Improving the performance of Gaussian process regression for mutation on SARS-CoV-2

Content

1. Abstract
2. introduction
3. Basic GP model
    - 3.1 Principle
    - 3.2 Data Preprocessing
    - 3.3 Kernel
    - 3.4 Random search for parameters
    - 3.5 Comparison with other ML methods
    - 3.6 Analysis shortage
4. Improved GP model:
    - 4.1 Summary of improvements
    - 4.2 Feature engineering
    - 4.3 Bayesian optimization for parameters
    - 4.4 Manage overfitting
    - 4.5 data augmentation
5. Result and analysis
6. Discussion
7. Referee

1. Abstract:

Regression for protein mutation on a 3D protein structure is a crucial and meaningful task in biostatistics research. In this article, we present a method to improve the performance of Gaussian Process (GP) models for this task. GP models are well-known for their ability to capture complex relationships and provide probabilistic predictions. However, due to the limited size of the dataset (less than 1000 rows) and the basic features derived from the three coordinates, the basic GP model often exhibits poor performance.

To address these challenges, we employ feature engineering techniques to extract and select more informative features from the original dataset. This helps to enhance the representation of the protein mutation data. Additionally, we utilize Bayesian optimization to determine better parameters for the kernel function, which further improves the model's performance.

Overfitting, a common issue in regression tasks, is mitigated through regularization techniques and data augmentation. We explore two data augmentation methods and demonstrate that augmenting the entire dataset can lead to a significant improvement in the regression results.

To evaluate the effectiveness of our approach, we compare the results obtained from different models and datasets, as well as results from other machine learning (ML) models. We aim to overcome the limitations of the basic GP model and enhance the accuracy and reliability of protein mutation regression.

2. Introduction:

Gaussian process is suitable for protein spatial analysis for its flexibility, uncertainty estimation, and robustness to noise and outliers. In the article "Bayesian Modeling of spatial molecular profiling data via Gaussian Process" by Li et al, the Gaussian process (GP) model is incorporated to identify genes that have significant spatial variation (SV genes), and clusters them into different patterns using a GP mixture model (GPMM). It is also used for analyzing multidimensional neuronal and behavioral relationships in Parametric Copula-GP model for analyzing multidimensional neuronal and behavioral relationships (Li et al). In general, GO model has versatile function in analyzing protein.

3. Basic GP method

3.1 Principle:

Gaussian Process (GP) is a flexible statistical model that can capture complex relationships in data. It assigns a probability distribution to each point in the dataset, allowing for interpolation and extrapolation. The mean value function and covariance function of the target variable to describe the overall distribution are found during the process. Gaussian Process (GP) regression is often chosen for its probabilistic predictions, which enable us to assess the confidence interval. We can learn how GP derived from normal distribution from the graph1.

$$x \sim \mathcal{N}(\mu, \sigma^2)$$
$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')))$$

Graph1: how GP derived form normal distribution

Definition 2.1 *A* Gaussian process *is a collection of random variables, any finite number of which have a joint Gaussian distribution.* □

A Gaussian process is completely specified by its mean function and covariance function. We define mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$ as

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned} \tag{2.13}$$

and will write the Gaussian process as

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big). \tag{2.14}$$

Graph2: Formal definition of GP

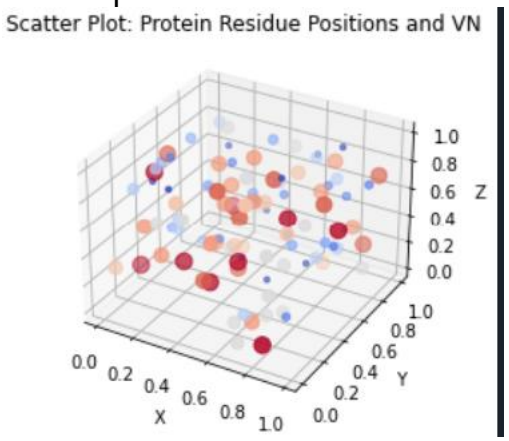
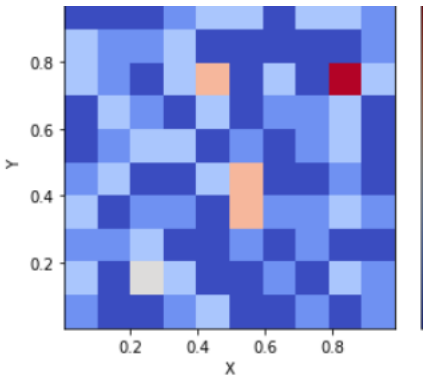
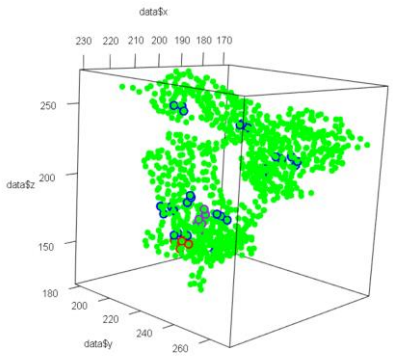
One of the challenges in this analysis is the distribution of the object

