WEP
Warwick
Evans
Publishing

Transactions on Economics, Business and Management Research
ISSN: 2960-1789, eISSN: 2960-2254 | Volume 14
AFTEM 2024

A Gaussian additive model based on the stacking framework and its application in stock price forecasting

Kai Chen [1, #], Mengyao Sun [2, #], Shuaichen Ge [3, #, *]

[1] Faculty of Science, Hangzhou Dianzi University, Hangzhou, China, 310018

[2] Management Science and Engineering, Tianjin University of Finance and Economics, Tianjin, China, 300221

[3] Faculty Office of Science and Engineering, University of Nottingham Ningbo, Ningbo, China,315100

* Corresponding author: g2044426910@163.com

[#]These authors contributed equally.

Abstract. With the rapid development of big data technologies, employing stock price prediction models has become crucial in quantitative finance. Among the various prediction models, selecting appropriate features, mining factors, and quantifying the uncertainty of stock prices are essential challenges. To address these issues, in this paper proposes an improved Gaussian Process Additive Model based on the stacking method. The proposed approach introduces a "re-encoding and reduction" design in the first layer of the stacking framework, effectively filtering redundant features. Furthermore, the method uses the Gaussian Process Additive Model in the second layer of the stacking framework to capture interactions between features. By incorporating a Gaussian process prior, the model can provide uncertainty estimates for the predictions. Additionally, the flexibility in selecting sub-models and kernel functions within the Gaussian additive framework enhances the method's capacity to fit complex functions, offering a high degree of flexibility. Both simulation experiments and real-world data analysis demonstrate that the proposed method is highly competitive compared to other classical prediction models.

Keywords: Stock Price Prediction; Stacking Algorithm; Gaussian Process Additive Model; Uncertainty Quantification.

1. Introduction

In today's dynamic and ever-changing financial markets, stock price prediction has become a critical area of focus for researchers and practitioners in quantitative finance. Accurate predictions can provide valuable insights for investment decisions, playing a key role in financial derivatives pricing and risk management [1]. However, two significant challenges persist: navigating the high-dimensional factor space and effectively quantifying uncertainty. The high-dimensional factor space contributes to stock price volatility and complexity, making accurate predictions highly challenging. It not only increases the intricacy of predictive models but also heightens the risk of overfitting, ultimately undermining both the accuracy and reliability of forecasts [2-3]. Additionally, financial markets' inherent randomness and unpredictability add another layer of difficulty, highlighting the importance of incorporating uncertainty quantification into prediction models [4]. Given these challenges, this study aims to develop a stock price prediction model that not only ensures prediction accuracy but also ranks the importance of the predictive variables and quantifies the uncertainty of the results. By addressing both critical challenges, this model aims to improve both the robustness and the reliability of stock price forecasts.

Stock price prediction models generally fall into two categories: those based on stock price time series forecasting and those based on factor characteristic forecasting. For the time series approach, representative methods include the ARIMA model [5-6] and the GARCH model [7-8]. These models assume some linear dependence in stock price sequences and attempt to forecast future prices by

identifying patterns in historical price changes. However, the nonlinearity and complexity of financial markets often expose the limitations of these linear models in real-world applications [9-11]. To address these shortcomings, researchers have introduced more advanced time series techniques, such as the autoregressive integrated moving average (ARIMA) model [12] and the vector autoregressive (VAR) model [13]. While these methods show improvement, they still struggle with the challenges posed by high-dimensional factor spaces. Moreover, relying solely on historical price sequences to predict future trends can be inadequate in complex and volatile financial markets. Recognizing these limitations, this paper focuses primarily on the multi-factor prediction model [14]. With the continuous advancement of machine learning, models such as regularized multiple linear regression (e.g., LASSO and Ridge regression), support vector machine regression using kernel functions, decision tree regression based on the principle of entropy, and random forests based on the bootstrap method have become widely used in quantitative trading tasks. More advanced models like XGBoost and LightGBM, which are based on gradient boosting, also predict stock prices by leveraging feature factors [15-20]. These models, capable of capturing both linear and nonlinear relationships between predictor and response variables, offer more detailed insights into the factors influencing stock prices. However, they still face limitations when dealing with complex interactions among factors. Specifically, in high-dimensional scenarios, these models may suffer from slow convergence, increased computational costs, instability, and reduced robustness. In recent years, deep learning has been increasingly applied to stock price prediction, owing to its ability to perform powerful feature extraction and nonlinear fitting. Models like long short-term memory (LSTM) networks [21] and convolutional neural networks (CNNs) [22] can automatically extract meaningful features from vast amounts of historical data, enabling the construction of sophisticated predictive models. However, the "black-box" nature of deep learning models, along with their high demand for large-scale data, limits their application in finance—particularly when it comes to the interpretability of predictive variables and the quantification of uncertainty in forecasted results. To address these challenges, this paper proposes an enhanced Gaussian process additivity model within a stacking framework. The core objective is to improve both the accuracy and stability of stock price predictions through multi-level model integration and uncertainty analysis. The innovations of this paper include the following

(1) Introduction of a stacking framework incorporating re-encoding technology. The first innovation introduced is the "recording and reproduction" design at the initial layer of the stacking framework. The number of model parameters is successfully reduced by recoding high-dimensional factor spaces and applying dimensionality reduction techniques. This step effectively mitigates the overfitting issue typically caused by high-dimensional factors and enhances the model's generalization capabilities.

(2) Application of the Gaussian Process Additive Model. In the second layer of the stacking framework, a Gaussian process additive model is employed. This model not only ranks the importance of predictor variables and their interactions but also provides uncertainty estimates for the predicted results.

(3) Flexible Sub-model and Kernel Function Selection. The flexibility of choosing different kernel functions in the Gaussian process adds adaptability, allowing the model to handle complex data more effectively. This flexibility enables the model to adjust to varying market conditions and data characteristics, making it more robust.

