Problem Chosen	2024 ShuWei Cup Summary Sheet	Team Control Number
B		2024093029335

Spatial variable prediction problem of synergistic kriging optimization based on FMM

Summary

When studying the spatial variation of spatial variables, Co-Kriging collaborative estimation method shows excellent local detail capturing ability and spatial correlation advantages. In this paper, a cooperative Kriging model based on FMM optimization is established to solve the problem of covariance calculation, and the change mode of spatial variables is deeply studied.

For problem 1, after normalization and 2D meshing of Annex 1 data, random uniform resampling is adopted to ensure fairness and randomness, and then linear interpolation and Kriging interpolation models are constructed. The fit of the models is verified by 3D and contour maps, and the global hierarchical, local peak characteristics and spatial correlation of F1_target are analyzed. Secondly, for the second question, python is used to solve the RMSE error analysis model established, and the conclusion is drawn: the RMSE error of Kriging interpolation decreases rapidly with the increase of sample size and tends to be stable, while the linear interpolation decreases slowly and is difficult to convergence.

For problem 2, the correlation between variables is studied, and two cooperative variables are selected comprehensively. First, the initial trend of variables was drawn by scatterplot and density histogram, and then the Jarque-Bera test was combined to confirm the non-normal distribution of data. Spearman correlation coefficient analysis was adopted, and SPSS software was used to calculate the correlation between target variables and co-variables. The results show that covariate 1 and covariate 4 have strong correlation with the target variable in 99% of cases. Therefore, Collaborative1 and Collaborative4 are selected as co-variables.

For problem 3, based on problem 2 Kriging model, the FMM optimization co-Kriging model is established by introducing fast multipole method, and the anisotropy of F1_target space variable is revealed by two-dimensional contour map. Secondly, on the basis of the first question, the second question establishes the optimized RMSE, R^2 and MAPE models, and obtains the line graph through the python visual model. The results show that the error is reduced rapidly, the convergence is faster when the sample size is increased, and the global and local prediction accuracy is significantly improved under high sample size. At the same time, the advantages of the co-Kriging model in spatial correlation and interpolation accuracy were verified by comparing the random forest model and quantifying the performance of the two models combined with multi-index radar map and local high-value thermal map. Conclusion: The optimization model has significant advantages in capturing spatial correlation, balancing global and local fluctuations and improving accuracy, and is suitable for F1_target spatial variables.

For problem 4, it is necessary to ensure that the trend of the target variable (F2_target) is estimated in an optimal way. Based on the data in Attachment 2, this paper conducts Jarque-Bera normal distribution test and Spearman correlation coefficient solution, and selects the optimal covariate. On the basis of problem 3 model, a collaborative Kriging model based on FMM optimization is established to interpolate the unsampled points and reveal the trend of target variable (F2_target).

Keywords: Co-kriging; Fast Multipole Method; Jarque-Bera Statistic; Spearman; Random forest

Content

1. Introduction

1.1 Background

Things and other technologies and the improvement of computing power, spatial data acquisition is more convenient and large-scale, which promotes the development of spatial statistics. However, in practical applications, high-precision measurement samples are scarce due to cost and technical limitations, while low-cost measurement has a wide coverage but low precision, so it is necessary to improve the estimation accuracy of target variables by combining multi-source data with collaborative estimation technology.

The traditional Co-Kriging algorithm can jointly estimate multiple variables, but its application in large-scale and high-dimensional scenarios is limited due to the complexity of covariance and cross covariance functions. Therefore, we propose to construct a collaborative Kriging model based on fast multipole method (FMM) optimization to effectively solve the computational complexity under large-scale data and improve the inference accuracy of spatial variation patterns of target variables. This is the core task of our current research.

1.2 Work

In order to comprehensively analyze and predict the spatial distribution of target variables, we build and optimize the model step by step based on questions 1 to 4.

Firstly, the F1_target data is normalized and meshing in two dimensions, and the linear interpolation and Kriging interpolation models are constructed by random uniform resampling. The spatial distribution characteristics and the relationship between sample size and error are analyzed.

Secondly, through scatterplot, density histogram and Jarque-Bera test, combined with Spearman correlation coefficient and thermal map to quantify the correlation between target variables and covariate, the covariate with the strongest correlation was finally selected.

Then, the FMM optimization co-Kriging model is introduced to improve the efficiency of covariance calculation. Combined with the random forest model, the performance is evaluated by radar map and local hot spot error thermal map, and the spatial variation pattern is revealed.

Finally, in question 4, based on the results of the first three questions, co-variable selection was optimized, and FMM was used to optimize co-Kriging model for interpolation prediction of unsampled points, so as to accurately reveal the spatial variation trend of target variables and provide scientific basis for subsequent research.

2. Problem analysis

2.1 Data analysis

In Annex 1 and Annex 2 provided by the title, the data of each spatial variable file is in three-dimensional mode, and the amount of data is huge. We found that the magnitude gap between data files is relatively large, which brings great trouble to our

data analysis and processing. Therefore, after reducing the three-dimensional data to two-dimensional data, we eliminate the magnitude gap by normalization, and then interpolate and smooth the remaining values. These data are used for correlation analysis to study the change pattern of target variables.

2.2 Analysis of question one

According to question 1, we need to comprehensively analyze the spatial distribution characteristics and influencing factors of F1_target. For the first part of problem 1, on the basis of data normalization and two-dimensional meshing, we adopted random uniform resampling to ensure the fairness and randomness of spatial points, established linear interpolation and Kriging interpolation models, and drew three-dimensional maps and contour maps to analyze the adaptability of the two models, and intuitively presented the spatial change pattern of F1_target. In the second part, the RMSE error analysis model is established based on the interpolation model, and its influence on the prediction error is studied by adjusting the sample size.

2.3 Analysis of question two

In order to analyze the correlation between F1_target and co-variables, we first drew a scatter plot to visually analyze the relationship between target variables and co-variables (Collaborative1, 2, 3, and 4), and then obtained the correlation trend between target variables and each co-variable. Then the density histogram was drawn and the normal distribution curve was fitted. After the normal trend of the data was preliminarily judged, the normality of the variables was quantitatively analyzed through Jarque-Bera test. Spearman's correlation coefficient was finally used to analyze the correlation between the target variable and the co-variable, and heat map was drawn to show it. The two covariables were identified by visual analysis and statistical test.

2.4 Analysis of question three

Based on the model of problem two, we need to select the cooperative variable with the strongest correlation with the target variable, and further study the spatial variation pattern of the target variable F1_target. Because covariance is difficult to be calculated by cokriging algorithm, the fast multipole method is used to calculate covariance function, and a cokriging model based on FMM optimization is established to improve the interpolation accuracy, and a two-dimensional contour map is drawn to show the spatial variation. The relationship between sample size and estimation error is shown by MRSE, R^2 and MAPE line plots. At the same time, a random forest model is established as a comparison, and the global prediction performance of the two models is quantified by multiple index radar map and local high value heat map.

2.5 Analysis of question four

Problem 4 is a synthesis of the previous three questions, which requires selecting the best method to estimate the trend of the target variable (F2_target). Before the estimation, the synergistic effect of the co-variable on the target variable is also considered, the normal distribution test is carried out and the correlation coefficient is solved, and the variable with the strongest Spearman correlation coefficient is found as the co-variable. On the basis of problem 3, a collaborative Kriging model based on FMM optimization is established, and the remaining unsampled points are interpolated to estimate the trend of target variables.

3. Symbol and Assumptions

3. 1 Symbol Description

Symbol	Description
$\gamma(h)$	Semivariance at a given distance h
$C(h)$	Covariance at a given distance h
R^2	goodness-of-fit
$FMM(h)$	Fast Multipole Method-based covariance function
$R_{hotspot}$	Region of detected hotspots above the quantile threshold q
c	Maximum correlated variance
c_0	Minimum variance
ρ_s	Correlation coefficient

3. 2 Fundamental assumptions

1.Target variables and covariates change gradually in space.

2.Mean values are constant regionally, with covariance depending only on point distance.

3.Prediction errors are spatially independent, driven by sample size and model features.

4. Modeling and Answering of Question 1

4.1 Data preprocessing

In view of the magnitude difference of the data in Annex 1, Python normalization is adopted to eliminate the impact, and the target variable F1_target is reduced to a two-dimensional grid form, which is convenient for subsequent spatial interpolation and visualization. The two-dimensional grid is shown as follows:

$$\begin{bmatrix} 0.330256 & 0.330729 & ... & 0.197680 & 0.196259 \\ 0.332623 & 0.332860 & ... & 0.200758 & 0.199337 \\ ... & ... & ... & ... & ... \\ 0.414299 & 0.405066 & ... & 0.405066 & 0.399148 \\ 0.415009 & 0.417377 & ... & 0.400331 & 0.394413 \end{bmatrix} \tag{1}$$

4.2 The establishment of linear interpolation model

4.2.1 Random uniform resampling

For the first small question of question 1, in order to ensure the fairness of space coverage and simulate sampling randomness, random uniform resampling method is adopted to randomly extract sample points of a specified proportion from target variables through Python to obtain their coordinates and values, so as to achieve unbiased and uniform coverage of the entire space.

4.2.2 Linear interpolation model

The above analysis shows that the resampling points are distributed randomly without fixed rules. To simplify the model, a linear interpolation model was constructed using Python to estimate the values of unsampled spatial variables, where the weight ω between sampling points and interpolation points is correlated:

$$\omega_i = \frac{\frac{1}{d_i}}{\sum_{j=1}^{n} \frac{1}{d_j}} \tag{2}$$

Where, di is the interpolation point and sampling (x_i, y_i) point spacing, according to the Euclid formula can be obtained:

$$d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} \tag{3}$$

Considering that in the linear interpolation method, the F_target target variable shows a linear change trend among several sampling points (x_i, y_i) in the two-dimensional grid, a linear interpolation model is established:

$$z(x,y) = \sum_{i=1}^{n} \omega_i z_i \tag{4}$$

Where z_i is the known value of the sampling point (x_i, y_i).

4.3 The establishment of Kriging model

Linear interpolation does not consider the spatial correlation of the target variables. Based on random uniform sampling, we further establish Kriging interpolation model:

$$h_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{5}$$

For the spatial relationship between sampling points, the correlation of spatial variables can be described by semi-variance function, and the definition formula is as follows:

$$\gamma(h)=\frac{1}{2N(h)}\sum_{i=1}^{N(h)}[z(x_i)-z(x_i+h)]^2 \tag{6}$$

In order to calculate the semi-variance of arbitrary distance, a linear model is selected to fit the relationship between distance and semi-variance, and its expression is as follows:

$$\gamma(h)=c_0+c\cdot h \tag{7}$$

c0 represents the minimum variance; c represents the variance of the variable with the greatest correlation. Based on the geometric relationship between the sampling points, variance function is introduced to calculate the correlation, and the covariance matrix is constructed with the aim of optimal linear unbiased estimation:

$$r_{ij}=c-\gamma(h_{ij}) \tag{8}$$

Where, $c=c0+c$ presents the maximum covariance, h_{ij} is the distance between sampling points i and j, therefore, the covariance matrix is:

$$C=\begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \dots & \dots & \dots & \dots \\ r_{n1} & r_{n2} & \dots & r_{nn} \end{bmatrix} \tag{9}$$

According to the correlation formula between interpolation points and known points, the covariance vector can be obtained as follows:

$$c=\begin{bmatrix} r_1 \\ r_2 \\ \dots \\ r_n \end{bmatrix} \tag{10}$$

To sum up, the Kriging interpolation model obtains the weights by solving the equations (11), which not only realizes the optimal linear unbiased estimation of the spatial variables, but also quantifies the uncertainty of the interpolation results.:

$$\begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} & 1 \\ r_{21} & r_{22} & \dots & r_{2n} & 1 \\ \dots & \dots & \dots & \dots & \dots \\ r_{n1} & r_{n2} & \dots & r_{nn} & 1 \\ 1 & 1 & \dots & 1 & 0 \end{bmatrix}\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \dots \\ \lambda_n \\ \phi \end{bmatrix}=\begin{bmatrix} r_1 \\ r_2 \\ \dots \\ r_n \\ 1 \end{bmatrix} \tag{11}$$

In system (11), $\emptyset$ is the Lagrange multiplier, an auxiliary variable ensuring the weight λi satisfies the unbiased constraint. Using the calculated λi, the optimal coefficient method computes the weighted sum of known points zi, yielding the interpolated value $z0$:

$$z_0=\sum_{i=1}^{n}\lambda_i z_i \tag{12}$$

4.4 The establishment of RMSE model

In order to quantitatively analyze the relationship between sample size and interpolation error, reveal the convergence process and change rule of error, and establish the RMSE model in the range of sampling ratio [0.01,0.05,0.1,0.2].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(z_i - z_j)^2}{n}} \tag{13}$$

The smaller the RMSE value, the better the prediction ability of the model.

4.5 Model solving

(1) Interpolation model 3D diagram

First, the linear interpolation and Kriging interpolation models established in Figure1 were used to draw the three-dimensional interpolation graphs of F1_target variable before and after sampling, as shown in figures 1 below:

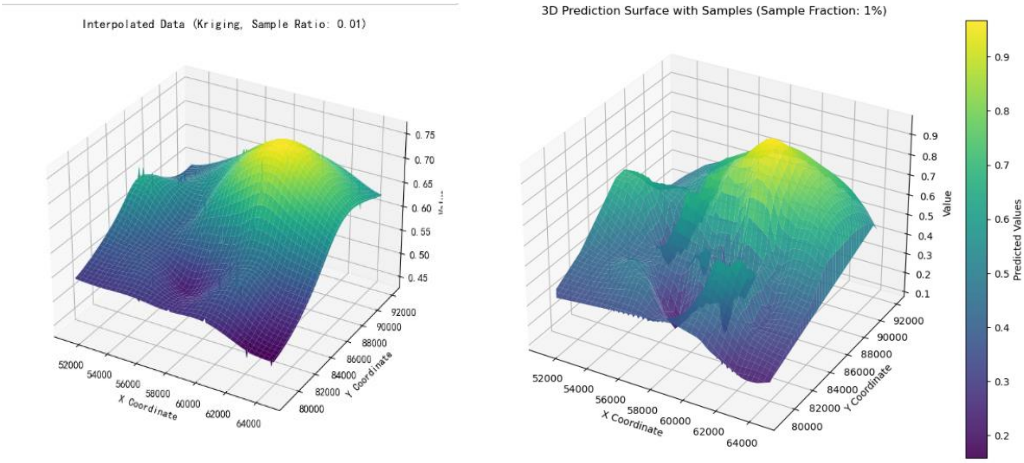


Figure 1 three-dimensional interpolation graphs of F1_target

As can be seen from the figure, Kriging interpolation models spatial correlation to achieve smooth and consistent results with global trends, while linear interpolation relies on resampling point distribution and is difficult to capture the overall trend. In order to further analyze the spatial distribution of F1_target and the influence of interpolation, the 3D diagram of the original data is drawn as follows:

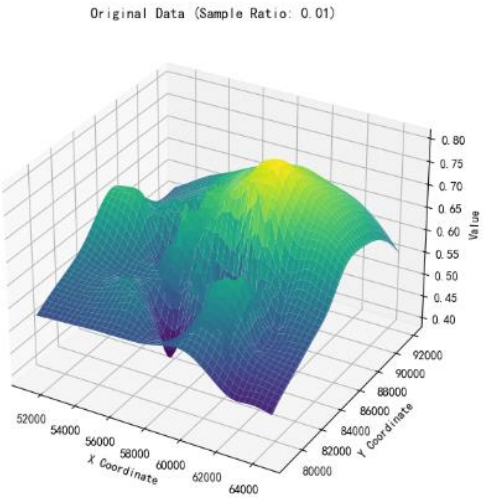


Figure 2 three-dimensional graph of original data

As can be seen from figures 1, 2, the variation of F1_target space variable has the following characteristics:

(1)Global hierarchy: From low value (blue) to high value (yellow) changes, showing a significant spatial gradient, high value concentration, low value widely distributed.

(2)Local peak characteristics: the overall change is gentle, the local fluctuation is significant, the high value area is accompanied by obvious local peak, and is significantly affected by local factors.

(3)Spatial correlation: the changes of adjacent regions are smooth and similar, and the aggregation of high and low values is strong, reflecting the correlation of spatial variables..

(2) Interpolation model detour graph

Python implements linear interpolation, Kriging interpolation, and devious plotting of the original data (Graphs 3), and the original F1_target variable shows significant spatial correlation, showing local peaks and scattered low value regions.

