Enhancing Regression Accuracy and Reliability with Ensemble Learning and Iterative Hyperparameter Optimization

Thi Giang Thanh Tran
Faculty of Electrical Engineering Technology, Industrial University of Ho Chi Minh City, Vietnam
tranthigiangthanh@iuh.edu.vn

Thanh Ngoc Tran
Faculty of Electrical Engineering Technology, Industrial University of Ho Chi Minh City, Vietnam
tranthanhngoc@iuh.edu.vn (corresponding author)

Received: 3 April 2025 | Revised: 23 April 2025, 5 May 2025, and 12 May 2025 | Accepted: 15 May 2025

 DOI: https://doi.org/10.48084/etasr.11266

ABSTRACT

Ensemble learning models such as XGBoost, LightGBM, and CatBoost are gaining popularity for regression predictions. As their accuracy significantly depends on hyperparameters, identifying the optimal hyperparameters plays a crucial role. This study proposes a new approach by integrating ensemble learning models with hyperparameter optimization algorithms to assess their effectiveness compared to traditional machine learning models such as RF, SVR, and KNN. This study also conducted multiple repetitions of the hyperparameter optimization process on the Boston Housing Price and Bike Sharing Demand datasets, supported by statistical analyses of the results to enhance the reliability of the findings compared to a single run. The results demonstrate that applying ensemble learning models helps reduce prediction errors, increasing the models' accuracy compared to traditional ML models. Furthermore, comparisons indicate that repeating iterations *n* times improves the reliability of the results.

Keywords-XGBoost; LightGBM; CatBoost; HPO Algorithms; ML

I. INTRODUCTION

In recent decades, Machine Learning (ML) has undergone rapid development and has attracted considerable interest from across the research community. ML models include supervised learning (where input data are labeled with known outcomes), unsupervised learning (where input data lack labels or known outcomes), and semi-supervised learning, which involves a mixture of labeled and unlabeled examples [1-2]. ML models have proven to be powerful for a wide range of tasks, including classification, clustering, and regression. In particular, regression tasks have benefited from various ML models, notably Random Forest (RF) [3, 4], Support Vector Regression (SVR) [5, 6], and K-Nearest Neighbors (KNN) [7, 8], which have been effectively utilized to model and predict continuous results.

Recently, there has been a surge in interest in ensemble learning models for regression, with algorithms such as XGBoost, LightGBM, and CatBoost demonstrating promising improvements in predictive performance [9-11]. These gradient-boosting-based ensemble models are based on the principle of combining multiple weak learners to form a stronger predictive model. In [9], a comprehensive comparison of XGBoost, LightGBM, and CatBoost was made on a credit risk dataset, highlighting the relative strengths of each approach. In [10], XGBoost and LightGBM were integrated to predict the temperatures of molten steel in manufacturing, achieving high precision. Furthermore, in [11], the effectiveness of an XGBoost model optimized with a genetic algorithm was demonstrated for load forecasting, achieving superior accuracy compared to traditional methods. Collectively, these studies underscore the adaptability and strong performance of ensemble models across diverse application domains, leveraging their unique strengths to capture complex patterns.

A key aspect affecting the accuracy of ML models is their dependence on hyperparameters, which are configuration parameters that govern the learning process. Identifying optimal hyperparameters is essential to enhance model performance. A variety of algorithms exist for Hyperparameter Optimization (HPO), which can be broadly classified into three categories: Model-free Algorithms (including Random Search and Grid Search), Bayesian Optimization Algorithms (such as Bayesian Optimization with Gaussian Processes and Tree-structured Parzen Estimators), and Metaheuristic Algorithms (such as the Genetic Algorithm and Particle Swarm Optimization). Recent studies have successfully applied these HPO algorithms to improve traditional regression models, such as RF, SVR, and KNN, achieving significant performance

gains [12, 13]. These results show that appropriate hyperparameter tuning can significantly improve the accuracy of ML models in regression tasks.

This study examines the performance of three state-of-the-art ensemble regression models, XGBoost, LightGBM, and CatBoost, in combination with common HPO algorithms: Random Search (RS), Bayesian Optimization using Tree-structured Parzen Estimators (BO-TPE), and Particle Swarm Optimization (PSO). The predictive accuracy of these optimized ensemble models is evaluated and compared to that of traditional ML regression models, including RF, SVR, and KNN. It is important to note that the results of HPO can vary between runs due to the inherent randomness in the search process [13]. This variability arises from the use of random initial seeds and the stochastic nature of procedures such as RS, BO-TPE, and PSO. Therefore, to ensure the reliability and robustness of the results, this study follows an algorithmic workflow in which each HPO algorithm is executed multiple times (n independent runs) for every optimizer–model combination. Statistical analyses are subsequently performed on the distribution of results obtained from these repeated executions. By comparing the aggregated results of multiple HPO algorithms runs with those derived from a single execution, the stability of the optimization process can be evaluated, thereby improving the credibility of the findings. This study employs two benchmark datasets commonly used in ML research: the Boston Housing Price dataset and the Bike Sharing Demand dataset. To statistically evaluate the results across different experimental scenarios, boxplot charts are employed to illustrate their distribution and variability.