(4) Validation through Simulated and Real Data. The validity of this proposed method is demonstrated through experimental analysis of both simulated and real-world stock price data. Compared to traditional time series models, machine learning approaches, and other classical prediction models, this model demonstrates superior accuracy and stability, particularly when dealing with high-dimensional factor spaces and quantifying prediction uncertainty.

In conclusion, the Gaussian process additivity model based on the stacking framework proposed in this paper offers a novel and effective approach for stock price forecasting. It not only outperforms existing models in terms of accuracy and robustness but also shows significant practical potential for application in real-world financial markets.

2. Stacking Framework

2.1. Bootstrap Sampling and Feature Selection

First, we employ the Bootstrap Sampling method to randomly draw samples from the dataset. For each base learner, we randomly select a specific feature factor column X_j(where $j \in \{1, 2,...,p\}$) from the feature matrix $X = [X_1, X_2 \ldots X_P]]$ to construct multiple distinct training subsets. Assume the training dataset is (X_{train}, y_{train}), where X_{train} is an $n \times p$ matrix representing n samples and p features. For the i^{th} base learner, we perform the following steps: Randomly sample mmm instances from X_{train} to form a subset $X_{sub}^{(i)}$ and its corresponding target values subset $y_{sub}^{(i)}$:$X_{sub}^{(i)}, y_{sub}^{(i)} = \mathrm{BootstrapeSample}(X_{train}, y_{train})$.Select a specific feature column $X_j^{(i)}$ from $X_{sub}^{(i)}$ for training:

$$X_j^{(i)} = \mathrm{SelectFeature}\left(X_j^{(i)}, j\right) \tag{1}$$

Each base learner is trained on its respective subset, with its utilized feature column recorded. In this manner, each base learner focuses on learning from different samples and a single feature factor, resulting in the generation of multiple diverse models.

In this study, Ridge regression is employed as the base learner. The objective of the Ridge regression model is to minimize the following loss function:

$$\mathcal{L}(w) = \frac{1}{2m}\sum_{k=1}^{m}\left(y_k^{(i)} - X_k^{(i)}w\right)^2 + \frac{\alpha}{2}||w||_2^2 \tag{2}$$

Where w is the weight vector of the model, α is the regularization parameter, $y_k^{(i)}$ is the target value for the k^{th}instance in the subset, and $X_k^{(i)}$ is the corresponding feature value. The inclusion of the regularization term $\frac{\alpha}{2}||w||_2^2$ effectively mitigates the impact of multicollinearity, improving the model's stability.

During the model evaluation phase, we use a validation set (X_{val}, y_{val}), to assess the performance of each base learner. The predictive accuracy of each model is quantified by calculating the correlation coefficient $\rho^{(i)}$ between the base leaner's prediction $\hat{y}_{val}^{(i)}$ and the true values in the validation set:

$$\rho^{(i)} = \frac{Cov(y_{val}, \hat{y}_{val})}{\sigma_{y_{val}}\sigma_{\hat{y}_{val}^{(i)}}} \tag{3}$$

Where $Cov(\cdot)$represents the covariance, and $\sigma_{y_{val}}$ and $\sigma_{\hat{y}_{val}^{(i)}}$ are the standard deviations of the true values and predictions, respectively. Based on the magnitude of the correlation coefficient $\rho^{(i)}$, we retain the top-performing base learners to construct the final ensemble model.

In the final prediction phase, we apply the selected base learners to the test dataset X_{test}, generating multiple predictions: $\hat{Y}_{test} = \left[\hat{y}_{test}^{(1)}\ \hat{y}_{test}^{(2)}\ \ldots\ \hat{y}_{test}^{(k)}\right]$, where k is the number of retained optimal base learners. The predictions from the training and validation sets are then combined to form a new training dataset $\hat{Y}_{train\_val}$,enabling further optimization and analysis of the model. Finally, the predictions from the combined training, validation, and test sets are saved for subsequent analysis.

2.2. Gaussian Process Additive Model

2.2.1. Gaussian Process Regression (GPR).

Gaussian Process Regression (GPR) is a nonparametric regression method grounded in Bayesian inference, widely applied in prediction and uncertainty quantification tasks. The fundamental concept

of GPR is to treat all possible fitting functions as a collection of random functions mapping the input space to the output space. By selecting an appropriate kernel function (i.e., covariance function), GPR can effectively capture complex relationships between input features, enabling accurate predictions of the target variable in regression problems. In GPR, it is assumed that the input vector XXX and the output variable y are related through the following functional relationship:

$$y = f(X) + \epsilon \tag{4}$$

Among them, ϵ is independent and identically distributed Gaussian noise, typically assumed to follow$\epsilon \sim \mathcal{N}(0\,, \sigma_n^2)$. Within the Bayesian framework, a Gaussian process assumes that the function f(X) follows a Gaussian process. This means that for any given input point X, the distribution of the function value f(X) is a multivariate Gaussian distribution:

$$\begin{gathered} f(\mathbf{X}) \sim \mathcal{GP}\big(m(\mathbf{X}), k(\mathbf{X}, \mathbf{X}')\big) \\ m(X) = \mathbb{E}[f(X)] \\ k(X, X') = \mathrm{Cov}\big(f(X), f(X')\big) \end{gathered} \tag{5}$$

Here, $m(\mathrm{X})$is the mean function, and $k(\mathrm{X}, \mathrm{X}')$ is the kernel function, which measures the similarity between different points in the input space. Typically, the data can be centered so that the mean function is zero, i.e., $(m(\mathrm{X}) = 0)$, leaving the model primarily dependent on the kernel function $k(\mathrm{X}, \mathrm{X}')$. Given a training dataset X and corresponding target values y, the goal of GPR is to infer the distribution of the function value $f(\mathrm{X}')$ at a new test point X', based on the training data. Specifically, this involves first calculating the covariance matrix between the training data and the test point:

$$\begin{gathered} \mathbf{K} = k(\mathbf{X}, \mathbf{X}) \\ \mathbf{K}' = k(\mathbf{X}', \mathbf{X}') \\ \mathbf{K}_{\text{cross}} = k(\mathbf{X}, \mathbf{X}') \end{gathered} \tag{6}$$

