Problem Chosen	2024 ShuWei Cup Summary Sheet	Team Control Number
B		2024100129342

Research on the Collaborative Estimation Method for Spatial Variables

Summary

In this paper, we focus on the integrated application of spatial interpolation, relational measurement, and machine learning techniques. We recognize the significant interactions among spatial data sampling points and their propensity to show specific patterns. We emphasize the need for in-depth analysis and integration of spatial trends for predicting target variables. Initially, data preprocessing is carried out. We construct a 266×266 matrix grid for the specified txt file, linking each matrix unit to the grid column, and then optimize the data for visualization.

For problem 1, the research focuses on the change rule of F1 target variable. In this study, random uniform sampling strategy is adopted to randomly select sample points from the original data set, which is used as the basis of Kriging interpolation algorithm to estimate the value of unsampled spatial variables. The result of interpolation is visualized by contour map. By adjusting the sample size, this study further explores the dynamic relationship between the sample size and the estimation error, and reveals the specific relationship between the two by drawing the correlation line chart.

For problem 2 , it is necessary to explore the relationship between the target variable and the co-variable. Ben, we will use Pearson or Spearman correlation coefficients for correlation tests, or use indicators such as the Gini coefficient to measure the relationship between the two. Finally, the two covariables that are most correlated with the target variables are selected as the estimated covariables.

For problem 3, we adopt a similar method to problem 1, by randomly and evenly sampling target variables and covariables, and use these sampled data to estimate the values of spatial variables at unsampled locations. The results will be shown in contour map. At the same time, we will adjust the sample size and study its association with the estimation error. To evaluate the effectiveness of different methods, plan to compare at least two different algorithms. These algorithms may include the Kriging interpolation used in continuation problem 1, or the adoption of machine learning algorithms to evaluate the effect of the covariate on the target variable.

For problem 4, we'll utilize the optimal estimation method from Question 3 to predict the trend of the target variable, given Annex 2's limited data. Analyzing associated covariables and their spatial patterns, we'll apply this model to forecast the target's unsampled values, visualizing the forecasts as contour maps.

Keywords: Machine learning, random uniform sampling, Kriging interpolation, contour map visualization, target variable prediction

Contents

1.Introduction

1.1 Background

This study explores the application of spatial statistical techniques in estimating spatial variables, especially for target variables with sparse samples. The core goal is to significantly improve the accuracy of estimates by implementing collaborative estimation techniques. The research covers the following aspects:

2. Problem analysis and restatement

2.1 Restatement of the problem

Question 1: Based on the data provided in Annex 1, this study will dig deeper into the spatial variation rule of the target variable (F1). This includes a random uniform resampling of F1 to assess the value of the variable at the unsampled points and to analyze the correlation between the sample size and the estimated error.

Question 2: We will conduct a detailed analysis of the spatial correlation between the target variable (F1) in Annex 1 and other potential covariables, and then select the two most critical covariables as auxiliary information sources in the F1 estimation process.

Question 3: Based on the analysis results of question 2, we will select one or two covariables to study the spatial change pattern of F1. In this process, we will apply random uniform resampling technique to estimate the spatial variable values of unsampled points, and systematically compare different estimation methods.

Question 4: For the target variable (F2) in Annex 2, given the scarcity of its sample data, we will adopt the method that has the best verification effect in question 3 to estimate the trend of F2, and show its application effect through concrete results.

2.2 Problem analysis

The purpose of this paper is to study the application of spatial interpolation algorithm, correlation analysis method and machine learning method in spatial data prediction. The research background involves the interdependence and tendency of spatial variables, which need to be considered in the prediction of target values. The following is a detailed analysis of the topic:

Data preprocessing : The problem requires the data to be organized into 266x266 matrix grid data format and visualized to provide an intuitive basis for subsequent analysis.

Question 1: Research on the change mode of F1 target variable Random uniform resampling method is used to select sample data, and the spatial variable values of unsampled locations are estimated based on Kriging interpolation algorithm. The relationship between sample size and estimation error is analyzed, and the optimal sample size is found by drawing line graph.3. Symbol and Assumptions.

Question 2: Study on the correlation between target variables and co-variables Correlation analysis is performed using Pearson or Spearman correlation coefficients, or the relationship

between the covariate and the target variable is analyzed based on methods such as Gini coefficient. The two covariables with the strongest correlation with the target variable are selected as the estimated covariables.

Question 3: Spatial interpolation and estimation of covariate and target variable .The target variables and covariables are randomly and evenly resamped, and estimated based on Kriging interpolation algorithm or machine learning algorithm. The relationship between sample size and estimation error is analyzed, and the performance of different methods is compared.

Question 4: Target variable trend estimation. The unsampled value of the target variable is estimated by using the optimal method determined in problem 3, combined with the covariables related to the target variable and their spatial distribution. The results are presented in the form of contour map to provide a visual basis for subsequent analysis.

3. Symbol and Assumptions

3.1 Symbol Description

symbol	implication
$Z(X)$	Regionalized variables
m	Mathematical expectations of regionalized variables
$c(h)$	Covariance function
$r(h)$	Semi-variance function
λi	The weight of the sampling point
μ	Global mean
$\sigma 2$	Estimation variance
MAE	Mean absolute error
MSE	Mean square error
$RMSE$	Root mean square error
$R2$	Coefficient of determination, representing goodness of fit
θ	The threshold of the neuron
ωi	The connection weights of neurons
F	Scaling factor
cr	Crossover probability

η	Learning rate
Xi	Characteristic matrix
Y	Target matrix
ϵ	Random error term
p	Regression model parameter

3.2 Fundamental assumptions

In order to simplify the given problem and modify it to a more suitable simulation. In reality, we make the following basic assumptions. Every assumption has a valid reason.

(1) It is assumed that spatial variables are interdependent in distribution, and this dependence shows a specific spatial pattern. This reflects the core principle of spatial statistics, namely that observation points of spatial data do not exist in isolation, but exhibit significant spatial autocorrelation characteristics.

(2) It is assumed that the results of different measurement methods for the same physical quantity are different, but these differences remain spatially consistent. This shows that the data obtained follow some common spatial structure even with various measurement methods, thus providing the possibility of integration for collaborative estimation.

(3) It is assumed that there is some degree of linear or nonlinear relationship between the selected covariate (that is, other spatial variables associated with the target variable) and the target variable, which helps to estimate the spatial distribution of the target variable more accurately. Further, we believe that the precision of the target variable space estimation can be significantly improved through carefully selected covariables.

4.Data preprocessing

Firstly, the data in Annex I is preprocessed. In view of the fact that the provided data are not clearly divided according to the 266x266 grid matrix format, this study carried out detailed regional division and arrangement for each data set. After this processing step, the distribution of each data set can be visually presented, and the specific distribution diagram is shown below.

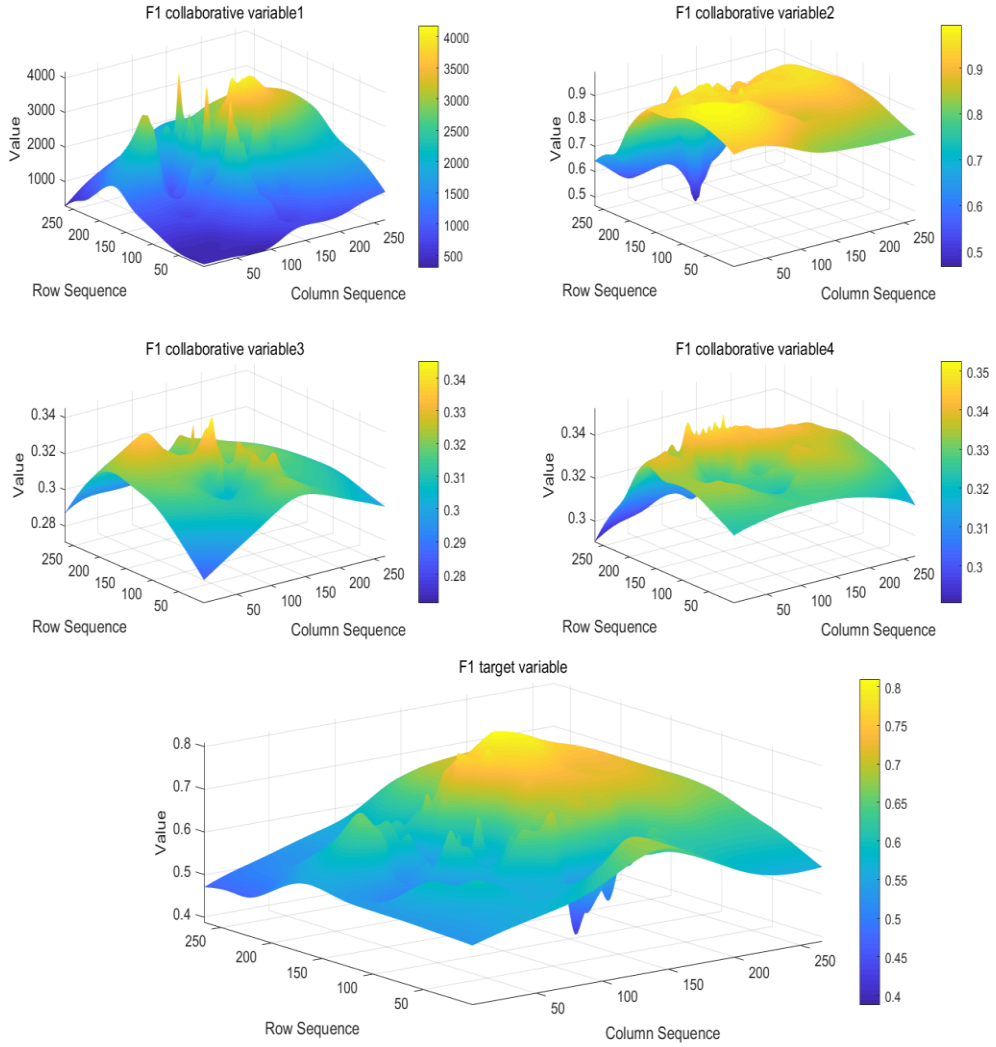

Fig. 1 Annex 1 Visualization

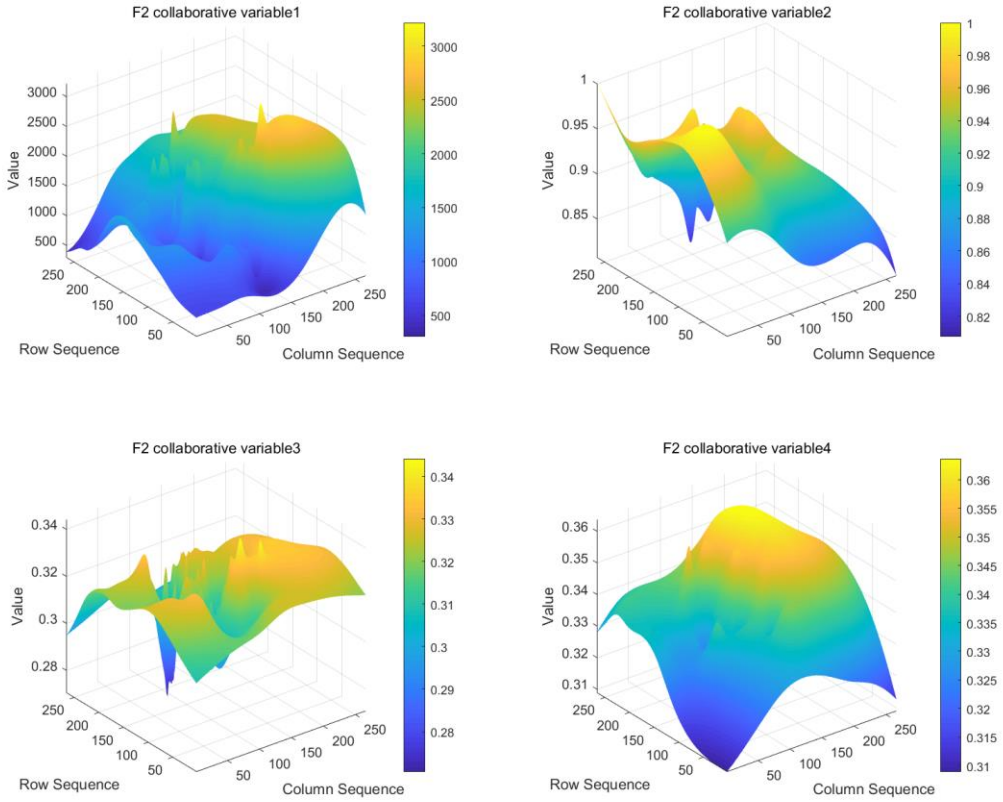

Fig 2 Appendix 2 Visualization

By visualizing the collated data, we observe that the data exhibits a continuous distribution and reveals underlying trends. The surface plot clearly shows the distribution shape of the data points as the variables continuously change, and the trend of increase, decrease or stability shown in the data series. These findings lay a solid foundation for further exploring the inherent laws and patterns of the data. In addition, it is noted that the dimension of variable 1 is significantly higher than that of the other variables, and with the exception of variable 1, the value range of the other variables is generally concentrated between 0 and 1.

5. Model establishment and solution

5.1 The establishment and solution of problem 1

5.1.1Random and uniform resampling

First, the data set is resampled randomly and evenly. The data provided in the problem form a 266x266 data matrix. Next, apply the following pseudo-code for resampling:

Resampling algorithm
function resample_data(data, sample_rate): [rows, cols] = size(data)

```
    total_elements = rows * cols
    num_to_set_nan = round((1 - sample_rate) * total_elements)

    elements_per_row = floor(num_to_set_nan / rows)
    remaining_elements = num_to_set_nan - elements_per_row * rows

    nan_mask = true(rows, cols)    # Initialize NaN mask

    for i = 1 to rows:
        current_col_index = 1
        for j = 1 to (cols - elements_per_row):
            if current_col_index <= cols:
                nan_mask[i, current_col_index] = false
                if random() < 0.5:
                    current_col_index = current_col_index + floor(cols / (cols - elements_per_row))
                else:
                    current_col_index = current_col_index + ceil(cols / (cols - elements_per_row))
                if current_col_index > cols:
                    break

        if random() < 0.5:    # Randomly reverse row order
            nan_mask[i, :] = flip(nan_mask[i, :])

    # Apply the NaN mask to the original data matrix
    new_data = data
    new_data[nan_mask] = NaN

    return new_data
```

This algorithm aims to realize data resampling by randomly replacing some elements of the data matrix with NaN values, while maintaining the original shape and layout of the matrix. The specific operation process is as follows:

Input: The algorithm receives the data matrix and the resampling rate (sample_rate) as input parameters. This ratio specifies the proportion of data that should be retained in the matrix, and unselected elements will be assigned the value NaN.