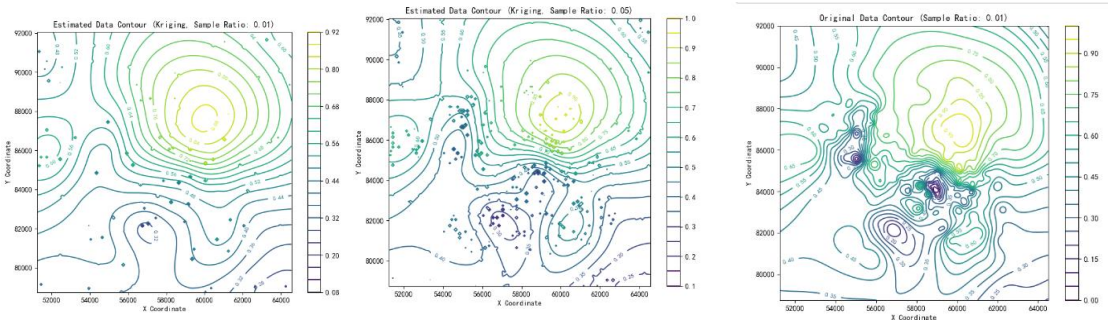


Figure 3 original data Contour

The Kriging interpolation model has high adaptability, accurately portrays the global trend and local details of F1_target, and shows strong prediction ability in sparse regions. The results of the roundabout map are highly consistent with those of the three-dimensional map. The roundabout map reveals the boundary of the high-value hot spot, and the three-dimensional map highlights the peak shape, which verifies the transition of F1_target from low value to high value and the concentration of the high-value region. In summary, F1_target has both global gradient and local fluctuation characteristics, and Kriging interpolation fully reflects its rule, providing reliable support for subsequent analysis.

(3) Error analysis of RMSE

Compared with MSE, RMSE can more directly reflect the difference of prediction performance in sparse regions. The RMSE of the interpolation method was analyzed with Python as the sample size changed, and the sample volume-error relationship graph was drawn (Figure 4、5).：

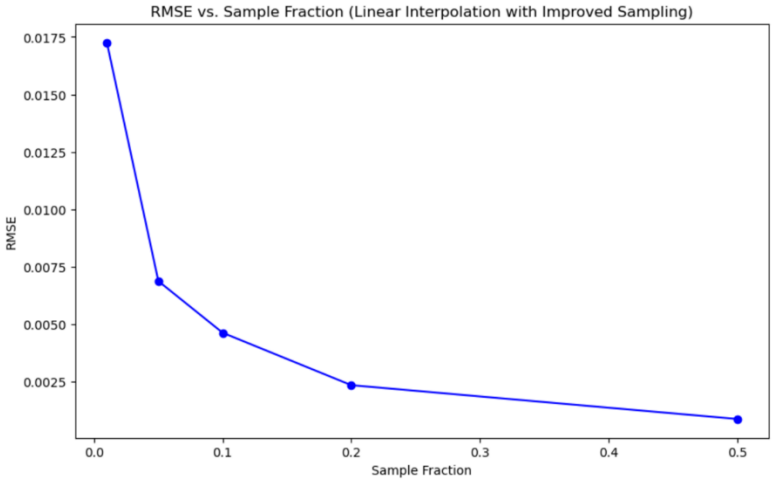


Figure 4 RMSE vs Sample Fraction(Linear)

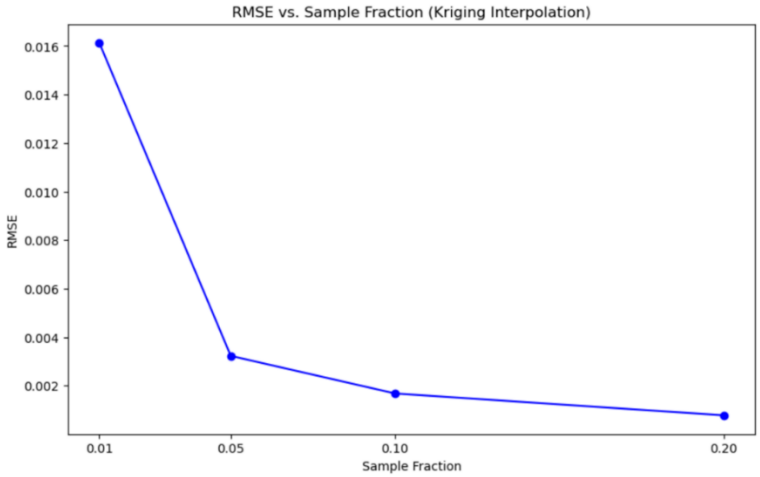


Figure 5 RMSE vs Sample Fraction(Kriging)

As can be seen from Figures 4 and 5, RMSE of Kriging interpolation decreases rapidly with the increase of sample size and becomes stable soon, while RMSE of linear interpolation decreases slowly with the increase of sample size and is difficult to converge.

(4) Result analysis

① Overall trend:

RMSE decreases nonlinearly with the increase of sample size. The error is significant at low sample size and decreases rapidly after increasing sample size, but the marginal benefit decreases and eventually becomes stable.

② Low sample size:

The error of linear interpolation is large, it only relies on local information, and it is difficult to capture the global trend. Kriging interpolation uses spatial correlation modeling, with significantly lower errors and more accurate predictions.

③ Sample size increase:

When the sample size of linear interpolation is 10%-20%, the error gradually decreases but does not converge, and the dependence is strong. Kriging interpolation can be stable with 10%-20% sample size, and has high precision under low sample size.

④ High sample size:

The error of linear interpolation decreases slowly and its ability is limited. Kriging interpolation error is stable, high precision display, especially suitable for sparse data scenarios.

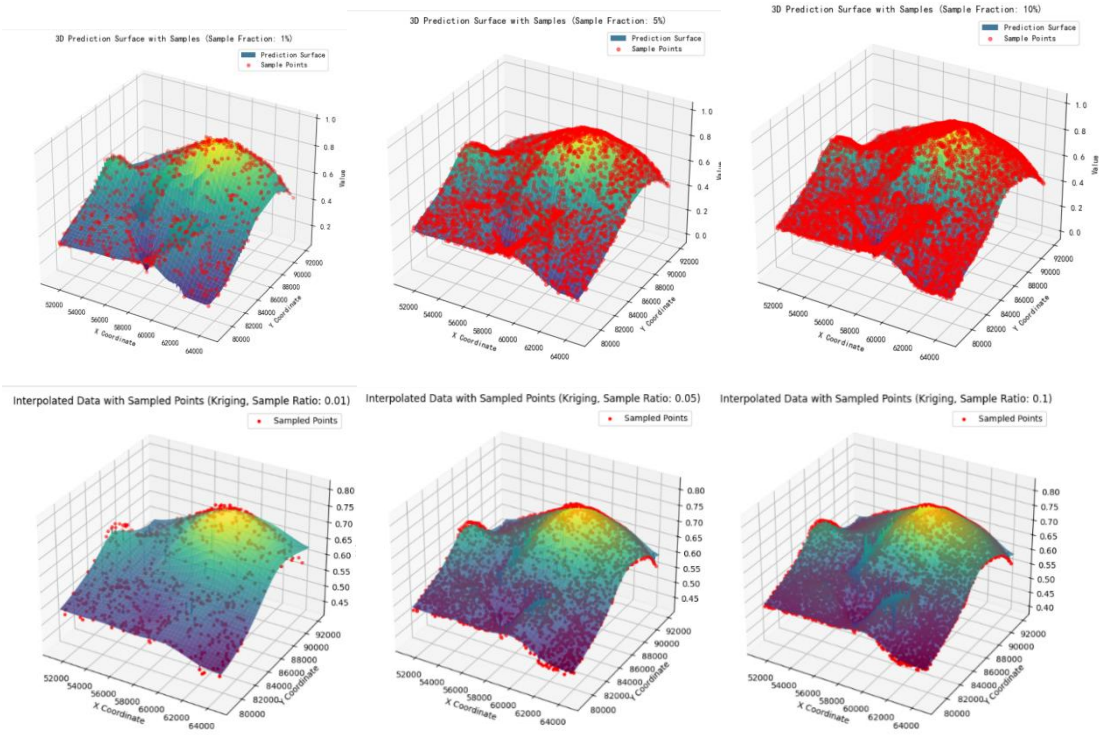
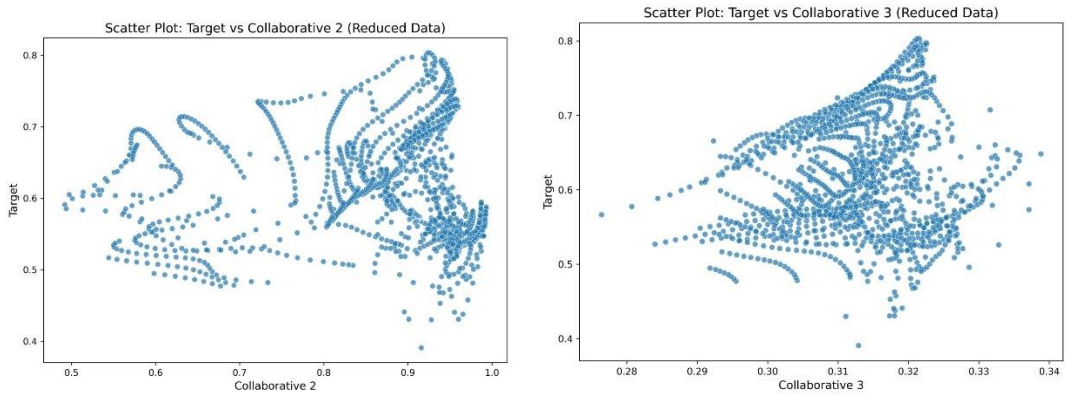


Figure 6 sample size variation

5. Modeling and Answering of Question 2

5.1 Scatter plot analysis

Problem 2 requires us to analyze the correlation between the target variable and the auxiliary variable.



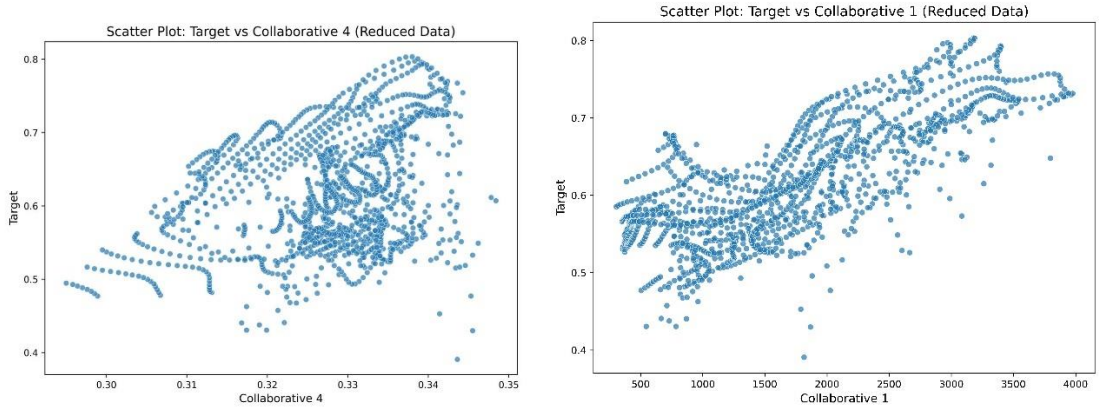


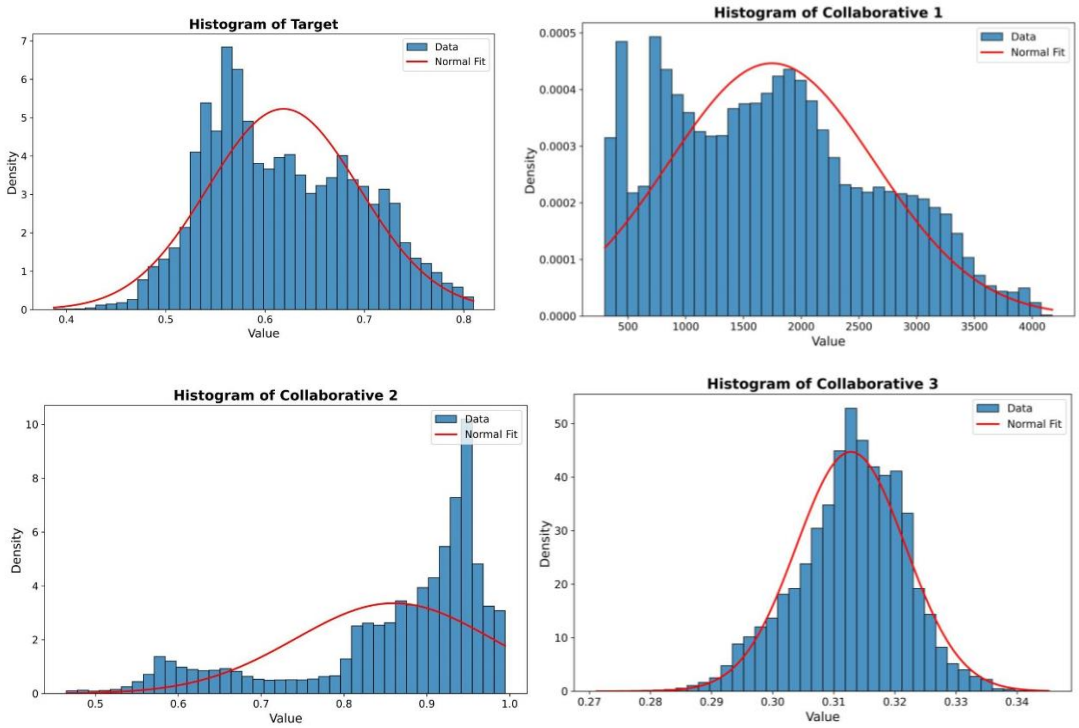
Figure 7 Scatter plot analysis

From the above figure, it can be found that the correlation between Target variable and co-variable (Collaborative1, 2, 3, and 4) is limited. Except for some correlation trends between Target and Collaborative1, the scatter distribution between other relevant variable-co-variable pairs is relatively irregular.

5.2 Correlation analysis

5.2.1 Density - normal graph

In order to explore whether the five spatial variables conform to the normal distribution characteristics, the density histogram is drawn for each variable in turn, and the normal distribution curve is superimposed for preliminary fitting analysis.



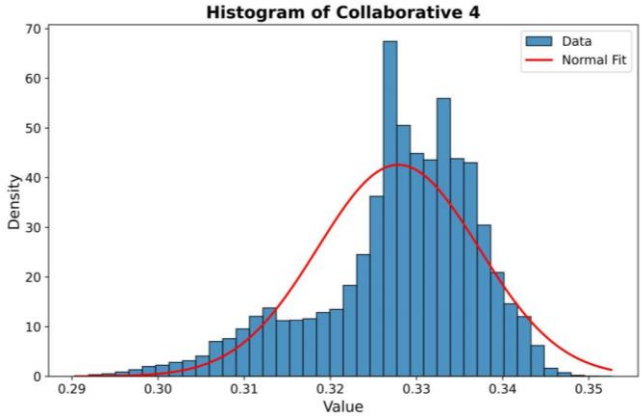


Figure 8 Density - normal graph(all)

(1)F1_target data is nearly symmetric as a whole, but the tail is deviated and the right tail is biased, which does not fully conform to the normal distribution.

(2)F1_Collaborative1 has a significant right-sided, left-sided multimodal feature, far from normality.

(3)F1_Collaborative2 presents a bimodal distribution with a large peak interval and a serious deviation from normality.

(4)F1_Collaborative3 is symmetrical, and the peak value fits normally, but the right tail is slightly heavier, and the normality is not fully satisfied.

(5)F1_Collaborative4 is close to the normal distribution shape, but slightly skewed to the right and has poor tail fitting, which does not fully conform to normal.

5.3 Jarque-Bera Normal distribution test

The histogram and line chart show that there is a limited correlation between variables. In order to further verify the correlation between the target variable and the co-variable, the Jarque-Bera test was used for quantitative analysis of normality based on sufficient sample size (>30), providing a reliable basis for statistical analysis.