Then, the predicted distribution of the test point is calculated using the following formula:

$$\mathbf{f}' \sim \mathcal{N}(\mathbf{K}_{\text{cross}}^{\top}(\mathrm{K} + \sigma_n^2 \mathrm{I})^{-1}\mathbf{y}, \mathbf{K}' - \mathbf{K}_{\text{cross}}^{\top}(\mathrm{K} + \sigma_n^2 \mathrm{I})^{-1}\mathbf{K}_{\text{cross}}) \tag{7}$$

In this context, the mean value $\mathrm{K}_{\text{cross}}^{\top}(\mathrm{K} + \sigma_n^2 \mathrm{I})^{-1}\mathrm{y}$ represents the predicted value at a new test point in GPR, while the variance $\mathrm{K}' - \mathrm{K}_{\text{cross}}^{\top}(\mathrm{K} + \sigma_n^2 \mathrm{I})^{-1}\mathrm{K}_{\text{cross}}$ reflects the uncertainty of the prediction. Through the above formulas, Gaussian Process Regression can perform predictions while accounting for model uncertainty. By deriving the mean and variance of the posterior distribution from the Gaussian distribution, we can make statistical inferences about the data, giving GPR a distinct advantage in handling small sample sizes, complex models, and quantifying uncertainty.

2.2.2. Gaussian Process Regression (GPR).

An additive model (AM) represents the output variable as a weighted sum of the independent contributions of multiple input variables. This approach effectively captures the nonlinear effects of various input variables and the interactions between them while reducing model complexity and mitigating the curse of dimensionality. By combining Gaussian Process Regression (GPR) with an additive model, we can form a Gaussian Process Additive Model (GPAM). In this model, each feature of the input variable is modeled as a Gaussian process, and the predictions of these sub-models are aggregated to construct a comprehensive global model. In the Gaussian additive model, it is assumed that the relationship between the target variable y and the input variable X can be decomposed into the sum of component functions for each dimension, along with interaction terms between dimensions. The specific form is as follows:

$$y = f(\mathbf{x}) + \epsilon = \sum_{j=1}^{d} f_j(x_j) + \sum_{1 \le j < k \le d} f_{jk}(x_j, x_k) + \cdots + \epsilon \tag{8}$$

Where d denotes the dimension of the input variables, $f_j(x_j)$ represents the univariate function on the j-th dimension, and $f_{jk}(x_j, x_k)$ represents the bivariate interaction function between the j-th and k-th dimensions. ϵ denotes the independent and identically distributed Gaussian noise. Each function $f_j(x_j)$, $f_{jk}(x_j, x_k)$, etc., in the Gaussian additive model is modeled as a Gaussian process. For each univariate or multivariate function $f(\cdot)$, the covariance between function values is determined by a corresponding kernel function $k(\cdot,\cdot)$. Specifically, $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$, where the overall kernel function is the sum of the kernel functions of each variable. The overall kernel function $k(\mathbf{x}, \mathbf{x}')$ is composed of the kernel functions of individual dimensions and the interaction terms across dimensions, which can be expressed as:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{d} k_j(x_j, x_j') + \sum_{1 \le j < k \le d} k_{jk}(x_j, x_k; x_j', x_k') + \cdots \tag{9}$$

In Gaussian process regression, the kernel function determines how the model maps the input data to the function space that will be fitted. However, using traditional kernel functions can lead to difficulties in identifying the function—specifically, it can be challenging to clearly distinguish the independent contributions of input variables to the output. The orthogonal kernel (OAK) addresses this issue by constructing a set of mutually orthogonal function families. The different function spaces corresponding to the orthogonal kernel function do not overlap or interfere with each other, allowing us to identify the independent impact of each input feature or feature combination on the output through analysis of variance (ANOVA).

For each input feature x_i, we expect its corresponding sub-function $f_i(x_i)$ to satisfy the following orthogonal constraints:

$$\int_{\mathcal{X}_i} f_i(x_i) p_i(x_i) dx_i = 0 \tag{10}$$

In this context, $\mathcal{X}_i$ represents the sample space of the feature x_i, and $p_i(x_i)$ is the probability density function of the input feature x_i. This condition ensures that each sub-function $f_i(x_i)$ is orthogonal to its distribution $p_i(x_i)$ over its defined domain, thereby preventing interference between the sub-functions. To construct a kernel function that satisfies the orthogonality condition, we need to adjust the base kernel accordingly. This new kernel function can be obtained through the following equation:

$$\tilde{k}_i(x_i, x_i') = k_i(x_i, x_i') - \mathbb{E}[S_i f_i(x_i)] \mathbb{E}[S_i^2]^{-1} \mathbb{E}[S_i f_i(x_i')] \tag{11}$$

Where,

$$\mathbb{E}[S_i f_i(x_i)] = \int p_i(x_i) k_i(x_i, x_i) dx_i \tag{12}$$

$$\mathbb{E}[S_i^2] = \int \int p_i(x_i) p_i(x_i') k_i(x_i, x_i') dx_i dx_i' \tag{13}$$

By making these adjustments, we can ensure that each kernel function $k_i(x_i, x_i')$ satisfies the orthogonality condition, thereby enabling effective decomposition of the model. This method is particularly useful in constructing complex models, especially when it is necessary to ensure the independence and interpretability of each component of the model.

