse Ridge Regression," *Advances in neural information processing systems*, vol. 36, pp. 41 076–41 258, Dec. 2023.

[25] C.-T. Li, Y.-C. Tsai, C.-Y. Chen, and J. C. Liao, "Graph neural networks for tabular data learning: A survey with taxonomy and directions," *ACM Computing Surveys*, vol. 58, no. 1, pp. 1–51, 2025.

[26] Y. Song, Y. Gu, T. Li, J. Qi, Z. Liu, C. S. Jensen, and G. Yu, "Chgnn: A semi-supervised contrastive hypergraph learning network," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 9, pp. 4515–4530, 2024.

[27] D. Qi, W. Zheng, and J. Wang, "FeatAug: Automatic feature augmentation from one-to-many relationship tables," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 1805–1818.

[28] G. Xiong, S. Sinha, and A. Zhang, "ProtoNAM: Prototypical neural additive models for interpretable deep tabular learning," *ACM Transactions on Knowledge Discovery from Data*, vol. 19, no. 9, pp. 1–25, 2025.

[29] Y. Shi, W. Xu, M. Jin, H. Zhang, Q. Wu, Y. Zhang, and M. Xu, "Beyond KAN: Introducing KarSein for adaptive high-order feature interaction modeling in CTR prediction," *arXiv preprint arXiv:2408.08713*, 2024.

[30] H. Julkunen and J. Rousu, "Comprehensive interaction modeling with machine learning improves prediction of disease risk in the UK biobank," *Nature Communications*, vol. 16, no. 1, p. 6620, 2025.

[31] A. Kleinau, B. Preim, and M. Meuschke, "FINCH: Locally visualizing higher-order feature interactions in black box models," *arXiv preprint arXiv:2503.16445*, 2025.

[32] R. Nasiboglu and E. Nasibov, "FyzzyGBR—A gradient boosting regression software with fuzzy target values," *Software Impacts*, vol. 14, p. 100430, Dec. 2022.

[33] J. L. C. Bárcena, P. Ducange, F. Marcelloni, and A. Renda, "Increasing trust in ai through privacy preservation and model explainability: Federated learning of fuzzy regression trees," *Information Fusion*, vol. 113, p. 102598, 2025.

[34] M. Yao, T. Zhao, J. Cao, and J. Li, "Hierarchical fuzzy topological system for high-dimensional data regression problems," *IEEE Transactions on Fuzzy Systems*, 2025, early Access.

[35] C. Xue and M. Mahfouf, "Encfis: An exclusionary neural complex fuzzy inference system for robust regression learning," *IEEE Transactions on Fuzzy Systems*, vol. 32, no. 3, pp. 1539–1552, 2024.

[36] G. Hesamian, A. Johannssen, and N. Chukhrova, "Fuzzy nonlinear regression modeling with radial basis function networks," *IEEE Transactions on Fuzzy Systems*, vol. 32, no. 4, pp. 1733–1742, 2024.

[37] X. Zhu, X. Hu, L. Yang, W. Pedrycz, and Z. Li, "A development of fuzzy-rule-based regression models through using decision trees," *IEEE Transactions on Fuzzy Systems*, vol. 32, no. 5, pp. 2976–2986, 2024.

[38] R. Boloix-Tortosa, J. J. Murillo-Fuentes, I. Santos, and F. Pérez-Cruz, "Widely Linear Complex-Valued Kernel Methods for Regression," *IEEE Transactions on Signal Processing*, vol. 65, no. 19, pp. 5240–5248, Oct. 2017.

[39] N. Chepurko, K. Clarkson, L. Horesh, H. Lin, and D. Woodruff, "Quantum-inspired algorithms from randomized numerical linear algebra," in *International Conference on Machine Learning*. PMLR, 2022, pp. 3879–3900.

[40] L. Zhou, "Ordered pair of normalized real numbers," *Information Sciences*, vol. 538, pp. 290–313, Oct. 2020.

[41] H. Cui, H. Zhang, L. Zhou, C. Yang, B. Li, and X. Zhao, "Fuzzy analytic hierarchy process with ordered pair of normalized real numbers," *Soft Computing*, vol. 27, no. 17, pp. 12 267–12 288, Sep. 2023.

[42] Y. Zheng, X. Ding, X. Zhao, X. Pan, and L. Zhou, "K-Nearest Neighbor Algorithm Based on the Framework of Ordered Pair of Normalized Real Numbers," *IEEE Transactions on Artificial Intelligence*, pp. 1–16, 2025.

[43] M. Chen, Y. Zheng, X. Pan, and L. Zhou, "Generalized-metric-based pattern recognition using ordered pair of normalized real numbers," *Applied Artificial Intelligence*, vol. 39, no. 1, p. 2590815, 2025.

[44] Y. Huang, Y. Zheng, X. Pan, and L. Zhou, "Stepwise regression algorithm based on the ordered pair of normalized real numbers framework," *The Journal of Supercomputing*, vol. 81, no. 8, p. 900, May 2025.

[45] A. Hu, D. Prasad, L. Pizzato, N. Foord, A. Abrahamyan, A. Leontjeva, C. Doyle, and D. Jermyn, "Featurecuts: Feature selection for large data by optimizing the cutoff," *arXiv preprint arXiv:2508.00954*, 2025.

[46] A. Starkey, U. I. Akpan, O. AL Hosni, and Y. Pullissery, "Class-level feature selection method using feature weighted growing self-organising maps," *arXiv preprint arXiv:2503.11732*, 2025.

[47] R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, Jan. 1996.

[48] C.-H. Liu and G. Novikov, "Robust sparse regression with non-isotropic designs," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, 2024, pp. 270–305.

[49] G. Chandrasekaran, R. Meka, and K. Stavropoulos, "Sparse linear regression is easy on random supports," *arXiv preprint arXiv:2511.06211*, 2025.

[50] H.-J. Ye, H.-H. Yin, D.-C. Zhan, and W.-L. Chao, "Revisiting Nearest Neighbor for Tabular Data: A Deep Tabular Baseline Two Decades Later," in *The Thirteenth International Conference on Learning Representations*, Jan. 2025.

[51] H.-J. Ye, S.-Y. Liu, H.-R. Cai, Q.-L. Zhou, and D.-C. Zhan, "A closer look at deep learning on tabular data," *arXiv preprint arXiv:2407.00956*, 2024.

[52] S.-Y. Liu, H.-R. Cai, Q.-L. Zhou, H.-H. Yin, T. Zhou, J.-P. Jiang, and H.-J. Ye, "Talent: A tabular analytics and learning toolbox," *Journal of Machine Learning Research*, vol. 26, no. 226, pp. 1–16, 2025.

[53] S.-Y. Liu, H.-R. Cai, Q.-L. Zhou, and H.-J. Ye, "Talent: A tabular analytics and learning toolbox," *arXiv preprint arXiv:2407.04057*, 2024.

[54] V. Dentamaro, P. Giglio, D. Impedovo, G. Pirlo, and M. D. Ciano, "An Interpretable Adaptive Multiscale Attention Deep Neural Network for Tabular Data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 4, pp. 6995–7009, Apr. 2025.