Gaussian Process Regression Model Based on Random Sampling and Secondary Encoding Techniques

Yu Miao

School of Economics and Management, Beijing University of Posts and Telecommunications, Beijing, China, 100876

lianlangx@gmail.com

Abstract. Gaussian Process Regression (GPR) is a flexible non-parametric method that has been widely used in various prediction tasks due to its superior performance in fitting nonlinear functions. However, as the sample size increases, the computational complexity of GPR models grows exponentially, limiting their application to large-scale datasets. To address this issue, this paper proposes a GPR model based on the Stacking framework. The core innovation of the model consists of two parts: first, random sampling techniques are employed to extract multiple subsamples from the original dataset, and independent GPR models are trained for each subsample. Since the subsample sizes are relatively small, this strategy effectively reduces computational complexity and further improves efficiency through parallel processing of multiple models. Second, to overcome the performance variance among different submodels, a model fusion mechanism is adopted. The predictions from the individual submodels are treated as new features, and a secondary GPR model is trained as a combiner to optimize the aggregation of these predictions. This two-layer structural design not only significantly reduces the computational cost of GPR but also enhances the generalization capability of the predictive model through model fusion. Simulation experiments and real-world data analyses demonstrate that the proposed method exhibits a clear competitive advantage over traditional regression models.

Keywords: Gaussian Process Regression, Random Sampling, Model Fusion.

1. Introduction

Regression models are important tools in statistical analysis, with the most common form being linear regression, which assumes a linear relationship between the independent and dependent variables. In practical applications, various extensions of linear regression models have been developed, such as multiple linear regression, LASSO regression, and ridge regression [1]. However, these models assume that there is a linear relationship between the predictor variables and the response variable, which limits the flexibility of the model when dealing with complex nonlinear relationships.

To address this issue, non-parametric regression models have emerged. Non-parametric statistical models provide more flexibility as they do not rely on predefined functional forms but instead estimate the relationship between variables through the data itself. For example, polynomial regression captures nonlinear relationships between independent and dependent variables by introducing higher-order terms [2]. Although nonlinear regression improves model flexibility to some extent, these models still rely on predefined functional forms and face limitations when handling complex, unknown nonlinear relationships. Kernel regression models use kernel functions to weight nearby data points and estimate the regression value at the target point, while local polynomial regression extends kernel regression by fitting a polynomial in the neighborhood of each target point to capture local trends [3]. However, these methods face the "curse of dimensionality" problem, where the sparsity of local data points increases as the data dimensionality grows, leading to a loss of distinction between data points and a reduction in model estimation accuracy [4, 5]. This paper turns its attention to Gaussian Process Regression (GPR), a model that maintains good performance in high-dimensional data [6, 7]. GPR defines the correlation between data points through a kernel function, allowing it to flexibly capture complex relationships between independent and dependent variables [8]. One significant advantage of GPR is that it not only generates predictions but also

provides uncertainty estimates for those predictions, making it highly reliable and flexible in dealing with complex and unknown data. However, GPR has high computational complexity, especially as the sample size increases, since the inversion of the covariance matrix has a complexity of $O(n^3)$, limiting its application to large-scale datasets.

To solve this problem, several methods have been proposed. For example, the Bootstrap method generates multiple subsamples through random sampling with replacement. Due to the high randomness of sampling, this method improves model stability and prevents it from getting stuck in local optima [9]. However, after generating multiple Gaussian Process Regression (GPR) models based on the Bootstrap concept, how to effectively combine these models becomes a new issue. Bagging offers a potential solution, as it is a direct application of Bootstrap. By training multiple models and averaging their predictions, Bagging enables model fusion. While Bagging can reduce variance to some extent, it performs simple averaging, which may not yield optimal results for model fusion. Model averaging methods provide another approach by assigning weights to each model and combining them through weighted averaging. Although this method can improve performance to some extent, it is limited by searching for the optimal solution in a linear space, which poses challenges in more complex nonlinear scenarios [10]. Finally, we turn to the layered concept of Stacking, which provides greater flexibility [11]. Our approach introduces a new Gaussian Process model during the model fusion process, allowing model fusion to break free from the linear space.

In summary, this paper proposes a Gaussian Process Regression model based on the Stacking concept. First, random sampling is applied to the samples and their features, generating several subsamples. After training these subsamples with Gaussian Process Regression models, each subsample produces a prediction result. These predictions are then used as new input features to construct a new Gaussian Process Regression model, which is used to fuse the outputs of the previous models. The innovation of this paper lies in two aspects: first, random sampling reduces computational complexity, making each training model smaller and enhancing the model's adaptability to large samples and high-dimensional datasets; second, the introduction of a new Gaussian Process model for fusion effectively improves the overall model performance. To validate the model's effectiveness, we conducted simulation experiments and real-world data analysis, testing model performance across different experimental settings such as varying sample sizes, feature counts, and functions connecting independent and dependent variables. The experimental results demonstrate that the proposed method shows strong competitiveness compared to classic models such as BP neural networks, XGBoost, random forest models, and Lasso regression.