3. Analysis of data

3.1. Data sources and experimental Settings

In this paper, the comprehensive performance of the proposed method will be illustrated from two perspectives: simulation experiments and actual data analysis. For the simulation data analysis, the

specific experimental settings are as follows: The simulation dataset is generated by the following six formulas, where the input variables $X_1, X_2, \ldots, X_{10}$ are randomly generated from the Gaussian distribution, and y is the output variable calculated according to the formulas. The first layer of all formulas, except for Result.2, selects the five features with the largest correlation coefficients, while Result.2 selects the three features with the largest correlation coefficients. These formulas consider various nonlinear and interactive effects to test the model's performance in complex situations:

$$\text{Result.1} \quad y = x_1^2 - 2x_2 + \cos(3x_3) \cdot \sin(5x_4) + e^{x_5} + \xi \tag{16}$$

$$\text{Result.2} \quad y = x_1^2 - 2x_2 + cos(3x_3) \cdot \sin(5x_4) + e^{x_5} + \log(|x_6|) + \xi \tag{17}$$

$$\text{Result.3} \quad y = x_1^2 - 2x_2 + cos(3x_3) \cdot \sin(5x_4) + e^{x_5} + \log(|x_6|) + \xi \tag{18}$$

$$\text{Result.4} \quad y = \log(x_1 + 2) + arcsin(\frac{x_2}{2}) + x_3^2 \cdot e^{-x_4} + sin(2\pi \cdot x_5) + \xi \tag{19}$$

$$\text{Result.5} \quad y = x_1^2 + sin(x_2) + \log(x_3 + 1) + e^{x_4} + cos(x_5) + \sqrt{x_6 + 4} + tan(x_7) + arcsin(\frac{x_8}{2}) + x_9^3 - 2x_{10} + \xi \tag{20}$$

Result.6 Based on Result 5, the data size is expanded from 100 to 10000.

where ξ follows a normal distribution. We set the number of samples corresponding to each function to be 100 and the data characteristics to be 10. We used Python's method, specifically 'np.random.normal',to generate a Gaussian distribution with a mean of 0 and a variance of 0.01, and used the above function formula to generate the corresponding y. Then, all the data (100 samples) were divided into training verification sets (70%) and test sets (30%). Then the training validation set is further divided into training set (70%) and validation set (30%), in which the training set data is used to train the ridge regression model, the validation set data is used to select the appropriate features according to the correlation coefficient ranking, and the test set data is used for prediction. Finally, we combine the output of the training set (49%) and the validation set (21%) with y as the training set for the oak model, and the output of the test set with y as the test set for the oak model.

For the real data analysis in this study, the data is sourced from the Tushare Pro website and provided by the Shenzhen Stock Exchange. The dataset includes daily trading information from the Shenzhen stock market, such as basic company data, stock price trends, and other key financial indicators. The research focuses on several companies listed on the Shenzhen Stock Exchange, specifically within the stock code range 000001.SZ to 000017. SZ. Each stock contains 242 samples, representing 242 days of stock price rise and fall data, forming a complete time series that can fully reflect the performance of stocks under different market conditions. By training independent models for these 17 stocks, each model can capture the unique behavior and characteristics of each stock. These stocks cover different industries and can reflect the overall dynamics of the market, so they are sufficient to represent broader market trends. The purpose of this study is not to comprehensively analyze stocks in all securities markets, but to verify the effectiveness and applicability of the model by analyzing stocks on the Shenzhen Stock Exchange. These stocks span different sectors of the market, making them broadly representative and ideal for testing the model’s performance under various market conditions. The selected predictive variables include daily closing price, opening price, highest price, lowest price, turnover, turnover rate, and other relevant financial metrics. These variables are commonly used in stock price prediction models as they effectively reflect the stock’s market performance and liquidity. The proposed model ranks the importance of these predictive factors within the high-dimensional factor space and provides quantitative estimates of uncertainty in stock price predictions. To evaluate the model's performance, the root mean square error (RMSE) is calculated on the test set. RMSE measures the difference between the predicted and actual values, with a lower RMSE indicating better prediction accuracy. Through this approach, the model’s predictive capability and stability in real-world stock market scenarios are rigorously assessed. The calculation formula is:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{21}$$

For horizontal analysis of the performance of the proposed methods, comparison methods in this paper include K-nearest neighbor regression (KNN) [23], LASSO regression (LASSO) [24], and random forest model(RF) [25].

3.2. Simulation experiment and empirical analysis

For the simulation experiment, we employed the OAK model, k-nearest neighbor regression, lasso regression, and random forest regression. The error metrics are presented in Table 1 and Figure 1. In this experiment, different models were applied to predict five distinct functions, and the RMSE values for each model under various scenarios were calculated.

The OAK model consistently performed well across most results (such as Result1, Result2, Result3, and Result4), particularly excelling in Result2 and Result4, where it achieved RMSEs of 0.14157 and 0.105972, respectively, significantly outperforming the other models. This superior performance can be largely attributed to the OAK model's ability to capture complex nonlinear relationships among the features. However, in Result 6, the OAK model's performance deteriorated notably, with an RMSE of 0.308975. This decline is likely due to the substantial increase in data volume (from 100 to 10,000), which elevated the computational complexity and memory demands of the OAK model. Consequently, this may have led to overfitting or an overly complex model. In contrast, simpler models like k-nearest neighbor and random forest demonstrated more robust performance under this scenario, achieving lower RMSEs of 0.138626 and 0.12853, respectively.

Table 1. Error results under different results

Method	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6
OAK	0.1023	0.5371	0.1415	0.1059	0.1729	0.3089
KNN	0.1950	0.9097	0.1668	0.1333	0.1882	0.1386
LASSO	0.4681	0.6555	0.2066	0.2678	0.3268	0.3089
RF	0.2060	0.5546	0.1359	0.1199	0.1872	0.1285

Overall, the performance of different models varies significantly across outcomes. The OAK model demonstrates clear advantages in capturing complex nonlinear relationships, but its effectiveness may be compromised when handling large datasets. In contrast, models like k-nearest neighbor and random forest exhibit robust performance, particularly when dealing with simpler or larger datasets. The Lasso regression model, however, consistently underperforms across all cases, with especially poor results in Result 1, where the RMSE reaches 0.90978. This indicates that the Lasso model may be inadequate for datasets with highly nonlinear relationships. In summary, each model has its strengths and weaknesses depending on the scenario. The OAK model is generally better suited for addressing complex nonlinear problems, but caution is required as the data volume increases.