Setup phase: First, the algorithm calculates the total number of elements in the matrix and determines the number of elements to be replaced with NaN.

Distribution strategy: Next, the algorithm determines the number of elements that should be removed per row, aiming to distribute NaN values evenly between rows and columns. Ensure a balanced distribution by adjusting the allocation of remaining NaN.

Row inversion randomness: When processing each row, the algorithm randomly flips the

order of NaN values in the row with a 50% probability, introducing randomness into the resampling process.

Apply NaN values: Finally, the algorithm applies the generated NaN mask to the original data matrix, replacing the data at the specified location with NaN.

Result: After processing, the algorithm outputs a new data matrix containing NaN values, maintaining the shape of the original matrix while introducing ordered missing parts in the data

In this study, a variety of different sampling rates are applied to the matrix to achieve resampling effect. The sampling rates used include: 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, and 0.1. The following is the image display of the corresponding resampling results:

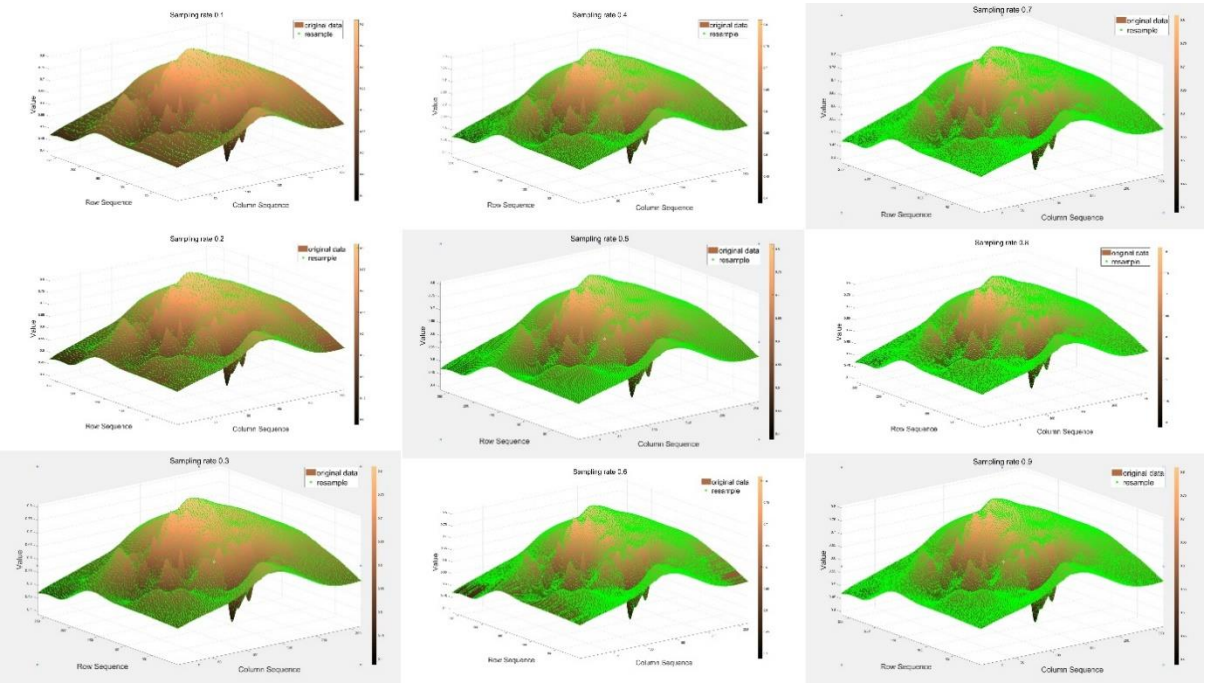

Fig. 3 Resampling diagram

It can be seen from the images that the resampling method proposed in this study performs well and can evenly perform the sampling operation at the preset sampling rate.

5.1.2 Spatial interpolation based on Kriging interpolation algorithm

5.1.2.1Establishment of model

Kriging interpolation [1], also known as spatial autocovariance optimal interpolation, is an advanced linear optimal and unbiased estimation technique specifically for the processing of spatially distributed data. The technique was developed in the 1950s by South African engineer D.G. Krieg and statistician H.S. Sischer co-founded. The core of the method is to calculate the sample average value of unknown space points by dynamic weighted average technique based on the spatial position and correlation of samples. This method was further developed by Matheron, a French geostatistician, in recognition of Kriging's pioneering work in introducing statistical theory to the field of geological assessment in 1951. Through in-depth analysis of the shape, size, layout and interaction of samples, Krigin interpolation technique integrates the complex structure information of variance function, so as to achieve accurate linear unbiased optimal prediction of unsampled points.

This method is suitable for regionalized variables with spatial correlation and makes full

use of spatial sampling information. It takes into account not only the data of the point to be estimated, but also the information of the neighboring points and the spatial relationship between these points, and makes use of the distribution structure of the observed values, so it is more accurate than other methods and can provide estimation errors.

The key concepts of Kriging interpolation include: regionalized variable, variance function and covariance function.

(1) Regionalized variable theory

When a variable's value in space is associated with its location, we call it a regionalized variable. Such variables reveal the characteristics of spatial property distribution, for example, in the study of natural gas hydrate resources, such resource distribution also shows clear spatial properties. The regionalized variable has two characteristics: before observation, it appears as a random field; After observation, it is expressed as a specific space point function value. The features such as the number and density of interpolation points belong to the category of regionalized variables, and their spatial structure and dependency can be explored through the theory of regionalized variables.

(2) Variance function

The function of variation, also known as the function of variation or the moment of spatial variation, mainly serves to reveal the spatial correlation and internal structure of regionalized variables. In one-dimensional space, the function specifically shows how the regionalized variable $\boldsymbol{z(x)}$ varies between two different points $\boldsymbol{x}$ and $\boldsymbol{x+h}$ as the space point x moves along the X-axis. For each distance vector h, the variation of variable $\boldsymbol{z(x)}$ between point x and point $\boldsymbol{x+h}$ $\boldsymbol{z(x)-z(x+h)}$ exhibits a constant variance that does not change with position $\boldsymbol{x}$. Based on this, we can measure the degree of spatial variation of the regionalized variable over the distance $\boldsymbol{h}$ by calculating the expected value of the square of this change - the variance:

$$r(h)=\frac{1}{2}\mathrm{var}[z(x)-z(x+h)]$$

$\boldsymbol{r(h)}$ is known as a semi-variance function. The graph of the function depicts the relationship between the semi-variance function $\boldsymbol{r(h)}$and the distance $\boldsymbol{h}$, which contains three key characteristic parameters: the base value, the variation range, also known as the space dependent range, and the nugget value, also known as the regional discontinuity value. Together, these parameters shape the overall properties of the semi-variance function. For the observational data series $\boldsymbol{z(x)}$, where i takes the value 1, 2, 3, up to n, the value of the sample semi-variance function can be calculated by the following formula:

$$R(h)=\frac{1}{2}N(h)\sum[z(x_i)-z(x_i+h)]^2$$

In the formula, $\boldsymbol{N(h)}$ represents the number of data pairs (x_i, x_i+h) separated by distance $\boldsymbol{h}$; $\boldsymbol{z(x_i)}$ and $\boldsymbol{z(x_i+h)}$ are sample measurements at positions $\boldsymbol{(x_i)}$ and $\boldsymbol{(x_i+h)}$, respectively; And $\boldsymbol{h}$ represents the distance between the samples. The variance function can show the whole picture of spatial variation at different scales, but its significance is mainly manifested at half of the maximum spacing distance.

(3) Covariance function

The Kriging method is a geostatistical tool used to estimate the internal interpolation between observed sample points. This method is based on the theory of regionalized variables. Once the semi-variance function model of a variable is established, the data of observed sample points can be used to estimate the regionalized variable values of unsampled points in the study area. Assuming that the regional variation $\boldsymbol{Z(x)}$ satisfies the second-order stationarity and the eigenhypothesis, the mathematical expectation is $\boldsymbol{m}$, the covariance is $\boldsymbol{c(h)}$, and the variance function $\boldsymbol{r(h)}$ exists, that is:

$$\left.\begin{aligned} &E[Z(x)] = m \\ &c(h) = E[Z(x)Z(x+h)] - m^2 \\ &r(h) = \frac{1}{2}E[Z(x) - Z(x+h)]^2 \end{aligned}\right\}$$

The estimate at the predicted point$\boldsymbol{x_0}$can be obtained by summing the observed value $\boldsymbol{Z(x_i)}$ of the $\boldsymbol{n}$ sampling points around it with linear weights, as follows:

$$Z(x_0) = \sum_{i=1}^{n} \lambda_i Z(x_i)$$

In this formula, the weight of the sampling point $\boldsymbol{x_i}$ is different from that of the reciprocal distance interpolation method: the weight here is not only based on the distance between the prediction point and the sampling point, but also considers the spatial distribution relationship between the prediction point and the sampling point, as well as between the sampling points.

To ensure an unbiased optimal estimate, two key conditions need to be followed:

(1) Unbiased estimation, that is

$$E[z'(x) - z(x)] = 0$$

(2) The estimation variance is minimum, that is, $\boldsymbol{var[z'(x) - z(x)] \rightarrow min}$ on the premise of the existence of different functions, the ordinary Kriging model and its estimation variance can be expressed with the help of the relationship between covariance and variance function $\boldsymbol{C(h) = C(O) - r(h)}$:

$$\sum_{i=1}^{n} \lambda_i \gamma(x_i, x_j) + \mu = \gamma(x_i, x), \sum_{i=1}^{n} \lambda_i = 1, (i = 1,2, \cdots n)$$

$$\sigma_k^2 = \sum_{i=1}^{n} \lambda_i \gamma(x_i, x) - \gamma(x, x) + \mu$$

And the weight $\boldsymbol{\lambda_i}$ is required to satisfy the above equation.

Where $\boldsymbol{\gamma(x_i, x_j)}$ is the half-variation between the observation point $\boldsymbol{x}$ and $\boldsymbol{x}$, $\boldsymbol{\gamma(x_i, x)}$is the half-variation between the sampling point$\boldsymbol{x_i}$ and the interpolation point $\boldsymbol{x}$, and $\boldsymbol{\mu}$ is the Lagrange multiplier related to the variance minimization. The weight value can be calculated from this equation, and the internal interpolation $\boldsymbol{z(x)}$ at the point $\boldsymbol{x}$ to be estimated can be obtained by substituting it into the above equation. The equation can also be expressed as a matrix:

Where, $\boldsymbol{\gamma(x_i, x_j)}$ represents the half-variation between the observation point $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$, $\boldsymbol{\gamma(x_i, x)}$ is the half-variation between the sampling point$\boldsymbol{x_i}$ and the interpolation point $\boldsymbol{x}$, and $\boldsymbol{\mu}$ is the Lagrange multiplier associated with the variance minimization. By calculating the weight of this equation, and then substituting the weight value, the internal interpolation $\boldsymbol{z(x)}$ of the point $\boldsymbol{x}$ to be estimated can be obtained. The equation can also be expressed in matrix form.

$$K = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} & 1 \\ c_{21} & c_{21} & \cdots & c_{21} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}, \lambda = \begin{bmatrix} \lambda_1 \\ \lambda 2 \\ \vdots \\ \lambda n \\ -\mu \end{bmatrix}, D = \begin{bmatrix} c(x_1, x) \\ c(x_2, x) \\ \vdots \\ c(x_n, x) \\ 1 \end{bmatrix}$$

Then the Kriging equation is:

$$K\lambda = D$$

Its estimated variance is:

$$\sigma^2 = c(x, x) - \lambda^T D$$

5.1.3 Model solution

The core principle of Kriging interpolation involves in-depth analysis of the probability distribution of spatial attributes with spatial location, and then defining the proximity range that is critical to the target interpolation point. On this basis, the technique uses the sampled data within this distance range to calculate the attribute estimate of the target region by accurate weighted average method. This method pursues an optimal linear unbiased interpolation estimation, which not only considers the geometric characteristics, size and spatial layout of samples, but also assigns appropriate weight coefficients to each sample to ensure the accuracy and rationality of attribute estimation.

The calculation process includes:

1.Enter the sampled data.

2. Determine the area scope and mesh size for meshing.

3. Test the data and remove the outliers.

4. Calculate histogram and decide data preprocessing.

5. Apply variance function calculation to explore the spatial structure of variables.

6. Perform Kriging interpolation estimation

① Estimation of weight coefficient of the point to be estimated

When the polygon estimation method is adopted, the weight of the sampling point nearest to the point to be estimated is first calculated. The specific estimation method follows the following formula

$$\lambda_i = \frac{\frac{1}{c + d_i^w}}{\sum_{i=1}^{n} c + d^w}$$

After the weight of the nearest sampling point is obtained, the weight of the point to be estimated is determined according to the established method.

② According to the search strategy, select the appropriate reference points.

③ Based on the known variance function and the number of sampling points (such as three sampling points), three equations are constructed and the coefficients are obtained by solving the equations as follows:

$$\begin{bmatrix} C(1,1) & C(1,2) & C(1,3) \\ C(2,1) & C(2,2) & C(2,3) \\ C(3,1) & C(3,2) & C(3,3) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} C(0,1) \\ C(0,2) \\ C(0,3) \end{bmatrix}$$

④ The influence of anisotropy change on the weight value when the nugget value is fixed and the effect of isotropy on the weight value when the nugget value changes are discussed.

⑤ By using the weight value obtained and substituting the corresponding formula, the linear combination of n sample values in the evaluation area can be calculated.