2. Theory and Methods

To address the computational complexity of Gaussian Process models and provide a solution for regression problems in large-scale datasets, this paper proposes a Gaussian Process Regression model method based on the Stacking framework. This method resamples the data and features to generate multiple subsample sets, trains several Gaussian Process models for each subsample, and then fuses the output results of these models using a new Gaussian Process model.

Suppose we have a dataset $\{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in R^d$ represents the input features of the i-th sample, $y_i \in R$ represents the corresponding output, and n is the total number of samples. To fully utilize the data and reduce the risk of model overfitting, we apply Bootstrap resampling to these n samples, generating k subsample sets from the original data. Let the j-th subsample set be $\left\{\left(x_{i_j}, y_{i_j}\right)\right\}_{i_j=1}^{n_j}$, where n_j represents the number of samples in the j-th subsample set.

Gaussian Process Regression is a non-parametric Bayesian method. A Gaussian Process model assumes that the outputs follow a Gaussian Process, which is uniquely determined by the mean function and covariance function, and satisfies:

$$f(x) \sim GP\big(m(x), k(x, x')\big) \tag{1}$$

where $m(x)$ is the mean function, which is typically set to 0, and $k(x, x')$ is the covariance function, represented by a selected kernel function. Taking the Radial Basis Function (RBF) kernel as an example:

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{\|x-x'\|^2}{2l^2}\right) \tag{2}$$

Where σ_f^2 represents the amplitude parameter, which controls the maximum value of the kernel function, and l is the length scale, determining the sensitivity of the output to changes in the input variables. For each subsample $\left\{\left(x_{i_j}, y_{i_j}\right)\right\}_{i_j=1}^{n_j}$, a Gaussian Process model is trained using the input set $X_j = \left\{x_{i_j}\right\}_{i_j=1}^{n_j}$ and the corresponding output set $Y_j = \left\{y_{i_j}\right\}_{i_j=1}^{n_j}$. Given an input point x_*, the distribution of the predicted output y_* is assumed to be:

$$p\left(y_* | x_*, X_j, Y_j\right) \sim N\left(\mu_j(x_*), \sigma_j^2(x_*)\right) \tag{3}$$

The predicted mean $\mu_j(x_*)$ and the predicted variance $\sigma_j^2(x_*)$ are given by:

$$\mu_j(x_*) = k\left(x_*, X_j\right)\left(K_j + \sigma_n^2 I\right)^{-1} Y_j \tag{4}$$

$$\sigma_j^2(x_*) = k(x_*, x_*) - k\left(x_*, X_j\right)\left(K_j + \sigma_n^2 I\right)^{-1} k\left(x_*, X_j\right) \tag{5}$$

Where K_j is the covariance matrix calculated using the kernel function, and σ_n^2 represents the variance of the noise.

Following the steps above, after obtaining k Gaussian Process models, each model generates a predicted result $\mu_j(x_*)$ for the input point x_*. These prediction results from all models are then used as new input features to construct a new input feature matrix Z:

$$Z = \begin{bmatrix} \mu_1(x_1) & \cdots & \mu_k(x_1) \\ \vdots & \ddots & \vdots \\ \mu_1(x_n) & \cdots & \mu_k(x_n) \end{bmatrix} \tag{6}$$

Next, using this new feature matrix Z, a new Gaussian Process Regression model is trained to fuse the predictions of the previous models. For a given input Z_*, the input vector is represented as:

$$Z_* = [\mu_1(x_*), \mu_2(x_*), \ldots, \mu_k(x_*)] \tag{7}$$

The distribution of the predicted output $f(Z_*)$ is then given by:

$$p(f(Z_*)|Z, Y, Z_*) \sim N\left(\mu(Z_*), \sigma^2(Z_*)\right) \tag{8}$$

The predicted mean $\mu(Z_*)$ and the predicted variance $\sigma^2(Z_*)$ are given by:

$$\mu(Z_*) = k(Z_*, Z)[K(Z, Z) + \sigma_n^2 I]^{-1} Y \tag{9}$$

$$\sigma_j^2(Z_*) = k(Z_*, Z_*) - k(Z_*, Z)(K(Z, Z) + \sigma_n^2 I)^{-1} k(Z_*, Z) \tag{10}$$

Where $K(Z, Z)$ is the covariance matrix calculated using the kernel function, and σ_n^2 is the noise variance.