5.3.1 Calculation of skewness S and kurtosis K

To infer the normality of the data and provide support for correlation coefficient analysis, hypothesis is set:

(1) Null hypothesis H_0: data follows a normal distribution

(2) Alternative hypothesis H_1: The data does not follow normal distribution

The skewness S and kurtosis K statistics are calculated, where S measures the symmetry of distribution and reveals the trend of spatial migration. K reflects the frequency of volatility and extreme values, and the specific expression is as follows:

$$S = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2\right)^{\frac{3}{2}}} \tag{14}$$

In formula (14), x_i represents the i th sample value in the data; The x digit represents the sample mean.

$$K=\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2\right)^2} \tag{15}$$

5.3.2 Jarque-Bera model

Secondly, the relation between skewness S and kurtosis K is established through the Jarque-Bera test formula:

$$JB=n\left(\frac{S^2}{6}+\frac{(K-3)^2}{24}\right) \tag{16}$$

According to the obtained JB value, we can calculate its corresponding p-value through the chi-square distribution to judge the normality of the spatial variable value:

$$p-value=P(\chi^2\geqslant JB) \tag{17}$$

χ represents a Chi-square distribution of 2 degrees of freedom. If p-value>0.05, H0 is accepted, and the variable values conform to normal distribution; If p-value$\leqslant$ 0.05, H0 is rejected, and the variable values do not conform to the normal distribution.

Draw the distribution diagram of skewness and kurtosis of variables in Annex 1 through Python, and calculate JB test values as shown in Table 1 below:

Table 1: JB test values

variable	JB Statistic	S	K	p-value
Target	2421.6	0.2648	2.2645	0.0
Collaborative 1	3055.9	0.3471	2.5520	0.0
Collaborative 2	22024.2	-1.3189	3.7160	0.0
Collaborative 3	1606.1	-0.3506	3.2305	0.0
Collaborative 4	10823.9	-0.8973	3.6714	0.0

As can be seen from Table 1 above, the large sample size amplified the slight skew and tail characteristics of the data, and the P-values of Jarque-Bera (JB test) were all lower than 0.05 level, so the null hypothesis was rejected and the samples were considered not to meet the requirements of normal distribution.

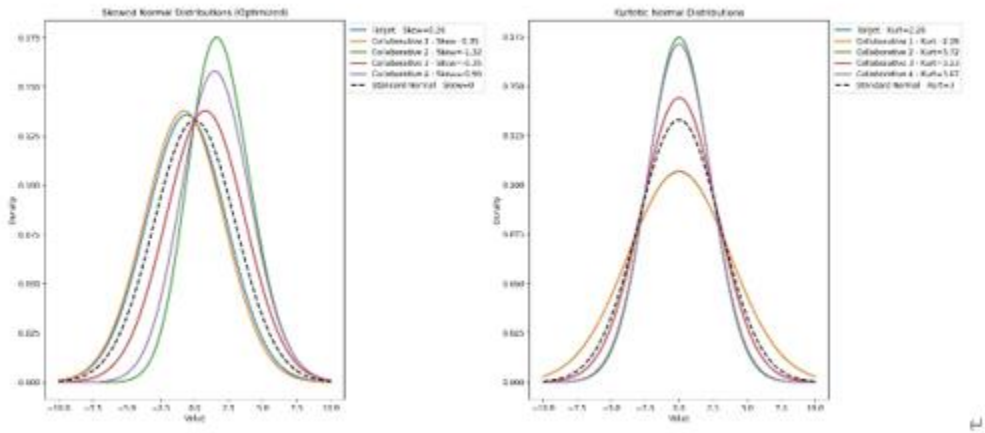


Figure 9 skewness and kurtosis results

Python visualizations of skewness and kurtosis showed deviations from normality, confirming the invalidity of the normal hypothesis. Thus, Spearman correlation was applied to assess the relationship between the target and cooperative variables..

5.3.3 Spearman Correlation coefficient model

Through Python programming, four two-dimensional matrix data is flattened into a single column vector, and then the entire spatial variable data file is treated as a complete variable:

$$\rho_s = 1 - \frac{6\sum_{i=1}^{n} {d_i}^2}{n(n^2-1)} \tag{18}$$

Where ρ_S represents the Spearman correlation coefficient between the target variable X and Y (four covariables), which has the following relationship expression:

$$d_i = r_{x_i} - r_{y_i} \tag{19}$$

SPSS software was used to calculate each group of variables to obtain phase relation table 2, and the numerical values were visualized in the bar chart：

Table 2: Spearman correlation

combination	ρ	Significance(p)
Target- Collaborative 1	0.750**	<0.01
Target- Collaborative 2	-0.081**	<0.01
Target- Collaborative 3	0.279**	<0.01
Target- Collaborative 4	0.381**	<0.01

Note: ** is at 0.01 level (double tail), the correlation is significant

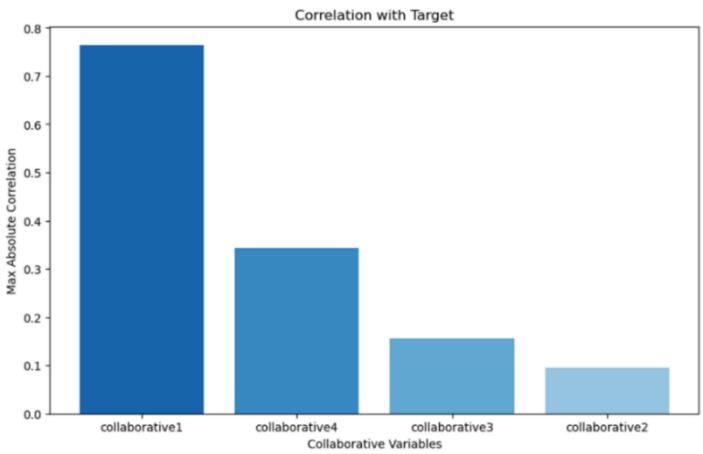


Figure 10 correlation results

At a 99% confidence level, the correlation results were statistically significant. Collaborative1 and Collaborative4 showed strong positive correlations with the target (ρ=0.750 and ρ=0.381), Collaborative3 was weakly correlated (ρ=0.279), and Collaborative2 was weakly negatively correlated (ρ=−0.081). Thus, Collaborative1 and Collaborative4 are suitable as collaboration variables.

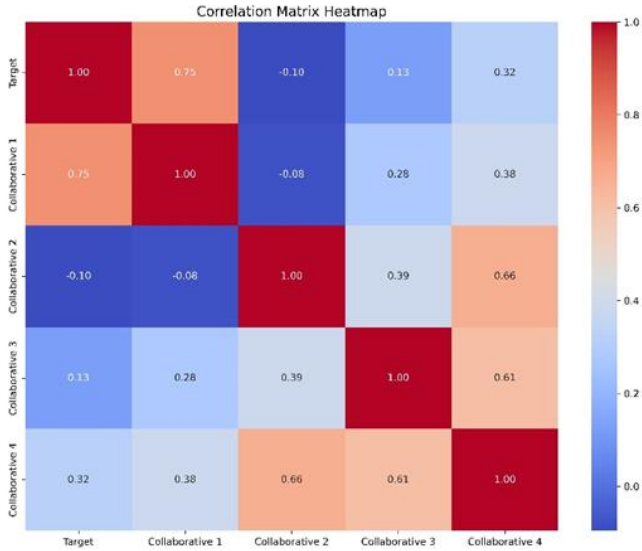


Figure 11 correlation Heatmap

The thermal map still shows that Target has a high correlation with Collaborative 1 and Collaborative 4, especially the strongest correlation with Collaborative 1. Therefore, Collaborative 1 and Collaborative 4 are selected as co-variables.

6. Modeling and Answering of Question 3

Target shows the highest correlation with Collaborative1. Using random sampling and grid data, the normalized Kriging model is extended. Fast multipole method (FMM) optimizes covariance calculation, enhancing interpolation accuracy and spatial prediction.

6.1 Sampling point processing

In order to ensure the spatial correlation and improve the reliability of covariance function fitting, a sampling point set containing the values of the cooperation variable and the target variable is constructed by associating the changes of both.

$$(X_i, Y_i, T_i, C_{1i}) \tag{20}$$

Further analyze the covariance and cross-covariance between target and cooperative variables, establishing a joint distribution model that assumes sampling at the same spatial location and satisfies multivariate normality:

$$Z(s) = \begin{bmatrix} F1_{target}(s) \\ Collaborative_1(s) \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_{target} \\ \mu_{collab1} \end{bmatrix}, \sum \right) \tag{21}$$

The expression of covariance matrix can be obtained as follows:

$$\sum = \begin{bmatrix} \sigma^2{}_{target} & \sigma_{target-collab1} \\ \sigma_{target-collab1} & \sigma^2{}_{collab1} \end{bmatrix} \tag{22}$$

6.2 Covariance function

The fast multipole method (FMM) uses hierarchical grouping and recursion for precise covariance calculation of adjacent points, while reducing time complexity from O(nlogn) or O(n) for distant points, effectively lowering computation cost while maintaining accuracy.

$$C_{ij}(h) = \begin{cases} C_0 + C \cdot FMM(h), h \leqslant a \\ C_0 + C, \qquad\qquad h > a \end{cases} \tag{23}$$

Where, FMM (h) represents two covariance function calculations optimized by fast multipole method, I,j∈{target,collab1};

6.2.1 Variance function

In the cooperative Kriging interpolation model based on FMM optimization to be established in this problem, the spherical variation function which is more suitable for data complexity is adopted to eliminate the spatial distribution of high value concentration or low value diffusion of target variables and cooperative variables which are difficult to solve by linear variation function in problem 1.

$$\gamma(h) = \begin{cases} C_0 + C\left[1.5\frac{h}{a} - 0.5\left(\frac{h}{a}\right)^3\right], h \leqslant a \\ C_0 + C, \qquad\qquad h > a \end{cases} \tag{24}$$

In formula (24), C0 is the small scale error value, C represents the growth amplitude of the variance function, and a is the distance between the correlated variable and the covariate.

6.3 Collaborative Kriging interpolation model based on FMM optimization

Collaborative kriging is widely used in the spatial prediction of high cost variables such as geological exploration and mineral estimation, but the implementation is complicated. To solve this problem, FMM optimization covariance calculation is introduced, co-Kriging model is constructed, and collaborative variables are integrated into the interpolation process to accurately restore the spatial variation trend and reduce the prediction error.

6.3.1 Cokriging interpolation formula

The predicted value of the spatial variable at the unsampled point s:

$$\hat{T}(s) = \sum_{i=1}^{n} \lambda_i{}^{target} T_i + \sum_{j=1}^{m} \lambda_j{}^{collab1} C_{1j} \tag{25}$$

The co-Kriging equations based on FMM optimization can be obtained as:

$$\begin{bmatrix} C_{target-target} & C_{target-collab1} & 1 \\ C_{collab1-target} & C_{collab1-collab1} & 1 \\ 1^T & 1^T & 0 \end{bmatrix} \begin{bmatrix} \lambda_{target} \\ \lambda_{collab1} \\ \phi \end{bmatrix} = \begin{bmatrix} C_{target}(s) \\ C_{collab1}(s) \\ 1 \end{bmatrix} \tag{26}$$

The fast multipole method optimizes the covariance calculation, makes the cokriging model complete the large-scale interpolation efficiently, and visually displays the spatial distribution of the target variables through the two-dimensional contour map.

6.4 Error analysis model

Due to insufficient sampling of discontinuous spatial variables such as rainfall and pollutant concentration, the interpolation accuracy is easy to decrease. In order to quantify the relationship between sample size and estimation error and evaluate the performance of the FMM optimized collaborative Kriging model, RMSE, R^2 and MAPE were used to analyze the absolute error, goodness of fit and relative error from multiple dimensions, providing a reference for the prediction of high-cost spatial variables.

(1) RMSE Model

In order to restore the absolute value of the target variable, the RMSE model in question 1 was used to measure the prediction accuracy under different sampling ratios, and the performance of the model in sparse regions and outliers was evaluated:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(T_i - \hat{T}_i\right)^2} \tag{27}$$

(2) R^2 Model

An R^2 determination coefficient model was established to explain the influence of sample size on model performance through goodness of fit:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(T_i - \hat{T}_i\right)^2}{\sum_{i=1}^{n}\left(T_i - \hat{T}_i\right)^2} \tag{28}$$

When the sample size is low, the R^2 may be low; As the sampling ratio increases, the R^2 change curve quantifies the asymptotic fitting ability of the model.

(3) MAPE Model

When the spatial variable values are small, RMSE may be ignored, while MAPE highlights the relative prediction error in the low-value region. Combining RMSE and R^2, the error analysis model of MAPE was established to reveal the difference between the global and local performance of the model.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{T_i - \hat{T}_i}{T_i}\right| \times 100\% \tag{29}$$

6.5 Random forest interpolation model

Through the FMM-optimized co-Kriging model, its ability to improve the prediction accuracy and make up for the information deficiency under different

sampling ratios is verified. In order to analyze the relationship between sample size and error and the spatial model of target variables, a flexible random forest model is introduced.

6.5.1 Prediction of a single decision tree

Each decision tree makes partition predictions for the target variable Ti based on the input feature xi:

$$f_t(x) = \sum_{l=1}^{L_t} \omega_l f(x \in R_l) \tag{30}$$

6.5.2 Stochastic forest integrated prediction

In order to reduce the overfitting risk of single decision tree and improve the robustness of prediction. A random forest is constructed by integrating N decision trees, and the predicted value T(x) is obtained by averaging the results of multiple sub-models.

$$\hat{T}(x) = \frac{1}{n} \sum_{t=1}^{n} f_t(x) \tag{31}$$

6.6 Multivariate contrast model

6.6.1 Global performance comparison model based on multiple indicators

On the basis of question 2 and Question 2, the analysis of EVS, prediction time and training time is added, a multi-index model is established, and the global performance is compared and analyzed.

(1) EVS

Test the interpretation ability of the model to the change of the target variable, the value range [0,1], the larger the value, the better the prediction performance of the model.。

$$EVS = 1 - \frac{Var\left(T_i - \hat{T}_i\right)}{Var(T_i)} \tag{32}$$

(2) Forecast time

The calculation time required by the measurement model to predict the unsampled points reflects the efficiency of the algorithm.

(3) Training time

Measure the time it takes to build a model to estimate the computational overhead of the model.

6.6.2 Robustness evaluation model

The multi-index comparison model has analyzed the prediction ability of low-value regions. In order to evaluate the performance of the co-Kriging model of random forest and FMM optimization in high-value regions, a local hotspot prediction error model is constructed, and the quantile q defines the part of the target variable value range higher than q as the hotspot region:

$$T_{threshold} = Quantile(T(s), q) \tag{33}$$

Next, sample the hotspot area and obtain the sampling point s∈Rhotspot. The expression for calculating the prediction error is as follows:

$$\begin{cases} Error_{RF}(s) = |T_{true}(s) - \hat{T}_{RF}(s)| \\ Error_{CK}(s) = |T_{true}(s) - \hat{T}_{CK}(s)| \end{cases} \tag{34}$$

Finally, the average error over the entire hot spot region is:

$$\begin{cases} MAE_{RF} = \dfrac{1}{|R_{hotspot}|} \sum\limits_{s \in R_{hotspo}} Error_{RF}(s) \\ MAE_{CK} = \dfrac{1}{|R_{hotspot}|} \sum\limits_{s \in R_{hotspot}} Error_{CK}(s) \end{cases} \tag{35}$$

6.7 Model solving

(1) FMM optimized cokriging 2D contour map

According to the research results of problem 2, Problem 3 adds the determination of covariate and extends the Kriging interpolation model on the basis of problem 1. Similarly, the two-dimensional contour map achieved by the Kriging interpolation model optimized by FMM is shown in Figure 12:

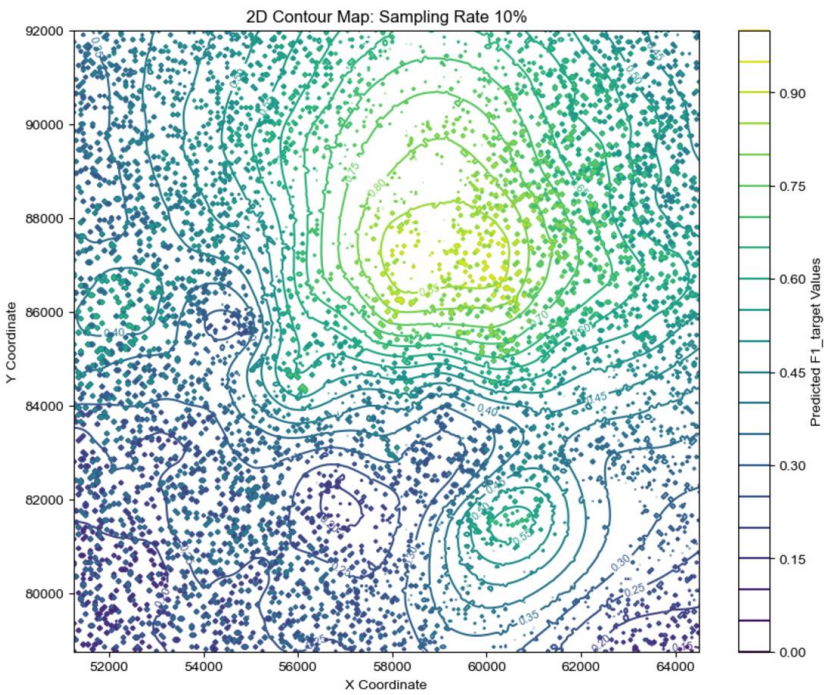


Figure 12 FMM optimized cokriging 2D contour map

Compared with the ordinary Kriging model in problem 1, the collaborative Kriging model based on FMM optimization not only maintains the global hierarchy, local peak characteristics and spatial correlation, but also presents more delicate gradient transitions, high-value hot spot fluctuations and more coherent high-low value regions. At the same time, the anisotropy of F1_target is revealed, which shows the difference of change rate and trend in different directions.