variable (VN). It is extremely right skewed, while normal distribution is preferred for GP model. Moreover, the scarcity of data poses the main challenge. With only 972 data points available for the A chain, the patterns we need to study are highly complex, with infrequent but influential high VN points.

Our objective is not only to develop a model that can accurately fit the mutation, or Virus Number (VN) at various residuals positions, but also to analyze the pattern of the confidence interval in the region of interest and account for possible dynamic movements and changes.

3.2 Data explore and preprocessing.

Explore: code: #%% Data exploration

Scatterplot:	Heatmap: VN on xy-plane	Hotspots: VN hotspots
		

Preprocessing:

1.Scaling: code: #%%log transformation of y

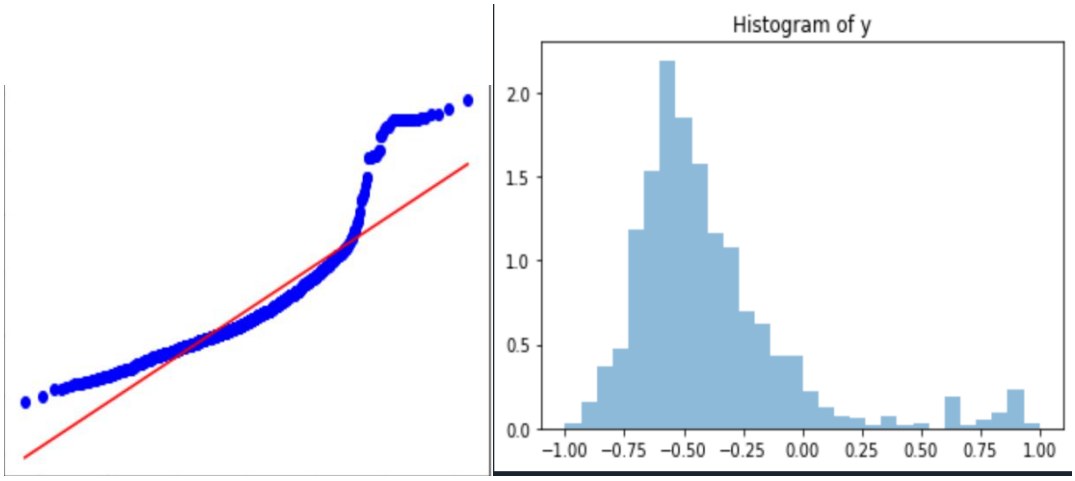
To manage the right skewed of y value, we first take a log10 transformation. MinMaxScaler method(linear) is then took to VN and 'x','y',and 'z', which confirm all the variables in [-1,1]. StandardScaler method, which minus the average x before dividing the standard deviation, is another approach. Kernels of GP model, a distance-based function, are sensitive to the scale of the features, absolute magnitude of the features is crucial here. Consequently, relative positions could not be changed here, which could vary after StandardScaler, making MinMaxScaler a more suitable choice.

Standard Scaler	$\frac{x_i - \text{mean}(\boldsymbol{x})}{\text{stdev}(\boldsymbol{x})}$
MinMax Scaler	$\frac{x_i - \min(\boldsymbol{x})}{\max(\boldsymbol{x}) - \min(\boldsymbol{x})}$
Robust Scaler	$\frac{x_i - Q_1(\boldsymbol{x})}{Q_3(\boldsymbol{x}) - Q_1(\boldsymbol{x})}$

Figure3: scaler method

2.transformation of y: code: #%%log transformation of y

y is preferred to be normal distributed. We use QQ-plot, histogram polt and standard statistical test method include Kolmogorov-Smirnov, Shapiro-Wilk, and Anderson-Darling to test it.