This fusion model allows for the flexible combination of outputs from multiple base learners, enabling it to capture complex relationships between the individual base models, rather than relying on simple weighted averaging. As a result, this approach offers greater flexibility and representational power compared to traditional Bagging ensemble methods. The primary advantage of the proposed method lies in the significant reduction of computational complexity during the training process, achieved by resampling the data and features, allowing each Gaussian Process model to handle smaller subsamples, thereby effectively avoiding computational bottlenecks in large-sample scenarios. Furthermore, during the model fusion stage, another Gaussian Process Regression is

employed to learn the complex nonlinear relationships between the outputs of multiple models, further improving the generalization ability and predictive accuracy of the model. This approach avoids the simple linear weighting method commonly seen in traditional ensemble learning and is better suited for handling nonlinear problems. The specific algorithm is shown in Figure 1.

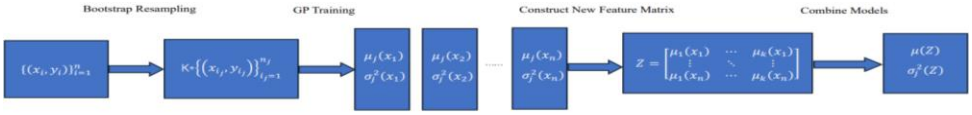

Figure 1. Algorithm Flowchart

3. Results

3.1. Simulation Experiment

In this study, we first conducted a simulation experiment by generating various datasets and systematically evaluating the predictive performance of multiple models. Initially, n samples were randomly generated, with each sample consisting of m features forming the input x. The target variable y in this study is defined by the following equation:

$$y_i = \mu_i + \epsilon_i = x_1 + \sin(x_2) + \cos(x_3) + x_4^2 + \exp(x_5) + \log(x_6 + 1) + \sqrt{x_7} + \tanh(x_8) + x_9^3 + \frac{1}{x_{10}+1} + x_{11}^4 + \sin(x_{12}) + \cos(x_{13}) + \tanh(x_{14}) + \frac{1}{x_{15}+1} + \epsilon_i \quad (11)$$

Here, x_i represents the feature variables, which follow a uniform distribution over $[0, 1]$. The random error term $\epsilon_i \sim N(0, \eta^2)$ is Gaussian noise with a mean of zero and a variance of η^2. The sample sizes are set to $n = 50,100,150$, and the number of features is set to $m = 5,10,15$. The proposed model, Stacking GP, is compared with BP neural network, XGBoost model, random forest model, and Lasso model under different experimental conditions [12-15]. Each experimental condition is repeated $t = 100$ times to ensure the stability and reliability of the results. The root mean square error (RMSE) is used as the evaluation metric:

$$RSME = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (12)$$

The dataset is divided into a training set, validation set, and test set in a ratio of 4: 3: 3 to evaluate the model's generalization ability. The experimental conditions are shown in Table 1, and the experimental results are presented in Table 2 and Figures 2 through 7.

Table 2 presents the mean RMSE results for each model across 100 experiments under different experimental conditions. As shown in the table, the Stacking GP model performs the best under most conditions, especially as the number of features increases, demonstrating better adaptability and robustness. Compared to other models, Stacking GP reduces the RMSE by approximately 20-40% on average across Conditions 1 to 6, indicating its superior performance in handling high-dimensional data. In particular, under Condition 2, the RMSE of Stacking GP is around 1.6052, significantly outperforming BP (2.1510), XGBoost (2.0749), Random Forest (1.8562), and Lasso (2.2226). BP neural network and XGBoost models perform similarly under some conditions, while the Lasso model performs poorly in all cases.

Table 1. Simulated Data Experimental Conditions

condition	n	features	expression
1	100	10	Formula with 10 features
2	50	10	Formula with 10 features
3	150	10	Formula with 10 features
4	100	5	Formula with 5 features
5	100	15	Formula with 15 features
6	100	10	Formula with 5 features

Table 2. Simulated Data Experimental Results

condition	RMSE				
	stacking_gp	BP	XGBoost	random_tree	Lasso
1	2.2841	3.2295	3.1314	2.9057	4.1690
2	1.6052	2.1510	2.0749	1.8562	2.2226
3	2.7461	3.0665	3.6569	2.9879	4.1950
4	0.7755	0.8800	1.6169	1.6378	2.9023
5	2.2611	3.6301	2.8948	3.0334	4.3286
6	1.4911	1.9561	2.3267	1.8399	3.3688