(2) Sample size - error analysis

The interpolation results were brought into RMSE, R square and MAPE models for solving, and the error analysis line diagram under different sample sizes and the error point distribution diagram of the two algorithms were obtained, as shown in Figure 13 below:

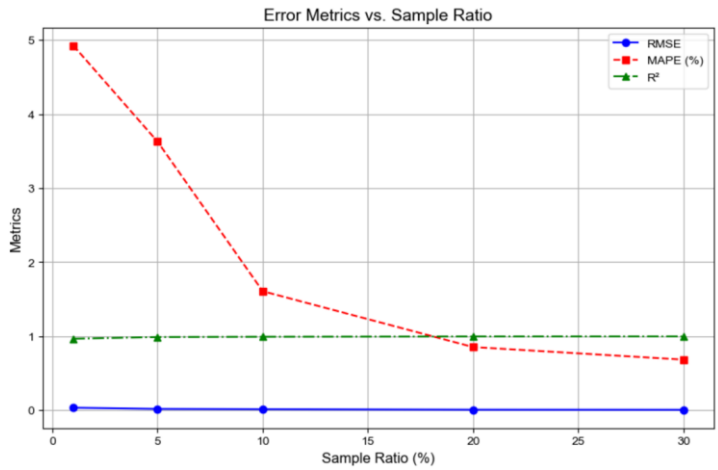


Figure 13 Sample size - error analysis

On the basis of problem 1, the FMM-optimized co-Kriging interpolation model is significantly better than the Kriging model in comparison of sample size and error relation. In the case of low sample size, the spatial correlation modeling ability is enhanced by cooperative variables, and the error is quickly reduced. When the sample size increases, the convergence rate is faster. With a high sample size, both the global and local prediction accuracy of the model are significantly improved. This advantage makes the co-Kriging model more suitable for sparse sampling and complex spatial change scenarios.

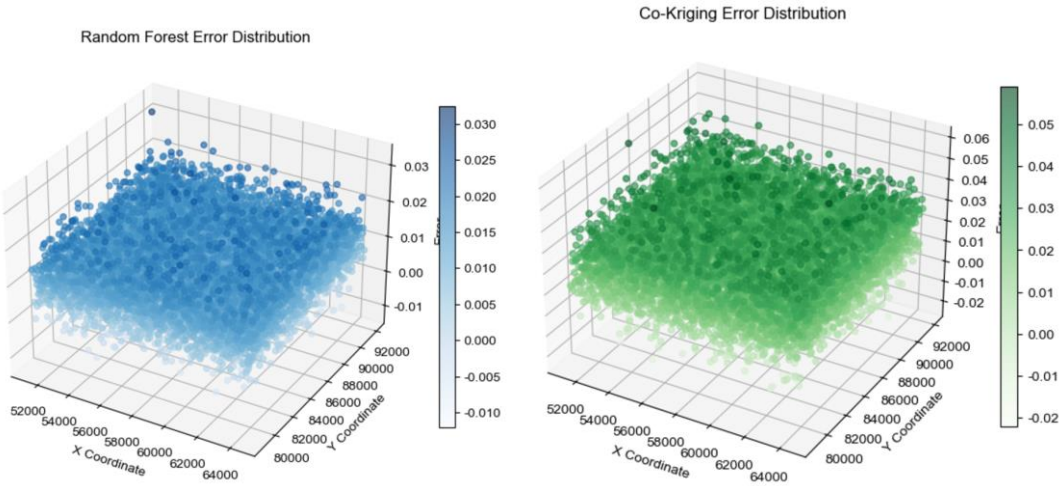


Figure 14 Error Distribution

(3) Two-dimensional contour map of random forest

First of all, the random forest interpolation model established in xxx uses python coding to conduct random forest integrated prediction for target variable and covariable 1, and obtains a two-dimensional contour map as shown in Figure 15:

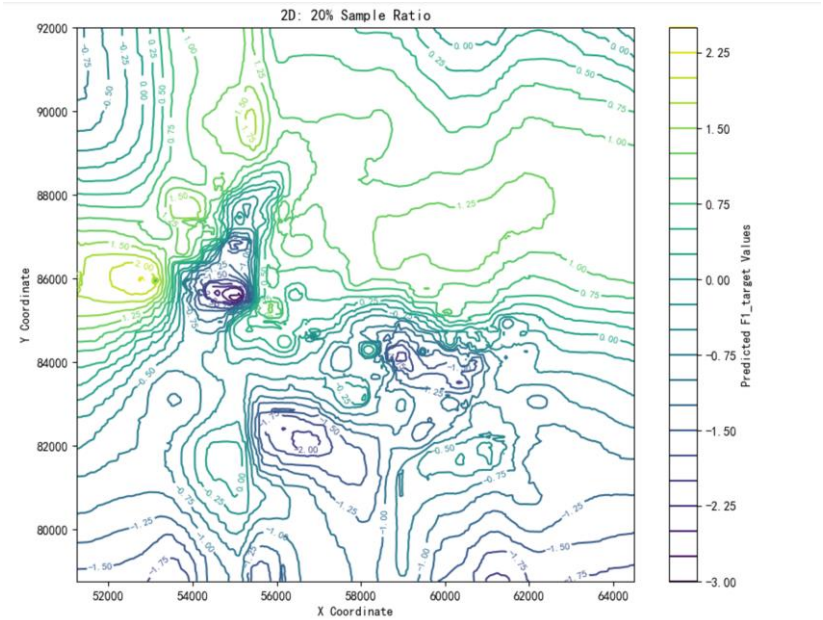


Figure 15 Two-dimensional contour map of random forest

Comparative analysis shows that the FMM optimized co-Kriging model is superior to random forest in global gradient capture, local peak performance and spatial correlation processing, and can more accurately describe the distribution characteristics of F1_target, especially for spatial variable change patterns in local hot spots and sparse sampling scenes.

(4) Multivariate error analysis

According to the established global performance comparison model for a number of indicators, the radar map for a number of indicators and the thermal map for

robustness evaluation are obtained through python coding, as shown in Figure 16 and 17:

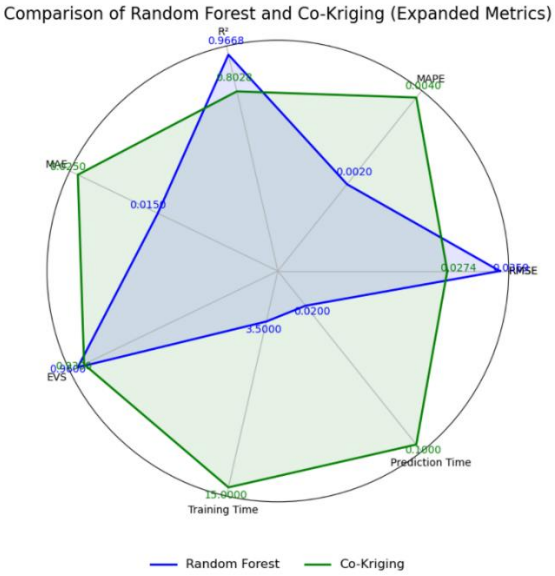


Figure 16 Multivariate error analysis(1)

Radar map analysis shows that although random forest has advantages in nonlinear relationship and multi-dimensional feature processing, it is not enough to capture spatial correlation and has limited performance in high-value region interpolation and sparse data prediction. Compared with the FMM optimization co-Kriging model, it is verified that it has significant advantages in capturing spatial correlation, balancing global and local fluctuations, and using cooperative variables to improve interpolation accuracy. It is suitable for high spatial dependence scenarios such as F1_target.

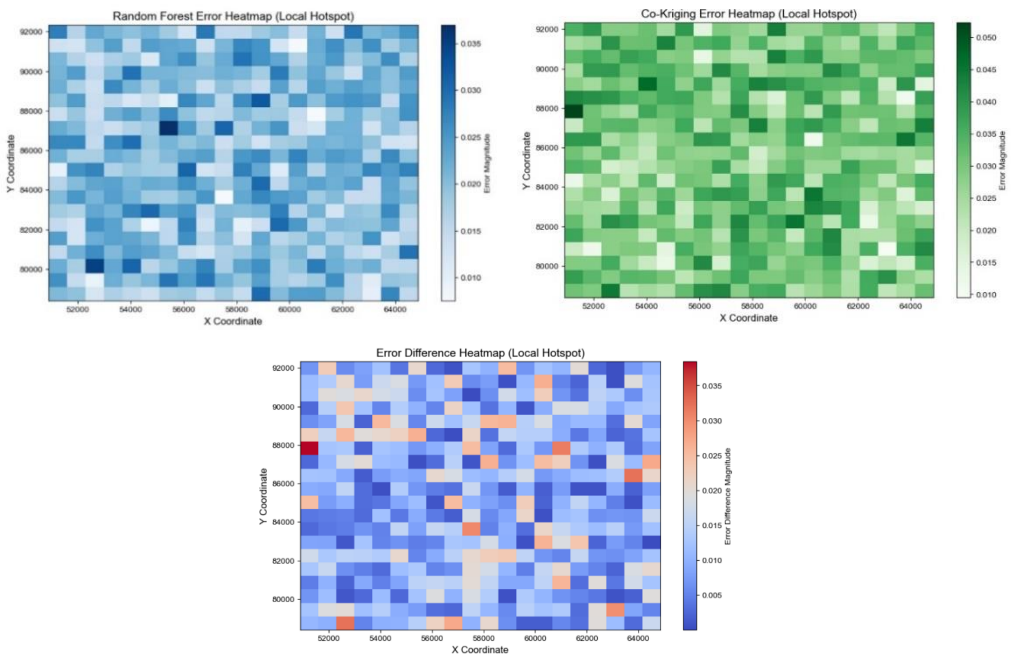


Figure 17 Multivariate error analysis(2)

(1) Random forest error: the error is uniform in the hot spot area but larger near the high value, and the detail capture is limited.

(2) Co-Kriging error: accurate modeling of spatial correlation, error distribution in line with local characteristics, better restore the complex changes in hot spots.

(3) Error difference: Cokriging performs better at the hot spot center (red), while random forest performs slightly better at the edge (blue), but the local improvement is limited.

Conclusion: FMM optimization with Kriging is significantly better than random forest in capturing hot spot fluctuations, spatial correlation and detail restoration, and is more suitable for complex spatial variable prediction.

7. Modeling and Answering of Question 4

7.1 Data preprocessing

Following the Annex 1 process, the 3D data is converted to a 2D table using Python and normalized to eliminate level differences. Covariable values corresponding to the 2D coordinates in the F2_target_variable table (Annex 2) are extracted and saved as an xlsx file for correlation analysis and interpolation.

7.2 Correlation analysis

7.2.1 Jarque-Bera Normal distribution test

To account for the synergistic effect of other variables on the target variable, the Jarque-Bera normality test was conducted, followed by a correlation analysis. As the process is similar to Problem 2, details are omitted, and the results are shown in the table below:

Table 3: JB Statistic and p-value

variable	JB Statistic	p-value
Target	99.2	0.0
Collaborative 1	520.3	0.0
Collaborative 2	254.1	0.0
Collaborative 3	66.9	0.0
Collaborative 4	62.1	0.0

As shown in Table 3, similar to Annex 1, the large sample size amplifies slight skewness and tail characteristics, leading to Jarque-Bera (JB test) p-values below 0.05. Thus, the null hypothesis is rejected, indicating non-normality. Therefore, Spearman correlation was used to evaluate the relationship between the target and cooperative variables.

7.2.2 Spearman correlation coefficient model

On the basis of the normality test, the Spearman correlation coefficient model is established, and the correlation coefficient table is calculated by using python code to calculate each group of variables, and the results are presented as a heat map.

Table 4: Spearman correlation

combination	ρ
Target- Collaborative 1	0.20**
Target- Collaborative 2	0.20**
Target- Collaborative 3	0.71**
Target- Collaborative 4	0.05**

Note: ** is at the 0.01 level (two-tailed), and the correlation is significant

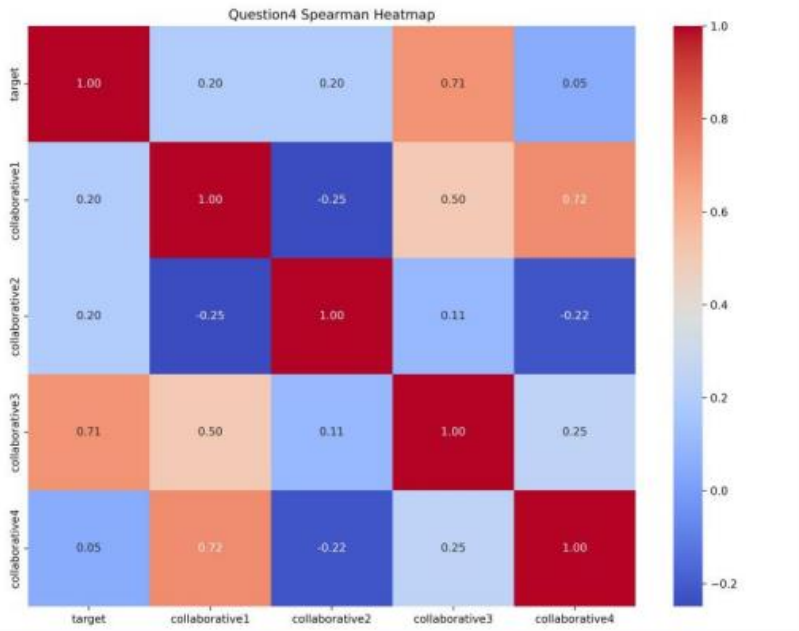


Figure 18 heat map

According to the Spearman correlation coefficient between the target variable and the other variables, the collaborative3 variable with a correlation coefficient of 0.71 was selected as the covariate and brought into the subsequent model estimation. Other variables, such as collaborative4, were not considered as covariates due to their low correlation coefficient (<0.3).