II. AN OVERVIEW OF ML MODELS AND HPO ALGORITHMS

A. Machine Learning (ML) Models

ML models, a subfield of artificial intelligence, involve algorithms that can learn from data and make predictions or decisions. ML encompasses various models, broadly categorized into classification, regression, clustering, and other specialized types. In regression problems, where the task is to predict continuous numerical values based on input data, several classical ML models have been widely used, such as RF, SVR, and KNN, which are investigated in this study.

1) Random Forest

RF is a versatile ensemble learning method that constructs many decision trees during training. Introduced by Leo Breiman, it builds each tree independently while randomly selecting features for each split, thereby reducing overfitting and increasing robustness. By combining the predictions of multiple decision trees, RF provides accurate and stable predictions for classification and regression tasks. Several hyperparameters of RF are crucial in determining its performance, including [13-15]:

- *n_estimators* is the number of decision trees generated within the ensemble. Each decision tree represents a submodel in the RF. Increasing this value may lead to better performance but also increases computational complexity.
- *max_depth* denotes the maximum depth of each decision tree in the ensemble. This determines the complexity level of each tree and influences the model's learning and decision-making abilities. A large max_depth value may lead to overfitting.
- *min_samples_leaf* is the minimum number of samples required for a node to be considered a leaf node.
- *criterion* denotes the measure of impurity for a split. Two common values for this hyperparameter are "gini" (using the Gini index) and "entropy" (using information entropy).
- *max_features* is the maximum number of features to consider when searching for the best split at each decision tree node.

2) Support Vector Regression (SVR)

SVR is an effective technique for regression tasks that extends the principles of Support Vector Machines (SVMs) to continuous target variables. Developed by Vladimir Vapnik, SVR aims to find the optimal hyperplane that maximizes the margin while minimizing the error between the predicted and actual values. SVR can capture nonlinear relationships between features and targets by employing a kernel function to map data into a higher-dimensional space. The key hyperparameters of SVR include [13, 15-17]:

- *C* is one of the most important hyperparameters in SVR. This hyperparameter acts as a trade-off between prediction error and model complexity, balancing the degree of penalty imposed on errors during the learning process.
- *Kernel*: This hyperparameter determines the type of kernel function used to find the optimal hyperplane. Common options include linear, polynomial, and Radial Basis Function (RBF).
- Epsilon: This hyperparameter represents the acceptable error level in the optimization process. It defines a tolerance margin where no penalty is given for errors. During optimization, SVR seeks a hyperplane such that all training data points lie within this epsilon distance, effectively controlling the width of the margin.

3) K-Nearest Neighbors (KNN)

KNN is a simple but effective nonparametric classification and regression algorithm, which operates on the principle of similarity, where the prediction for a new data point is based on the majority vote for classification tasks or the average, for regression tasks, of its k nearest neighbors in the feature space [13, 15, 18]. The important hyperparameter of KNN is *n_neighbors*, which determines the number of nearest neighbors the model considers when predicting the label or value of a new data point.

B. Hyperparameter Optimization (HPO) Algorithms

Hyperparameters play a crucial role in optimizing the performance of ML models. Selecting appropriate hyperparameters can significantly improve accuracy, generalization ability, and model efficiency. Therefore, HPO algorithms are essential for building and applying ML models.

Numerous HPO algorithms exist within the field of ML. This study investigates the following algorithms: RS, BO-TPE, and PSO algorithms.

C. Random Search (RS)

RS is a method used for HPO in ML. It involves randomly selecting hyperparameters for a model from a predefined search space. Instead of exhaustively searching through all possible combinations, RS samples a fixed number of sets to evaluate the model's performance. The RS process consists of four key steps [13, 19]:

- Defining the search space: Define the range of possible values for each hyperparameter.
- Random sampling: Sample sets of hyperparameters from the search space.
- Model evaluation: Evaluate the model's performance on each sampled set.
- Selection: Select the set that yields the best performance on the validation dataset.

RS is simple to implement and easy to understand, making it suitable for beginners in hyperparameter optimization. Compared to exhaustive search methods such as Grid Search (GS), it can efficiently find good configurations with fewer trials. Additionally, it is flexible and can be applied to a wide range of ML models.

D. Bayesian Optimization Using Tree-structured Parzen Estimators (BO-TPE)

BO-TPE leverages Bayesian optimization principles and Tree-structured Parzen Estimator probabilistic modeling. By iteratively selecting hyperparameter configurations based on TPE's distributions of regions of high and low performance, BO-TPE efficiently explores the hyperparameter space, seeking and exploiting promising values to find even better configurations. This approach is efficient in high-dimensional search spaces or when evaluating the objective function, which is computationally expensive. BO-TPE offers efficiency and effectiveness for hyperparameter optimization tasks by adaptively sampling hyperparameters and updating its model with observed performance to guide further exploration toward the most promising regions [13, 20].