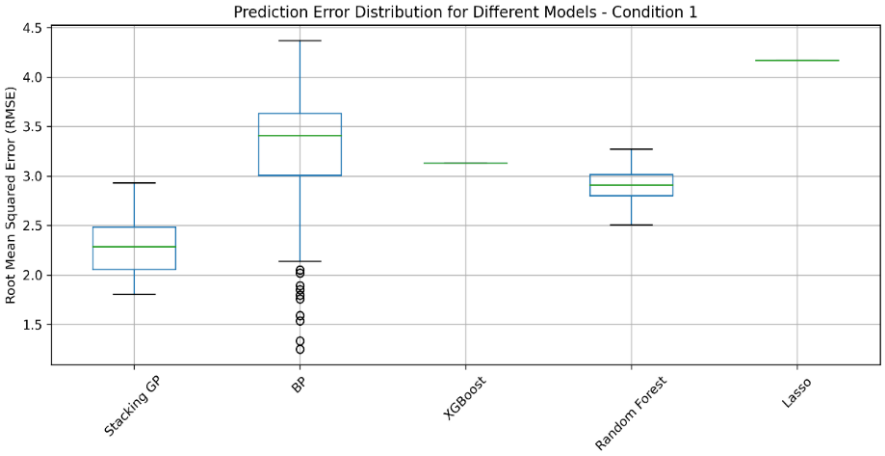

Figure 2. Box Plot of Results for Condition 1

From the box plots in Figures 2 to 7, we can observe the error distribution characteristics of different models under varying experimental conditions. In most cases, the box plots of the Stacking GP model show a narrower RMSE distribution range, indicating better stability and robustness. Additionally, the median is significantly lower than that of other models, suggesting that Stacking GP has a lower average error and smaller error fluctuations. In contrast, the box plot for the BP neural network shows more outliers and a wider error range, indicating higher sensitivity to noise and sample variations. The distributions in the box plots for XGBoost and Random Forest are more concentrated, but their medians are higher than that of Stacking GP, especially as the number of features increases, where the error range gradually expands. The box plots for the Lasso model exhibit the largest error range and the highest median in all conditions, significantly higher than the other models.