In the empirical analysis, this study selected stocks from the Tushare website, ranging from 000001.SZ to 000017.SZ, with approximately 242 data records for each stock. The data features include opening price (open), highest price (high), lowest price (low), closing price (close), previous day's closing price (pre_close), trading volume (vol), and trading amount (amount). These features are used as input variables (X) for the model, while the price change (change) is used as the target variable (y).

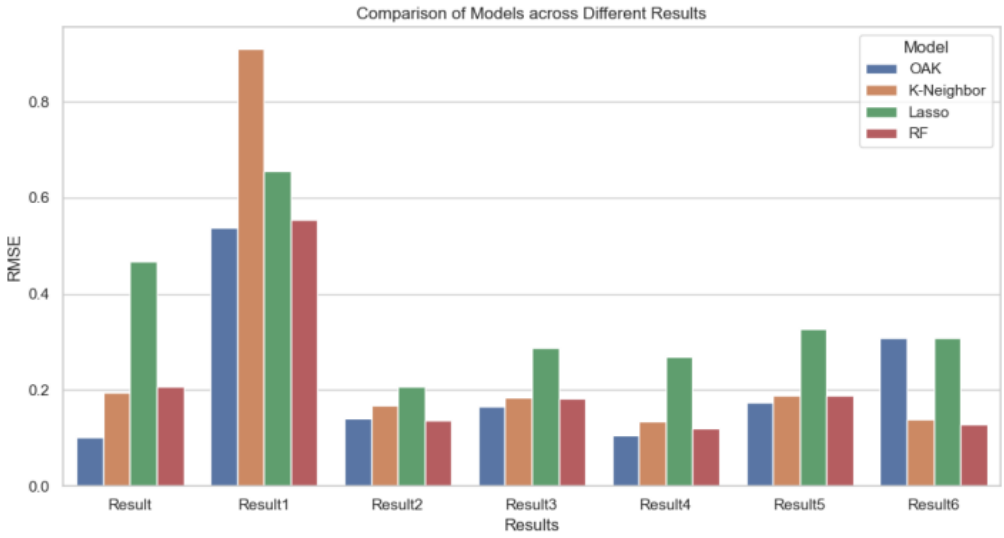

Figure 1. Visual comparison of error results under different results

In the experimental design, these input variables were first passed through a ridge regression model in the first layer for feature selection, identifying the features most correlated with the target variable. The selected features were then fed into the second-layer OAK model for further modeling and prediction. To compare the performance of different models, the second layer of the stacking model was also replaced with k-nearest neighbor regression (K-Neighbor), Lasso regression (Lasso), and random forest regression (RF). The effectiveness of each model was evaluated by comparing the RMSE values across different stocks to determine the optimal model. The results of the empirical analysis are presented in Table 2 below:

Table 2. Comparison of error results of different stocks using different methods

Stock name	OAK	KNN	LASSO	RF
000012.SZ	0.0632	0.0710	0.0595	0.0696
000011.SZ	0.1302	0.2083	0.2031	0.1670
000010.SZ	0.1096	0.1008	0.1097	0.0962
00009.SZ	0.0867	0.1834	0.1636	0.1645
00008.SZ	0.0014	0.0196	0.0356	0.0170
00007.SZ	0.1234	0.1138	0.1205	0.0867
00006.SZ	0.0099	0.1487	0.1619	0.0848
00005.SZ	0.0140	0.0328	0.0353	0.0311
00004.SZ	0.3713	0.4776	0.5538	0.4464
00002.SZ	0.2038	0.1779	0.2046	0.1763
00001.SZ	0.1221	0.1712	0.1229	0.1673
000016.SZ	0.0112	0.0796	0.0858	0.0684
000017.SZ	0.0416	0.0743	0.0742	0.0507

The results of multiple experimental data are visualized in Figures 2 and 3. The histogram illustrates the RMSE performance of different models across various stocks. It is evident that the OAK model consistently achieves lower RMSE values for most stocks, particularly for stocks like 000012.SZ and 000006.SZ, where it demonstrates significant advantages. However, on certain stocks, such as 000004.SZ, the OAK model does not perform as well as other models, resulting in a relatively high RMSE. This variability may be due to the complexity of the data or the correlation among features. Stock prices that fluctuate more, have a higher degree of complexity, and it is limited to simply capture their characteristics. However, compared with some existing classic methods, the orthogonal Gaussian additive kernel model still has better results.

In contrast, the Lasso model exhibits higher RMSE values on some stocks, suggesting a weaker ability to capture specific data characteristics. The KNN and RF models show considerable variation in

RMSE across different stocks, with more pronounced fluctuations, highlighting the instability of these models when handling diverse datasets.

The box plot with discrete points is to study the robustness of the proposed method and the comparison method From the actual experiment and the figure, we can see that only one stock shows a discrete state in each method, which shows that the models have good stability. The box plot reveals that the RMSE distribution for the OAK model has a lower median and a narrower range of fluctuations compared to other models, indicating more stable performance and generally lower errors across multiple stocks. While the OAK model does have a few large outliers (e.g., points exceeding 0.3), most of its predictions maintain a low error range. In contrast, the KNN, Lasso, and RF models have higher medians and wider fluctuation ranges, particularly in the KNN and Lasso models, where significant error values (e.g., points near 0.5) are evident. This suggests that although these models may perform well in specific instances, they lack the stability and consistency demonstrated by the OAK model.