E. Particle Swarm Optimization (PSO)

PSO is a metaheuristic optimization algorithm inspired by the social behavior observed in bird flocks or fish schools. PSO operates by maintaining a population of particles, each representing a potential solution to the optimization problem. These particles traverse the search space, exchanging information among themselves and adjusting their positions based on their own best-known positions (p_{best}) and the globally best-known position (g_{best}) among their neighbors. The PSO process consists of the following main steps [13, 21]:

- Initialization: Initialize a population of particles with random positions and velocities.
- Evaluation: Evaluate the fitness of each particle using the objective function.
- Velocity update: Update the velocity of each particle based on its personal best and global best positions.
- Position update: Update the position of each particle based on its velocity and current position.
- Repeat: Repeat steps 2-4 until a termination criterion is met.

III. PROPOSED METHOD

A. The Proposed Ensemble Learning Model

Ensemble learning combines diverse ML models to create a more accurate predictive model [22, 23]. Among ensemble learning models, XGBoost, LightGBM, and CatBoost are prominent and widely used due to their high performance, flexibility, and ease of use.

1) XGBoost

XGBoost, introduced in 2016 by Tianqi Chen, harnesses the power of gradient-boosting principles to deliver exceptional performance across a wide range of ML tasks [24, 25]. Incorporating a second-order Taylor expansion for the loss function and integrating regularization techniques leads to faster convergence, mitigates overfitting, and enhances forecasting accuracy. Its computational efficiency, bolstered by sophisticated data preprocessing and result caching, enables parallel execution. These features make XGBoost exceptionally suited for handling large and complex datasets. Its versatility across classification, regression, and ranking tasks cements its status as a top choice within the ML community.

2) LightGBM

Developed by Microsoft Research Asia in 2017, LightGBM significantly enhances the performance of Gradient Boosting Decision Trees (GBDT) by introducing two novel techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) [24, 26]. GOSS prioritizes data points with more significant gradients during training, leading to more efficient data utilization and accelerated learning. EFB, on the other hand, addresses computational complexity by grouping highly correlated features, resulting in streamlined operations and improved model efficiency. These advances render LightGBM particularly well-suited for large-scale datasets and computationally intensive tasks.

3) CatBoost

Introduced by Prokhorenkova et al. in 2018, CatBoost is a robust gradient-boosting algorithm designed for efficiently handling categorical features [24, 27]. Utilizing binary decision trees as base learners, it employs advanced techniques such as permutation methods, one-hot-max-size encoding, and target-based statistics. These techniques improve the handling of categorical data, thus boosting prediction accuracy.

4) Hyperparameters of Ensemble models

Like the hyperparameters of traditional ML models, ensemble learning models also have hyperparameters that can significantly impact their effectiveness. The important hyperparameters of XGBoost, LightGBM, and CatBoost models used for HPO algorithms in this study are [28]:

- *n_estimators*: The number of decision trees in the model. A higher value typically increases accuracy but also raises the risk of overfitting. Conversely, a lower value decreases accuracy but can simplify and speed up the model.
- *max_depth*: The maximum depth of a decision tree. It determines the complexity of the generated decision trees.
- *learning_rate*: Determines the model's sensitivity to weight updates.
- *subsample*: The ratio of the data sample used to train each tree. It aids in reducing overfitting by introducing randomness into the training process.
- *colsample_bytree*: Denotes the fraction of features used to train each decision tree. This parameter is utilized in XGBoost and LightGBM models.
- *colsample_bylevel*: The ratio of features used to split each node in the decision tree. This is a specific feature of CatBoost models.

This study investigates the effectiveness of ensemble learning models compared to traditional ML approaches. Traditional and ensemble ML models were integrated with various HPO algorithms to achieve this. Figure 1 presents a schematic diagram that illustrates the HPO algorithms. The inputs to the HPO algorithms consist of the dataset (X, y), the structure of the ML models (mdl), and the range of values for the corresponding hyperparameters. The HPO algorithms are integrated with a cross-validation procedure (k-fold = 3). This approach ensures that the models are validated across multiple data subsets, thus mitigating overfitting and providing a more reliable estimate of their generalization capacity [10]. By analyzing and comparing the MSE values, the effectiveness of the ensemble learning models can be compared with that of traditional models. The mathematical equation for MSE can be formulated as follows [29]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (1)$$

where n is the number of observations, y_i is the actual value of the target variable for observation i, and $\hat{y}_i$ is the predicted value of the target variable for observation.

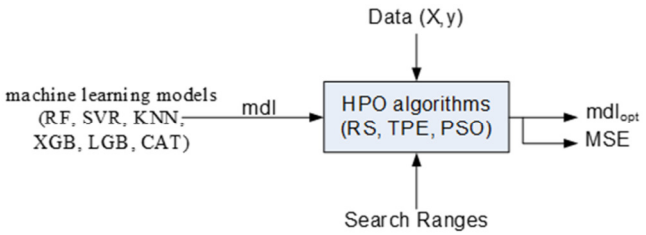

Fig. 1. Schematic diagram of HPO algorithms.