Graph4: QQ-plot and histogram of y before Box-Cox

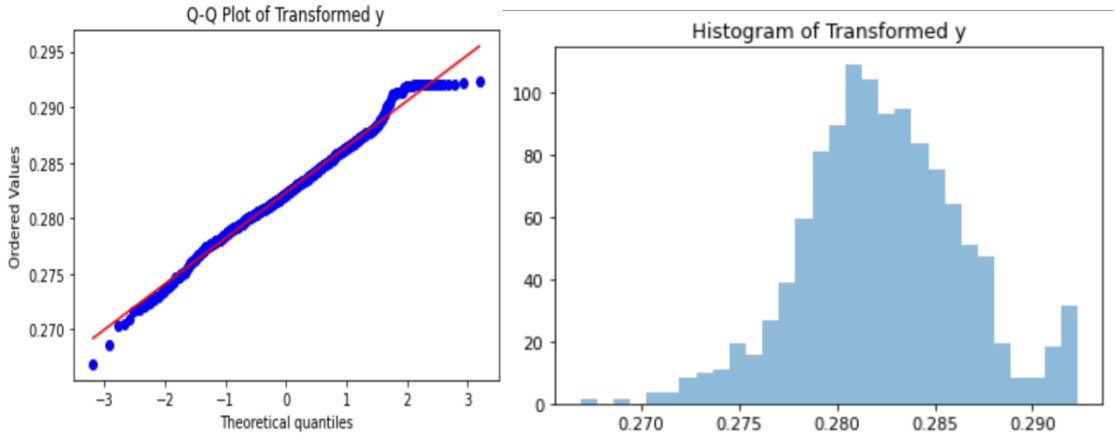
Shapiro-Wilk test p-value: 7.724715987445591e-31

Anderson-Darling test statistic: 40.17281670422699

Anderson-Darling test critical values: [0.574 0.653 0.784 0.914 1.088]

[code: #%%Box-Cox for further transform y]

It turns out that the log transformation could not give a normal distribution to y. Thus, further method is needed. We use Box-Cox method here. The Box-Cox method transforms data to approximate normality. It applies a power transformation to the data, estimating the optimal lambda value that maximizes normality.



Graph5: QQ-plot and histogram of y after Box-Cox

Shapiro-Wilk test p-value: 5.193222386878915e-05

Anderson-Darling test statistic: 1.513998405351117

Anderson-Darling test critical values: [0.574 0.653 0.784 0.914 1.088]

After this, we confirm that y value now is like normally distribution.

3.3 Kernel

Selecting the best kernel and tuning hyperparameters is crucial in protein mutation regression. A kernel, the covariance function of the GP, captures relationships between data points, allowing us to measure similarity and generate regression results. Finding the right kernel involves exploring options and adjusting hyperparameters to improve model performance. It is a critical task that significantly impacts predictive accuracy. We have tried RBF, Matern and combination of both kernels. It turns out that using Matern(a generalization of RBF) itself as the kernel give a best balance between model complexity and accuracy.

Usual covariance functions [edit]

There are a number of common covariance functions:[9]

- Constant : $K_{\mathrm{C}}(x, x') = C$
- Linear: $K_{\mathrm{L}}(x, x') = x^T x'$
- white Gaussian noise: $K_{\mathrm{GN}}(x, x') = \sigma^2 \delta_{x,x'}$
- Squared exponential: $K_{\mathrm{SE}}(x, x') = \exp\left(-\frac{|d|^2}{2\ell^2}\right)$
- Ornstein–Uhlenbeck: $K_{\mathrm{OU}}(x, x') = \exp\left(-\frac{|d|}{\ell}\right)$
- Matérn: $K_{\mathrm{Matern}}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}|d|}{\ell}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}|d|}{\ell}\right)$
- Periodic: $K_{\mathrm{P}}(x, x') = \exp\left(-\frac{2\sin^2\left(\frac{d}{2}\right)}{\ell^2}\right)$
- Rational quadratic: $K_{\mathrm{RQ}}(x, x') = (1 + |d|^2)^{-\alpha}$, $\alpha \geq 0$

Graph6: typical kernel choice

3.4 Random search for parameters

we use the random search (search randomly samples a specified number of parameter settings from the defined distribution) to find the best parameters for the kernel. Here, we need to find 'length_scale' and 'nu' of the kernel, control the flexibility and smoothness respectively.