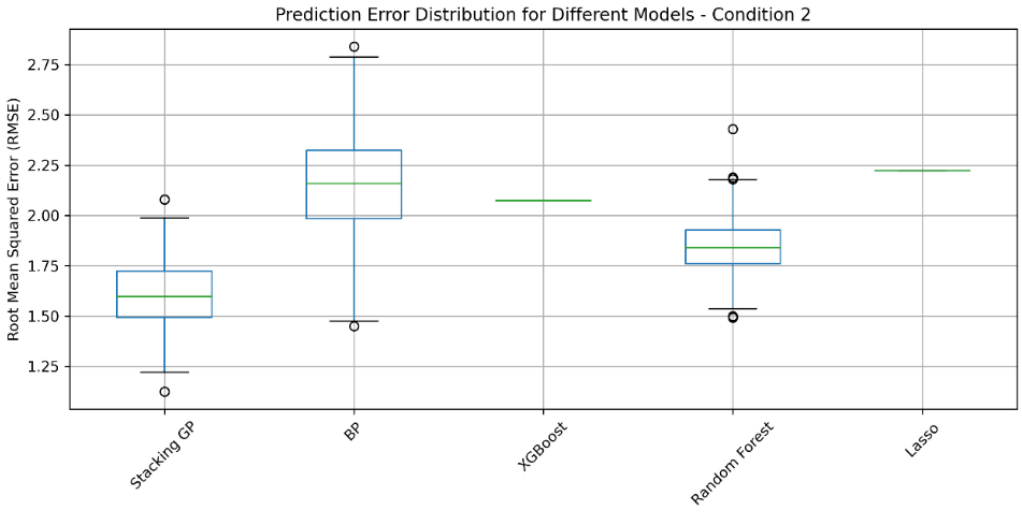

Figure 3. Box Plot of Results for Condition 2

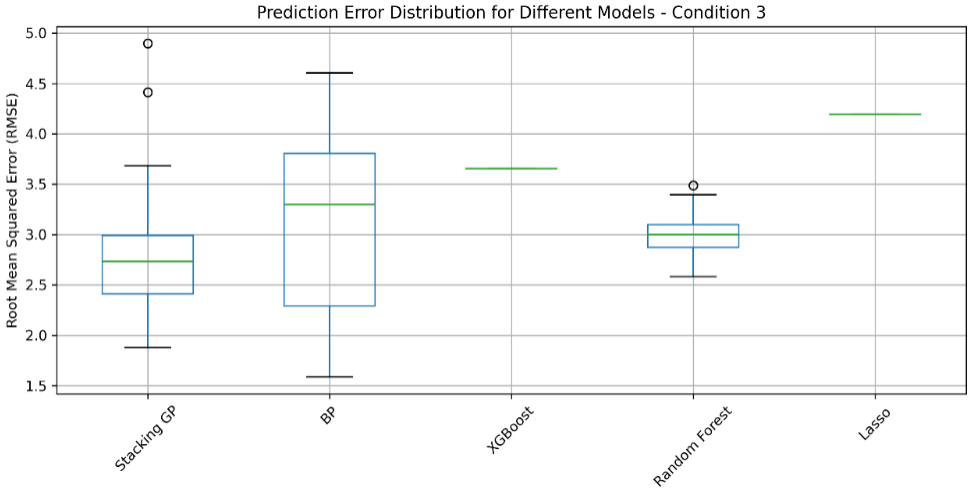

Figure 4. Box Plot of Results for Condition 3

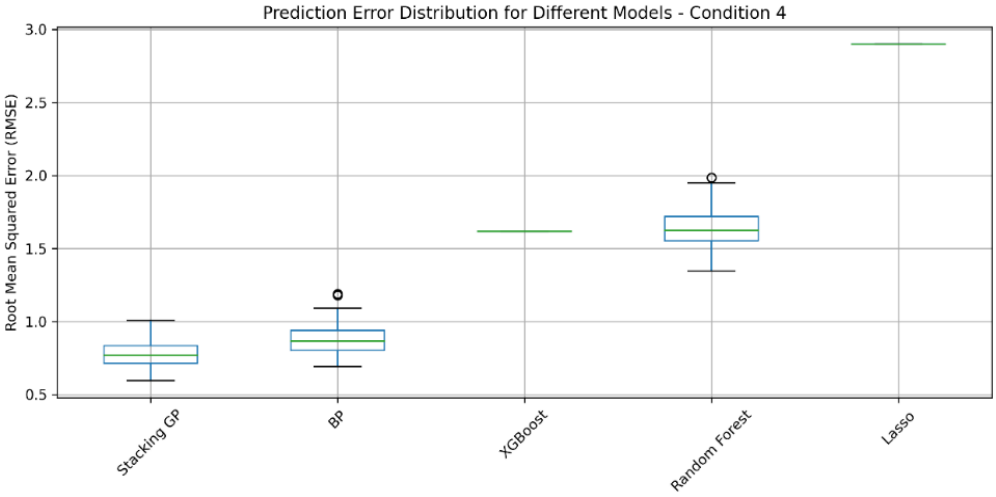

Figure 5. Box Plot of Results for Condition 4

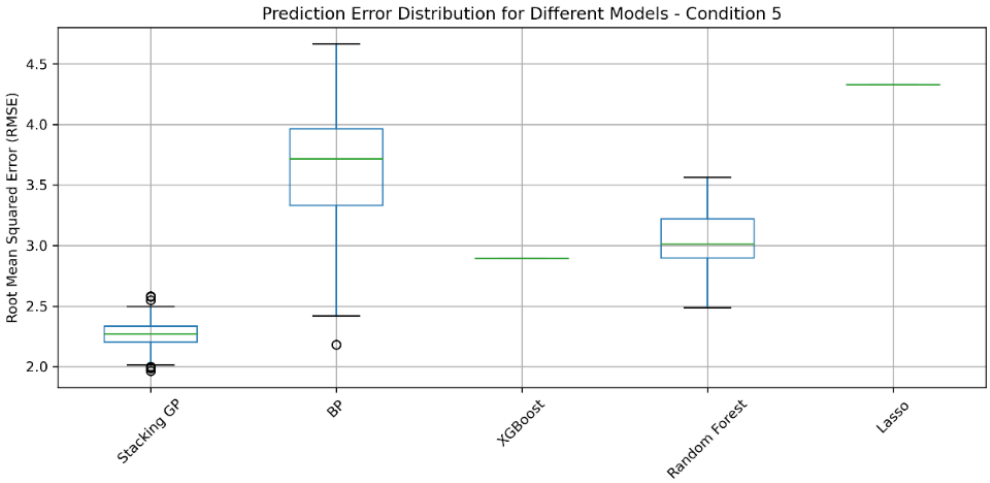

Figure 6. Box Plot of Results for Condition 5

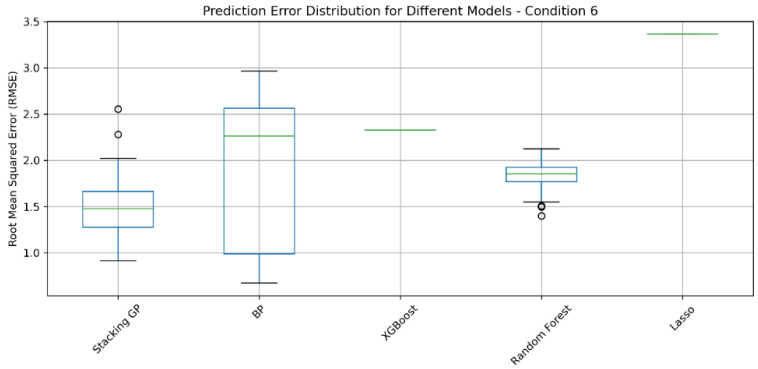

Figure 7. Box Plot of Results for Condition 6

We also believe that because our model incorporates randomization of samples and features, it not only reduces the computational complexity of the Gaussian Process model but also generates a larger search space. In this larger search space, it is more likely to find better solutions compared to the original one. Therefore, the proposed model exhibits a competitive advantage over other models.

3.2. Real-World Data Experiment

In this section, we apply our model to real-world datasets for testing and compare it with other models. By evaluating the performance on real-world data, we test the model's generalization ability and its behavior in practical scenarios. Additionally, this allows us to further assess the model's stability and accuracy when confronted with different types of data.

Table 3. Real-World Data Experiment Results

y	RMSE				