B. Repetition and Analysis of Statistical Data n Times

One of the characteristics of HPO algorithms such as RS, BO-TPE, and PSO is that the results may vary for each program run, leading to fluctuations in the obtained results. Therefore, to improve reliability, a statistical analysis over multiple iterations of the HPO algorithms was used, as shown in Figure 2(a) with n times repetition. After completing all iterations, the n values of the MSE were obtained, represented as [MSE_1, MSE_2, ..., MSE_n]. A boxplot, depicted in Figure 2(b), was then utilized to perform statistical analyses of the results, presenting the minimum, 25th percentile, median (50th percentile), 75th percentile, and maximum values [30].

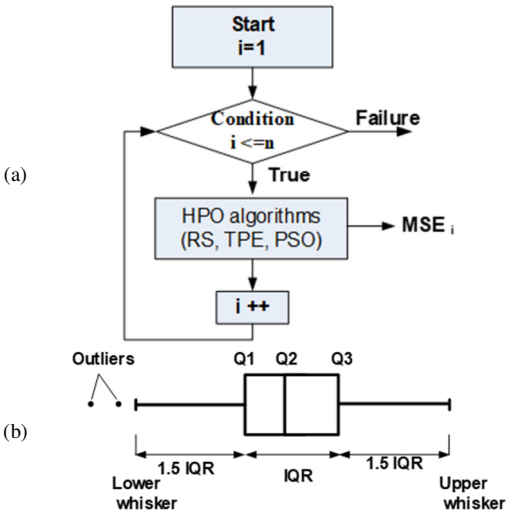

Fig. 2. The multiple iterations of the HPO algorithms: (a) repeat loop, (b) boxplot statistics.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

This study investigated the performance of the models by considering two commonly used datasets: the Boston Housing dataset, containing information regarding housing prices and related features for 506 Housings in the suburbs of Boston, Massachusetts [13], and the publicly available Bike-sharing Demand dataset, containing information about bike sharing demand in Washington, D.C [31]. Table I below lists the variables and data types for these two datasets.

An important parameter that needs to be determined is the hyperparameter search range, which serves as one of the inputs for HPO algorithms, as depicted in Figure 1. Table II illustrates the search range for each hyperparameter considered in this study, providing their types and ranges accordingly.

This study goes beyond evaluating the performance of ensemble models against traditional ML models. It also explores the effectiveness of the iterative process with each process repeated n times. Therefore, the cases examined in this study are as follows:

- Case 1 – Baseline: Each HPO algorithm is run only once for each associated ML model, as shown in Figure 1.
- Case 2 - Iterative HPO - In contrast to the baseline, each HPO algorithm is repeated 20 times (n = 20) for each associated ML model, as illustrated in the cycle of Figure 2(a).

TABLE I. BOSTON HOUSING DATASET AND BIKE SHARING DEMAND DATASET DESCRIPTION

Boston Housing Dataset			**Bike Sharing Demand Dataset**		
Variable	Description	Type	Variable	Description	Type
crim	Per capita crime rate by town	float	datetime	Date and time of bike rental	datetime
zn	Proportion of residential land zoned for lots over 25,000 sq.ft.	float	season	Season (1 = spring, 2 = summer, 3 = fall, 4 = winter)	int
indus	Proportion of non-retail business acres per town	float	holiday	Holiday (1 = yes, 0 = no)	int
chas	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)	int	workingday	Working day (1 = yes, 0 = no)	int
nox	Nitric oxides concentration (parts per 10 million)	float	weather	Weather condition (1 = Clear, 2 = Nearly Clear, 3 = Mostly Cloudy, 4 = Cloudy)	int
rooms	Average number of rooms per Housinghold	float	temp	Temperature (C)	float
age	Proportion of owner-occupied units built prior to 1940	float	atemp	Apparent Temperature (C)	float
dis	Weighted distance to five Boston employment centers	float	humidity	Humidity (%)	float
rad	Index of accessibility to radial highways	float	windspeed	Windspeed (m/s)	float
tax	Full-value property-tax rate per $10,000	float	count	Total number of bikes rented	int
ptratio	Pupil-teacher ratio by town	float			
black	$1000(Bk - 0.63)^2$ where Bk is the proportion of black people by town	float			
lstat	Lower status of the population (percent)	float			
medv	Median value of owner-occupied homes in $1000s	float			

B. Experimental Results

Table III presents the experimental results for Case 1. These values represent the MSE of the RS, TPE, and PSO algorithms for both traditional ML models (RF, SVR, KNN) and the proposed ensemble models (XGB, LGB, CAT) applied to the two datasets. Note that the values for the traditional ML models (RF, SVR, KNN) of the Boston Housing dataset are reference values from [10] and are formatted in bold italics in Table III.