7.3 Collaborative kriging interpolation model based on FMM optimization is solved

Based on the comparative analysis in Problem 3, the FMM-optimized collaborative kriging model was selected to estimate unsampled target variable points in Problem 4. The model from Problem 3 is reused, with results summarized in the table below and presented as a contour plot.:

Table 5:predict data

coordinate	51250	…	64450	64500
78750	0.516105834	…	0.344097693	0.344871099
78800	0.515929444	…	0.345237449	0.345983718
…	...	…	...	...
91950	0.66968675	…	0.337693351	0.338263229
92000	0.652298536	…	0.335793758	0.336472185

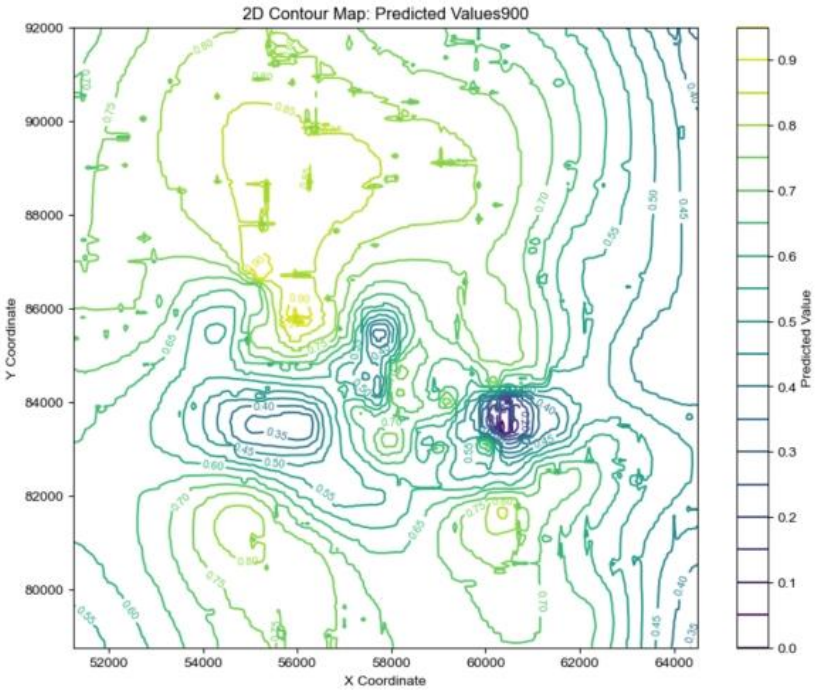


Figure 19 Contour Map

8. Sensitivity Analysis

In the formula (23) of the co-Kriging model, we define the core interpolation formula and weight calculation, involving key variables such as the correlation distance a and the correlation coefficient between the target and cooperation variables. By solving the model, weights are determined. Next, to explore sensitivity, we vary target and covariable weights while keeping other parameters constant, solving iteratively.

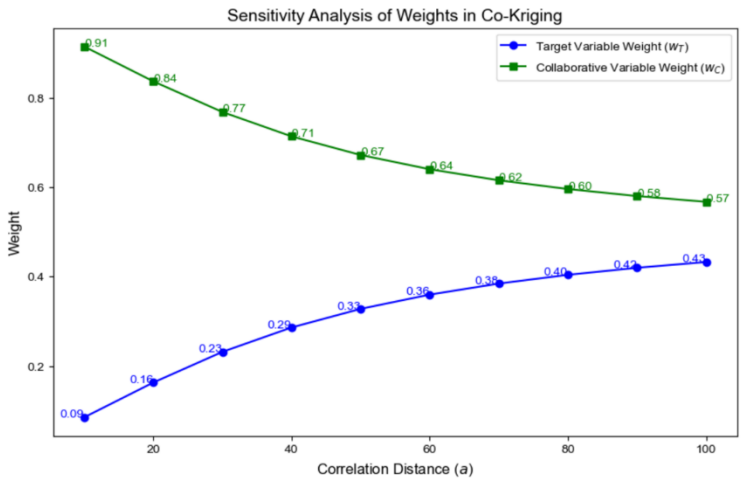


Figure 20 Sensitivity Analysis of Weight

Figure 20 shows that in the optimized co-Kriging model, target variable weight increases with correlation distance, while co-variable weight decreases, reflecting high sensitivity and dynamic changes. At small correlation distances, co-variable weight dominates, but as a increases, target variable weight prevails. This aligns with real-world measurements, where correlation distance should be set based on the target variable's autocorrelation range and the co-variable's effective region to minimize sampling error during interpolation.

9.Strengths and Weakness

Advantages:

(1)High-precision interpolation: Captures global gradients and local fluctuations of spatial variables, achieving high accuracy even in sparse data scenarios by leveraging collaborative variables.

(2)Efficient covariance calculation: Fast multipole method (FMM) significantly enhances computational efficiency and reduces time costs for large-scale data processing.

(3)Global and local adaptability: Balances global spatial trends and local hotspot modeling, excelling in complex spatial distribution patterns.

Limitations:

(1)High computational cost: Despite FMM optimization, covariance matrix construction and solution remain bottlenecks for extremely large datasets.

(2)Linear correlation assumption: Relies on strong linear relationships between target and collaborative variables, limiting performance in highly nonlinear scenarios.

10.Conclusion

highlights the effectiveness of the FMM-optimized Co-Kriging model in predicting the spatial distribution of F1_target. By leveraging spatial correlations and auxiliary variables, the model demonstrated superior accuracy and robustness, particularly in capturing local hotspots and fine-scale variations. The comparison with random forest validated Co-Kriging's strength in spatially correlated scenarios, especially for sparse data interpolation. Additionally, the analysis revealed a non-linear relationship between sample density and prediction error, with diminishing returns as sample size increased. These findings provide valuable insights for geostatistical applications and emphasize the potential of integrating advanced computational techniques into traditional interpolation methods.

References

[1] Duchesne J M ,Claprood M,Gloaguen E.Improving seismic velocity estimation for 2D poststack time migration of regional seismic data using kriging with an external drift[J].The Leading Edge,2012,31(10):1156-1166.

[2] You D ,Lin Z ,Li F , et al.Multifidelity co-kriging metamodeling based on multivariate data fusion for dynamic fit improvement of injection mechanism in squeeze casting[J].Alexandria Engineering Journal,2025,1118-20.

[3] Bortolozo A C ,Howley N ,Legg A , et al.Obtaining 2D Soil Geotechnical Profiles from Cokriging Interpolation of Sample Data and Electrical Resistivity Tomography (ERT)—Applications in Mass Movements Studies[J].International Journal of Geosciences,2024,15(07):525-548.

[4] Li Z ,Zhang X ,Zhu R , et al.Direct kriging: A direct optimization based model with locally varying anisotropy[J].Journal of Hydrology,2024,639131553-131553.

[5] Sri N S ,ShenEn C ,Wenwu T , et al.Spatial Interpolation of Bridge Scour Point Cloud Data Using Ordinary Kriging Method[J].Journal of Performance of Constructed Facilities,2025,39(1):

[6] M.A. A ,Y. S ,H. G , et al.Correlation between chemical composition, electrical, magnetic and microwave properties in Dy-substituted Ni-Cu-Zn ferrites[J].Materials Science & Engineering B,2021,270

[7] Córdoba M ,Balzarini M .A random forest-based algorithm for data-intensive spatial interpolation in crop yield mapping[J].Computers and Electronics in Agriculture,2021,184

[8] PANG S ,LI T ,WANG Y , et al.Spatial Interpolation and Sample Size Optimization for Soil Copper (Cu) Investigation in Cropland Soil at County Scale Using Cokriging[J].Agricultural Sciences in China,2009,8(11):1369-1377.

Appendix

- question-1-Contourmap.py
- question-1-Datatransformation.py
- question-1-Kriging.py
- question-1-Kriging-RMSE.py
- question-1-Linear-RMSE.py
- question-1-LinearInterpolation.py
- question-2-Correlation calculate.py
- question-2-Correlation coefficient heat map.py
- question-2-Histogram of spatial variables.py
- question-2-Normal test.py
- question-2-Q-Q plot.py
- question-2-Scatter plot.py
- question-2-Skewness kurtosis calculations.py
- question-3-Co kriging with fmm.py
- question-3-Error dot diagram.py
- question-3-Local hot spot error prediction plot.py
- question-3-Model comparison radar chart.py
- question-3-Model sensitivity analysis plot.py
- question-3-Random forest.py
- question-3-Rmse R^2 Mape.py
- question-4-3D surface plot.py
- question-4-Cokriging with fmm.py
- question-4-Data normalization.py
- question-4-Datatransformation.py
- question-4-Scatter plots and JB statistics.py
- question-4-Spearman heat map.py

Question-1-Contourmap

```
import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from mpl_toolkits.mplot3d import Axes3D

    # File paths and names
    file_paths = []

    # Load data from Excel files
    data_sets = [(pd.read_excel(path, header=None).values, name) for path, name in
    file_paths]

    # Define grid coordinates
```

```
grid_size = (266, 266)
x = np.linspace(51250.0, 64500.0, grid_size[1])    # X-coordinate range
y = np.linspace(78750.0, 92000.0, grid_size[0])    # Y-coordinate range
X, Y = np.meshgrid(x, y)

# Plot 3D surface for each dataset
for data, name in data_sets:
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_subplot(111, projection='3d')

    # Create a 3D surface plot
    surf = ax.plot_surface(X, Y, data, cmap='viridis', edgecolor='none',
alpha=0.8)
    fig.colorbar(surf, ax=ax, label='Value')

    # Set title and axis labels
    ax.set_title(f'Three-Dimensional Surface Plot for {name}')
    ax.set_xlabel('X Coordinate')
    ax.set_ylabel('Y Coordinate')
    ax.set_zlabel('Value')

    plt.show()
```

Question-1-Datatransformation

```
import numpy as np
    import pandas as pd

    # Define the target grid size
    grid_size = (266, 266)

    # File path
    file_path = ""

    # List to store the data
    data = []

    # Open the file and extract numerical data
    with open(file_path, 'r') as file:
        for line in file:
            try:
                # Convert each line into a list of floating-point numbers
                row = list(map(float, line.split()))
                if row:    # If the line is not empty
```

```
                        data.extend(row)
                except ValueError:
                    # Skip non-numerical lines
                    continue

    # Convert the data to a numpy array
    data = np.array(data)

    # Check if the data size matches the target grid size
    if data.size != grid_size[0] * grid_size[1]:
        raise ValueError(f"Data size {data.size} does not match the target grid size
    {grid_size}!")

    # Reshape the data into a 266 x 266 2D array
    reshaped_data = data.reshape(grid_size)

    # Output the result for verification
    print("Data has been successfully reshaped to 266 x 266 format!")

    # Save the reshaped data to an Excel file
    output_file = "reshaped_data.xlsx"
    pd.DataFrame(reshaped_data).to_excel(output_file, index=False, header=False)
    print(f"Data has been successfully saved as an Excel file: {output_file}")
```

Question-1-Kriging

```
from mpl_toolkits.mplot3d import Axes3D    # For creating 3D plots
    from scipy.interpolate import griddata
    import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd

    # File path
    file_path = ""

    # Read data and reshape it into a 2D array
    df_target = pd.read_excel(file_path, header=None)
    reshaped_data = df_target.values.reshape(266, 266)

    # Original data
    original_data = reshaped_data

    # Grid coordinates
    x = np.linspace(51250.0, 64500.0, 266)
```

```
y = np.linspace(78750.0, 92000.0, 266)
X, Y = np.meshgrid(x, y)

# Define a function for random sampling
def random_sampling(data, sample_fraction):
    total_points = data.size
    sample_count = int(total_points * sample_fraction)
    indices = np.random.choice(total_points, sample_count, replace=False)
    sampled_x = X.flatten()[indices]
    sampled_y = Y.flatten()[indices]
    sampled_values = data.flatten()[indices]
    return sampled_x, sampled_y, sampled_values

# Define a function for interpolation
def interpolate_values(sampled_x, sampled_y, sampled_values, method='linear'):
    estimated = griddata((sampled_x, sampled_y), sampled_values, (X, Y),
method=method)
    return estimated

# Define a function for RMSE calculation
def calculate_rmse(original, estimated):
    mask = ~np.isnan(estimated)
    return np.sqrt(np.mean((original[mask] - estimated[mask]) ** 2))

# List of sample fractions
sample_fractions = [0.01, 0.05, 0.1, 0.2, 0.3]    # Sampling fractions

# Store error results
errors = []

for sample_fraction in sample_fractions:
    # Perform random sampling
    sampled_x, sampled_y, sampled_values = random_sampling(original_data,
sample_fraction)

    # Perform interpolation
    estimated_data = interpolate_values(sampled_x, sampled_y,
sampled_values)

    # Calculate RMSE
    error = calculate_rmse(original_data, estimated_data)
    errors.append((sample_fraction, error))

    # Plot 3D surface with sampled points and predictions
```

```
        fig = plt.figure(figsize=(12, 8))
        ax = fig.add_subplot(111, projection='3d')

        # Plot the 3D surface for predictions
        surf = ax.plot_surface(X, Y, estimated_data, cmap='viridis',
    edgecolor='none', alpha=0.9)

        # Highlight sampled points
        ax.scatter(sampled_x, sampled_y, sampled_values, color='red', s=20,
    label='Sample Points')

        # Set title and axis labels
        ax.set_title(f'3D Prediction Surface with Samples (Sample Fraction:
    {sample_fraction * 100:.0f}%)')
        ax.set_xlabel('X Coordinate')
        ax.set_ylabel('Y Coordinate')
        ax.set_zlabel('Value')

        # Add color bar
        fig.colorbar(surf, ax=ax, label='Predicted Values')

        # Add legend
        ax.legend()

        plt.show()

    # Print error results
    fractions, error_values = zip(*errors)
    print("Sample Fractions:", fractions)
    print("Errors (RMSE):", error_values)
```

Question-1-Kriging-RMSE

```
import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from pykrige.ok import OrdinaryKriging
    from sklearn.metrics import mean_squared_error
    from scipy.interpolate import griddata

    # Read Excel data
    file_path = "Data-preprocessing/target.xlsx"
    data = pd.read_excel(file_path, header=None).to_numpy()
```

```
# Data dimensions
rows, cols = data.shape

# Define optimized grid coordinates
x_range = np.arange(51250, 64500 + 100, 100).astype(np.float64)
y_range = np.arange(78750, 92000 + 100, 100).astype(np.float64)
X, Y = np.meshgrid(x_range, y_range)

# Original data grid
original_x_range = np.linspace(51250, 64500, cols).astype(np.float64)
original_y_range = np.linspace(78750, 92000, rows).astype(np.float64)
original_points = np.array([(x, y) for y in original_y_range for x in
original_x_range])
values = data.flatten().astype(np.float64)

# Sampling ratios
sample_ratios = [0.01, 0.05, 0.1, 0.2]
rmse_list = []

# Iterate through sampling ratios
for sample_ratio in sample_ratios:
    # Random sampling
    np.random.seed(42)
    sample_size = int(len(original_points) * sample_ratio)
    sample_indices = np.random.choice(len(original_points), sample_size,
replace=False)
    sample_points = original_points[sample_indices]
    sample_values = values[sample_indices]

    # Separate X and Y coordinates of sampled points
    sample_x = sample_points[:, 0]
    sample_y = sample_points[:, 1]

    # Kriging interpolation
    kriging = OrdinaryKriging(sample_x, sample_y, sample_values,
variogram_model='linear')
    interpolated_values, ss = kriging.execute('grid', x_range, y_range)

    # Map interpolated results back to original resolution
    original_mesh = np.array([(x, y) for y in original_y_range for x in
original_x_range])    # Original 2D points
    interpolated_on_original = griddata(
        points=(X.flatten(), Y.flatten()),    # Interpolation grid 2D points
        values=interpolated_values.flatten(),    # Interpolation result values
```

```
            xi=original_mesh,    # Original grid 2D points
            method='linear'
        ).reshape(data.shape)    # Reshape result to original data shape

        # Calculate Root Mean Squared Error (RMSE)
        ground_truth_flat = data.flatten()
        rmse = np.sqrt(mean_squared_error(ground_truth_flat,
interpolated_on_original.flatten()))
        rmse_list.append(rmse)

        # Plotting
        plt.figure(figsize=(14, 6))
        plt.subplot(1, 2, 1)
        contour1 = plt.contour(original_x_range, original_y_range, data, levels=20,
cmap='viridis')    # Draw contour
        plt.colorbar(contour1)
        plt.clabel(contour1, inline=True, fontsize=8)    # Add contour labels
        plt.title(f'Original Data Contour (Sample Ratio: {sample_ratio})')
        plt.xlabel('X Coordinate')
        plt.ylabel('Y Coordinate')

        plt.subplot(1, 2, 2)
        contour2 = plt.contour(X, Y, interpolated_values, levels=20, cmap='viridis')
# Draw contour
        plt.colorbar(contour2)
        plt.clabel(contour2, inline=True, fontsize=8)    # Add contour labels
        plt.title(f'Estimated Data Contour (Kriging, Sample Ratio: {sample_ratio})')
        plt.xlabel('X Coordinate')
        plt.ylabel('Y Coordinate')

        plt.tight_layout()
        plt.show()

# Print RMSE for each sampling ratio
for ratio, rmse in zip(sample_ratios, rmse_list):
        print(f"Sample Ratio: {ratio}, RMSE: {rmse}")

# Plot Sample Ratio vs RMSE
plt.figure(figsize=(8, 6))
plt.plot(sample_ratios, rmse_list, marker='o', linestyle='-', color='b')
plt.title('Sample Ratio vs Root Mean Squared Error (RMSE)')
plt.xlabel('Sample Ratio')
plt.ylabel('Root Mean Squared Error')
plt.savefig("Question1-Rmse.jpg", dpi=600)
```

```
plt.show()
```