	stacking gp	BP	XGBoost	random tree	Lasso
moisture	1.3635	6.5930	1.0893	1.0736	1.3979
oil	0.5621	1.3358	0.6284	0.5962	0.6282
protein	2.0855	5.4024	2.6902	2.1312	1.8607
starch	3.6654	22.0429	4.9432	3.9438	3.5753

The corn dataset used in this study is sourced from https://lib.stat.cmu.edu/datasets/tecator, collected by the Cargill company [16]. The dataset consists of spectral data for $n = 80$ corn samples, recording data from the 1100 to 2498 nanometer rane at 2-nanometer intervals, resulting in a total of 700 spectral features. The dataset also includes corresponding measurements of moisture content, oil content, protein content, and starch content for each corn sample. In the experiment, we use the 700 spectral features as the independent variables $\{x_i\}_{i=1}^{700}$, and $\{y_i\}_{i=1}^{4}$ represent the dependent variables, where $i = 1,2,3,4$ correspond to moisture content, oil content, protein content, and starch content, respectively. Similar to the simulation experiment, the dataset is divided into training, validation, and test sets in a ratio of 4:3:3. The experiment is repeated $t = 10$ times and compared with other models. Root mean square error (RMSE) is again used as the evaluation metric, with the experimental results shown in Table 3 and Figures 8 to 11.