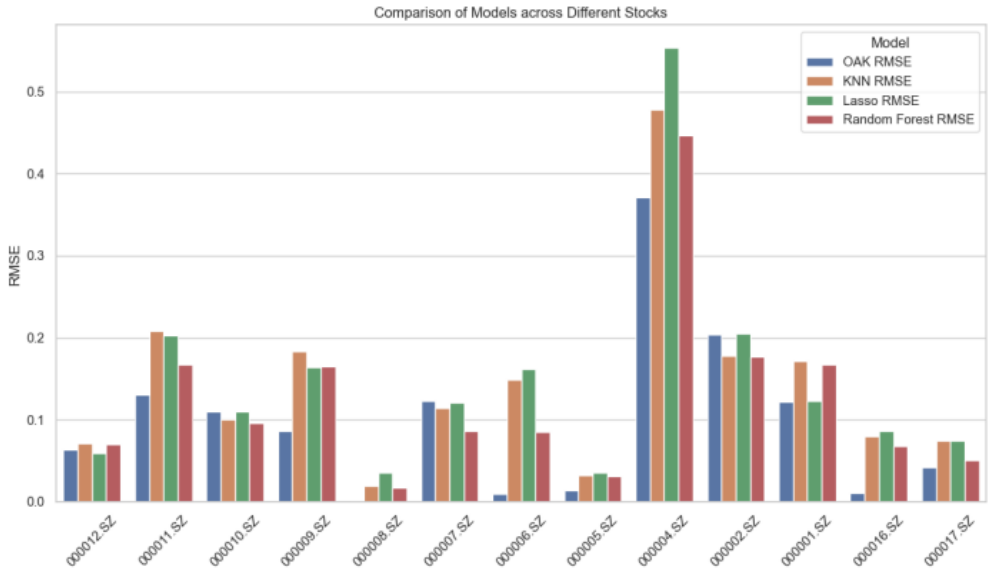

Figure 2. Visual comparison of the error of different methods for different stocks

In summary, the OAK model demonstrated relatively stable and consistent predictive capabilities in this empirical analysis. Although it encountered some challenges with certain specific stocks, it generally outperformed the other models. The performance of the KNN and RF models varied significantly across different stocks, indicating a degree of instability, while the Lasso model exhibited slightly inferior overall performance. These findings suggest that the OAK model offers high adaptability and robustness when processing financial time series data.

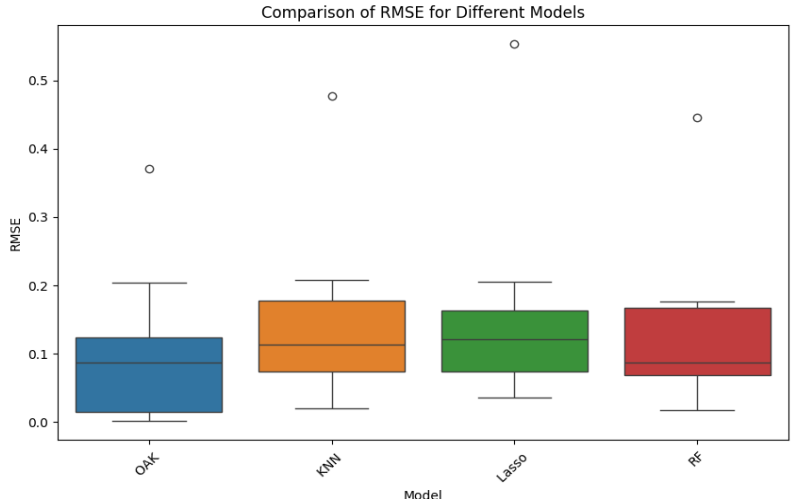

Figure 3. Visual comparison of error fluctuation results of different methods under different stocks

To enhance the interpretability of the model, this paper focuses on stocks No. 11 and No. 12 (stock codes 000011. SZ and 000012. SZ). The Sobol index is employed to understand the model's behavior, providing an explanatory analysis of the variables. In the first layer, the four features with the highest correlation coefficients are selected, while the remaining features are filtered out. For stock No. 11, the retained features are [low, open, pre_close, amount], and for stock No. 12, the features are [high, close, vol, amount]. These recorded data are then imported into the orthogonal Gaussian additive model (OAK). After model training, the Sobol index of each feature is calculated within the OAK model, taking into account the interaction effects across all dimensions. From the 15 possible effects, the three most significant single or interaction effects are selected for visualization, aiming to identify how interactions between features influence the model's output. The interpretability results are displayed in Figure 4. The three figures on the left of Figure 4 illustrate the impact of interactions between the low price (low) and other variables (volume, opening price, and amount) on the price change (change) for stock No. 11.

1. Low & Vol (R = 0.430): This figure shows the interaction between low price and trading volume. A clear nonlinear relationship is evident between the two variables, demonstrating a strong explanatory power for the price change (R = 0.430). The contour lines in the figure indicate that when both the low price and trading volume are simultaneously low or high, the fluctuation range of the price change is substantial. This suggests that changes in trading volume amplify the influence of low prices on price fluctuations to a certain degree.

2. Low & Open (R = 0.358): The interaction between low price and opening price exhibits a moderately strong nonlinear relationship (R = 0.358). The contour distribution shows that changes in the opening price have a more pronounced impact on price fluctuations when the low price is higher. Specifically, when the opening price is low, there is a tendency for the price change to experience significant negative fluctuations. This suggests that the synergy between the opening price and the low price can better explain the trend in price changes.

3. Low & Amount (R = 0.078): The interaction between low price and transaction amount is relatively weak (R = 0.078), as indicated by the relatively flat contour lines in the figure. This suggests that the relationship between these two variables is not as pronounced as in the previous two cases. Nevertheless, within certain transaction amount ranges (particularly lower amounts), fluctuations in low prices may still influence the price change to some extent. However, overall, this interaction effect provides limited explanatory power for price changes.

In summary, the interaction effects between low price and trading volume, as well as opening price, have a more significant impact on price changes, while the interaction between low price and transaction amount is relatively weaker. This indicates that low price is a crucial feature, and its interaction with trading volume and opening price plays a significant role in explaining variations in price changes.