TABLE II. RANGES FOR HYPERPARAMETERS

Models	**Hyperparameter**	***Type***	**Search Range**
RF	*n_estimators*	Discrete	[10, 100]
	max_depth	Discrete	[5, 50]
	min_samples_split	Discrete	[2, 11]
	min_samples_leaf	Discrete	[1, 11]
	criterion	Categorical	['gini', 'entropy']
	max_features	Discrete	[1, 64]
SVR	*C*	Continuous	[0.1, 50]
	kernel	Categorical	['linear', 'poly', 'rbf', 'sigmoid']
	epsilon	Continuous	[0.001, 1]
KNN	*n_neighbors*	Discrete	[1, 20]
XGBoost, LightGBM, Catboost	*n_estimators*	Continuous	[50, 500]
	max_depth	Discrete	[1, 16]
	learning_rate	Continuous	[0.01, 0.3]
	subsample	Continuous	[0.1, 1]
XGBoost, LightGBM	*colsample_bytree*	Continuous	[0.1, 1]
Catboost	*colsample_bylevel*	Continuous	[0.1, 1]

TABLE III. THE RESULTS OF CASE 1

Data	**Random Search**					
	RF	**SVR**	**KNN**	**XGB**	**LGB**	**CAT**
Boston	***27.9***	***61.4***	***80.8***	27.2	24.7	26.0
Bike	9,730	23,483	21,056	8,933	7,565	6,951
	Tree-structured Parzen Estimator					
Boston	***25.4***	***59.4***	***80.8***	27.4	25.1	23.4
Bike	9,683	23,099	21,121	8,079	7,405	6,871
	Particle Swarm Optimization					
Boston	***25.7***	***58.7***	***80.7***	24.5	25.1	25.0
Bike	9,582	23,664	21,056	8,851	7,659	7,175

TABLE IV. THE RESULTS OF CASE 2, BOSTON HOUSING DATASET

Statistics	**RF**	**SVR**	**KNN**	**XGB**	**LGB**	**CAT**
	Random Search					
mean	27	60.4	81	26	24.6	25.3
std	0.6	1	0.3	1	0.5	1.2
min	25.9	59.0	80.7	23.4	23.7	23.9
0.25	26.6	60.0	80.7	25.4	24.3	24.3
0.5	26.9	60.5	80.8	26.0	24.7	25.1
0.75	27.5	60.8	81.3	26.9	24.9	25.6
max	28.1	62.4	81.6	27.3	25.8	28.5
	Tree-structured Parzen Estimator					
mean	26.7	60.7	81	25.7	24.5	24.7
std	0.6	1.2	0.3	0.7	0.6	1.3
min	25.3	58.6	80.7	23.9	22.9	22.4
0.25	26.4	59.9	80.7	25.3	24.1	23.7
0.5	26.7	60.3	80.8	25.9	24.6	24.8
0.75	27.1	61.3	81.3	26.1	25.0	25.6
max	28.2	63.4	81.8	26.9	25.4	26.5
	Particle Swarm Optimization					
mean	27	60	80.8	26.1	24.5	25.1
std	0.4	0.6	0.2	1.1	0.4	1.1
min	25.1	58.5	80.7	23.6	23.8	23.2
0.25	26.4	59.9	80.7	25.5	24.2	24.4
0.5	26.7	60.0	80.7	26.1	24.4	25.0
0.75	26.9	60.2	80.7	26.9	24.6	25.7
max	27.6	61.2	81.3	27.4	25.6	27.5

Figure 3 presents the experimental results for Case 2. Each subfigure (a-f) displays a boxplot that depicts the MSE distribution obtained from applying the corresponding HPO algorithm (RS, TPE, PSO) to all six models (RF, SVR, KNN, XGB, LGB, CAT). The boxplots for the Boston Housing

dataset are shown in subfigures (a-c), while those for the Bike Sharing Demand dataset are shown in subfigures (d-f). Tables IV and V present the corresponding statistical data for the Boston Housing and Bike Sharing Demand datasets, respectively, as outlined in subfigures (a-c) and (d-f). Notably, the values in each column of the boxplots correspond to the minimum, 25th percentile, median, 75th percentile, and maximum values, as detailed in Tables IV and V.

TABLE V. THE RESULTS OF CASE 2, BIKE SHARING DEMAND DATASET

Statistics	RF	SVR	KNN	XGB	LGB	CAT
	Random Search					
mean	10022	23539	21046	8775	7705	7246
std	355	70	48	722	180	274
min	9463	23429	21009	7704	7447	6885
0.25	9787	23482	21009	8123	7548	6980
0.5	9975	23548	21009	8677	7698	7314
0.75	10289	23590	21072	9433	7823	7429
max	10700	23678	21124	9911	8021	7869
	Tree-structured Parzen Estimator					
mean	9920	23082	21050	8518	7613	7210
std	368	137	47	393	126	195
min	9467	22797	21009	7618	7388	6813
0.25	9680	23005	21009	8334	7530	7074
0.5	9757	23047	21053	8591	7597	7201
0.75	10113	23144	21056	8729	7690	7332
max	10833	23343	21176	9177	7856	7641
	Particle Swarm Optimization					
mean	9739	23551	21035	8675	7683	7156
std	224	107	30	630	168	146
min	9379	23425	21009	7912	7472	6929
0.25	9567	23468	21009	8254	7567	7077
0.5	9730	23523	21031	8428	7650	7151
0.75	9887	23609	21053	9077	7752	7197
max	10295	23790	21121	10300	8079	7600