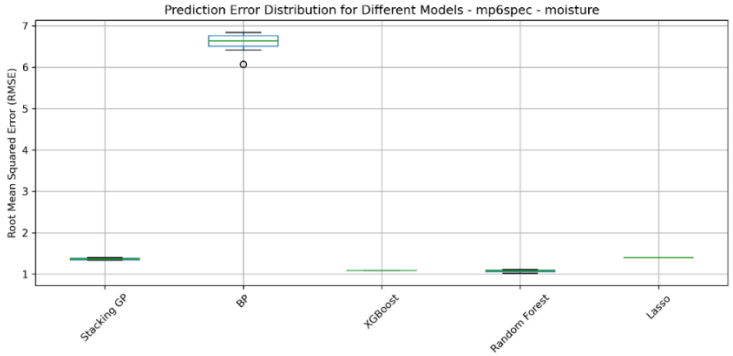

Figure 8. Box Plot of Results for moisture

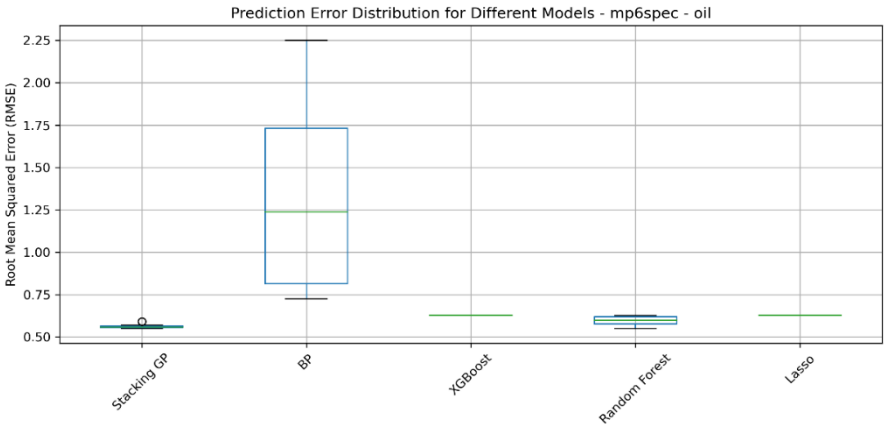

Figure 9. Box Plot of Results for oil

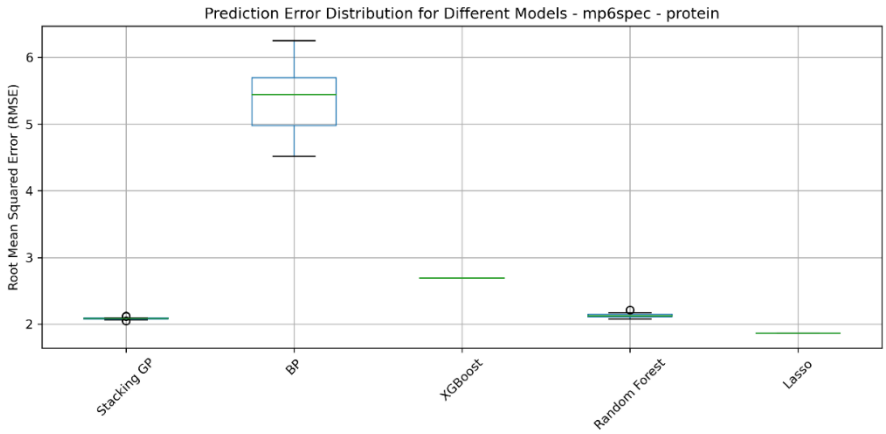

Figure 10. Box Plot of Results for protein

The results show that in the prediction of oil content, the Stacking GP model significantly outperforms the other models, with a root mean square error (RMSE) of approximately 0.5621. Compared to the BP neural network's RMSE of 1.3359, the error is reduced by about 58%. The box plots also reveal that the error distribution of the Stacking GP model is highly concentrated and lower than that of the other models, indicating high stability and accuracy. For the predictions of moisture, protein, and starch content, the Stacking GP model performs comparably to the best-performing models, without significant differences. For instance, in the starch content prediction task, the RMSE of Stacking GP is 3.6655, only about 2.5% higher than the best-performing Lasso model (RMSE of 3.5754), while the overall error distribution remains at a low level, demonstrating good robustness and predictive capability.

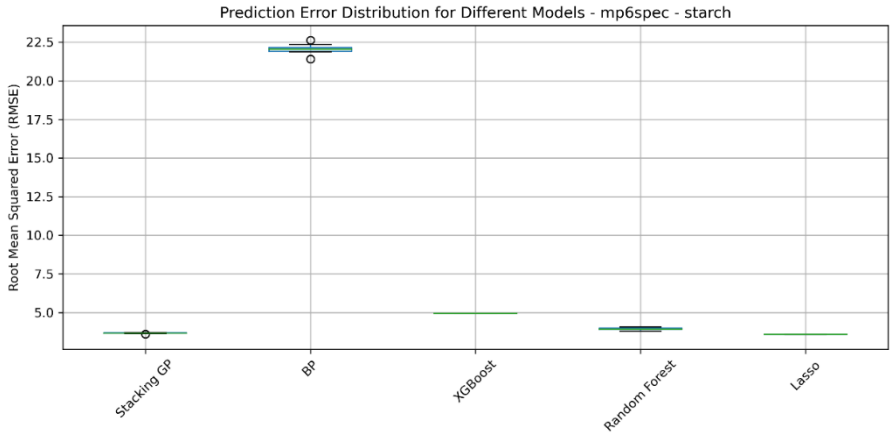

Figure 11. Box Plot of Results for starch

In this experiment, the dataset contains 700 features, which is a high dimensionality that could easily lead to the curse of dimensionality. However, the results indicate that the Stacking GP model still performs well when handling high-dimensional data. In the prediction of oil content, Stacking GP has the lowest RMSE, outperforming the other models. In the predictions of moisture, protein, and starch content, while some models have a slight advantage, the performance of Stacking GP is very close to the best, and its error distribution remains stable. This suggests that Stacking GP demonstrates competitiveness and strong generalization ability even with high-dimensional and complex data.