The three pictures on the right side of Figure 4 show the interaction effect between the low price (low) of stock No. 12 and other variables (volume, opening price, and transaction amount) on the change. First High & Close (R = 0.923): This chart shows the interaction effect between high prices and closing prices. It can be seen that there is a significant linear relationship between the two variables, and this relationship has a very strong explanatory power on the increase and decrease (R = 0.923). The contour lines marked in the figure show that when the high price and closing price are both high at the same time, the fluctuation range of the rise and fall increases significantly, indicating that the linkage relationship between these two variables has a very strong explanatory power in explaining market price changes. Secondly, High & Amount (R = 0.038): The interaction effect between high price and turnover shows a weak non-linear relationship (R = 0.038). It can be seen from the contour distribution that although high prices and trading volume will affect the rise and fall in some cases, overall, this interactive effect has relatively low explanatory power on the rise and fall. Especially when the transaction volume is high, the impact on market price fluctuations is relatively limited. Finally, Close & Amount (R = 0.027), the interaction effect between closing price and trading volume

is also weak (R = 0.027). The contour lines shown in the figure are relatively flat, indicating that the linkage of these two variables has a positive impact on the rise.

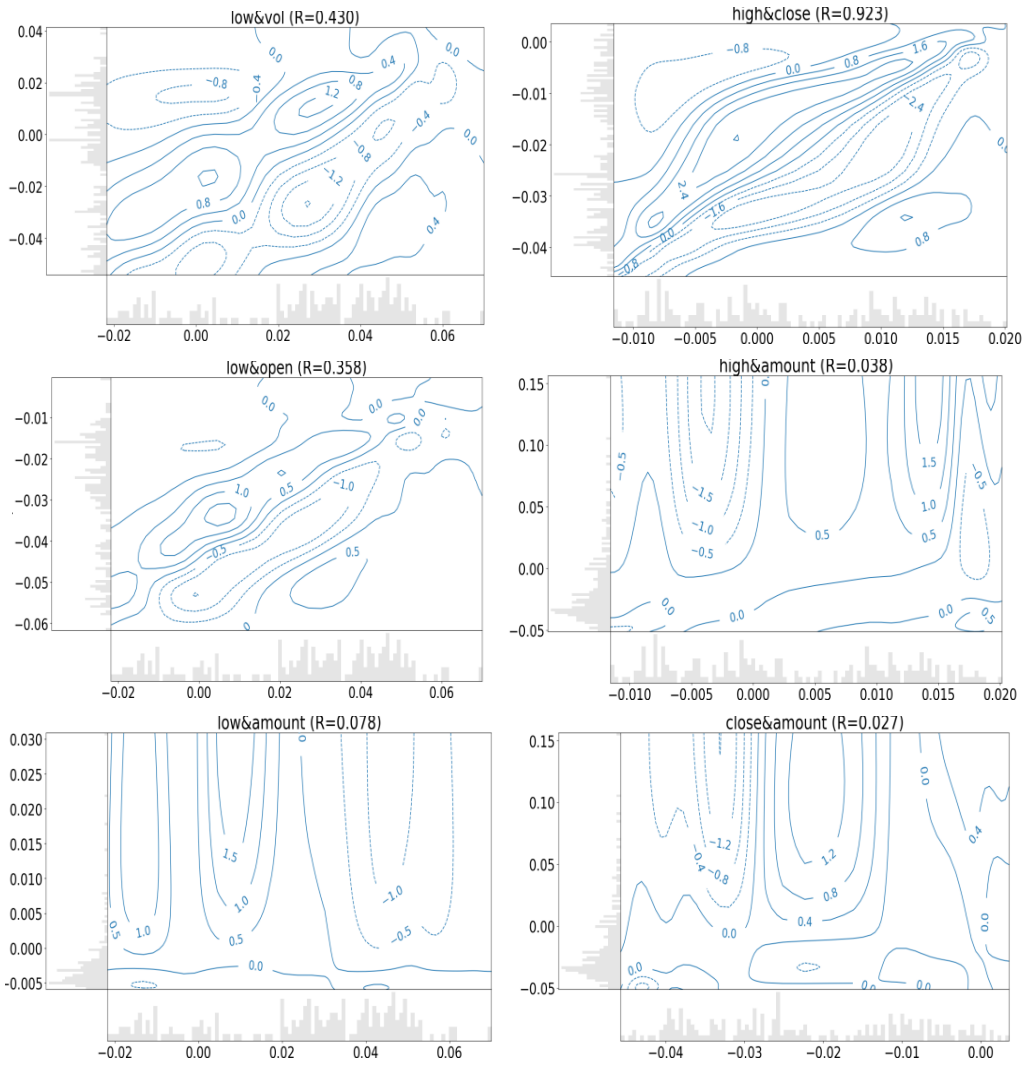

Figure 4. Contour plot of interpretability results (number 000011. SZ (left) and 000012. SZ (right))

The impact of the decline is small. Nonetheless, in some cases, this interactive effect may still explain part of the market volatility, but the overall impact is low. In summary, the interactive effect between high price and the closing price has the most significant impact on the increase and decrease, while the interactive effect between high price and transaction volume, closing price, and transaction volume is relatively weak. When explaining market fluctuations, priority should be given to the linkage relationship between high prices and closing prices, which is of great significance to understanding market price changes.

4. Conclusions and Extension

This paper presents an improved Gaussian Process Additive Model based on the Stacking framework, which integrates Ridge Regression with the Orthogonal Additive Kernel Gaussian Process (OAK-GP). The aim is to efficiently handle high-dimensional linear data and complex nonlinear relationships. By employing Ridge Regression in the first layer, we perform independent regression analyses for each input dimension and use correlation coefficients to filter models that contribute significantly to the predictions, thereby enhancing overall model performance. In the second layer, the OAK-GP model captures the intricate interactions between input features and utilizes Sobol index analysis to assess the contribution of individual features and their interactions, providing strong interpretability and the ability to conduct variable importance analysis. Experimental results demonstrate the exceptional performance of this model in stock price prediction tasks. Not only does

it accurately forecast stock trends, but the calculation of Sobol indices also enables an in-depth analysis of the contribution of each feature to the model's output. Furthermore, the additive Gaussian process framework provides uncertainty estimation capabilities, offering decision-makers more detailed reference materials. The model has potential applications in other fields requiring high-precision prediction and uncertainty quantification.