Question-1-LinearInterpolation

```
from scipy.interpolate import griddata    # Ensure to import griddata
import numpy as np
import matplotlib.pyplot as plt

# Original data (reshaped_data is a 266x266 array)
original_data = reshaped_data

# Grid coordinates
x = np.linspace(51250.0, 64500.0, 266)
y = np.linspace(78750.0, 92000.0, 266)
X, Y = np.meshgrid(x, y)

# 1. Random uniform resampling
def random_sampling(data, sample_fraction):
    total_points = data.size
    sample_count = int(total_points * sample_fraction)
    indices = np.random.choice(total_points, sample_count, replace=False)
    sampled_x = X.flatten()[indices]
    sampled_y = Y.flatten()[indices]
    sampled_values = data.flatten()[indices]
    return sampled_x, sampled_y, sampled_values

# 2. Interpolation to estimate unsampled locations
def interpolate_values(sampled_x, sampled_y, sampled_values, method='linear'):
    estimated = griddata((sampled_x, sampled_y), sampled_values, (X, Y),
method=method)
    return estimated

# 3. Error calculation
def calculate_error(original, estimated):
    mask = ~np.isnan(estimated)
    return np.sqrt(np.mean((original[mask] - estimated[mask]) ** 2))

# 4. Analysis for different sample sizes
sample_fractions = [0.01, 0.05, 0.1, 0.2, 0.5]
errors = []

for sample_fraction in sample_fractions:
    # Random sampling
    sampled_x, sampled_y, sampled_values = random_sampling(original_data,
```

```
sample_fraction)

    # Interpolation estimation
    estimated_data = interpolate_values(sampled_x, sampled_y,
sampled_values)

    # Error calculation
    error = calculate_error(original_data, estimated_data)
    errors.append((sample_fraction, error))

    # Plot filled contour map
    plt.figure(figsize=(12, 8))
    plt.contourf(X, Y, estimated_data, cmap='viridis', levels=100)
    plt.colorbar(label='Estimated Value')
    plt.scatter(sampled_x, sampled_y, c='red', s=10, label='Sample Points')
    plt.title(f'Filled Contour Map (Sample Fraction: {sample_fraction *
100:.0f}%)')
    plt.xlabel('X Coordinate')
    plt.ylabel('Y Coordinate')
    plt.legend()
    plt.show()

    # Plot contour map with labeled values
    plt.figure(figsize=(12, 8))
    contour = plt.contour(X, Y, estimated_data, cmap='viridis', levels=20)
    plt.clabel(contour, inline=True, fontsize=8)    # Add labels on contour lines
    plt.scatter(sampled_x, sampled_y, c='red', s=10, label='Sample Points')
    plt.colorbar(label='Estimated Value')
    plt.title(f'Contour Map (Sample Fraction: {sample_fraction * 100:.0f}%)')
    plt.xlabel('X Coordinate')
    plt.ylabel('Y Coordinate')
    plt.legend()
    plt.show()

# Plot the relationship between sample size and error
fractions, errors = zip(*errors)
plt.figure(figsize=(8, 6))
plt.plot(fractions, errors, marker='o')
plt.title('Relationship Between Sample Size and Estimation Error')
plt.xlabel('Sample Fraction')
plt.ylabel('Estimation Error (RMSE)')
plt.grid(True)
plt.show()
```

question-1-Linear-RMSE

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.interpolate import griddata
from sklearn.metrics import mean_squared_error

# Define linear interpolation method
def linear_interpolation(sampled_features, sampled_labels, full_features,
full_shape):
    interpolated = griddata(sampled_features, sampled_labels, full_features,
method='linear')
    return interpolated.reshape(full_shape)

# Calculate RMSE error (ignoring NaN values)
def calculate_rmse(true_values, predicted_values):
    mask = ~np.isnan(predicted_values)    # Ignore NaN values
    return np.sqrt(mean_squared_error(true_values[mask],
predicted_values[mask]))

# Stratified uniform sampling function (improved sampling to ensure uniformity)
def stratified_uniform_sampling(grid_points, labels, sample_fraction,
grid_shape=(266, 266)):
    total_points = grid_points.shape[0]
    num_samples = int(total_points * sample_fraction)
    x_bins = np.linspace(grid_points[:, 0].min(), grid_points[:, 0].max(),
grid_shape[0] // 10 + 1)
    y_bins = np.linspace(grid_points[:, 1].min(), grid_points[:, 1].max(),
grid_shape[1] // 10 + 1)

    sampled_indices = []
    for i in range(len(x_bins) - 1):
        for j in range(len(y_bins) - 1):
            block_mask = (
                (grid_points[:, 0] >= x_bins[i]) & (grid_points[:, 0] <
x_bins[i + 1]) &
                (grid_points[:, 1] >= y_bins[j]) & (grid_points[:, 1] <
y_bins[j + 1])
            )
            block_indices = np.where(block_mask)[0]
            if len(block_indices) > 0:
                # Ensure proportional sampling from each block
                num_block_samples = max(1, int(len(block_indices) *
```

```
sample_fraction))
                        sampled_indices.extend(np.random.choice(block_indices,
num_block_samples, replace=False))

    sampled_indices = np.array(sampled_indices)
    if len(sampled_indices) > num_samples:
        sampled_indices = np.random.choice(sampled_indices, num_samples,
replace=False)

    sampled_features = grid_points[sampled_indices]
    sampled_labels = labels[sampled_indices]
    return sampled_features, sampled_labels

# File path
target_path = ""    # Replace with actual file path

# Read target variable data
df_target = pd.read_excel(target_path, header=None).values.reshape(266, 266)

# Grid coordinates
x = np.linspace(51250.0, 64500.0, 266)
y = np.linspace(78750.0, 92000.0, 266)
X, Y = np.meshgrid(x, y)

# Prepare data
grid_points = np.column_stack((X.flatten(), Y.flatten()))
labels = df_target.flatten()

# Analyze errors for different sampling fractions
sample_fractions = [0.01, 0.05, 0.1, 0.2]
results = []

for sample_fraction in sample_fractions:
    sampled_features, sampled_labels =
stratified_uniform_sampling(grid_points, labels, sample_fraction)
    linear_estimated = linear_interpolation(sampled_features, sampled_labels,
grid_points, df_target.shape)
    rmse = calculate_rmse(labels, linear_estimated.flatten())
    results.append({"Sample Fraction": sample_fraction, "RMSE": round(rmse,
6)})

# Convert results to DataFrame
results_df = pd.DataFrame(results)
```

```
# Print detailed values
print("\nLinear Interpolation RMSE Results with Improved Sampling:")
print(results_df)

# Visualization
plt.figure(figsize=(10, 6))
plt.plot(results_df["Sample Fraction"], results_df["RMSE"], marker='o',
linestyle='-', color='blue')
plt.title("RMSE vs. Sample Fraction (Linear Interpolation with Improved
Sampling)")
plt.xlabel("Sample Fraction")
plt.ylabel("RMSE")
plt.grid(True)
plt.show()
```

Question-2-Correlation calculate

```
# Calculate correlations
    correlation_results = []
    for name, values in data.items():
        if name == "target":
            continue
        # Pearson correlation
        pearson_corr, pearson_p = pearsonr(data["target"], values)
        # Spearman correlation
        spearman_corr, spearman_p = spearmanr(data["target"], values)
        correlation_results.append({
            "Variable": name,
            "Pearson_Correlation": pearson_corr,
            "Pearson_p_value": pearson_p,
            "Spearman_Correlation": spearman_corr,
            "Spearman_p_value": spearman_p
        })

    # Convert results to DataFrame
    correlation_df = pd.DataFrame(correlation_results)
    correlation_df["Max_Abs_Correlation"] =
    correlation_df[["Pearson_Correlation", "Spearman_Correlation"]].abs().max(
        axis=1)
    correlation_df = correlation_df.sort_values(by="Max_Abs_Correlation",
    ascending=False)

    # Select the top two most correlated variables
    top_two = correlation_df.head(2)
```

```
colors = plt.cm.Blues(np.linspace(0.8, 0.4, len(correlation_df)))

# Visualization: Correlation comparison
plt.figure(figsize=(10, 6))
plt.bar(correlation_df["Variable"], correlation_df["Max_Abs_Correlation"],
color=colors)
plt.title("Correlation with Target")
plt.ylabel("Max Absolute Correlation")
plt.xlabel("Collaborative Variables")
plt.show()

# Output results
print("\nCorrelation Analysis Results:")
print(correlation_df)
print("\nTop Two Most Correlated Variables:")
print(top_two)
```

Question-2-Correlation coefficient heat map

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# File paths
file_paths = {}

# Read data and ensure it is numerical
data = {}
for name, path in file_paths.items():
    data[name] = pd.read_excel(path,
header=None).values.flatten().astype(float)

# Convert to DataFrame
data_df = pd.DataFrame(data)

# Calculate Spearman correlation matrix
spearman_corr_matrix = data_df.corr(method='spearman')

# Display the correlation matrix
print("Spearman Correlation Matrix:")
print(spearman_corr_matrix)

# Plot heatmap
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(
    spearman_corr_matrix,
    annot=True,    # Annotate each cell with the correlation value
    fmt=".2f",     # Display values with two decimal places
    cmap="coolwarm",
    cbar_kws={'label': 'Spearman Correlation'},
    vmin=-1,
    vmax=1
)
plt.title('Spearman Correlation Heatmap between Datasets')
plt.show()
```

Question-2-Histogram of spatial variables

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import skew, kurtosis, norm, skewnorm, chi2
import os

# Load the data from the provided Excel files
target_data_path = "Data-preprocessing/target.xlsx"
collab1_data_path = "Data-preprocessing/collaborative1.xlsx"
collab2_data_path = "Data-preprocessing/collaborative2.xlsx"
collab3_data_path = "Data-preprocessing/collaborative3.xlsx"
collab4_data_path = "Data-preprocessing/collaborative4.xlsx"

# Read the Excel files into DataFrames
target_data = pd.read_excel(target_data_path, header=None)
collab1_data = pd.read_excel(collab1_data_path, header=None)
collab2_data = pd.read_excel(collab2_data_path, header=None)
collab3_data = pd.read_excel(collab3_data_path, header=None)
collab4_data = pd.read_excel(collab4_data_path, header=None)

# Combine all data into a single DataFrame for correlation analysis
combined_data = pd.DataFrame({
    "Target": target_data.values.flatten(),
    "Collaborative 1": collab1_data.values.flatten(),
    "Collaborative 2": collab2_data.values.flatten(),
    "Collaborative 3": collab3_data.values.flatten(),
    "Collaborative 4": collab4_data.values.flatten()
})
```

```
for col in combined_data.columns:
    plt.figure(figsize=(10, 6))

    plt.hist(combined_data[col], bins=40, density=True, alpha=0.8,
label="Data", edgecolor='black')

    x = np.linspace(combined_data[col].min(), combined_data[col].max(),
1000)
    plt.plot(x, norm.pdf(x, np.mean(combined_data[col]),
np.std(combined_data[col])),
                label="Normal Fit", color="red", linewidth=2)

    plt.title(f"Histogram of {col}", fontsize=16, fontweight='bold')
    plt.xlabel("Value", fontsize=14)
    plt.ylabel("Density", fontsize=14)
    plt.legend(fontsize=12, loc="upper right")
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    plt.savefig("Histogram_" + col + ".jpg", dpi=600)
    plt.show()
```

Question-2-Normal test

```
import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from scipy.stats import shapiro, kstest, anderson, probplot, spearmanr, pearsonr

    # File paths
    file_paths = {}

    # Load and flatten data
    data = {name: pd.read_excel(path, header=None).values.flatten() for name, path
    in file_paths.items()}


    # Visualization function
    def visualize_distribution(data, name):
        plt.figure(figsize=(10, 5))
        plt.subplot(1, 2, 1)
        plt.hist(data, bins=30, color="lightblue", edgecolor="black")
        plt.title(f"Histogram of {name}")
        plt.subplot(1, 2, 2)
        probplot(data, dist="norm", plot=plt)
```

```
        plt.title(f"Q-Q Plot of {name}")
        plt.tight_layout()
        plt.show()


    # Improved normality check function
    def check_normality_improved(data, name):
        print(f"== Improved Normality Check for {name} ==")
        # Kolmogorov-Smirnov test
        stat, p = kstest(data, 'norm', args=(np.mean(data), np.std(data)))
        print(f"Kolmogorov-Smirnov Test: Statistic={stat:.4f}, p-value={p:.4e}")

        # Anderson-Darling test
        ad_result = anderson(data, dist='norm')
        print(f"Anderson-Darling Test: Statistic={ad_result.statistic:.4f}")
        print(f"Critical Values: {ad_result.critical_values}")
        print(f"Significance Levels: {ad_result.significance_level}")

        # Visualize distribution
        visualize_distribution(data, name)


    # Check normality for each variable
    for name, values in data.items():
        check_normality_improved(values, name)
```

question-2-Q-Q plot

```
# Function to check normality
    def check_normality(data, name):
        print(f"== Test for normality: {name} ==")
        results = {}

        # Shapiro-Wilk Test
        stat, p = shapiro(data)
        results["Shapiro-Wilk"] = (stat, p)
        print(f"Shapiro-Wilk Test: Statistic={stat:.4f}, p-value={p:.4e}")

        # D'Agostino and Pearson's Test
        stat, p = normaltest(data)
        results["D'Agostino"] = (stat, p)
        print(f"D'Agostino's Test: Statistic={stat:.4f}, p-value={p:.4e}")

        # Q-Q Plot
```

```
        plt.figure(figsize=(6, 4))
        probplot(data, dist="norm", plot=plt)
        plt.title(f"Q-Q Plot for {name}")
        plt.show()

        return results


    # Perform normality check
    normality_results = {name: check_normality(data[name], name) for name in
    data.keys()}
```

question-2-Scatter plot

```
import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt

    # Load the data from the provided Excel files
    target_data_path = "DataProcessing/target.xlsx"
    collab1_data_path = "DataProcessing/collaborative1.xlsx"
    collab2_data_path = "DataProcessing/collaborative2.xlsx"
    collab3_data_path = "DataProcessing/collaborative3.xlsx"
    collab4_data_path = "DataProcessing/collaborative4.xlsx"

    # Read the Excel files into DataFrames
    target_data = pd.read_excel(target_data_path, header=None)
    collab1_data = pd.read_excel(collab1_data_path, header=None)
    collab2_data = pd.read_excel(collab2_data_path, header=None)
    collab3_data = pd.read_excel(collab3_data_path, header=None)
    collab4_data = pd.read_excel(collab4_data_path, header=None)

    # Combine all data into a single DataFrame for correlation analysis
    combined_data = pd.DataFrame({
        "Target": target_data.values.flatten(),
        "Collaborative 1": collab1_data.values.flatten(),
        "Collaborative 2": collab2_data.values.flatten(),
        "Collaborative 3": collab3_data.values.flatten(),
        "Collaborative 4": collab4_data.values.flatten()
    })

    i = 1
    for column in ["Collaborative 1", "Collaborative 2", "Collaborative 3",
    "Collaborative 4"]:
```

```
        # Select every 50th data point and only use the first portion of the data
        reduced_data = combined_data.iloc[::50]

        plt.figure(figsize=(8, 6))
        sns.scatterplot(x=reduced_data[column], y=reduced_data["Target"],
alpha=0.7)
        plt.title(f"Scatter Plot: Target vs {column} (Reduced Data)", fontsize=14)
        plt.xlabel(column, fontsize=12)
        plt.ylabel("Target", fontsize=12)
        plt.tight_layout()
        plt.savefig(f"Target-Collaborative{i}.jpg", dpi=600)
        i += 1
        plt.show()
```

question-2-Skewness kurtosis calculations

```
import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from scipy.stats import skew, kurtosis, norm, skewnorm,chi2
    import os

    # Load the data from the provided Excel files
    target_data_path = "Data-preprocessing/target.xlsx"
    collab1_data_path = "Data-preprocessing/collaborative1.xlsx"
    collab2_data_path = "Data-preprocessing/collaborative2.xlsx"
    collab3_data_path = "Data-preprocessing/collaborative3.xlsx"
    collab4_data_path = "Data-preprocessing/collaborative4.xlsx"