C. Analysis and Discussion

1) Analysis of Ensemble Models' Performance

By analyzing Tables III, IV, V, and Figure 3, it becomes evident that applying ensemble models yields smaller errors than those obtained by traditional ML models, indicating an improvement in the accuracy of prediction results. The detailed analyses clarify the following aspects:

- Case 1: Table III reveals that when evaluating the RS algorithm on the Boston Housing dataset, the MSE values for ensemble models (XGB=27.2, LGB=24.7, CAT=26) are consistently lower than those for traditional models (RF=27.9, SVR=61.4, KNN=80.8). Similar results are observed for the TPE and PSO algorithms. This trend holds for the Bike dataset as well.
- Case 2: Considering both Tables IV and V, the statistical analysis of MSE for the TPE algorithm on the Boston Housing dataset reveals that the minimum values for ensemble models (XGB=23.9, LGB=22.9, CAT=22.4) are consistently lower than those for traditional models (RF=25.3, SVR=58.6, KNN=80.7). Similar results are observed across the 25th, 50th, 75th, and maximum percentiles. This pattern is consistent with RS and PSO, and similar trends are found for the Bike dataset.

In conclusion, the results strongly suggest that ensemble models offer a significant advantage in terms of prediction accuracy compared to the traditional models in this study.

2) Analysis of Results After n Iterations

To evaluate the effectiveness of *n* iterations, it is necessary to analyze and compare the data presented in Table III (Single Random Run) and Tables IV and V (n-Iterations).

- The analysis of the results indicates that the MSE values in each run appear randomly. Therefore, conducting only one run and using its results for analysis may lead to unreliable conclusions. For instance, consider Table IV. In the case of applying the optimized RS algorithm, a single run might yield the corresponding MSE values as RF = 25.9 (min), XGB = 26.0 (50th percentile), LGB = 24.9 (75th percentile), and CAT = 28.5 (max). This could lead to the conclusion that the RF model performs better than XGB and CAT. However, compared with the statistical analysis conducted in multiple iterations, where ensemble algorithms outperformed the traditional algorithm after various iterations, this result is unreliable. Similarly, analyzing the cases of TPE and PSO algorithms and the Bike dataset in Table V yields the same conclusions.
- The values in Table III (single random run) all fall within the range of distributions after *n* iterations in Table IV (Boston Housing dataset) and Table V (Bike dataset). Taking the case of the RS algorithm applied to the Boston Housing dataset, the values in Table III (RF=27.9, SVR=61.4, KNN=80.8, XGB=27.2, LGB=24.7, CAT=26), all lie within the fluctuation range (min-max) of Table IV as follows: RF=[25.9-28.1], SVR=[59.0-62.4], KNN=[80.7-81.6], XGB=[23.4-27.3], LGB=[23.7-25.8], CAT=[23.9-28.5]. The results for other algorithms such as TPE and PSO show a similar pattern. Notably, similar results are also obtained when analyzing the Bike dataset.

This analysis clearly demonstrate that applying *n* iterations, followed by the statistical analysis and boxplot visualization, significantly enhances the reliability of the research results, eliminating random factors analysis.

V. CONCLUSIONS

This study addresses the improvement of precision in load forecasting cases by proposing a workflow that combines ensemble learning models with multiple executions of HPO algorithms (RS, BO-TPE, PSO). Unlike previous studies that relied on single optimization runs, this approach emphasizes stability through *n* independent repetitions, supported by statistical analyses of the error rate distributions. The results on the Boston Housing and Bike Sharing Demand datasets demonstrate that the ensemble models (XGBoost, LightGBM, and CatBoost) consistently outperformed traditional models (RF, SVR, and KNN). Furthermore, repeated HPO iterations significantly reduce stochastic influences inherent in single optimization trials, enhancing the reliability of the findings. Overall, this study provides a methodological framework for future research that can be extended to applying other state-of-the-art ML models and advanced HPO algorithms in regression forecasting problems.