The flexibility inherent in the model's kernel functions allows for the incorporation of both Euclidean and non-Euclidean data types. By selecting appropriate non-Euclidean kernel functions, the model can effectively handle diverse input variables, such as text data alongside traditional numerical datasets. This capability broadens the scope of potential inputs and enhances the model's applicability. Furthermore, the original kernel function can be transformed into a kernel matrix that captures not only the relationship between inputs and outputs but also the interdependencies among multiple outputs. By employing maximum likelihood estimation to assess the parameters within this matrix, we can derive a multi-output orthogonal additive model. This comprehensive framework enhances the model's interpretability and enables a more nuanced understanding of the interactions between different input variables and their collective impact on the predicted outcomes.

References

[1] Anand C. Comparison of stock price prediction models using pre-trained neural networks [J]. Journal of Ubiquitous Computing and Communication Technologies, 2021, 3 (2): 122 - 134.

[2] Gao Z, Tsay R S. A structural-factor approach to modeling high-dimensional time series and space-time data [J]. Journal of Time Series Analysis, 2019, 40 (3): 343 - 362.

[3] Ando T, Bai J. Clustering huge number of financial time series: A panel data approach with high-dimensional predictors and factor structures [J]. Journal of the American Statistical Association, 2017, 112 (519): 1182 - 1198.

[4] Ekblom J. Decision Making under Uncertainty in Financial Markets: improving decisions with stochastic optimization[M]. Linköping University Electronic Press, 2018.

[5] Khanderwal S, Mohanty D. Stock price prediction using ARIMA model [J]. International Journal of Marketing & Human Resource Research, 2021, 2 (2): 98 - 107.

[6] Mahadik A, Vaghela D, Mhaisgawali A. Stock price prediction using lstm and arima [C]//2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC). IEEE, 2021: 1594 - 1601.

[7] Naik N, Mohan B R, Jha R A. GARCH model identification for stock crises events [J]. Procedia Computer Science, 2020, 171: 1742 - 1749.

[8] Gao Z, He Y, Kuruoglu E E. A hybrid model integrating LSTM and GARCH for Bitcoin price prediction [C]//2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP). IEEE, 2021: 1 - 6.

[9] Ma Q. Comparison of ARIMA, ANN and LSTM for stock price prediction [C]//E3S Web of Conferences. EDP Sciences, 2020, 218: 01026.

[10] Ho M K, Darman H, Musa S. Stock price prediction using ARIMA, neural network and LSTM models [C]//Journal of Physics: Conference Series. IOP Publishing, 2021, 1988 (1): 012041.

[11] Inglada-Perez L. A comprehensive framework for uncovering non-linearity and Chaos in financial markets: Empirical evidence for four major stock market indices[J]. Entropy, 2020, 22 (12): 1435.

[12] Hossain M R, Ismail M T, Karim S A B A. Improving stock price prediction using combining forecasts methods [J]. IEEE Access, 2021, 9: 132319 - 132328.

[13] Hushani P. Using autoregressive modelling and machine learning for stock market prediction and trading [C]//Third International Congress on Information and Communication Technology: ICICT 2018, London. Springer Singapore, 2019: 767 - 774.

[14] Wei Z, Dai B, Lin D. Factor investing with a deep multi-factor model [J]. arXiv preprint arXiv: 2210.12462, 2022.

[15] Cheng P, Mao F, Zhao H. Relationship Study of Meltblowning Variables Based on Machine Learning Algorithm [J]. Automation and Machine Learning, 2022, 3 (2): 17 - 26.

[16] Wu, Y. Stock Price Prediction Based on Simple Decision Tree, Random Forest, and XGBoost [J]. BCP Business & Management, 2023.

[17] Jiang M, Liu J, Zhang L, et al. An improved Stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms [J]. Physica A: Statistical Mechanics and its Applications, 2020, 541: 122272.

[18] Xiaosong Z, Qiangfu Z. Stock prediction using optimized LightGBM based on cost awareness [C]//2021 5th IEEE International Conference on Cybernetics (CYBCONF). IEEE, 2021: 107 - 113.

[19] Costa A B R, Ferreira P C G, Gaglianone W P, et al. Machine learning and oil price point and density forecasting [J]. Energy Economics, 2021, 102: 105494.

[20] Ampomah E K, Qin Z, Nyame G. Evaluation of tree-based ensemble machine learning models in predicting stock price direction of movement [J]. Information, 2020, 11 (6): 332.

[21] Chong L S, Lim K M, Lee C P. Stock market prediction using ensemble of Deep Neural Networks [C]//2020 IEEE 2nd international conference on artificial intelligence in engineering and technology (IICAIET). IEEE, 2020: 1 - 5.

[22] Yang C, Zhai J, Tao G. Deep learning for price movement prediction using convolutional neural network and long short-term memory [J]. Mathematical Problems in Engineering, 2020, 2020 (1): 2746845.

[23] Barrash S, Shen Y, Giannakis G B. Scalable and adaptive KNN for regression over graphs [C]//2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP). IEEE, 2019: 241 - 245.

[24] Ranstam J, Cook J A. LASSO regression [J]. Journal of British Surgery, 2018, 105 (10): 1348 - 1348.

[25] Yuchi W, Gombojav E, Boldbaatar B, et al. Evaluation of random forest regression and multiple linear regression for predicting indoor fine particulate matter concentrations in a highly polluted city [J]. Environmental pollution, 2019, 245: 746 - 753.