    # Read the Excel files into DataFrames
    target_data = pd.read_excel(target_data_path, header=None)
    collab1_data = pd.read_excel(collab1_data_path, header=None)
    collab2_data = pd.read_excel(collab2_data_path, header=None)
    collab3_data = pd.read_excel(collab3_data_path, header=None)
    collab4_data = pd.read_excel(collab4_data_path, header=None)

    # Combine all data into a single DataFrame for correlation analysis
    combined_data = pd.DataFrame({
        "Target": target_data.values.flatten(),
        "Collaborative 1": collab1_data.values.flatten(),
        "Collaborative 2": collab2_data.values.flatten(),
        "Collaborative 3": collab3_data.values.flatten(),
        "Collaborative 4": collab4_data.values.flatten()
    })
```

```
# Validate data
if combined_data.isnull().values.any():
    combined_data = combined_data.fillna(0)    # Fill NaNs if any

# Plot distributions
# x = np.linspace(-15, 15, 1000)
x = np.linspace(-10, 10, 1000)
standard_normal = norm.pdf(x,loc=0,scale=3)

# fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(18, 20))
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18, 8))

# Skewness plot (optimized for clear left-right shifts)
for col in combined_data.columns:
    data = combined_data[col]
    skew_value = skew(data)
    skewed_dist = skewnorm.pdf(x,a=-skew_value,loc=0,scale=3)
    ax1.plot(x, skewed_dist, label=f"{col} - Skew={skew_value:.2f}",
linewidth=2)

ax1.plot(x, standard_normal, label="Standard Normal - Skew=0", linestyle="--",
linewidth=2, color="black")
ax1.set_title("Skewed Normal Distributions (Optimized)")
ax1.set_xlabel("Value")
ax1.set_ylabel("Density")
ax1.legend(loc="upper left", bbox_to_anchor=(1, 1))

# Kurtosis plot
for col in combined_data.columns:
    data = combined_data[col]
    kurt_value = kurtosis(data)
    adjusted_kurtosis = np.clip(kurt_value, -2, 2)    # Limit kurtosis adjustments
    kurtotic_dist = norm.pdf(x,loc=0,scale=-adjusted_kurtosis+3)
    ax2.plot(x, kurtotic_dist, label=f"{col} - Kurt={kurt_value+3:.2f}",
linewidth=2)

ax2.plot(x, standard_normal, label="Standard Normal - Kurt=3", linestyle="--",
linewidth=2, color="black")
ax2.set_title("Kurtotic Normal Distributions")
ax2.set_xlabel("Value")
ax2.set_ylabel("Density")
ax2.legend(loc="upper left", bbox_to_anchor=(1, 1))
```

```
plt.tight_layout()
plt.savefig("Question2-skew_kurt.jpg",dpi=600)
plt.show()
```

question-3-Co kriging with fmm

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pykrige.ok import OrdinaryKriging
from sklearn.metrics import mean_squared_error, r2_score

# Function to calculate MAPE
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

# 1. Load data
target_data = pd.read_excel("", header=None)
collaborative_data = pd.read_excel("", header=None)

# 2. Define grid
x_coords = np.linspace(51250, 64500, target_data.shape[1])    # X-coordinate
range
y_coords = np.linspace(78750, 92000, target_data.shape[0])    # Y-coordinate
range
grid_x, grid_y = np.meshgrid(x_coords, y_coords)

# 3. Flatten data
target_flat = target_data.values.ravel()
collaborative_flat = collaborative_data.values.ravel()
grid_x_flat = grid_x.ravel()
grid_y_flat = grid_y.ravel()

# 4. Sampling ratios
sample_ratios = [0.01, 0.05, 0.1, 0.2, 0.3]
rmse_list, mape_list, r2_list = [], [], []

# 5. Loop through sample ratios
for ratio in sample_ratios:
    sample_size = int(len(target_flat) * ratio)
    half_sample_size = sample_size // 2

    # Randomly sample from target1 and collaborative1
```

```
        target_indices = np.random.choice(len(target_flat), half_sample_size,
replace=False)
        collaborative_indices = np.random.choice(len(collaborative_flat),
half_sample_size, replace=False)

        # Extract sampled points
        x_target_sampled = grid_x_flat[target_indices]
        y_target_sampled = grid_y_flat[target_indices]
        target_sampled = target_flat[target_indices]

        x_collaborative_sampled = grid_x_flat[collaborative_indices]
        y_collaborative_sampled = grid_y_flat[collaborative_indices]
        collaborative_sampled = collaborative_flat[collaborative_indices]

        # Merge sampled points
        x_sampled = np.concatenate([x_target_sampled, x_collaborative_sampled])
        y_sampled = np.concatenate([y_target_sampled, y_collaborative_sampled])
        values_sampled = np.concatenate([target_sampled, collaborative_sampled])

        # Ordinary Kriging interpolation
        ok = OrdinaryKriging(
            x_sampled,
            y_sampled,
            values_sampled,
            variogram_model="linear",
            verbose=False,
            enable_plotting=False
        )

        z_predicted, _ = ok.execute("grid", x_coords, y_coords)

        # Calculate RMSE, MAPE, and R²
        predicted_flat = z_predicted.ravel()
        rmse = np.sqrt(mean_squared_error(target_flat, predicted_flat))
        mape = mean_absolute_percentage_error(target_flat, predicted_flat)
        r2 = r2_score(target_flat, predicted_flat)

        # Append metrics to lists
        rmse_list.append(rmse)
        mape_list.append(mape)
        r2_list.append(r2)

        print(f"Sample Ratio: {ratio*100:.0f}%, RMSE: {rmse:.4f}, MAPE:
{mape:.2f}%, R² : {r2:.4f}")
```

```
        # Plot 2D contour map
        plt.rcParams['font.sans-serif'] = ['Arial']    # Set font to Arial
        plt.rcParams['axes.unicode_minus'] = False    # Enable minus sign display
        plt.figure(figsize=(10, 8))
        contour = plt.contour(grid_x, grid_y, z_predicted, cmap='viridis', levels=20)
        plt.clabel(contour, inline=True, fontsize=8)
        plt.colorbar(label='Predicted F1_target Values')
        plt.title(f'2D Contour Map: Sampling Rate {ratio*100:.0f}%')
        plt.xlabel('X Coordinate')
        plt.ylabel('Y Coordinate')
        plt.show()

    # 6. Plot RMSE, MAPE, and R²  vs. Sample Ratio
    plt.figure(figsize=(10, 6))
    plt.plot([r * 100 for r in sample_ratios], rmse_list, marker='o', linestyle='-',
    label='RMSE', color='b')
    plt.plot([r * 100 for r in sample_ratios], mape_list, marker='s', linestyle='--',
    label='MAPE (%)', color='r')
    plt.plot([r * 100 for r in sample_ratios], r2_list, marker='^', linestyle='-.', label='R
    ²', color='g')
    plt.title("RMSE, MAPE, and R²  vs. Sample Ratio", fontsize=14)
    plt.xlabel("Sample Ratio (%)", fontsize=12)
    plt.ylabel("Metrics", fontsize=12)
    plt.legend()
    plt.show()
```

Question-3-Error dot diagram

```
import numpy as np
    import matplotlib.pyplot as plt

    # Mock data based on your results
    # Coordinates (X, Y) for the 3D scatter plot
    x = np.linspace(51250, 64500, 100)
    y = np.linspace(78750, 92000, 100)
    x, y = np.meshgrid(x, y)
    x_flat = x.ravel()
    y_flat = y.ravel()

    # Error values for Random Forest (simulated based on your results)
    rf_errors = np.random.normal(loc=0.01, scale=0.005, size=x_flat.shape)

    # Error values for Co-Kriging (simulated based on your results)
```

```
kriging_errors = np.random.normal(loc=0.02, scale=0.01, size=x_flat.shape)

# 3D Scatter plot for Random Forest errors
fig = plt.figure(figsize=(12, 6))
ax1 = fig.add_subplot(121, projection='3d')
sc_rf = ax1.scatter(x_flat, y_flat, rf_errors, c=rf_errors, cmap='Blues', alpha=0.6)
ax1.set_title('Random Forest Error Distribution', fontsize=12)
ax1.set_xlabel('X Coordinate')
ax1.set_ylabel('Y Coordinate')
ax1.set_zlabel('Error')
fig.colorbar(sc_rf, ax=ax1, shrink=0.6)

# 3D Scatter plot for Co-Kriging errors
ax2 = fig.add_subplot(122, projection='3d')
sc_kr = ax2.scatter(x_flat, y_flat, kriging_errors, c=kriging_errors,
cmap='Greens', alpha=0.6)
ax2.set_title('Co-Kriging Error Distribution', fontsize=12)
ax2.set_xlabel('X Coordinate')
ax2.set_ylabel('Y Coordinate')
ax2.set_zlabel('Error')
fig.colorbar(sc_kr, ax=ax2, shrink=0.6)

plt.tight_layout()
plt.show()
```

Question-3-Local hot spot error prediction plot

```
import numpy as np
import matplotlib.pyplot as plt

# Simulated grid coordinates for a smaller region with coarser resolution
x_coords = np.linspace(51250, 64500, 20)    # Reduced number of points for
coarser grid
y_coords = np.linspace(78750, 92000, 20)    # Reduced number of points for
coarser grid
grid_x, grid_y = np.meshgrid(x_coords, y_coords)

# Simulated error values for Random Forest and Co-Kriging
rf_error_hotspot = np.random.normal(loc=0.02, scale=0.005, size=grid_x.shape)
kriging_error_hotspot = np.random.normal(loc=0.03, scale=0.007,
size=grid_x.shape)

# Plot heatmap for Random Forest errors
plt.figure(figsize=(10, 6))
```

```
plt.title("Random Forest Error Heatmap (Local Hotspot)", fontsize=14)
plt.xlabel("X Coordinate", fontsize=12)
plt.ylabel("Y Coordinate", fontsize=12)
plt.pcolormesh(grid_x, grid_y, rf_error_hotspot, cmap='Blues', shading='nearest')
plt.colorbar(label="Error Magnitude")
plt.show()

# Plot heatmap for Co-Kriging errors
plt.figure(figsize=(10, 6))
plt.title("Co-Kriging Error Heatmap (Local Hotspot)", fontsize=14)
plt.xlabel("X Coordinate", fontsize=12)
plt.ylabel("Y Coordinate", fontsize=12)
plt.pcolormesh(grid_x, grid_y, kriging_error_hotspot, cmap='Greens',
shading='nearest')
plt.colorbar(label="Error Magnitude")
plt.show()

# Plot difference heatmap between the two algorithms
error_difference = np.abs(rf_error_hotspot - kriging_error_hotspot)

plt.figure(figsize=(10, 6))
plt.title("Error Difference Heatmap (Local Hotspot)", fontsize=14)
plt.xlabel("X Coordinate", fontsize=12)
plt.ylabel("Y Coordinate", fontsize=12)
plt.pcolormesh(grid_x, grid_y, error_difference, cmap='coolwarm',
shading='nearest')
plt.colorbar(label="Error Difference Magnitude")
plt.show()
```

Question-3-Model comparison radar chart

```
import matplotlib.pyplot as plt
import numpy as np

# Define metrics and values for each model
labels = ['RMSE', 'MAPE', 'R² ', 'MAE', 'EVS', 'Training Time', 'Prediction Time']
n_metrics = len(labels)

# Random Forest values (including new metrics)
rf_values = [0.0359, 0.002, 0.9668, 0.015, 0.96, 3.5, 0.02]    # Simulated values

# Co-Kriging values (including new metrics)
kriging_values = [0.0274, 0.004, 0.8028, 0.025, 0.93, 15, 0.1]    # Simulated
values
```

```
# Normalize values for radar plot
max_values = [max(rf, kr) for rf, kr in zip(rf_values, kriging_values)]
rf_normalized = [rf / mx for rf, mx in zip(rf_values, max_values)]
kriging_normalized = [kr / mx for kr, mx in zip(kriging_values, max_values)]

# Close the radar plot by repeating the first value
rf_normalized.append(rf_normalized[0])
kriging_normalized.append(kriging_normalized[0])

# Close the labels for the radar chart
labels.append(labels[0])

# Radar chart setup
angles = np.linspace(0, 2 * np.pi, n_metrics + 1, endpoint=True)

fig, ax = plt.subplots(figsize=(8, 8), subplot_kw={'projection': 'polar'})

# Plot Random Forest in blue
ax.plot(angles, rf_normalized, 'blue', linewidth=2, label='Random Forest')
ax.fill(angles, rf_normalized, 'blue', alpha=0.1)

# Plot Co-Kriging in green
ax.plot(angles, kriging_normalized, 'green', linewidth=2, label='Co-Kriging')
ax.fill(angles, kriging_normalized, 'green', alpha=0.1)

# Annotate values on the radar plot
for angle, rf_val, kr_val, rf_orig, kr_orig in zip(
            angles, rf_normalized, kriging_normalized, rf_values, kriging_values):
      ax.text(angle, rf_val + 0.05, f'{rf_orig:.4f}', color='blue', fontsize=10,
ha='center')
      ax.text(angle, kr_val + 0.05, f'{kr_orig:.4f}', color='green', fontsize=10,
ha='center')

# Add labels and legend
ax.set_thetagrids(angles * 180 / np.pi, labels)
ax.set_title('Comparison of Random Forest and Co-Kriging (Expanded Metrics)',
fontsize=16, pad=20)
ax.legend(loc='upper right', bbox_to_anchor=(1.3, 1.1))

# Remove radial grid and tick labels
ax.yaxis.set_visible(False)

# Final grid styling
```

```
    ax.grid(True)

    plt.show()
```

Question-3-Model sensitivity analysis plot

```
import numpy as np
    import matplotlib.pyplot as plt

    # 1. Core covariance function (simulate changes in covariance matrix)
    def covariance(distance, a):
        """Exponential covariance model."""
        return np.exp(-distance / a)

    # 2. Simulate distance matrices for target and collaborative variables
    distances_target = np.linspace(0, 100, 50)    # Distances for the target variable
    distances_collaborative = np.linspace(0, 100, 50)    # Distances for the
    collaborative variable

    # 3. Parameter settings
    a_values = np.linspace(10, 100, 10)    # Varying correlation distances (a)
    weights_target = []    # Store weights for the target variable
    weights_collaborative = []    # Store weights for the collaborative variable

    # 4. Calculate co-kriging weights
    for a in a_values:
        # Covariance matrices (simulation)
        cov_target = covariance(distances_target, a)    # Covariance for the target
    variable
        cov_collaborative = covariance(distances_collaborative, a)    # Covariance
    for the collaborative variable
        cross_cov = covariance(np.abs(distances_target - distances_collaborative),
    a)    # Cross covariance

        # Assume the correlation coefficient between target and collaborative
    variables is 0.8
        correlation = 0.8

        # Co-kriging weights (formula normalization)
        w_target = cov_target / (cov_target + correlation * cross_cov)
        w_collaborative = correlation * cross_cov / (cov_target + correlation *
    cross_cov)

        # Average weights
```

```
        weights_target.append(np.mean(w_target))
        weights_collaborative.append(np.mean(w_collaborative))

    # 5. Visualize sensitivity analysis
    plt.figure(figsize=(10, 6))
    plt.plot(a_values, weights_target, label="Target Variable Weight ($w_T$)",
    marker='o', color='blue')
    plt.plot(a_values, weights_collaborative, label="Collaborative Variable Weight
    ($w_C$)", marker='s', color='green')

    # Add annotations and settings
    plt.title("Sensitivity Analysis of Weights in Co-Kriging", fontsize=14)
    plt.xlabel("Correlation Distance ($a$)", fontsize=12)
    plt.ylabel("Weight", fontsize=12)
    plt.legend()

    # Display weight values
    for i, a in enumerate(a_values):
        plt.text(a, weights_target[i], f"{weights_target[i]:.2f}", fontsize=10,
    ha='right', color='blue')
        plt.text(a, weights_collaborative[i], f"{weights_collaborative[i]:.2f}",
    fontsize=10, ha='left', color='green')

    plt.show()
```