According to the above solution method, the comparison of contour maps before and after interpolation can be seen in the following figure

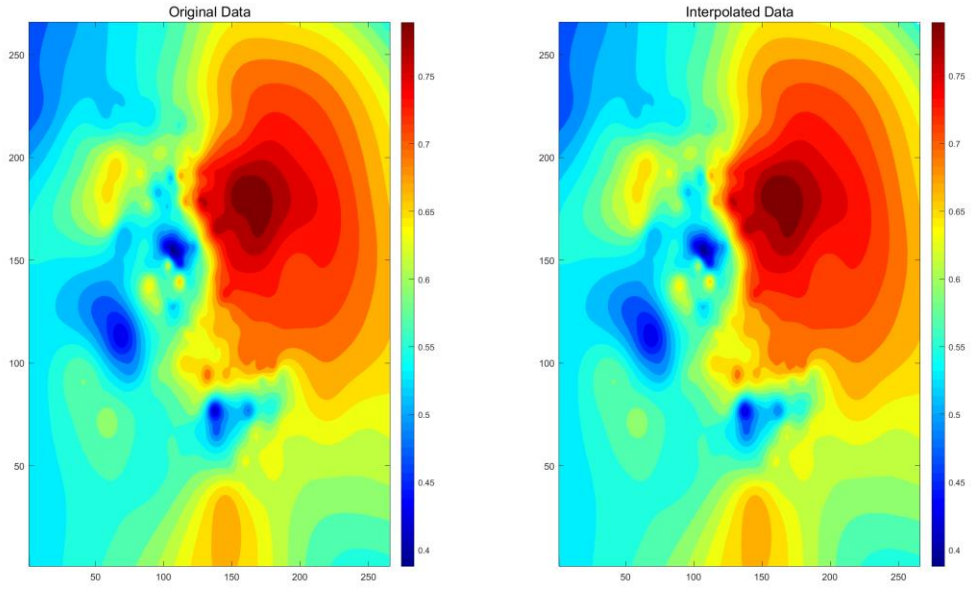

Fig. 4 Interpolation results with a sampling rate of 0.1

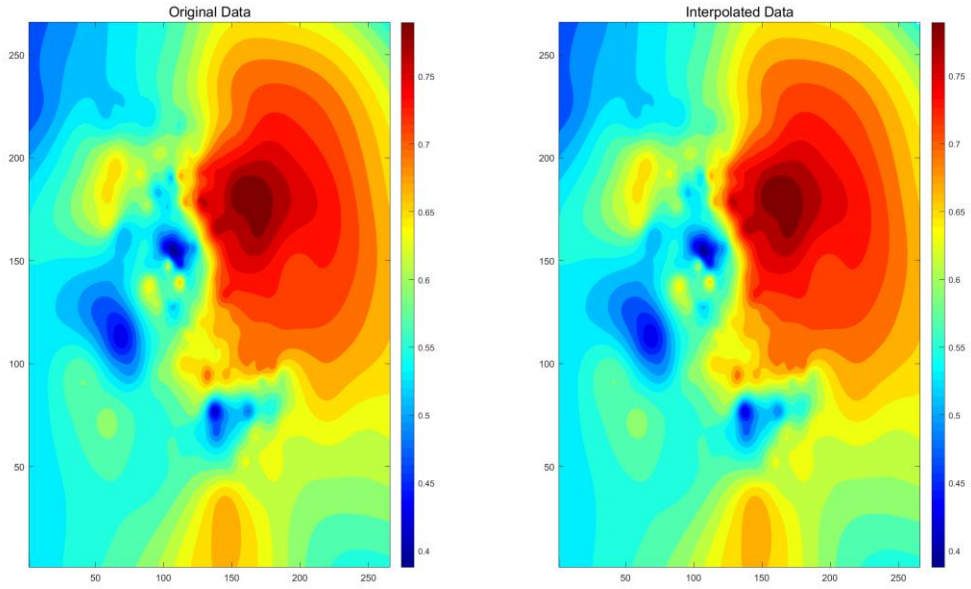

Fig. 5 Interpolation results with a sampling rate of 0.2

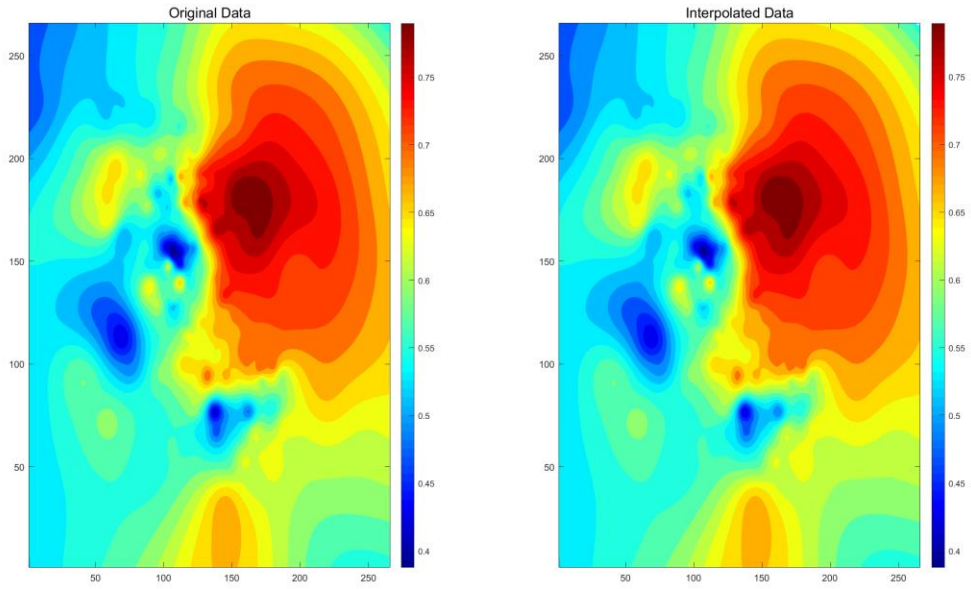

Fig. 6 Interpolation results with a sampling rate of 0.3

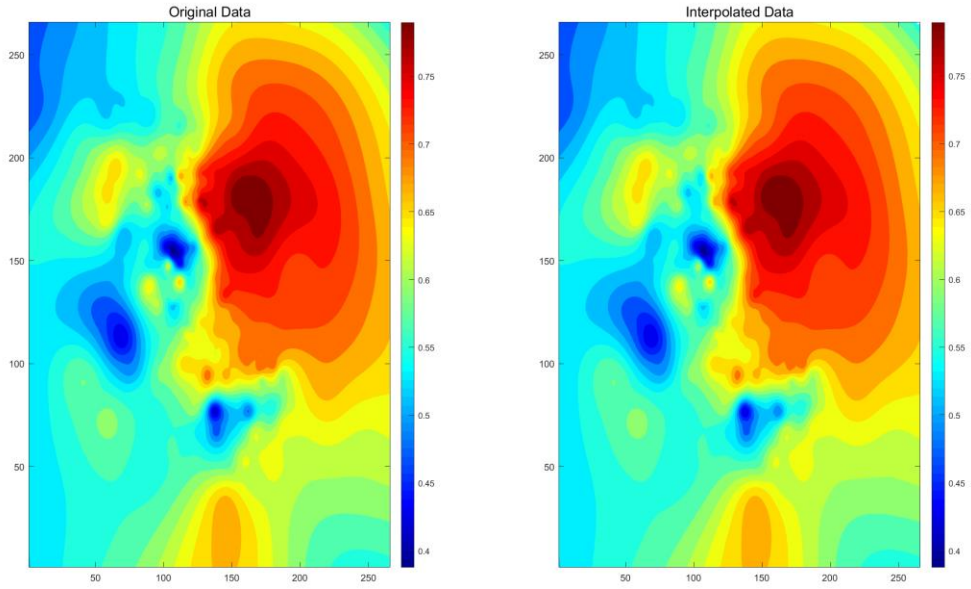

Fig. 7 Interpolation results with a sampling rate of 0.4

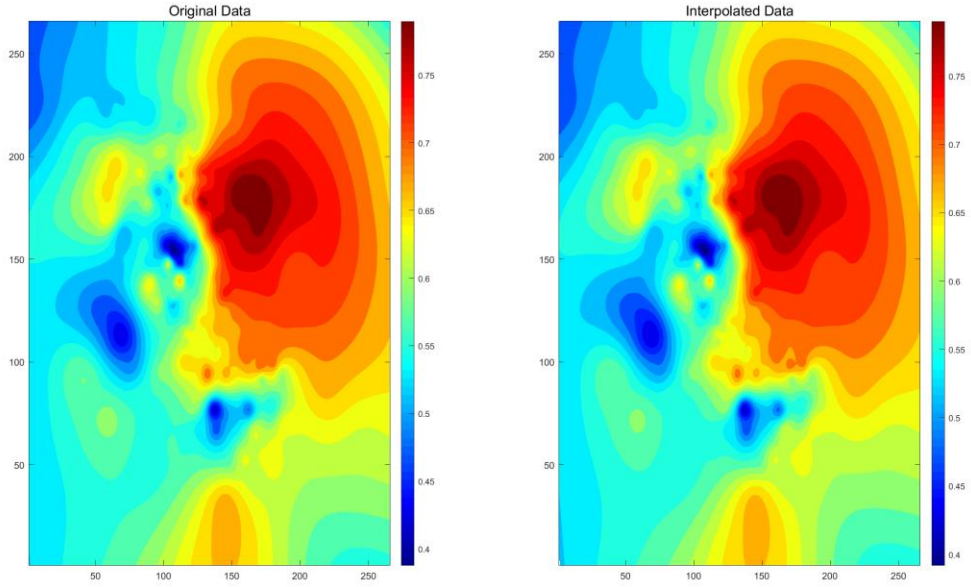

Fig. 8 Interpolation results with a sampling rate of 0.5

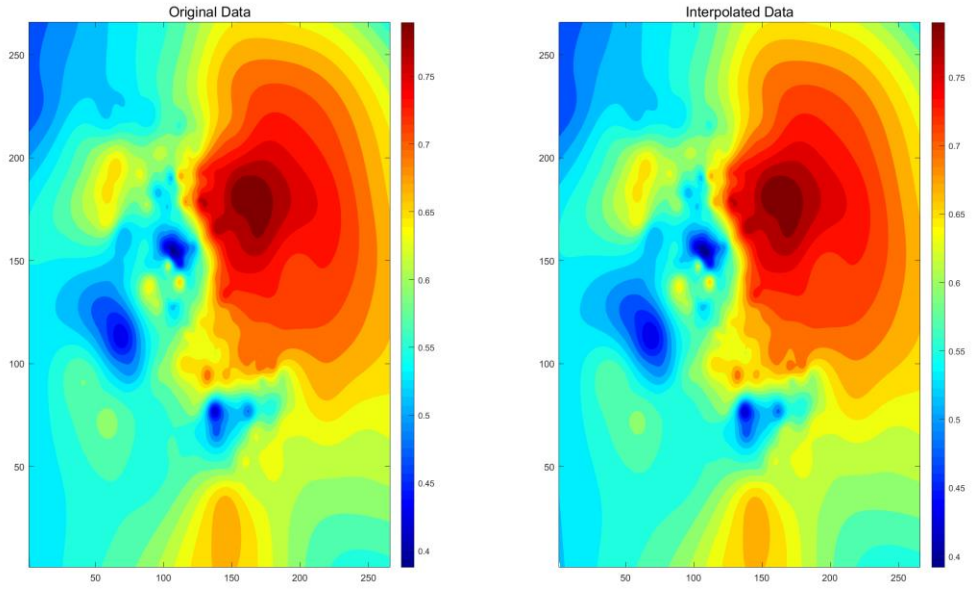

Fig. 9 Interpolation results with a sampling rate of 0.6

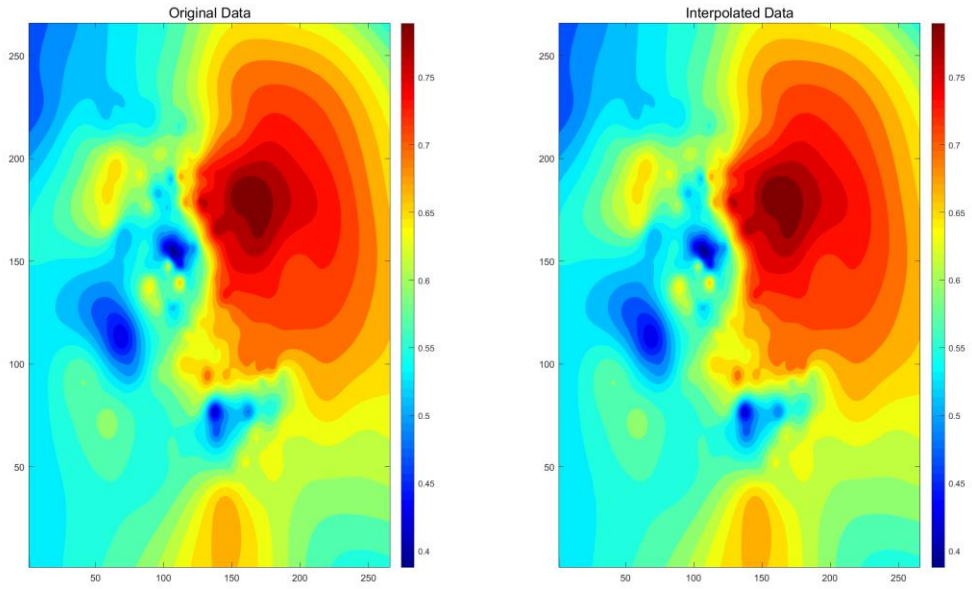

Fig. 10 Interpolation results with a sampling rate of 0.7

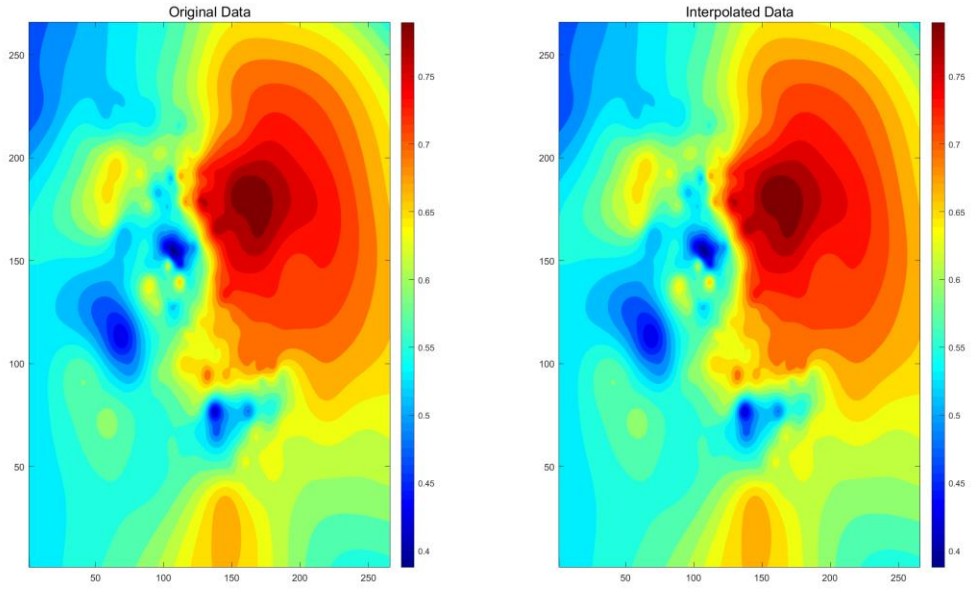

Fig. 11 Interpolation results with a sampling rate of 0.8

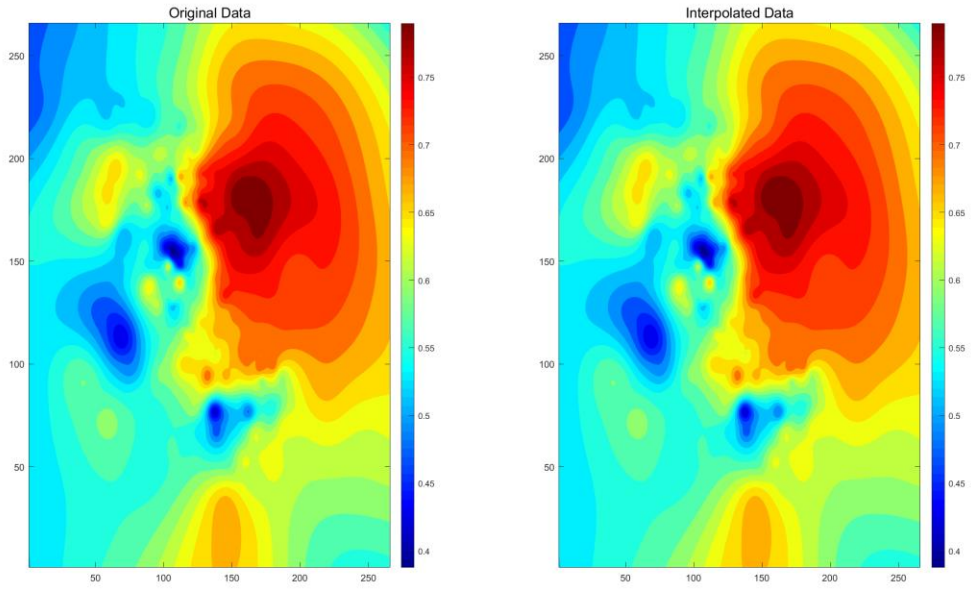

Fig. 12 Interpolation results with a sampling rate of 0.9

In order to comprehensively evaluate the performance of the interpolation algorithm, we use a number of quantitative indicators to measure the deviation between the interpolation result and the actual value. Specifically, we selected the mean absolute error (MAE), mean square error (MSE), root mean square error (RMSE) and R^2 coefficient for comprehensive evaluation. The following is the detailed calculation formula of each indicator

(1)Mean Absolute Error (MAE)

Represents the average absolute deviation between the interpolation result and the true value

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

(2)Mean Square Error (MSE)

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

Where n is the number of samples, y_i is the true value, and $\hat{y}_i$ is the corresponding interpolation result.

(3) Root mean square error (RMSE)

The square root of the mean square error, which measures the average error between the interpolated result and the true value:

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

(4) R^2（R-squared）

It is used to measure the degree of fit to the observed data, and its calculation formula is as follows:

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

The estimated errors of each sample rate are shown in the following table:

Table 2 Estimated errors at each sampling rate

Sampling rate	MAE	MSE	RMSE	R2
0.1	0.0006	0	0.0024	0.9990
0.2	0.0003	0	0.0025	0.9990
0.3	0.0002	0	0.0008	0.9999
0.4	0.0002	0	0.0014	0.9997
0.5	0.0001	0	0.0012	0.9997
0.6	0.0004	0	0.0052	0.9952
0.7	0.0001	0	0.0004	1
0.8	0.0001	0	0.0007	0.9999
0.9	0.0001	0	0.0004	1

With the increase of the sampling rate, MAE (mean absolute error) values generally show a downward trend, from 0.0006 of the low sampling rate to 0.0001 of the high sampling rate. This significant downward trend reveals that with the increase of the number of sampling points, the mean absolute deviation between the interpolation results and the actual data decreases substantially, thus significantly improving the accuracy of interpolation. RMSE (root mean square error), as the square root of MSE (mean square error), is equally effective in most cases as an indication of the accuracy of the interpolation result. Although the RMSE showed a relative peak value (0.0052) when the sampling rate was 0.6, the RMSE value remained at a

low level under other sampling rates, and continued to decline with the increase of sampling rate. These findings confirm that increasing the sampling points can effectively enhance the interpolation accuracy.

The R^2 coefficient, as an index to evaluate the accuracy of interpolation and the actual data, shows a remarkable effect. From the data observed in the table, although the R^2 value is slightly reduced to 0.9952 for a sample rate of 0.6, this result is still very close to the perfect fit standard of 1. In the remaining sample rate Settings, the R^2 values showed a tendency to be highly consistent with the ideal fit. In particular, when the sampling rate is increased to 0.7 and 0.9, the R^2 value reaches a perfect score of 1, indicating that the interpolation results are a perfect fit with the real data.

With the steady increase of sampling rate, the accuracy and fit of Kriging interpolation technology show an obvious trend of enhancement on the whole. However, it is important to note that when the sample rate is set at a specific level of 0.6, both the error evaluation index and the fit index show slight fluctuations. This fluctuation is mainly due to the random characteristics of random sampling itself.

5.2 The establishment and solution of problem 2

5.2.1 Correlation analysis between covariate and target variable based on tree model

Through continuous data segmentation strategy, decision tree algorithm can effectively identify and select the variables that are critical to the target prediction results, and this process continuously enhances the accuracy of prediction. The unique feature of this algorithm is that it can directly evaluate and quantify the importance of each feature. In the process of building the decision tree, each node splits the data set into two based on a certain feature, a core operation designed to improve the purity of the target variable in the subset through optimal segmentation, or to minimize the value of a certain loss function.

In the process of building the decision tree model [2], each node split splits the data set based on specific attributes, and its fundamental purpose is to maximize the purity gain of information, that is, significantly reduce the uncertainty of data. For each selected attribute, we assess its importance, which reflects the overall effect that the attribute contributes to reducing uncertainty throughout the decision tree structure.

$$\text{Importance}(f_j) = \sum_{t \in T_j} \left(\text{Impurity}_{\text{parent }(t)} - \text{Impurity}_{\text{left }(t)} - \text{Impurity}_{\text{right }(t)}\right)$$

Here, T_j refers to the entire set of nodes covered by the segmentation operation guided by the feature f_j. Impurity refers to indicators used to assess the purity of data, which may include different measures such as Gini index or information gain.

Consider n features labeled $f_1, f_2, \ldots, f_n$, the model is constructed by constructing m independent decision trees $T_1, T_2, \ldots, T_m$ to evaluate these features. For each feature f_j, we calculate its importance score:

$$Importance(f_j) = \frac{1}{m}\sum_{i=1}^{m}\sum_{t\in T_i}\delta(f_j, t)\cdot Impurity(t)$$

In the model, $\delta(f_j, t)$ is an indicator function that identifies whether node t adopts the feature f_j for data splitting. $\mathrm{Impurity}(t)$ has been used to measure the impurity level of node t.

By calculating the correlation scores of all features and ranking them in descending order, we select the features with the highest scores as the starting point for subsequent modeling.

The correlation analysis results of variables are shown in the figure below:

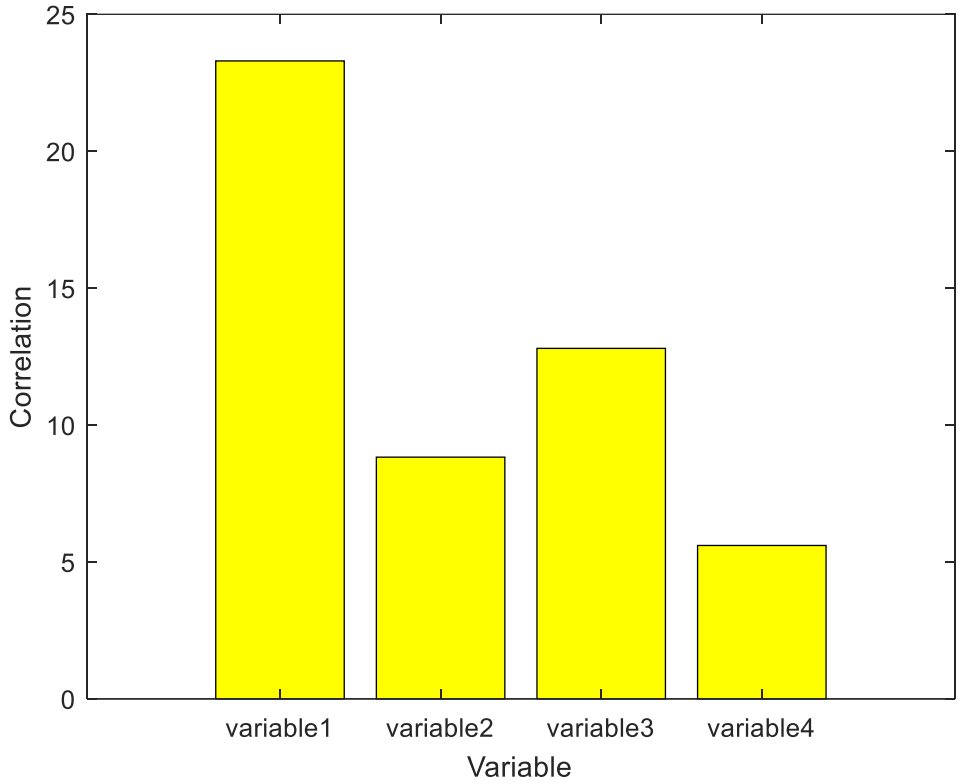

Fig. 13 Results of correlation analysis

The correlation values of each variable are shown in the following table:

Table 3 Correlation of variables

variable	Correlation to target
variable1	23.29633918
variable2	8.827019056
variable3	12.79924501
variable4	5.60173258

Observing the data in the table above, variable 1 and variable 3 show the most significant correlation. Therefore, we choose these two as co-variables.

5.3 The establishment and solution of problem 3

5.3.1 Target prediction based on combined Kriging-MLP neural network

5.3.2 Establishment of MLP neural network

Suppose the input from the I-th neuron is XI, and each input signal corresponds to a different connection weight, ωI. After weight adjustment, the sum of these input signals constitutes the total input of the neuron. The specific calculation is: $\sum_{i=1}^{n} \omega_i x_i$. On the basis of the total input, the neuron subtracts a threshold θ to complete the internal processing, resulting in the final input signal: $\sum_{i=1}^{n} \omega_i x_i - \theta$.

In a neural network architecture, each neuron relies on a unique conversion mechanism that takes an input signal as an input and generates an output signal as a result. In this model, the Sigmoid function is adopted as the function of this transformation mechanism. After the input data is processed by the function, it becomes the independent variable, and the output result of the function constitutes the output signal of the neuron. This mechanism provides a comprehensive picture of how neurons receive and process multiple input signals, which are ultimately converted into output signals. Thus, the output signal of the neuron (the result variable) is directly determined by the input signal (the independent variable).

Under the current model architecture, our core work shifts to optimizing the selection of the weight ratio ω_i and threshold θ. By precisely determining these parameters, the model will be able to achieve accurate predictions of the output signal. This involves accurately predicting the size of the output given the known input signal, i.e. predicting the actual quantity of goods based on the known information in the actual forecasting task. To this end, we adopted mean square error as a performance evaluation tool:

$$E_k = \frac{1}{2}\sum_{j=1}^{l}\left(y_j^k - y_j^k\right)^2$$

In the formula, the use of the coefficient $\frac{1}{2}$ is intended to simplify subsequent differential calculations. At the same time, the input of the JTH output neuron is defined as $\beta_j = \sum_{h=1}^{q} \omega_{hj} b_h$, and the threshold of each output neuron is set to θ_j. When the mean square error is the smallest, the corresponding weight ratio and threshold are the best parameters, and the prediction error is the smallest. Therefore, the next task is to calculate these parameters to minimize the mean square error.

Here, we adjust the parameters based on the gradient descent strategy (take the adjustment of ω as an example). After completing the k -round iteration, ω_k can be obtained. Using Taylor's formula, the error function $E(\omega)$ is expanded at ω_k, then:

Gradient descent method was used to adjust the parameters (taking ω as an example), and ω_k was obtained after K iterations. Applying Taylor expansion, the error function $E(\omega)$ is expanded at point ω_k, then:

$$E(\omega) = E(\omega_k) + E'(\omega_k)(\omega - \omega_k)$$

In the previous calculation, we omitted the remainder term R. At the completion of round $k+1$ iteration, let's say:

$$\omega = \omega_{k+1}$$

There is:

$$E(\omega_{k+1}) = E(\omega_k) + E'(\omega_k)(\omega_{k+1} - \omega_k)$$

In this study, we aim to minimize model errors through parameter optimization, so:

$$E(\omega_{k+1}) - E(\omega_k) = E'(\omega_k)(\omega_{k+1} - \omega_k) < 0$$

Here we find that we can find the special solution $\omega_{k+1} - \omega_k = -E'(\omega_k)$ to make the above formula true, whose substitution into the above formula is:

In this analysis, we observe that there is a special solution $\omega_{k+1} - \omega_k = -E'(\omega_k)$ that makes this equation valid. Substituting this particular solution into the original formula, we get:

$$E(\omega_{k+1}) - E(\omega_k) = -[E'(\omega_k)]2 < 0$$

Furthermore, the remaining term R(ω) is not considered in the aforementioned Taylor expansion, an omission based on the assumption that ω_{k+1} is close to ω_k. To reduce this bias, we introduce a small positive parameter $\eta > 0$, which helps to adjust the amplitude of change of the parameter, η is called the learning rate. Therefore, there are:

$$\omega_{k+1} - \omega_k = -\eta E'(\omega_k)$$

Therefore, there is an iterative formula:

$$\Delta\omega_{hj} = -\eta E'(\omega_{hj}) = -\eta \frac{\partial E_K}{\partial \omega_{hj}}$$

By the chain rule:

$$\frac{\partial k}{\partial \omega_{hj}} = \frac{\partial E_k}{\partial y_j^k} \cdot \frac{\partial y_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial \omega_{hj}}$$

Remember:

$$g_j = -\frac{\partial E_k}{\partial y_j^k} \cdot \frac{\partial y_j^k}{\partial \beta_j} = -(y_j^k - y_j^k)(y_j^k)'$$

Since the activation function uses the Sigmoid function, that is:

$$f(x) = \frac{1}{1 - e^{-x}}$$

There is:

$$(y_j^k)' = y_j^k(1 - y_j^k)$$

Further:

$$\frac{\partial E_k}{\partial y_j^k} \cdot \frac{\partial y_j^k}{\partial \beta_j} = (y_j^k - y_j^k) y_j^k (1 - y_j^k) = g_j$$

Defined by β_j:

$$\frac{\partial \beta_j}{\partial \omega_{hj}} = b_h$$

By combining the above formula, we get:

$$\Delta\omega_{hj} = \eta g_j b_h$$

Similar ones are:

$$\Delta\theta_j = -\eta g_j$$

Through this process, the model is continuously trained to determine the appropriate parameters and apply these parameters to actual predictions. This series of operations constitute the solution flow of the model.