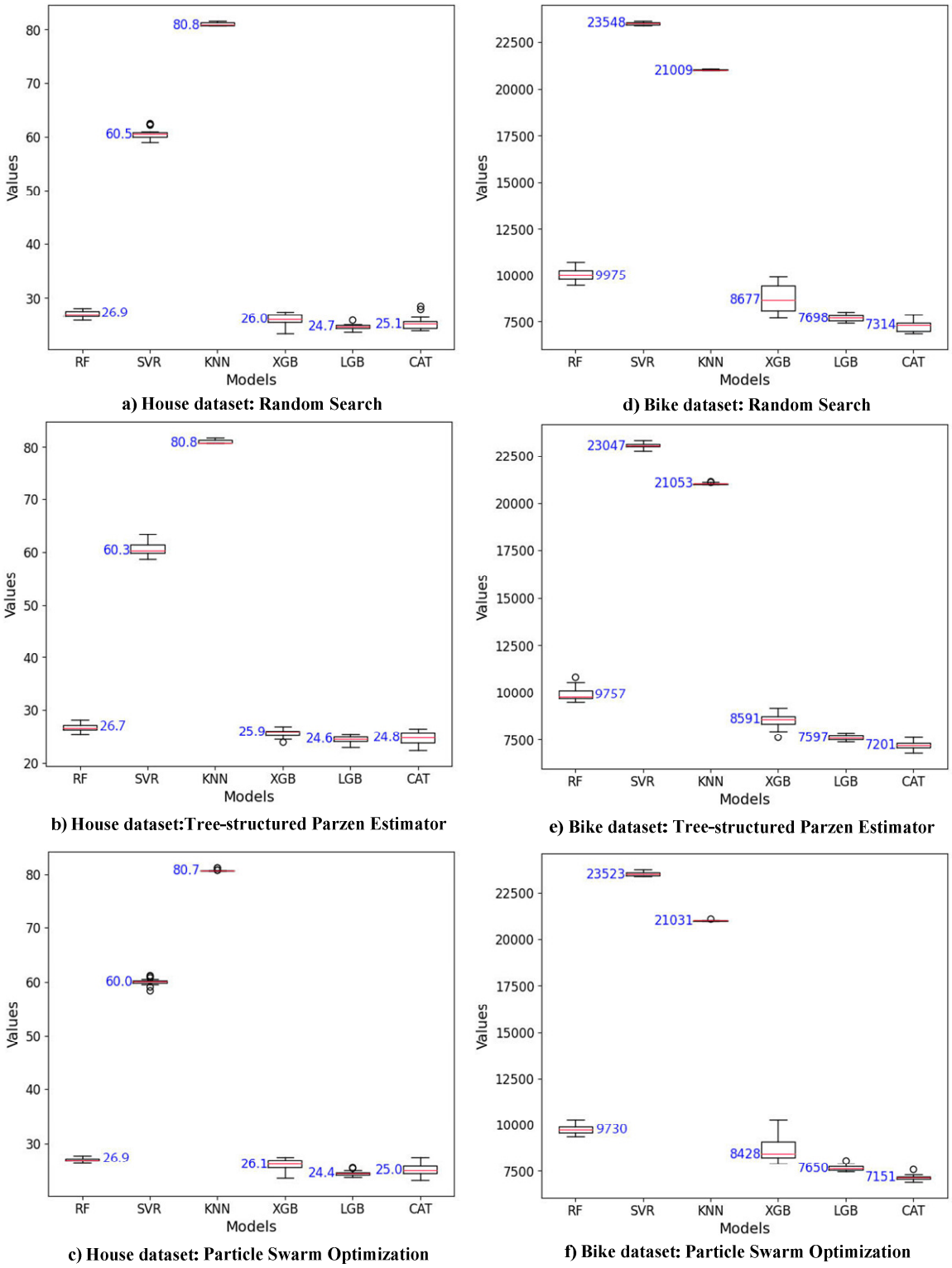

Fig. 3. The experimental results for Case 2.

REFERENCES

[1] J. Brownlee, Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch. Machine Learning Mastery, 2016.

[2] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, and V. Zocca, Python Deep Learning: Exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow, 2nd Edition. Birmingham, UK: Packt Publishing, 2019.

[3] N. Zhang, Z. Li, X. Zou, and S. M. Quiring, "Comparison of three short-term load forecast models in Southern California," *Energy*, vol. 189, Dec. 2019, Art. no. 116358, https://doi.org/10.1016/j.energy.2019.116358.

[4] Q. Zhao, Z. Ye, Y. Su, and D. Ouyang, "Predicting complexation performance between cyclodextrins and guest molecules by integrated machine learning and molecular modeling techniques," *Acta Pharmaceutica Sinica B*, vol. 9, no. 6, pp. 1241–1252, Nov. 2019, https://doi.org/10.1016/j.apsb.2019.04.004.

[5] Z. Tan, J. Zhang, Y. He, Y. Zhang, G. Xiong, and Y. Liu, "Short-Term Load Forecasting Based on Integration of SVR and Stacking," *IEEE Access*, vol. 8, pp. 227719–227728, 2020, https://doi.org/10.1109/ACCESS.2020.3041779.

[6] P. Meesad and R. I. Rasel, "Predicting stock market price using support vector regression," in *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, Dhaka, Bangladesh, May 2013, pp. 1–6, https://doi.org/10.1109/ICIEV.2013.6572570.

[7] F. Martínez, M. P. Frías, M. D. Pérez, and A. J. Rivera, "A methodology for applying k-nearest neighbor to time series forecasting," *Artificial Intelligence Review*, vol. 52, no. 3, pp. 2019–2037, Oct. 2019, https://doi.org/10.1007/s10462-017-9593-z.

[8] S. B. Imandoust and M. Bolandraftar, "Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background," *International Journal of Engineering Research and Applications*, vol. 3, no. 5, 2013.

[9] E. Al Daoud, "Comparison between XGBoost, LightGBM and CatBoost using a home credit dataset," *International Journal of Computer and Information Engineering*, vol. 13, no. 1, pp. 6–10, 2019.

[10] W. F. Zhou, J. G. Wang, L. F. Deng, Y. Yao, and J. L. Liu, "Terminal Temperature Prediction of Molten Steel in VD Furnace based on XGBoost and LightGBM Algorithms," in *2021 40th Chinese Control Conference (CCC)*, Shanghai, China, Jul. 2021, pp. 6289–6294, https://doi.org/10.23919/CCC52363.2021.9550444.