Results of the best combination of parameters is :

kernel: Matern

kernel__length_scale: 1.2927200214928352,

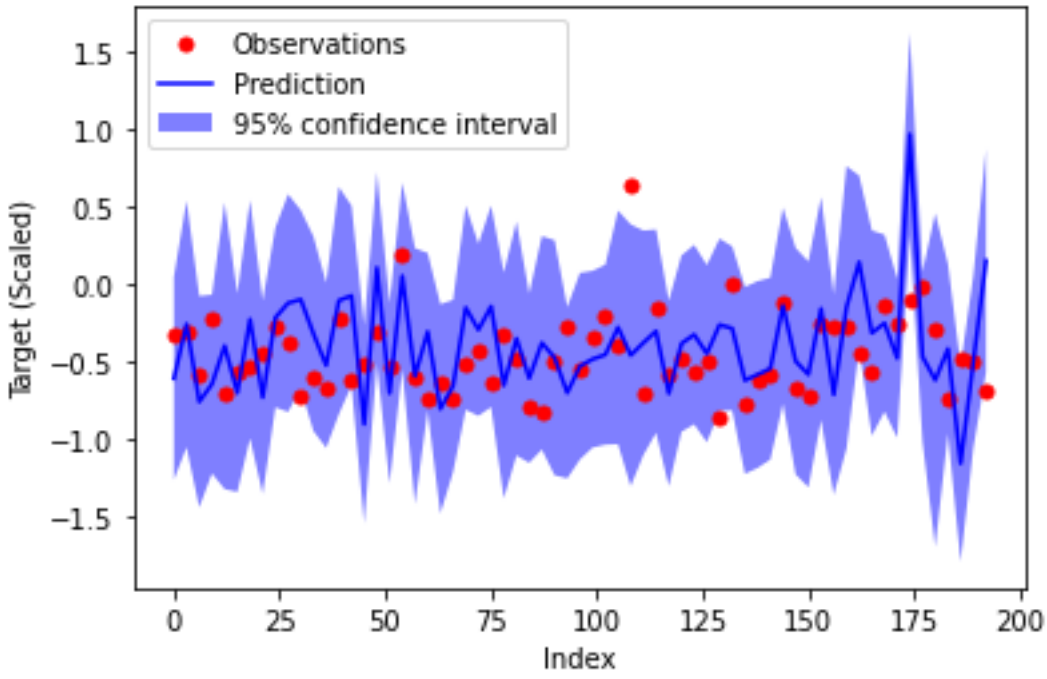
kernel__nu: 1.540932921169848}

3.5 Comparison with other ML methods

We show the result of the basic GP model and other ML model on the test set below. It should be notice that the graph only demonstrates some of the points. The width of the 95% Confidence Interval(CI) is calculated by 1.96*standard deviation of the predicted values from the Gaussian process regressor

GP:(for the above kernel and parameter)

[code: #%% Toy/basic model 2 (direct for CI)]



CI for GP

Mean Squared Error (MSE): 0.15166325180372806

Total Sum of Squares (TSS): 13.98577736718617

Regression Sum of Squares (RSS): 6535.900091768129

Mean Absolute Error (MAE): 0.28489766902220137

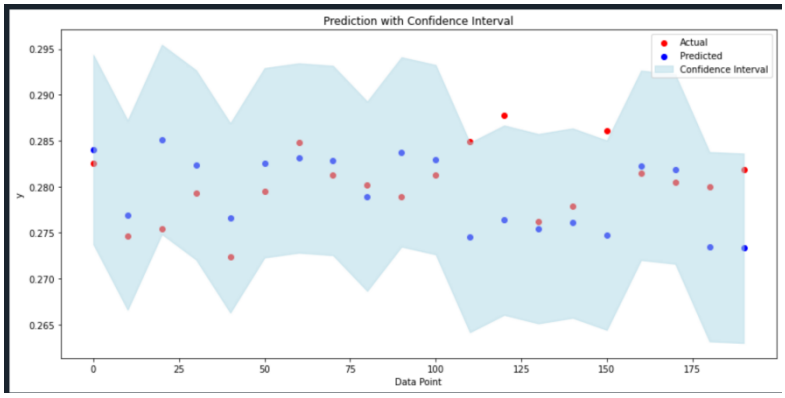
R-squared (R2): -1.1146006636080963

The model's performance, with an R2 of -1.11, suggests that it performs even worse than a null model, which simply takes the average of all y values. This poor performance can be attributed to the highly complex patterns of VN on the protein, which are challenging to learn. Particularly, the model struggles with predicting extremely high VN points, where the application of the Cox-Box transformation can be beneficial in improving predictions.