5.3.3 Design of combined Kriging-MLP neural network

In the initial setup, we have multiple spatial eigenmatrices $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ and a target matrix Y. The elements in these matrices represent values of variables at specific locations in space, and the Y matrix represents the target value we want to predict.

In spatial data, because sampling points can be unevenly distributed, Kriging interpolation technology can estimate the value of the unsampled location. This method uses the existing observation points to interpolate the eigenmatrix X_i and the target matrix Y to provide an estimate for each position. The standard form of the Kriging interpolation model is:

$$\hat{X}_i(s) = \mu_i(s) + \sum^{m} \lambda_j K(s, s_j)$$

Among them:

$\hat{X}_i(s)$ represents the eigenvalue obtained by interpolation at position s.

$\mu_i(s)$ is the global mean of a spatial feature.

$K(s, s_j)$ represents the covariance function between spatial locations.

λ_j is the weight calculated based on this covariance, while s_j is the position of the known observation point.

Similarly, the object matrix Y is interpolated with Kriging to obtain the interpolated object value $\hat{X}(s)$.

After obtaining the interpolated feature matrix and target matrix, these interpolated data are used as inputs to the multi-layer perceptron (MLP). MLP is a feedforward neural network composed of input layer, hidden layer and output layer. Let the eigenmatrix after interpolation be $\hat{X}_i$ and the target matrix be $\hat{Y}$, then the MLP model can be expressed as follows:

$$\mathbf{y} = f(\widehat{\mathbf{X}}, \mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_k)$$

Among them:

$\widehat{\mathbf{X}}$ is the eigenmatrix obtained by Kriging interpolation.

$\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_k$ is the weight of each layer of the neural network.

f stands for MLP activation function and interlayer conversion operation.

Combined with Kriging interpolation and MLP neural network, the model can capture the dependency relationship of spatial data more effectively, and improve the prediction accuracy with the help of the nonlinear characteristics of neural network. This combined model is particularly suitable for spatial data prediction and can effectively solve complex problems of spatial correlation.

5.3.4 Forecast Results

According to the sampling strategy in question 1, the data are randomly and evenly

extracted, and the Kriging-MLP [3]neural network model proposed in this paper is used for prediction. Specific results are detailed as follows.

Table 4 Estimation error of combined Kriging-MLP neural network

Sampling rate	MAE	MSE	RMSE	R2
0.1	0.000915159	9.38E-06	0.003062678	0.998738965
0.2	0.00021357	9.74E-07	0.000987052	0.999835514
0.3	0.000144215	6.62E-07	0.000813824	0.999890603
0.4	0.000314317	1.22E-05	0.003487058	0.996944357
0.5	0.000143965	3.51E-07	0.00059251	0.999940283
0.6	0.000229729	3.28E-06	0.001809848	0.999566596
0.7	8.50E-05	2.33E-08	0.000152546	0.999996248
0.8	9.31E-05	4.07E-06	0.002017527	0.999309132
0.9	0.000107652	3.45E-06	0.001858372	0.999438769

The tabular data show that the error index changes with the increase of sampling rate, and the trend is not single. This indicates that a moderate increase in data points can enhance the model performance, but an excessive increase may have limited effect. Generally, MAE decreases with the increase of sampling rate, indicating the improvement of prediction accuracy. However, at a sampling rate of 0.4, MAE appears abnormally high, which may be attributed to the randomness of random sampling.

The table shows that as the sampling rate increases, the R^2 value generally increases, indicating an improved model fit. However, at a sampling rate of 0.4, the R^2 value drops, which again correlates with random sampling.

The following describes the training process with a sampling rate of 0.1 as an example.

The convergence curve is shown below:

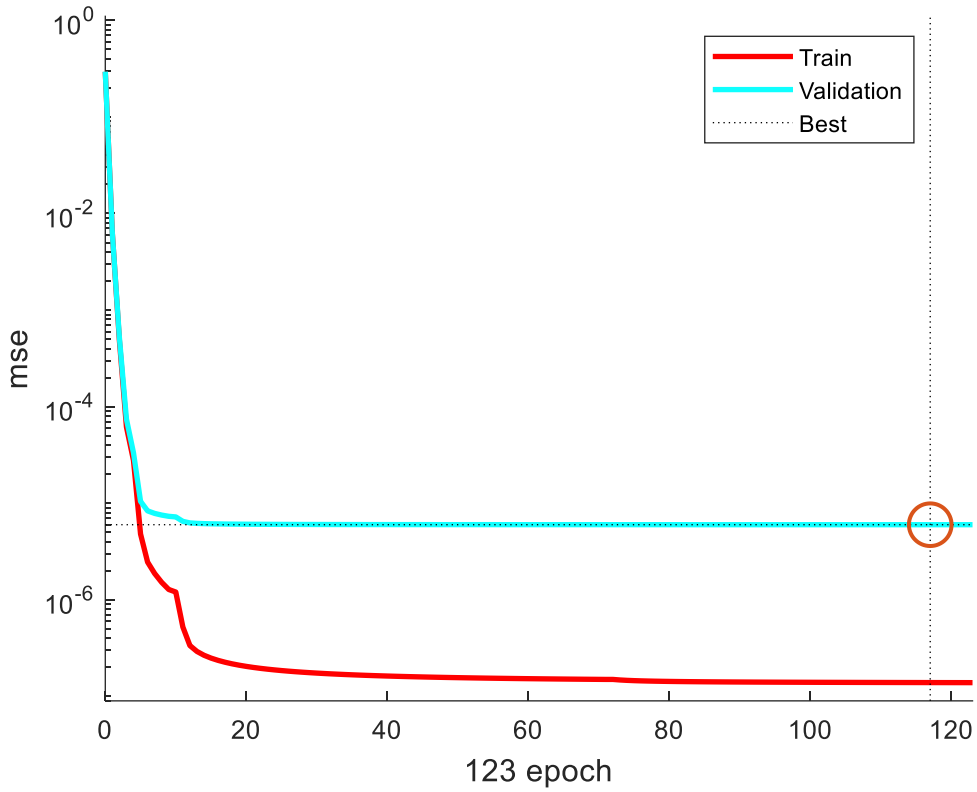

Fig. 14 Convergence curve of MLP with a sampling rate of 0.1

The figure above shows that MLP shows good convergence in predicting target variables. Its multi-layer nonlinear structure enables MLP to mine data features efficiently, adjust weights gradually, and reduce prediction errors gradually.

5.3.5 Target prediction based on the combined Kriging-Random Forest algorithm

Similar to the combined Kriging-MLP model, there are multiple spatial feature grid matrices $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ and a target grid matrix Y. The elements of these matrices correspond to variable values at different places in the space, where the Y matrix contains the prediction target value.

In the Kriging-random forest model[4], it is also necessary to implement Kriging interpolation on the eigenmatrix X_i and the target matrix Y. This interpolation method can estimate the value of the feature and the target variable in the blank space, and its mathematical formula is the same as previously mentioned:

$$\hat{X}_i(s) = \mu_i(s) + \sum^{m} \lambda_j K(s, s_j)$$

$$\hat{Y}(s) = \mu_Y(s) + \sum_{j=1}^{m} \lambda_j K(s, s_j)$$

At the position s, the Kriging estimates for the eigenvalues and the target values are denoted as $\hat{X}_i(s)$ and $\hat{Y}(s)$, respectively. In these estimates, $\mu_i(s)$ and $\mu_Y(s)$ represent the global average of the eigenvalues and the target values at their respective locations. $K(s, s_j)$

represents the covariance function describing the spatial correlation, and λ_j is the key weight coefficient derived from Kriging's estimation process.

After Kriging interpolation, the eigenmatrix $\hat{X}$ and the target matrix $\hat{Y}$ are obtained and then become the input data of Random Forest (RF) algorithm.

As an integrated learning strategy, random forests combine multiple decision trees to make predictions. Each tree is trained by randomly selecting data samples and features, and the prediction results are formed by aggregating the decisions of these trees. In the regression problem, the prediction formula for the random forest is:

$$\hat{y}_i = \frac{1}{T}\sum_{t=1}^{T} f_t(\hat{\mathbf{X}}_i)$$

$\hat{y}_i$ represents the forecast target at the point i,

T represents the total number of decision trees used in the random forest,

$f_t(\hat{\mathbf{X}}_i)$ is the prediction made by the t th tree for the input feature $\hat{\mathbf{X}}_i$.

During the training process, Random Forest uses a number of decision trees to learn from the data, each tree independently builds a model for the Kriging interpolated feature $\hat{X}$ and predicts the target value. The core of the training is to reduce the difference between the predicted result and the actual target value Y.

After the training, the model uses the feature matrix $\hat{X}$ obtained by Kriging interpolation as input to predict the target variables of the random forest model, thus generating predicted values at each point in the space.

According to the random uniform sampling method mentioned in question 1, the Kriging-random forest combined algorithm proposed in this paper is used for prediction, and the results are as follows.

Table 5 Estimation error of combined Kriging-random forest algorithm

Sampling rate	MAE	MSE	RMSE	R2
0.1	0.001064744	5.14E-06	0.002267713	0.999129105
0.2	0.000935165	4.04E-06	0.002008874	0.999303145
0.3	0.000914879	3.79E-06	0.001946207	0.999356969
0.4	0.000994067	4.66E-06	0.002158752	0.999195644
0.5	0.001033007	4.78E-06	0.002186443	0.999166646
0.6	0.001142722	1.02E-05	0.003194149	0.998240203
0.7	0.001160484	6.52E-06	0.002553349	0.99886532
0.8	0.001231527	6.16E-06	0.002481299	0.998965814
0.9	0.001127213	4.74E-06	0.002178036	0.999184107

5.3.6 Target prediction based on nonlinear regression algorithm

In this study, we can construct a regression model involving two co-variables and the target variable. After the values of covariables are determined by interpolation, the regression model can be used to calculate the corresponding target values. This relationship can be expressed as:

$$y_i = f\left(x_1^i, x_2^i, \cdots, x_j^i, \theta_1, \theta_2, \cdots, \theta_p\right) + \sigma_i \varepsilon \ (i = 1,2, \cdots, n)$$

Where y represents the true observed value, i identifies the group i sample. Function $f\left(x_1, x_2, \cdots, x_j, \theta_1, \theta_2, \cdots, \theta_p\right)$ is a multivariate nonlinear expression, on behalf of the deterministic part. $X_1, x_2, \cdots, x_j$ are independent variables, while $\theta_1, \theta_2, \cdots, \theta_p$ are unknown parameters of this function. The random part $\sigma_i \varepsilon$, ε is a random variable of the normal distribution N (0,1), and σ_i is the random standard deviation of group i data.

In current practice, the selection of nonlinear regression models is often based on empirical intuition or experimental exploration. While empirical selection may carry a higher risk of error, experimental selection, although time-consuming, can provide a more accurate model selection based on experimental data. Based on this consideration, the nonlinear regression model that is most suitable for the studied relationship is determined in detail through experimental means. The specific selection process is detailed as follows:

$$z = p1 + p2 * v1 + p3 * v3 + p4 * v3^2$$

For multiple parameters in a regression model, we usually rely on gradient descent or genetic algorithms for optimization. However, gradient descent may fall into local optimality, and genetic algorithms may encounter the same problem at a later stage. In order to overcome this limitation, differential evolution algorithm is used in this study, which can find the global optimal solution more effectively to optimize the parameters $\theta_1, \theta_2, \cdots, \theta_n$.

Here's how to solve it:

Population initialization

Set the population size to M, that is, 100 individuals, which are generated in a random uniform manner within the solution space, each individual consisting of a vector containing n elements.

$$X_i(0) = \left(x_{i,1}(0), x_{i,2}(0), x_{i,3}(0), \ldots, x_{i,n}(0)\right)$$

$$i = 1,2,3, \ldots, M$$

Among them,

$$x_{i,j}(0) = L_{j\_\min} + \text{rand}(0,1)\left(L_{j\_\max} - L_{j\_\min}\right)$$

$$i = 1,2,3, \ldots, M$$

$$j = 1,2,3, \ldots, n$$

Mutation

In the g iteration stage, we randomly select three members from the population, as $X_{p1}(g), X_{p2}(g), X_{p3}(g)$, and $1 \neq p2 \neq p3 \neq i$, and then construct variation vectors based on these members:

$$H_i(g) = X_{p1}(g) + F \cdot \left(X_{p2}(g) - X_{p3}(g)\right)$$

Where $\Delta_{p2,p3}(g) = X_{p2}(g) - X_{p3}(g)$ is the difference vector and F is the scaling factor.