[11] T. N. Tran and Q. D. Nguyen, "Research on the Influence of Genetic Algorithm Parameters on XGBoost in Load Forecasting," *Engineering, Technology & Applied Science Research*, vol. 14, no. 6, pp. 18849–18854, Dec. 2024, https://doi.org/10.48084/etasr.8863.

[12] N. Naineni, "State-of-the-Art MPI Allreduce Implementations for Distributed Machine Learning: A Survey." OSF, Sep. 24, 2024, https://doi.org/10.31219/osf.io/esm7q.

[13] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, https://doi.org/10.1016/j.neucom.2020.07.061.

[14] S. Punia, Nikolopoulos ,Konstantinos, Singh ,Surya Prakash, Madaan ,Jitendra K., and K. and Litsiou, "Deep learning with long short-term memory networks and random forests for demand forecasting in multi-channel retail," *International Journal of Production Research*, vol. 58, no. 16, pp. 4964–4979, Aug. 2020, https://doi.org/10.1080/00207543.2020.1735666.

[15] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," *SN Computer Science*, vol. 2, no. 3, Mar. 2021, Art. no. 160, https://doi.org/10.1007/s42979-021-00592-x.

[16] M. Jiang, S. Jiang, L. Zhu, Y. Wang, W. Huang, and H. Zhang, "Study on Parameter Optimization for Support Vector Regression in Solving the Inverse ECG Problem," *Computational and Mathematical Methods in Medicine*, vol. 2013, no. 1, 2013, Art. no. 158056, https://doi.org/10.1155/2013/158056.

[17] K. Smets, B. Verdonk, and E. M. Jordaan, "Evaluation of Performance Measures for SVR Hyperparameter Selection," in *2007 International Joint Conference on Neural Networks*, Dec. 2007, pp. 637–642, https://doi.org/10.1109/IJCNN.2007.4371031.

[18] T. Parhizkar, E. Rafieipour, and A. Parhizkar, "Evaluation and improvement of energy consumption prediction models using principal component analysis based feature reduction," *Journal of Cleaner Production*, vol. 279, Jan. 2021, Art. no. 123866, https://doi.org/10.1016/j.jclepro.2020.123866.

[19] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, pp. 281–305, Oct. 2012.

[20] K. Eggensperger, F. Hutter, H. Hoos, and K. Leyton-Brown, "Efficient Benchmarking of Hyperparameter Optimizers via Surrogates," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, Feb. 2015, https://doi.org/10.1609/aaai.v29i1.9375.

[21] M. Jain, V. Saihjpal, N. Singh, and S. B. Singh, "An Overview of Variants and Advancements of PSO Algorithm," *Applied Sciences*, vol. 12, no. 17, Jan. 2022, Art. no. 8392, https://doi.org/10.3390/app12178392.

[22] A. Kumar and M. Jain, Ensemble Learning for AI Developers: Learn Bagging, Stacking, and Boosting Methods with Use Cases. Berkeley, CA, USA: Apress, 2020.

[23] Z. Wan, Y. Xu, and B. Šavija, "On the Use of Machine Learning Models for Prediction of Compressive Strength of Concrete: Influence of Dimensionality Reduction on the Model Performance," *Materials*, vol. 14, no. 4, Jan. 2021, Art. no. 713, https://doi.org/10.3390/ma14040713.

[24] X. Zhao, N. Xia, Y. Xu, X. Huang, and M. Li, "Mapping Population Distribution Based on XGBoost Using Multisource Data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 11567–11580, 2021, https://doi.org/10.1109/JSTARS.2021.3125197.

[25] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, May 2016, pp. 785–794, https://doi.org/10.1145/2939672.2939785.

[26] G. Ke *et al.*, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems*, 2017, vol. 30.

[27] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: gradient boosting with categorical features support." arXiv, Oct. 24, 2018, https://doi.org/10.48550/arXiv.1810.11363.

[28] H. Alshari, A. Y. Saleh, and A. Odabaş, "Comparison of Gradient Boosting Decision Tree Algorithms for CPU Performance," *Journal of Institute Of Science and Technology*, vol. 37, no. 1, 2021.

[29] N. T. Tran, T. T. G. Tran, T. A. Nguyen, and M. B. Lam, "A new grid search algorithm based on XGBoost model for load forecasting," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 4, pp. 1857–1866, Aug. 2023, https://doi.org/10.11591/eei.v12i4.5016.

[30] M. Karasuyama and R. Nakano, "Optimizing SVR Hyperparameters via Fast Cross-Validation using AOSVR," in *2007 International Joint Conference on Neural Networks*, Orlando, FL, USA, Aug. 2007, pp. 1186–1191, https://doi.org/10.1109/IJCNN.2007.4371126.

[31] H. Fanaee-T and J. Gama, "Event labeling combining ensemble detectors and background knowledge," *Progress in Artificial Intelligence*, vol. 2, no. 2, pp. 113–127, Jun. 2014, https://doi.org/10.1007/s13748-013-0040-3.