To make comparisons with other ML models, we implement other three typical methods. All the model is tuned using relative methods. The results are dissatisfied either.

Neural network:

[code: #%% Neural Network]



CI for NN

MLPRegressor(activation='tanh', alpha=0.01, hidden_layer_sizes=(100,

100, 100), random_state=42)

Mean Squared Error (MSE): 0.07959919740006358

Total Sum of Squares (TSS): 13.98577736718617

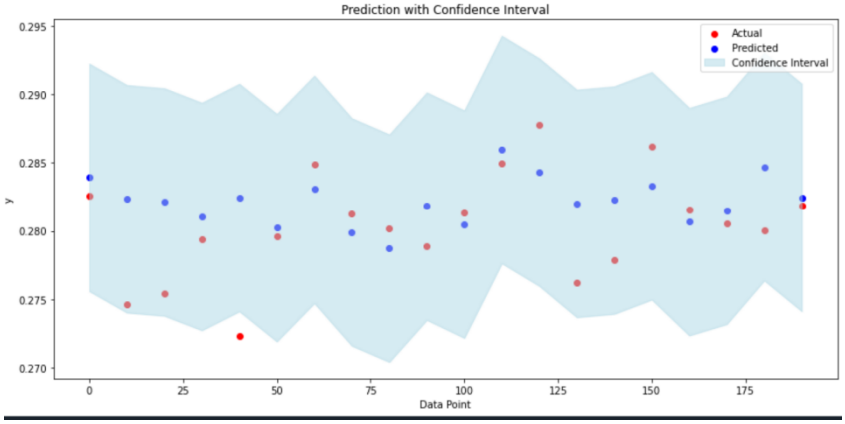
Regression Sum of Squares (RSS): 3336.138050659326

Mean Absolute Error (MAE): 0.22332365758639003

R-squared (R2): -0.10983058613747065

RF:

[code: #%% RF]



CI for RF

RandomForestRegressor(random_state=42)

Mean Squared Error (MSE): 0.09352669178606388

Total Sum of Squares (TSS): 13.98577736718617

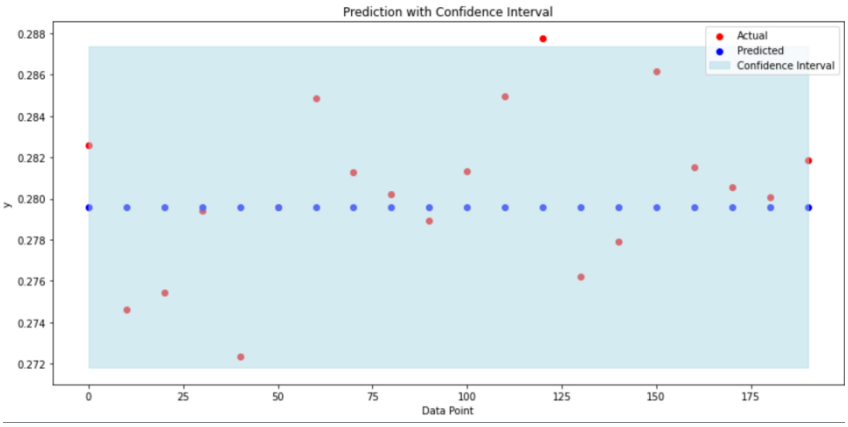
Regression Sum of Squares (RSS): 4385.181502616648

Mean Absolute Error (MAE): 0.2326191127107597

R-squared (R2): -0.3040179619241101

SVM:

[code: #%% SVM]



CI for SVM

SVR('C': 3, 'epsilon': 0.5, gamma at 'scale'(1 / (n_features * X.var())))