The three individuals randomly selected in the mutation operator are sorted from best to worst, and Xb,Xm,Xw, corresponding fitness fb,fm,fw are changed to:

$$V_i = X_b + F_i(X_m - X_w)$$

In addition, the value of the constant F adaptively adjusts according to the two individuals building the difference vector.

$$F_i = F_l + (F_u - F_l)\frac{f_m - f_b}{f_w - f_b}, F_l = 0.1, F_u = 0.9$$

The mutation strategy is:

$$DE/\text{rand} \quad /1 : V_i(g) = X_{p1}(g) + F\left(X_{p2}(g) - X_{p3}(g)\right)$$

$$DE/\text{best} \quad /1 : V_i(g) = X_{\text{best}}(g) + F\left(X_{p1}(g) - X_{p2}(g)\right)$$

$$DE/\text{current to best} \quad /1 \ : V_i(g) = X_i(g) + F\left(X_{\text{best}}(g) - X_i(g)\right) + F\left(X_{p1}(g) - X_{p2}(g)\right)$$

$$DE/\text{best} \quad /2 : V_i(g) = X_{\text{best}}(g) + F\left(X_{p1}(g) - X_{p2}(g)\right) + F\left(X_{p3}(g) - X_{p4}(g)\right)$$

$$DE/\text{rand} \quad /2 : V_i(g) = X_{p1}(g) + F\left(X_{p2}(g) - X_{p3}(g)\right) + F\left(X_{p4}(g) - X_{p5}(g)\right)$$

Intersect

$$v_{i,j} = \begin{cases} h_{i,j}(g), \text{rand}(0,1) \le cr \\ x_{i,j}(g), \text{ else} \end{cases}$$

Where $cr \in [0,1]$ is the crossing probability, cr=0.7.

Select

$$X_i(g+1) = \begin{cases} V_i(g), f\left(V_i(g)\right) < f\left(X_i(g)\right) \\ X_i(g), \text{ else} \end{cases}$$

The model parameters are set as follows:

argument	value
Number of population	100
Crossover rate	0.7
Variation rate	0.85
Allowable error of convergence	10^{-10}
Convergence allows the number of error judgments	1000
The maximum number of iterations allowed	30000

With the completion of the iteration, we successfully determined the optimal $p_1, p_2, \cdots, p_p$ combination.

After these iterative optimization steps, we get the optimal solution of the parameters. Specific results are as follows:

The sample rate is 0.1	
p1	-9.12422466469207
p2	6.78632633354305E-5
p3	62.6782496883493
p4	-101.932846759216

The sample rate is 0.2	
p1	-9. 26662354057229
p2	6.78072394557404E-5
p3	63.5638930156787
p4	-103.309587063647
The sample rate is 0.3	
p1	-9.29440879097552
p2	6. 74560438997271E-5
p3	63.7519349313116
p4	-103.614181119165
The sample rate is 0.4	
p1	-9.33301695575947
p2	6.74251893877463E-5
p3	63.9555506358299
p4	-103.871237595143
The sample rate is 0.5	
p1	-9.06015680794373
p2	6.75272072939776E-5
p3	62.2422719800425
p4	-90.5506303622446
The sample rate is 0.6	
p1	-7. 99647245985821
p2	6. 89384997474359E-5
p3	55.4960858404405
p4	-101.186965058368
The sample rate is 0.7	
p1	-9.25247002793595
p2	6.75879475419862E-5
p3	63.471237874186 3
p4	-103.148378509308
The sample rate is 0.8	
p1	-9.34392124957989
p2	6.79151119751503E-5
p3	64.0594287492016
p4	-104.098326544209
The sample rate is 0.9	
p1	-9.25467947621868
p2	6.7882250570275E-5
p3	63.499742498198
p4	-103.216103514737

The errors are shown in the following table.

Table 6 Estimation error of nonlinear regression

Sampling rate	MAE	MSE	RMSE	R2
0.1	0.03775441	0.00230566	0.048017293	0.677170703
0.2	0.038054024	0.002319881	0.048165146	0.687759263
0.3	0.037105322	0.001598422	0.039980268	0.602407589
0.4	0.040371886	0.002524414	0.050243543	0.674966212
0.5	0.040784489	0.002001258	0.044735426	0.561605809
0.6	0.043011678	0.002749791	0.052438451	0.641382672
0.7	0.04703728	0.002862928	0.053506337	0.57088066
0.8	0.044221894	0.002879645	0.053662326	0.626212154
0.9	0.042455376	0.002201498	0.046920122	0.54471578

5.3.7 Algorithm Performance Comparison

The target variables are estimated by the above three algorithms, and the performance of the algorithms under each sampling rate is presented as the average value. The corresponding box diagram is as follows.

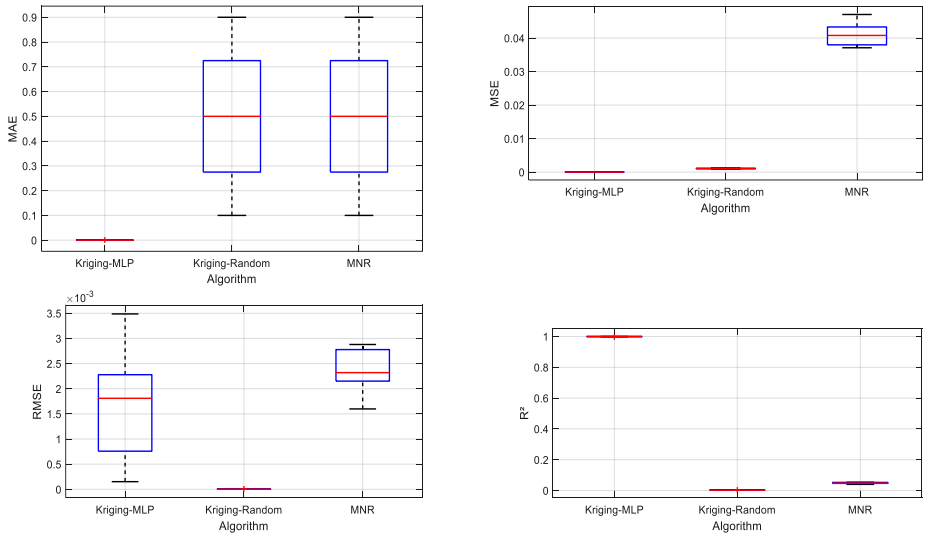

Fig. 15 Comparison of algorithm performance

The mean error is shown in the table below.

Table 7 Table of mean errors

	MAE	MSE	RMSE	R2
Kriging -MLP	0.00024963	3.81667E-06	0.001642379	0.999295608
Kriging - Random Forest	0.00106709	5.55883E-06	0.002330536	0.999045217
Nonlinear regression	0.041199596	0.002382611	0.048629879	0.620788982

Analysis of mean absolute error (MAE) results shows that the Kriging-MLP model has

the lowest MAE (0.00024963), indicating the smallest prediction error and high prediction accuracy. The Kriging-random forest model has a MAE of 0.00106709, which is slightly higher than the MLP model. The MAE of nonlinear regression model is the highest, reaching 0.041199596, and the prediction error is significant.

Examine the mean square error (MSE) and the root mean square error (RMSE), which reveal the square root of the model's prediction error and its sum of squares. The results show that both MSE and RMSE of Kriging-MLP model are the lowest, which are 3.81667E-06 and 0.001642379 respectively, which further confirms its superiority in prediction ability. The Kriging-random forest model has slightly higher MSE and RMSE, while the nonlinear regression model has the highest of these two indicators, indicating that its prediction error is more significant on these two measures.

In terms of coefficient of determination (R^2), both Kriging-MLP model and Kriging-random forest model have R^2 values close to 1, which are 0.999295608 and 0.999045217, respectively, showing excellent data fitting ability, and the predicted results are highly consistent with the real data. In contrast, the R^2 value of the nonlinear regression model is only 0.620788982, indicating that its fitting effect is poor, and there are obvious differences between the predicted results and the actual data.

Overall, the Kriging-MLP model showed superior predictive power, followed closely by the Kriging-random Forest model, while the nonlinear regression model showed less predictive power.

5.4 The establishment and solution of problem 4

5.4.1 target prediction based on combined Kriging-MLP neural network

In problem 3, this paper selects Kriging-MLP neural network as the optimal model, and predicts the target in Annex II based on this combined neural network.

The forecast results are shown in the figure below.

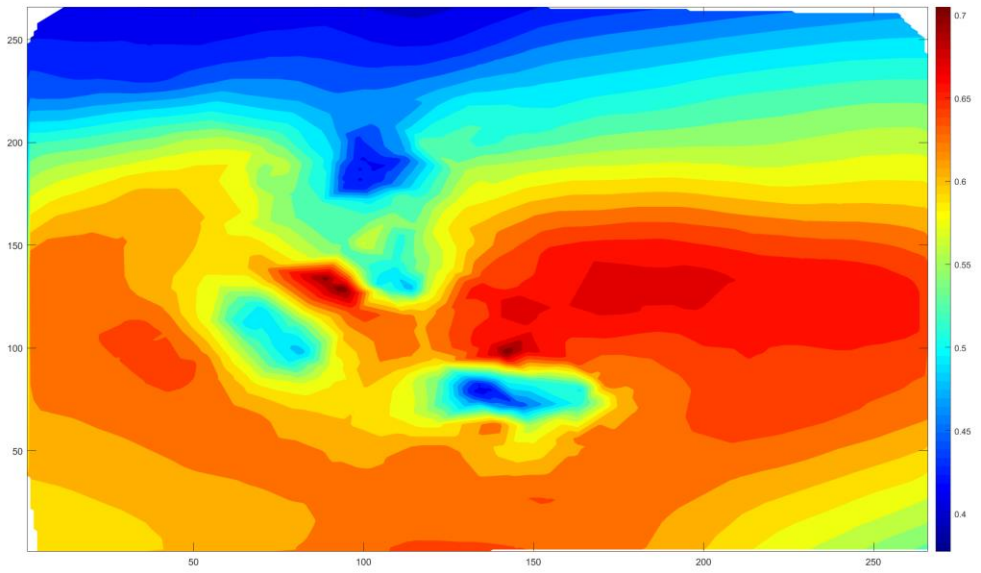

Fig. 16 Prediction results

To facilitate the observation of the prediction effect, the interpolation surface and the original data points are placed together in the same plot, as shown below.

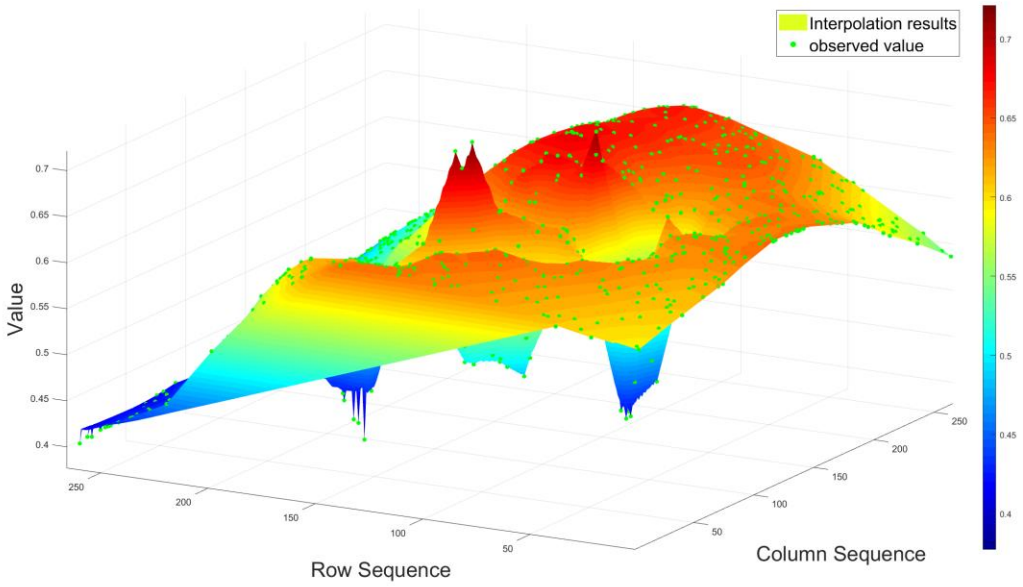

Fig. 17. Interpolation result diagram

The surface diagram shows that the proposed combined Kriging-MLP neural network model performs well in prediction. The prediction surface corresponds closely to the data points, accurately reproducing the overall trend and local details of the data, and demonstrating its powerful ability to handle complex dependencies and nonlinear relationships. This superior prediction ability demonstrates the effectiveness of combining Kriging with MLP and contributes an efficient and accurate new method to the field of prediction.

6. References

[1] Rao Zhijie, Zhang Qihui, Hu Xiaoqin, et al. Analysis of high temperature and dry weather in 2022 flood season in Nanchong City based on Kriging interpolation method [J/OL]. Anhui agricultural science, 1-6 [2024-11-16]. http://kns.cnki.net/kcms/detail/34.1076.S.20240924.1432.010.html.

[2] Gan Ying, Xiao Zhanhui, Liang Jiming, et al. Binary recursive segmentation detection of power communication network full link anomaly under data missing constraint [J/OL]. Automation technology and application of 1-5 [2024-11-16]. http://kns.cnki.net/kcms/detail/23.1474.TP.20241115.1545.006.html.

[3] Dou Miao, Hou Xiangning. Application of PCA-MLP neural network model in runoff prediction of Ningxia section of the Yellow River [J]. Water Conservancy Information Technology,2024,(04):49-53.DOI:10.19364/j.1674-9405.2024.04.009

[4] Li Yue, Yue Chunfang, Chen Dachun. Reference crop evapotranspiration prediction method based on combination of random forest feature selection and POA-LSTM [J/OL]. Water saving irrigation, 1-16 [2024-11-16]. http://kns.cnki.net/kcms/detail/42.1420.TV.20241114.0953.004.html.