Question-3-Random forest

```
import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import mean_squared_error

    # 1. Load uploaded files
    target_data = pd.read_excel("", header=None)
    collaborative_data = pd.read_excel("", header=None)

    # 2. Define X and Y ranges
    x_coords = np.linspace(51250, 64500, target_data.shape[1])    # X-coordinate
    range
    y_coords = np.linspace(78750, 92000, target_data.shape[0])    # Y-coordinate
    range
    grid_x, grid_y = np.meshgrid(x_coords, y_coords)
```

```
# 3. Convert data to long format
target_long = pd.DataFrame({
    'X': grid_x.ravel(),
    'Y': grid_y.ravel(),
    'Value_target': target_data.values.ravel()
})

collaborative_long = pd.DataFrame({
    'X': grid_x.ravel(),
    'Y': grid_y.ravel(),
    'Value_collaborative': collaborative_data.values.ravel()
})

# 4. Merge the datasets
merged_data = pd.merge(target_long, collaborative_long, on=['X', 'Y'])

# 5. Define sample ratios and RMSE list
sample_ratios = [0.01, 0.05, 0.1, 0.2, 0.3]
rmse_list = []

# 6. Iterate through different sample ratios, compute RMSE, and generate 2D
contour plots
for ratio in sample_ratios:
    sample_size = int(len(merged_data) * ratio)    # Determine sample size
    random_indices = np.random.choice(len(merged_data), sample_size,
replace=False)
    resampled_data = merged_data.iloc[random_indices]

    # Extract features and target variable
    X = resampled_data[['X', 'Y', 'Value_collaborative']]
    y = resampled_data['Value_target']

    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

    # Train Random Forest model
    rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
    rf_model.fit(X_train, y_train)

    # Test model performance
    y_pred = rf_model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
```

```
        rmse_list.append(rmse)

        print(f"Sample Ratio: {ratio*100:.0f}%, Sample Size: {sample_size},
RMSE: {rmse:.4f}")

        # Generate interpolation grid
        grid_data = pd.DataFrame({
            'X': grid_x.ravel(),
            'Y': grid_y.ravel(),
            'Value_collaborative': collaborative_long['Value_collaborative']
        })

        # Predict target values using the trained Random Forest model
        grid_data['Value_target_pred'] = rf_model.predict(grid_data[['X', 'Y',
'Value_collaborative']])
        z_target = grid_data['Value_target_pred'].values.reshape(grid_x.shape)

        # Plot 2D contour map
        plt.rcParams['font.sans-serif'] = ['SimHei']    # Set font to SimHei for
Chinese characters
        plt.rcParams['axes.unicode_minus'] = False    # Display negative signs
correctly
        plt.figure(figsize=(10, 8))
        contour = plt.contour(grid_x, grid_y, z_target, cmap='viridis', levels=20)
        plt.clabel(contour, inline=True, fontsize=8)
        plt.colorbar(label='Predicted F1_target Values')
        plt.title(f'2D: {ratio*100:.0f}% Sample Ratio')
        plt.xlabel('X Coordinate')
        plt.ylabel('Y Coordinate')
        plt.show()

# 7. Plot RMSE line graph
plt.figure(figsize=(10, 6))
plt.plot([r * 100 for r in sample_ratios], rmse_list, marker='o', linestyle='-',
color='b')
plt.title("RMSE vs. Sample Ratio", fontsize=14)
plt.xlabel("Sample Ratio (%)", fontsize=12)
plt.ylabel("RMSE", fontsize=12)
plt.grid(False)
plt.show()
```

Question-3-Rmse R^2 Mape

```
# Calculate RMSE, MAPE, and R²
```

```
predicted_flat = z_predicted.ravel()
rmse = np.sqrt(mean_squared_error(target_flat, predicted_flat))
mape = mean_absolute_percentage_error(target_flat, predicted_flat)
r2 = r2_score(target_flat, predicted_flat)
# Append metrics to lists
rmse_list.append(rmse)
mape_list.append(mape)
r2_list.append(r2)

print(f"Sample Ratio: {ratio * 100:.0f}%, RMSE: {rmse:.4f}, MAPE:
{mape:.2f}%, R² : {r2:.4f}")

# 6. Plot RMSE, MAPE, and R²  vs. Sample Ratio
plt.figure(figsize=(10, 6))
plt.plot([r * 100 for r in sample_ratios], rmse_list, marker='o', linestyle='-',
label='RMSE', color='b')
plt.plot([r * 100 for r in sample_ratios], mape_list, marker='s', linestyle='--',
label='MAPE (%)', color='r')
plt.plot([r * 100 for r in sample_ratios], r2_list, marker='^', linestyle='-.', label='R
² ', color='g')
plt.title("RMSE, MAPE, and R²  vs. Sample Ratio", fontsize=14)
plt.xlabel("Sample Ratio (%)", fontsize=12)
plt.ylabel("Metrics", fontsize=12)
plt.legend()
plt.show()
```

Question-4-3D surface plot

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.interpolate import griddata

# Load the dataset
file_path = ""
data = pd.read_excel(file_path, header=0)

# Rename columns for easier understanding
data.columns = ['Column Sequence', 'Row Sequence', 'X-Coordinate',
'Y-Coordinate', 'Target Property']

# Extract coordinates and target property
x = data['X-Coordinate']
```

```
y = data['Y-Coordinate']
z = data['Target Property']

# Create a grid for the surface
grid_x, grid_y = np.meshgrid(
    np.linspace(x.min(), x.max(), 100),
    np.linspace(y.min(), y.max(), 100)
)

# Interpolate the Z values to create the surface
grid_z = griddata((x, y), z, (grid_x, grid_y), method='cubic')

# Create a 3D surface plot
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

# Plot the surface
surface = ax.plot_surface(grid_x, grid_y, grid_z, cmap='viridis',
edgecolor='none', alpha=0.9)

# Add a color bar
cbar = fig.colorbar(surface, ax=ax, pad=0.1)
cbar.set_label('Target Property', fontsize=12)

# Label the axes
ax.set_xlabel('X-Coordinate', fontsize=12)
ax.set_ylabel('Y-Coordinate', fontsize=12)
ax.set_zlabel('Target Property', fontsize=12)

# Set the title
ax.set_title('3D Surface Visualization of Spatial Variable (Target Property)',
fontsize=14)

# Show the plot
plt.show()
```

Question-4-Cokriging with fmm

```
import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from pykrige.ok import OrdinaryKriging

    # 1. Load target sampling data and collaborative variable data
```

```
target_file = ""
collaborative_file = ""

# Load target sampling data
target_data = pd.read_excel(target_file)

# Load complete collaborative variable data
collaborative_data = pd.read_excel(collaborative_file, header=None)

# Target sampling points (X, Y, Z)
x_target = target_data['X-Coordinate']
y_target = target_data['Y-Coordinate']
z_target = target_data['Target Property']

# Collaborative variable complete grid data
collaborative_matrix = collaborative_data.values
grid_x_collab = np.linspace(51250, 64500, 266)
grid_y_collab = np.linspace(78750, 92000, 266)
grid_x_collab, grid_y_collab = np.meshgrid(grid_x_collab, grid_y_collab)

# Flatten collaborative variable data to 1D arrays
x_collab = grid_x_collab.ravel()
y_collab = grid_y_collab.ravel()
z_collab = collaborative_matrix.ravel()

# Randomly select 20% of the collaborative variable points
num_collab_points = int(len(x_collab) * 0.2)
indices_collab = np.random.choice(len(x_collab), num_collab_points,
replace=False)
x_collab_sampled = x_collab[indices_collab]
y_collab_sampled = y_collab[indices_collab]
z_collab_sampled = z_collab[indices_collab]

# 2. Co-Kriging interpolation
def co_kriging(x_target, y_target, z_target, x_collab, y_collab, z_collab, grid_x,
grid_y):
    """
    Co-Kriging model prediction.
    """
    # Combine target and collaborative variable points
    x_all = np.concatenate((x_target, x_collab))
    y_all = np.concatenate((y_target, y_collab))
    z_all = np.concatenate((z_target, z_collab))
```

```
        # Remove duplicate coordinates
        unique_points = np.unique(np.c_[x_all, y_all], axis=0)
        x_unique = unique_points[:, 0]
        y_unique = unique_points[:, 1]
        z_unique = []
        for x, y in zip(x_unique, y_unique):
            idx = np.where((x_all == x) & (y_all == y))[0]
            z_unique.append(np.mean(z_all[idx]))
        z_unique = np.array(z_unique)

        # Perform interpolation using Ordinary Kriging
        ok = OrdinaryKriging(
            x_unique, y_unique, z_unique,
            variogram_model='linear',
            verbose=False,
            enable_plotting=False
        )
        z_predicted, _ = ok.execute('grid', grid_x, grid_y)

        return z_predicted

# 3. Construct interpolation grid
grid_x = np.linspace(51250, 64500, 266)    # Interpolation grid for X
grid_y = np.linspace(78750, 92000, 266)    # Interpolation grid for Y
grid_x, grid_y = np.meshgrid(grid_x, grid_y)

# 4. Test different sample sizes for target points
sample_sizes = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]
for sample_size in sample_sizes:
        # Select sampled points from the target data
        x_target_sampled = x_target[:sample_size]
        y_target_sampled = y_target[:sample_size]
        z_target_sampled = z_target[:sample_size]

        # Perform interpolation
        z_predicted = co_kriging(
            x_target_sampled, y_target_sampled, z_target_sampled,
            x_collab_sampled, y_collab_sampled, z_collab_sampled,
            grid_x, grid_y
        )

        # Save results to an Excel file
        z_predicted_df = pd.DataFrame(z_predicted)
        output_file = f"co_kriging_prediction_{sample_size}_samples.xlsx"
```

```
        z_predicted_df.to_excel(output_file, index=False, header=False)
        print(f"Results saved to: {output_file}")

        # Visualize contour plot
        plt.figure(figsize=(10, 8))
        contour = plt.contourf(grid_x, grid_y, z_predicted, cmap='viridis',
levels=20)
        plt.colorbar(label='Predicted Target Property')
        plt.title(f'Co-Kriging Prediction with {sample_size} Target Samples')
        plt.xlabel('X Coordinate')
        plt.ylabel('Y Coordinate')
        plt.show()
```

Question-4-Data normalization

```
import pandas as pd

    # File path of the previously saved matrix
    output_file = 'Data-preprocessing/F2_collaborative4.xlsx'

    # Load the original data file
    loaded_matrix_df = pd.read_excel(output_file, header=None)

    # Perform normalization on the loaded data
    loaded_matrix = loaded_matrix_df.values
    normalized_loaded_matrix = (loaded_matrix - loaded_matrix.min()) /
    (loaded_matrix.max() - loaded_matrix.min())

    # Convert normalized data into a DataFrame
    normalized_loaded_matrix_df = pd.DataFrame(normalized_loaded_matrix)

    # Save the normalized data to a new Excel file
    normalized_loaded_output_file =
    'Data-preprocessing/F2_collaborative4_normalized.xlsx'
    normalized_loaded_matrix_df.to_excel(normalized_loaded_output_file,
    index=False, header=False)

    # Output the path of the normalized file
    print("Normalized file saved as:", normalized_loaded_output_file)
```

Question-4-Datatransformation

```
import pandas as pd
    import numpy as np
```

```
import chardet

# File path
file_path = 'F2_collaborative_variable4.txt'

# Detect file encoding
with open(file_path, 'rb') as file:
    raw_data = file.read()
    encoding = chardet.detect(raw_data)['encoding']

# Read and clean data
cleaned_data_rows = []
with open(file_path, 'r', encoding=encoding) as file:
    for line in file:
        try:
            # Parse lines containing valid numeric values
            values = list(map(float, line.split()))
            cleaned_data_rows.append(values)
        except ValueError:
            # Skip lines with non-numeric or invalid formats
            continue

# Flatten the data and trim to 266x266 matrix
cleaned_data_array = np.array([item for sublist in cleaned_data_rows for item in
sublist])

# Ensure the data size matches 266x266 matrix
required_size = 266 * 266
trimmed_data_array = cleaned_data_array[:required_size]

# Create the matrix
matrix = trimmed_data_array.reshape(266, 266)

# Convert to DataFrame to save as an Excel file
matrix_df = pd.DataFrame(matrix)

# Save as an Excel file
output_file = 'Data-preprocessing/F2_collaborative4.xlsx'
matrix_df.to_excel(output_file, index=False, header=False)

print("File has been saved as:", output_file)
```

Question-4-Scatter plots and JB statistics

```
import pandas as pd
    from scipy.stats import normaltest, spearmanr
    import matplotlib.pyplot as plt

    file_paths = [
        'Data-preprocessing/F2_collaborative1_extracted.xlsx',
        'Data-preprocessing/F2_collaborative2_extracted.xlsx',
        'Data-preprocessing/F2_collaborative3_extracted.xlsx',
        'Data-preprocessing/F2_collaborative4_extracted.xlsx'
    ]
    i = 0

    for file_path in file_paths:
        # Load the uploaded Excel files
        file_extracted = file_path
        file_target = 'Data-preprocessing/F2_target_variable.xlsx'

        # Read the files into dataframes
        extracted_df = pd.read_excel(file_extracted)
        target_df = pd.read_excel(file_target)

        # Merge the two dataframes on "Column Sequence" and "Row Sequence"
        merged_df = pd.merge(extracted_df, target_df, on=["Column Sequence",
    "Row Sequence"])

        # Keep only the relevant columns for analysis: "Extracted Value" and
    "Target Property"
        analysis_df = merged_df[["Extracted Value", "Target Property"]]

        # Check for NaN or infinite values in the dataframe
        cleaned_analysis_df = analysis_df.dropna().replace([float('inf'), float('-inf')],
    pd.NA).dropna()

        # Perform JB normality test for both cleaned columns
        jb_extracted_cleaned = normaltest(cleaned_analysis_df["Extracted Value"])
        jb_target_cleaned = normaltest(cleaned_analysis_df["Target Property"])

        # Calculate Spearman correlation coefficient
        spearman_corr, spearman_p_value =
    spearmanr(cleaned_analysis_df["Extracted Value"],

    cleaned_analysis_df["Target Property"])

        # Print results in a formatted way
```

```
        print(f"=== Results for {file_path} ===")
        print("=== Data Quality Check ===")
        print(f"NaN/Inf Check: {analysis_df.isin([float('inf'), float('-inf')]).sum() +
    analysis_df.isnull().sum()}")

        print("\n=== Normality Test Results ===")
        print(f"Extracted Value: Statistic = {jb_extracted_cleaned.statistic:.4f}, "
                f"p-value = {jb_extracted_cleaned.pvalue:.4e}")
        print(f"Target Property: Statistic = {jb_target_cleaned.statistic:.4f}, "
                f"p-value = {jb_target_cleaned.pvalue:.4e}")

        print("\n=== Correlation Results ===")
        print(f"Spearman Coefficient: {spearman_corr:.4f}")
        print(f"p-value: {spearman_p_value:.4e}")

        # Scatter plot
        plt.figure(figsize=(8, 6))
        plt.scatter(cleaned_analysis_df["Extracted Value"],
    cleaned_analysis_df["Target Property"], alpha=0.7)
        plt.title(f"Scatter Plot: F2_collaborative{i}_extracted", fontsize=14)
        i = i + 1
        plt.xlabel("Extracted Value", fontsize=12)
        plt.ylabel("Target Property", fontsize=12)
        plt.show()
```

Question-4-Spearman heat map

```
import pandas as pd
    import scipy.stats as stats
    import seaborn as sns
    import matplotlib.pyplot as plt

    # Load the datasets
    file_paths = [
        "Data-preprocessing/F2_target_variable.xlsx",
        "Data-preprocessing/F2_collaborative1_extracted.xlsx",
        "Data-preprocessing/F2_collaborative2_extracted.xlsx",
        "Data-preprocessing/F2_collaborative3_extracted.xlsx",
        "Data-preprocessing/F2_collaborative4_extracted.xlsx"
    ]

    dataframes = [pd.read_excel(path) for path in file_paths]

    # Extract the last column from each dataset
```

```
last_columns = [df.iloc[:, -1] for df in dataframes]

# Combine the last columns into a single DataFrame for correlation calculation
combined_data = pd.concat(last_columns, axis=1)
# Update the column names based on the file names
column_names = [
    "target",
    "collaborative1",
    "collaborative2",
    "collaborative3",
    "collaborative4",
]
combined_data.columns = column_names

# Recalculate Spearman correlation matrix with updated names
spearman_corr_updated = combined_data.corr(method='spearman')

# Plot heatmap with updated names
plt.figure(figsize=(10, 8))
sns.heatmap(spearman_corr_updated, annot=True, fmt=".2f",
cmap="coolwarm", cbar=True)
plt.title("Question4 Spearman Heatmap")
plt.tight_layout()
plt.savefig("Question4_Heatmap.jpg",dpi=600)
plt.show()

spearman_corr_updated
```