4. Conclusion

This paper proposes a Gaussian Process Regression model based on the Stacking framework, providing a solution to the problem of high computational complexity in traditional Gaussian Processes when applied to large-scale datasets. The core innovations of the model lie in two aspects: First, random sampling techniques are employed to partition the original dataset into multiple subsets, with each subset being used to train a Gaussian Process Regression model independently. This reduces computational complexity and improves efficiency. Second, a model fusion mechanism is introduced, where the predictions from each submodel are treated as new features, and a Gaussian Process fusion model is trained to effectively integrate the outputs of the submodels, thereby enhancing overall generalization ability. In the simulation experiments, the Stacking GP model outperforms other models in most conditions, especially when the number of features increases, demonstrating superior adaptability and robustness. In real-world data experiments, the Stacking GP model continues to perform well, with stable error distributions and high prediction accuracy, indicating that the proposed model maintains strong generalization ability and competitiveness when handling high-dimensional, complex data.

However, there are still several issues that need to be addressed. First, the choice of parameters such as the number of samples and the number of features to be selected in each sampling significantly affects the model's performance, and thus requires further experimentation and optimization. Second, the interpretability of the model is a key focus for future research. It is important to explore how to effectively interpret the influence of input variables x on the output y, making the model more transparent and reliable. Finally, for cases where the distance between certain data features cannot be adequately measured by Euclidean distance, introducing non-Euclidean kernel functions could be considered. This would allow for the exploration of their applicability in high-dimensional spaces, improving the model's performance and generalization ability across different types of data.

References

[1] Maulud D, Abdulazeez A M. A review on linear regression comprehensive in machine learning [J]. Journal of Applied Science and Technology Trends, 2020, 1 (2): 140-147.

[2] Neily N, Ammar B B, Kammoun H M. Prediction of COVID-19 active cases using polynomial regression and arima models [C] // International Conference on Intelligent Systems Design and Applications. Cham: Springer International Publishing, 2021: 1351-1362.

[3] Salibian-Barrera M. Robust nonparametric regression: review and practical considerations [J]. Econometrics and Statistics, 2023.

[4] Manzhos S, Ihara M. Degeneration of kernel regression with Matern kernels into low-order polynomial regression in high dimension [J]. The Journal of Chemical Physics, 2024, 160 (2).

[5] Cheung K Y, Lee S M S. High-dimensional local polynomial regression with variable selection and dimension reduction [J]. Statistics and Computing, 2024, 34 (1): 1.

[6] Di Bai. Gaussian process regression and extensions for stockmarket prediction and comparative analysis [D]. Shandong University, 2022. DOI: 10.27272/d.cnki.gshdu.2022.002950.

[7] Palar P S, Parussini L, Bregant L, et al. On kernel functions for bi-fidelity Gaussian process regressions [J]. Structural and Multidisciplinary Optimization, 2023, 66 (2): 37.

[8] Lyu C, Liu X, Mihaylova L. Review of Recent Advances in Gaussian Process Regression Methods [C] // UK Workshop on Computational Intelligence. Cham: Springer Nature Switzerland, 2022: 226-237.

[9] Ganaie M A, Hu M, Malik A K, et al. Ensemble deep learning: A review [J]. Engineering Applications of Artificial Intelligence, 2022, 115: 105151.

[10] LI Jiang-yun, DAI Wen-jiang, ZHANG Xuan-qing, et al. Distributed Hydrological Model Ensemble Simulation Based on Bayesian Model Average Method [J]. CHINA WATER & WASTEWATER, 2023, 39 (03): 116-122. DOI: 10.19853/j.zgjsps.1000-4602.2023.03.018.

[11] Ture B A, Akbulut A, Zaim A H, et al. Stacking-based ensemble learning for remaining useful life estimation [J]. Soft Computing, 2024, 28 (2): 1337-1349.

[12] Liu B. Review of swarm intelligence algorithm optimization of BP neural network [J]. Academic Journal of Computing & Information Science, 2023, 6 (6): 151-155.

[13] Asselman A, Khaldi M, Aammou S. Enhancing the prediction of student performance based on the machine learning XGBoost algorithm [J]. Interactive Learning Environments, 2023, 31 (6): 3360-3379.

[14] Antoniadis A, Lambert-Lacroix S, Poggi J M. Random forests for global sensitivity analysis: A selective review [J]. Reliability Engineering & System Safety, 2021, 206: 107312.

[15] McEligot A J, Poynor V, Sharma R, et al. Logistic LASSO regression for dietary intakes and breast cancer [J]. Nutrients, 2020, 12 (9): 2652.

[16] ZHU Rong, ZOU Guohua, ZHANG Xinyu. Optimal Model Averaging Estimation for Partial Functional LinearModels [J]. Journal of Systems Science and Mathematical Sciences, 2018, 38 (07): 777-800.