7.Appendix

Convert txt document to mat format of grid matrix

```
clc;                % Clear the command window
clear;              % Clear all variables from the workspace
data = xlsread('F1_target_variable.xlsx'); % Read data from the Excel file
new_set = []; % Initialize an empty array for the new dataset
temp_set = []; % Initialize an empty array to hold temporary data
num = 1;            % Initialize a counter for the number of data blocks
% Example of storing each block column-wise
for i = 1:size(data, 1) % Loop through each row of the data
    if isnan(data(i, 1)) % Check if the first column of the current row is NaN (empty)
        temp_set = [temp_set ; data(i, 2:end)']; % Append the rest of the row to temp_set
    else % If the first column is not empty
        temp_set = [temp_set ; data(i, 1)]; % Append the first column of the current row
to temp_set
        new_set(:, num) = temp_set; % Store temp_set as a new column in new_set
        num = num + 1; % Increment the block counter
        temp_set = []; % Reset temp_set for the next block
    end
end
```

Map making

```
clc;                % Clear the command window to remove previous commands
clear;              % Clear all variables from the workspace
close all;      % Close all open figure windows
load("F2_collaborative_variable1.mat"); % Load data from the .mat file for variable 1
subplot(2,2,1); % Create a subplot in a 2x2 grid and target the first position
surf = surf(new_set); % Create a 3D surface plot using the data
colormap('hot'); % Apply a hot color map to visualize the data intensity
% Other optional beautification steps can be added here
title('F2 collaborative variable1'); % Set the title of the subplot
xlabel('Column Sequence'); % Label the x-axis as 'Column Sequence'
ylabel('Row Sequence'); % Label the y-axis as 'Row Sequence'
zlabel('Value'); % Label the z-axis as 'Value'
axis tight; % Automatically adjust the axis limits to tightly fit the data
colorbar; % Add a color bar to interpret the data intensity
shading interp; % Remove the grid lines for a smoother appearance
%% % Comment separator for readability, not executed by MATLAB
```

```
clear;              % Clear all variables from the workspace
load("F2_collaborative_variable2.mat"); % Load data from the .mat file for variable 2
subplot(2,2,2); % Target the second position in the 2x2 grid
surf = surf(new_set); % Create a 3D surface plot using the data
colormap('hot'); % Apply a hot color map to visualize the data intensity
% Other optional beautification steps can be added here
title('F2 collaborative variable2'); % Set the title of the subplot
xlabel('Column Sequence'); % Label the x-axis
ylabel('Row Sequence'); % Label the y-axis
zlabel('Value'); % Label the z-axis
axis tight; % Adjust the axis limits
colorbar; % Add a color bar
shading interp; % Remove the grid lines
%% % Comment separator
clear;              % Clear all variables from the workspace
load("F2_collaborative_variable3.mat"); % Load data from the .mat file for variable 3
subplot(2,2,3); % Target the third position in the 2x2 grid
surf = surf(new_set); % Create a 3D surface plot using the data
colormap('hot'); % Apply a hot color map
% Other optional beautification steps
title('F2 collaborative variable3'); % Set the title
xlabel('Column Sequence');
ylabel('Row Sequence');
zlabel('Value');
axis tight;
colorbar;
shading interp;
%% % Comment separator
clear;              % Clear all variables from the workspace
load("F2_collaborative_variable4.mat"); % Load data from the .mat file for variable 4
subplot(2,2,4); % Target the fourth position in the 2x2 grid
surf = surf(new_set); % Create a 3D surface plot using the data
colormap('hot'); % Apply a hot color map
% Other optional beautification steps
title('F2 collaborative variable4'); % Set the title
xlabel('Column Sequence');
ylabel('Row Sequence');
zlabel('Value');
axis tight;
colorbar;
shading interp;
```

Problem one

The original matrix is sampled uniformly at random

```
clc; % Clear the command window to remove any previous commands
clear; % Clear all variables from the workspace
close all; % Close all open figure windows
load("F1_target_variable.mat"); % Load the data from the ".mat" file for the target variable
data = new_set; % Assign the loaded data to the variable 'data'
% Calculate the size of the matrix
[rows, cols] = size(data); % Get the number of rows and columns of the matrix
% Compute the total number of elements in the matrix
total_elements = rows * cols; % Multiply the number of rows by the number of columns
% Calculate the number of elements to set as NaN
sample_rate = 0.9; % Set the sample rate
filename = 'data_0.9.mat'; % Define the filename for the modified data
num_to_set_nan = round((1 - sample_rate) * total_elements); % Calculate the number of elements to be set as NaN
% Save the modified data to a new file
save(filename, 'new_set'); % Save the 'new_set' matrix to the specified file
%% Plotting
surf = surf(data); % Create a 3D surface plot of the data
colormap('gray'); % Apply a grayscale colormap to the plot
% Other optional beautification steps can be added here
title('Sampling rate 0.9', 'FontSize', 22); % Set the title of the plot with a font size of 22
xlabel('Column Sequence', 'FontSize', 22); % Label the x-axis as 'Column Sequence' with a font size of 22
ylabel('Row Sequence', 'FontSize', 22); % Label the y-axis as 'Row Sequence' with a font size of 22
zlabel('Value', 'FontSize', 22); % Label the z-axis as 'Value' with a font size of 22
axis tight; % Automatically adjust the axis limits to tightly fit the data
colorbar; % Add a color bar to the plot
shading interp; % Remove the grid lines for a smoother appearance
hold on; % Keep the current plot for adding new layers
new_rows = size(new_set, 1); % Get the number of rows of the new set
new_cols = size(new_set, 2); % Get the number of columns of the new set
% Initialize arrays to store the non-NaN points data
x_data = []; % Array to store x coordinates (column indices)
y_data = []; % Array to store y coordinates (row indices)
z_data = []; % Array to store z coordinates (values)
% Loop through the new_set matrix to find all non-NaN points
```

```
    for x = 1:new_rows
        for y = 1:new_cols
            if ~isnan(new_set(x, y))
                x_data = [x_data, y]; % Store the row index as x coordinate (switched x and y axes)
                y_data = [y_data, x]; % Store the column index as y coordinate
                z_data = [z_data, new_set(x, y)]; % Store the corresponding value as z coordinate
            end
        end
    end
    scatter3(x_data, y_data, z_data, 5, 'filled', 'g'); % Plot a scatter plot of the non-NaN points with green filled points
    legend('original data','resample', 'FontSize', 22); % Add a legend to the plot with a font size of 22
```

Kriging interpolation

```
    clc; % Clear the command window to remove any previous commands
    clear; % Clear all variables from the workspace
    close all; % Close all open figure windows
    %% Below is the code to plot the original data
    load("F1_target_variable.mat"); % Load the data from the ".mat" file for the target variable
    subplot(1,2,1); % Create a subplot in a 1x2 grid, first panel
    O_data = new_set; % Assign the loaded data to the variable 'O_data'
    contourf(O_data, 20, 'LineColor', 'none'); % Create a filled contour plot with 20 levels, no line color
    % Set the color map and color bar for the filled plot
    colormap(hot); % Use the hot color map for the plot
    colorbar; % Add a color bar to the plot
    title('Original Data', 'FontSize', 15); % Set the title of the plot with a font size of 15
    hold off; % Release the current plot for further customization
    load("data_0.9.mat"); % Load the modified data from the ".mat" file
    % Get the size of the matrix
    [m, n] = size(new_set); % Determine the number of rows and columns of the matrix
    % Create an index for all valid values
    [rows, cols] = find(~isnan(new_set)); % Find the positions where the values are not NaN
    % Get the non-NaN values
    valid_values = new_set(sub2ind([m, n], rows, cols)); % Extract the non-NaN values using linear indexing
```

```
% Treat the row and column indices of the valid values as scatter plot coordinates
x = cols; % x coordinates are the column indices
y = rows; % y coordinates are the row indices
z = valid_values; % z values are the corresponding valid values
% Create an interpolation object using scatteredInterpolant, selecting 'linear' interpolation
interp = scatteredInterpolant(x, y, z, 'linear', 'none'); % 'none' for NaN values outside the interpolation range
% Find the positions where the values are NaN
[rows_nan, cols_nan] = find(isnan(new_set)); % Determine the row and column indices of NaN values
new_set_nan = new_set; % Create a copy of the original set to fill in the interpolation results
% Use the interpolation object to fill in the NaN values
for i = 1:length(rows_nan)
    new_set_nan(rows_nan(i), cols_nan(i)) = interp(cols_nan(i), rows_nan(i));
end
%% Below is the code to plot the interpolated data
subplot(1,2,2); % Create a subplot in a 1x2 grid, second panel
contourf(new_set_nan, 20, 'LineColor', 'none'); % Create a filled contour plot with 20 levels, no line color
% Set the color map and color bar for the filled plot
colormap(hot); % Use the hot color map for the plot
colorbar; % Add a color bar to the plot
title('Interpolated Data', 'FontSize', 15); % Set the title of the plot with a font size of 15
hold off; % Release the current plot for further customization
%% Below is the code to evaluate the accuracy
true_values = []; % Initialize an array to store the true values
predicted_values = []; % Initialize an array to store the predicted values
for i=1:size(new_set,1) % Loop over the rows of the matrix
    for j=1:size(new_set,2) % Loop over the columns of the matrix
        if isnan(new_set(i,j)) % If the value at the current position is NaN
            true_values = [true_values; O_data(i,j)]; % Append the true value from the original data
            if isnan(new_set_nan(i,j)) % If the interpolated value is also NaN
                predicted_values = [predicted_values; nanmean(new_set(:))); % Use the mean of the entire set
            else
                predicted_values = [predicted_values; new_set_nan(i,j)]; % Append the interpolated value
            end
        end
    end
```

```
end
% Calculate the MAE (Mean Absolute Error)
MAE = mean(abs(true_values - predicted_values)); % Mean absolute error calculation
% Calculate the MSE (Mean Squared Error)
MSE = mean((true_values - predicted_values).^2); % Mean squared error calculation
% Calculate the RMSE (Root Mean Squared Error)
RMSE = sqrt(MSE); % Root mean squared error calculation
% Calculate the R² (Coefficient of Determination)
SS_tot = sum((true_values - mean(true_values)).^2); % Total sum of squares
SS_res =
```

Problem two

Feature correlation analysis

```matlab
clc; % Clear the command window to remove any previous commands
clear; % Clear all variables from the workspace
close all; % Close all open figure windows
% Load the collaborative variables from their respective .mat files
v1 = load("F1_collaborative_variable1.mat").new_set;
v2 = load("F1_collaborative_variable2.mat").new_set;
v3 = load("F1_collaborative_variable3.mat").new_set;
v4 = load("F1_collaborative_variable4.mat").new_set;
% Load the target variable from its .mat file
target = load("F1_target_variable.mat").new_set;
% Initialize an empty array to store the combined data
data = [];
% Loop over the rows and columns of the collaborative variables
for i = 1:size(v1, 1)
    for j = 1:size(v1, 2)
        % Append the collaborative variables and the target variable as a row to the data
array
        data = [data ; [v1(i, j), v2(i, j), v3(i, j), v4(i, j), target(i, j)]];
    end
end
% Shuffle the data to create a random permutation
TE = randperm(size(data, 1));
% Split the data into training set inputs (PN) and training set outputs (TN)
PN = data(TE, 1:end-1)'; % Training set inputs
TN = data(TE, end)'; % Training set outputs
% Normalize the input data to the range [0, 1]
```

```
[pn, ps_input] = mapminmax(PN, 0, 1); % Normalize inputs
pn = pn'; % Transpose the normalized inputs back to the original shape
% Normalize the target data to the range [0, 1]
[tn, ps_output] = mapminmax(TN, 0, 1); % Normalize targets
tn = tn'; % Transpose the normalized targets back to the original shape
% Set the parameters for the TreeBagger model
trees = 250; % Number of decision trees
leaf = 2; % Minimum number of leaves
OOBPrediction = 'on'; % Enable out-of-bag prediction
OOBPredictorImportance = 'on'; % Enable calculation of predictor importance
Method = 'regression'; % Specify the method as regression
% Train the TreeBagger model
net = TreeBagger(trees, pn, tn, 'OOBPredictorImportance', OOBPredictorImportance, ...
'Method', Method, 'OOBPrediction', OOBPrediction, 'minleaf', leaf);
% Extract the out-of-bag prediction error for each predictor
importance = net.OOBPermutedPredictorDeltaError;
% Plot the predictor importance as a bar chart
figure;
bar(importance);
% Set custom x-axis tick labels for the variables
set(gca, 'XTickLabel', {'variable1', 'variable2', 'variable3', 'variable4'});
% Set the axis labels
xlabel('Variable');
ylabel('Correlation');
% Beautify the graph
% Set the width of the bars (optional)
barWidth = 0.8; % Width value can be adjusted as needed
set(gca, 'BarWidth', barWidth);
```

Question three
Random uniform extraction

```
clc; % Clear the command window to remove any previous commands
clear; % Clear all variables from the workspace
close all; % Close all open figure windows

% Load the target variable from its .mat file
load("F1_target_variable.mat");

% Create a new_set object for the data
```

```
data = new_set;

% Get the size of the matrix
[rows, cols] = size(data);

% Calculate the total number of elements in the matrix
total_elements = rows * cols;

% Define the sample rate for the number of elements to be set as NaN
sample_rate = 0.5;
filename = 'data_0.5.mat'; % Set the filename for the modified data file

% Calculate the number of elements to be set as NaN
num_to_set_nan = round((1 - sample_rate) * total_elements);

% Keep the matrix shape intact by removing elements as evenly as possible
% Calculate the number of elements to remove per row (using floor to ensure it doesn't exceed the matrix boundary)
elements_per_row = floor(num_to_set_nan / rows);

% Calculate the remaining elements after distributing them evenly across rows
remaining_elements = num_to_set_nan - elements_per_row * rows;

% Initialize a logical matrix to mark which elements should be set as NaN
nan_mask = true(rows, cols);

% Uniformly set NaN in the matrix
for i = 1:rows
    current_col_index = 1; % Start from the first column
    for j = 1:(cols - elements_per_row) % Mark non-NaN values
        % Ensure we don't exceed the number of columns
        if current_col_index <= cols
            nan_mask(i, current_col_index) = false;
            if rand < 0.5
                current_col_index = current_col_index + floor(cols / (cols - elements_per_row)); % Skip columns uniformly
            else
                current_col_index = current_col_index + ceil(cols / (cols - elements_per_row)); % Skip columns uniformly
            end
            if current_col_index > cols
                break;
```

```
                    end
                end
            end
            % If we haven't set enough NaNs due to column limits, continue in subsequent iterations
        end

        % Randomly reverse the order of rows if a random flip is desired
        if rand < 0.5
            nan_mask(i, :) = fliplr(nan_mask(i, :)); % Reverse the order of the row vector, but flip is more appropriate
        end

        % If the current_col_index exceeds the number of columns and there are remaining NaNs to set,
        % we can reset it in the next row (but this typically won't happen in this example due to mod)

        % Apply the nan_mask to set the corresponding elements to NaN
        v1 = load("F1_collaborative_variable1.mat").new_set;
        v2 = load("F1_collaborative_variable2.mat").new_set;
        v3 = load("F1_collaborative_variable3.mat").new_set;
        v4 = load("F1_collaborative_variable4.mat").new_set;
        target = load("F1_target_variable.mat").new_set;

        % Set NaN in the collaborative variables and target variable based on the nan_mask
        v1(nan_mask) = NaN;
        v2(nan_mask) = NaN;
        v3(nan_mask) = NaN;
        v4(nan_mask) = NaN;
        target(nan_mask) = NaN;

        % Save the modified new_set matrices to the specified file
        save(filename, 'v1', 'v2', 'v3', 'v4', 'target');
```

Kriging -MLP

```
clc; % Clear the command window to remove any previous commands
clear; % Clear all variables from the workspace
close all; % Close all open figure windows
```

```
% Load the original target variable from its .mat file
O_target = load("F1_target_variable.mat").new_set;

% Load the sampled values from the 'data_0.9.mat' file
v1 = load("data_0.9.mat").v1;
v3 = load("data_0.9.mat").v3;
target = load("data_0.9.mat").target;

%% Create a mask (record where missing values are)
mask = zeros(size(v1, 1), size(v1, 2));
mask(isnan(v1)) = 1; % Set elements where v1 is NaN to 1 in the mask

%% Perform Kriging interpolation
[v1] = myKriging(v1);
[v3] = myKriging(v3);
[target] = myKriging(target);

%% Organize the data into a dataset. The first five columns are the Kriging interpolated values, and the last column is the true value
data = [];
for i = 1:size(v1, 1)
    for j = 1:size(v1, 2)
        if mask(i, j) == 1 % Check if the mask indicates a missing value
            data = [data ; [v1(i, j), v3(i, j), target(i, j), O_target(i, j)]]; % Append the values to the data array
        end
    end
end

%% Split the dataset
[MAE, MSE, RMSE, R2, net, mu, sigma] = myBP(data); % Assuming myBP is a function that performs some kind of analysis on the data

%% Randomly permute the data indices
R = randperm(size(data, 1));

%% Extract evaluation metrics
E = [MAE, MSE, RMSE, R2]; % Assuming these are the evaluation metrics from the myBP function

%% Save the results to a specified file
save('net', 'net', 'mu', 'sigma'); % Save the results as new_set matrices to the specified file
```

Kriging - Random Forest

```
clc; % Clear the command window to remove any previous commands
clear; % Clear all variables from the workspace
close all; % Close all open figure windows

% Load the original target variable from its .mat file
O_target = load("F1_target_variable.mat").new_set;

% Load the sampled values from the 'data_0.1.mat' file
v1 = load("data_0.1.mat").v1;
v3 = load("data_0.1.mat").v3;
target = load("data_0.1.mat").target;

%% Create a mask to identify missing values (records where data is NaN)
mask = zeros(size(v1, 1), size(v1, 2));
mask(isnan(v1)) = 1; % Set elements of the mask to 1 where v1 is NaN

%% Perform Kriging interpolation on the sampled values
[v1] = myKriging(v1);
[v3] = myKriging(v3);
[target] = myKriging(target);

%% Assemble the data into a dataset. The first five columns are the interpolated values, and the last column is the true value
data = [];
for i = 1:size(v1, 1) % Loop through each row
    for j = 1:size(v1, 2) % Loop through each column
        if mask(i, j) == 1 % Check if the mask indicates a missing value
            data = [data ; [v1(i, j), v3(i, j), target(i, j), O_target(i, j)]]; % Append the values to the data array
        end
    end
end

%% Call the MLP neural network function
[MAE, MSE, RMSE, R2] = myRF(data); % Assuming myRF is a function that trains an MLP and evaluates it

%% Combine the evaluation metrics into a single vector
```

```
E = [MAE, MSE, RMSE, R2]; % Create a vector containing the evaluation metrics
```

Obtain nonlinear regression data

```
clc; % Clear the command window to remove any previous commands
clear; % Clear all variables from the workspace
close all; % Close all open figure windows

% Load the original target variable from its .mat file
O_target = load("F1_target_variable.mat").new_set;

% Load the sampled values from the 'data_0.9.mat' file
v1 = load("data_0.9.mat").v1;
v3 = load("data_0.9.mat").v3;
filename = 'NLR_0.9.mat'; % Set the modified filename for the output .mat file

%% Create a mask to identify missing values (records where data is NaN)
mask = zeros(size(v1, 1), size(v1, 2));
mask(isnan(v1)) = 1; % Mark elements of the mask as 1 where v1 is NaN

%% Perform Kriging interpolation on the sampled values
[v1] = myKriging(v1);
[v3] = myKriging(v3);

%% Assemble the data into a dataset. The first five columns are the interpolated values, and the last column is the true value
data = [];
for i = 1:size(v1, 1) % Iterate over each row
    for j = 1:size(v1, 2) % Iterate over each column
        if mask(i, j) == 1 % Check if the mask indicates a missing value
            data = [data ; [v1(i, j), v3(i, j), O_target(i, j)]]; % Append the values to the data array
        end
    end
end

%% Randomly permute the indices of the data
R = randperm(size(data, 1));

%% Define the ratio of data to be used for training
train_ratio = 0.8; % Set the proportion of data to be used for training
```

```
%% Create the training dataset using the first 80% of the data
train = data(R(1:(floor(train_ratio * size(data, 1)))), :);

%% Create the test dataset using the remaining 20% of the data
test = data(((R(floor(train_ratio * size(data, 1)))) + 1):size(data, 1), :);

%% Save the training and test datasets to the specified file
save(filename, 'train', 'test'); % Save the datasets as variables 'train' and 'test' to the specified .mat file
```

Calculation of nonlinear regression results

```
clc; % Clear the command window to remove any previous commands
clear; % Clear all variables from the workspace
close all; % Close all open figure windows

% Load the test dataset from the 'NLR_0.9.mat' file
test = load('NLR_0.9.mat').test;
x = test(:, 1); % Extract the x-values from the test dataset
y = test(:, 2); % Extract the y-values from the test dataset

%% Constants for the polynomial model
% p1, p2, p3, p4 are the coefficients of the polynomial model
% These values are manually set for the model
% p1 for 0.1
% p2 for 0.1
% p3 for 0.1
% p4 for 0.1
% p1 for 0.2
% ...
% p1 for 0.9
p1 = -9.25467947621868;
p2 = 6.7882250570275e-5;
p3 = 63.499742498198;
p4 = -103.216103514737;

%% Fit the polynomial model to the data
% Calculate the predicted values using the polynomial model
z = p1 + p2 .* x + p3 .* y + p4 .* y.^2;
```

```
%% Evaluate the model
true_values = test(:, 3); % Extract the true values from the test dataset
predicted_values = z; % The predicted values from the model

% Calculate the Mean Absolute Error (MAE)
MAE = mean(abs(true_values - predicted_values));

% Calculate the Mean Squared Error (MSE)
MSE = mean((true_values - predicted_values).^2);

% Calculate the Root Mean Squared Error (RMSE)
RMSE = sqrt(MSE);

% Calculate the R-squared (R²) value
SS_tot = sum((true_values - mean(true_values)).^2); % Total Sum of Squares
SS_res = sum((true_values - predicted_values).^2); % Residual Sum of Squares
R2 = 1 - (SS_res / SS_tot); % R² calculation

%% Combine the evaluation metrics into a single vector
E = [MAE, MSE, RMSE, R2]; % Create a vector containing the evaluation metrics

Box plotting
clc; % Clear the command window to remove any previous commands
clear; % Clear all variables from the workspace
close all; % Close all open figure windows

% Load the error data from the '误差表.xlsx' file
data = xlsread("误差表.xlsx");
data(end,:) = []; % Remove the last row of data, if any

% Create a vector to group the data for boxplot, distinguishing different algorithms
groups = {'Kriging-MLP', 'Kriging-Random', 'MNR'};

%% Plot Boxplot 1
subplot(2,2,1)
% Combine the data into a matrix, with each column corresponding to an algorithm
r2_values = [data(:,2), data(:,8), data(:,15)];
h = boxplot(r2_values, 'Labels', groups, 'Orientation', 'vertical'); % Create a boxplot
set(h, 'LineWidth', 1.5); % Set the line width for the boxes
% Improve the appearance of the axes
set(gca, 'Box', 'on', ... % Turn on the box
'LineWidth', 1, ... % Set the line width
```

```
'XGrid', 'off', 'YGrid', 'off', ... % Turn off the grid
'TickDir', 'in', 'TickLength', [.015 .015], ... % Set tick direction and length
'XMinorTick', 'off', 'YMinorTick', 'off', ... % Turn off minor ticks
'XColor', [.1 .1 .1], 'YColor', [.1 .1 .1]); % Set axis colors

% Set the background color
set(gcf, 'Color', [1 1 1]);

% Improve the chart appearance
% Set the title and axis labels
ylabel('MAE', 'FontSize', 12, 'FontWeight', 'normal'); % Set y-axis label
xlabel('Algorithm', 'FontSize', 12, 'FontWeight', 'normal'); % Set x-axis label
% Set the font size and weight for the x-axis tick labels
set(gca, 'XTickLabel', groups, 'FontSize', 12, 'FontWeight', 'normal');
% Turn on the grid (optional)
grid on;

%% Plot Boxplot 2
subplot(2,2,2)
% Combine the data into a matrix, with each column corresponding to an algorithm
r2_values = [data(:,3), data(:,9), data(:,16)];
h = boxplot(r2_values, 'Labels', groups, 'Orientation', 'vertical');
set(h, 'LineWidth', 1.5);
set(gca, 'Box', 'on', ...
'LineWidth', 1, ...
'XGrid', 'off', 'YGrid', 'off', ...
'TickDir', 'in', 'TickLength', [.015 .015], ...
'XMinorTick', 'off', 'YMinorTick', 'off', ...
'XColor', [.1 .1 .1], 'YColor', [.1 .1 .1]);

set(gcf, 'Color', [1 1 1]);

% Improve the chart appearance
ylabel('MSE', 'FontSize', 12, 'FontWeight', 'normal');
xlabel('Algorithm', 'FontSize', 12, 'FontWeight', 'normal');
set(gca, 'XTickLabel', groups, 'FontSize', 12, 'FontWeight', 'normal');
grid on;

%% Plot Boxplot 3
subplot(2,2,3)
r2_values = [data(:,4), data(:,10), data(:,17)];
h = boxplot(r2_values, 'Labels', groups, 'Orientation', 'vertical');
```

```
set(h, 'LineWidth', 1.5);
set(gca, 'Box', 'on', ...
'LineWidth', 1, ...
'XGrid', 'off', 'YGrid', 'off', ...
'TickDir', 'in', 'TickLength', [.015 .015], ...
'XMinorTick', 'off', 'YMinorTick', 'off', ...
'XColor', [.1 .1 .1], 'YColor', [.1 .1 .1]);

set(gcf, 'Color', [1 1 1]);

ylabel('RMSE', 'FontSize', 12, 'FontWeight', 'normal');
xlabel('Algorithm', 'FontSize', 12, 'FontWeight', 'normal');
set(gca, 'XTickLabel', groups, 'FontSize', 12, 'FontWeight', 'normal');
grid on;

%% Plot Boxplot 4
subplot(2,2,4)
r2_values = [data(:,5), data(:,11), data(:,18)];
h = boxplot(r2_values, 'Labels', groups, 'Orientation', 'vertical');
set(h, 'LineWidth
```

Question four

```
Annex II Based on Kriging-MLP predictions
clc; % Clear the command window to remove any previous commands
clear; % Clear all variables from the workspace
close all; % Close all open figure windows

% Load the target variable data from the 'F2_target_variable.xlsx' file
O_target = xlsread("F2_target_variable.xlsx");

% Load collaborative variable 1 from the 'F2_collaborative_variable1.mat' file
v1 = load("F2_collaborative_variable1.mat").new_set;

% Load collaborative variable 3 from the 'F2_collaborative_variable3.mat' file
v3 = load("F2_collaborative_variable3.mat").new_set;

% Load the neural network structure from the 'net.mat' file
load("net.mat")

%% Prepare data from Attachment Two
target = NaN(266,266);
for i = 1:size(O_target,1)
```

```
        target(O_target(i,2)+1,O_target(i,1)+1) = O_target(i,5);
    end

    %% Create mask for recording missing values
    mask = zeros(size(target,1),size(target,2));
    mask(isnan(target)) = 1;

    %% Perform Kriging interpolation
    [Kriging_target] = myKriging(target);

    %% Prepare dataset. First two columns are collaborative variables and the last is Kriging interpolated value
    data = [];
    for i = 1:size(v1,1)
        for j = 1:size(v1,2)
            if mask(i,j) == 1
                data = [data ; [v1(i,j), v3(i,j), Kriging_target(i,j)]];
            end
        end
    end

    %% Predict using MLP
    x_test_normalized = (data - mu) ./ sigma;
    test_out = sim(net,x_test_normalized'); % Test
    test_out = test_out';

    %% Write the predictions back
    A_target = target;
    num = 0;
    for i = 1:size(v1,1)
        for j = 1:size(v1,2)
            if mask(i,j) == 1
                num = num + 1;
                A_target(i,j) = test_out(num);
            end
        end
    end

    %% Plotting
    contourf(A_target, 20, 'LineColor', 'none'); % 20 is the number of contour lines, 'LineColor', 'none' means no contour lines are drawn
    colormap(hot); % Use the hot color map
```

```
colorbar; % Add a color bar
hold off; % Release the current graph

%% Plotting 2
figure;
surf = surf(A_target);
colormap('gray'); % Use the grayscale color map
xlabel('Column Sequence', 'FontSize', 22); % X-axis label
ylabel('Row Sequence', 'FontSize', 22); % Y-axis label
zlabel('Value', 'FontSize', 22); % Z-axis label
axis tight; % Automatically adjust axis ranges to fit data tightly
colorbar; % Add a color bar
shading interp; % Remove the grid on the image, making it smooth
hold on;

% Initialize arrays to store non-NaN point data
new_rows = size(target, 1);
new_cols = size(target, 2);
x_data = []; % Store x coordinates (column index)
y_data = []; % Store y coordinates (row index)
z_data = []; % Store z coordinates (values)
% Traverse the new_set matrix to find all non-NaN points
for x = 1:new_rows
    for y = 1:new_cols
        if \~isnan(target(x, y))
            x_data = [x_data, y]; % Store row index as x coordinate, since MATLAB typically uses columns as the x-axis, but here we swapped x and y
            y_data = [y_data, x]; % Store column index as y coordinate
            z_data = [z_data, target(x, y)]; % Store corresponding value as z coordinate
        end
    end
end
scatter3(x_data, y_data, z_data, 20, 'filled', 'g'); % Plot scatter plot with filled green points
legend('Interpolation results', 'observed value', 'FontSize', 18); % Add a legend
```