Mean Squared Error (MSE): 0.11486673383110596

Total Sum of Squares (TSS): 13.98577736718617

Regression Sum of Squares (RSS): 5040.523119015015

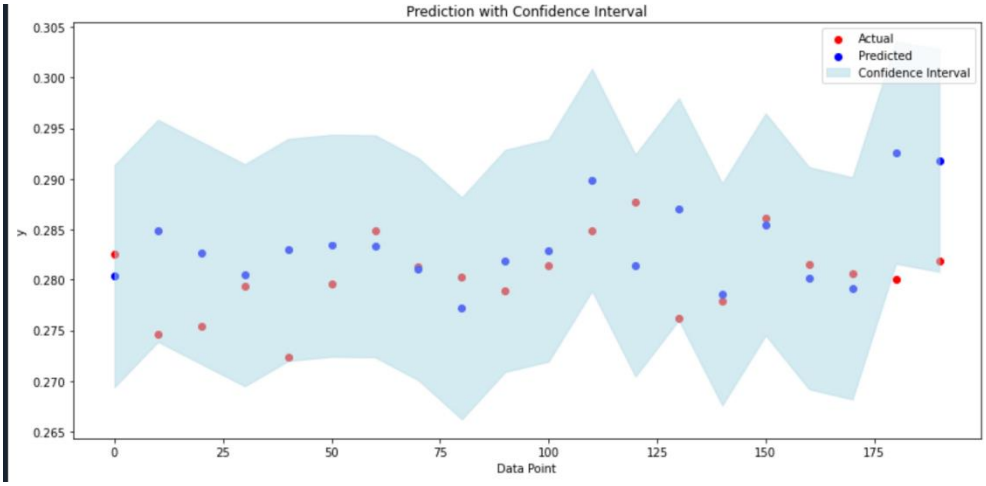
Mean Absolute Error (MAE): 0.28978665071046006

R-squared (R2): -0.6015565319678879

The most apparent thing to notice is that SVM give same prediction for all test point. It obviously means that the situation is too hard to split point use any pattern. Gaussian process performs the worse here and Neural network performs the best. Nevertheless, all for model perform worse than the null model, indicating failing to capture the pattern in text set..

We then try Ensemble learning method by stacking, which involves Combination of GP, NN, RF and SVM mentioned above to create a stronger and more accurate predictive model. However, the result does not show a dramatic progress.

[code: #%% 集成学习以上四种 model]



CI for Ensemble learning

Mean Squared Error (MSE): 0.15166325180372836

Total Sum of Squares (TSS): 13.98577736718617

Regression Sum of Squares (RSS): 29.57433410172703

Mean Absolute Error (MAE): 0.2848976690222018

R-squared (R2): -1.1146006636081007

3.6 Analysis the basic model

To further analysis the true performance of the basic model. After dividing the residues equally in three group according to VN, we design a classification model for predicting the VN.

The confusion matrix is following:

Confusion Matrix:

[[209 37 78]

 [144 60 120]

 [95 32 197]]

This shows that the basic GP model perform reasonable for the first quantile and the third quartile. At accuracy about 63%. However, the result is disastrous for the middle quantile, at accuracy for only 18.5%. It seems that the GP model is suitable to predict the polarized value. This will be considered below.

We believe the pattern is too hard for GP to study. We will show how to improve the model below.

4. Improved GP model

4.1 Summary for improvement:

1.To extract more information, we will try to process feature

engineering to develop more features (e.g: xy, x^2) and select the from them.

2. Use Bayesian method for better hyper parameter

3.Noise and regularization term might need to be considered to avoid overfitting.

4.Introduce data augmentation to improve the regression performance of the model.

4.2Feature engineering:

1.feature extraction:

[code: #%%Feature processing: Feature extraction]

our final features is:

x','y','z','distance_to_center', 'sum_xyz',x^2, y^2, z^2, xy, xz, yz

- 'distance_to_center': Calculate the distance from each point to the average position of all the points, providing insights into the spatial distribution.

'sum_xyz': Compute the total magnitude of the data along each axis.

- x^2, y^2, z^2, xy, xz, yz : Polynomial feature extraction with degree of 2. Generate new features by creating combinations involving different degrees of these variables.

These new features allow the model to capture complex patterns and interactions in the data, allowing it to fit non-linear relationships between the variables.

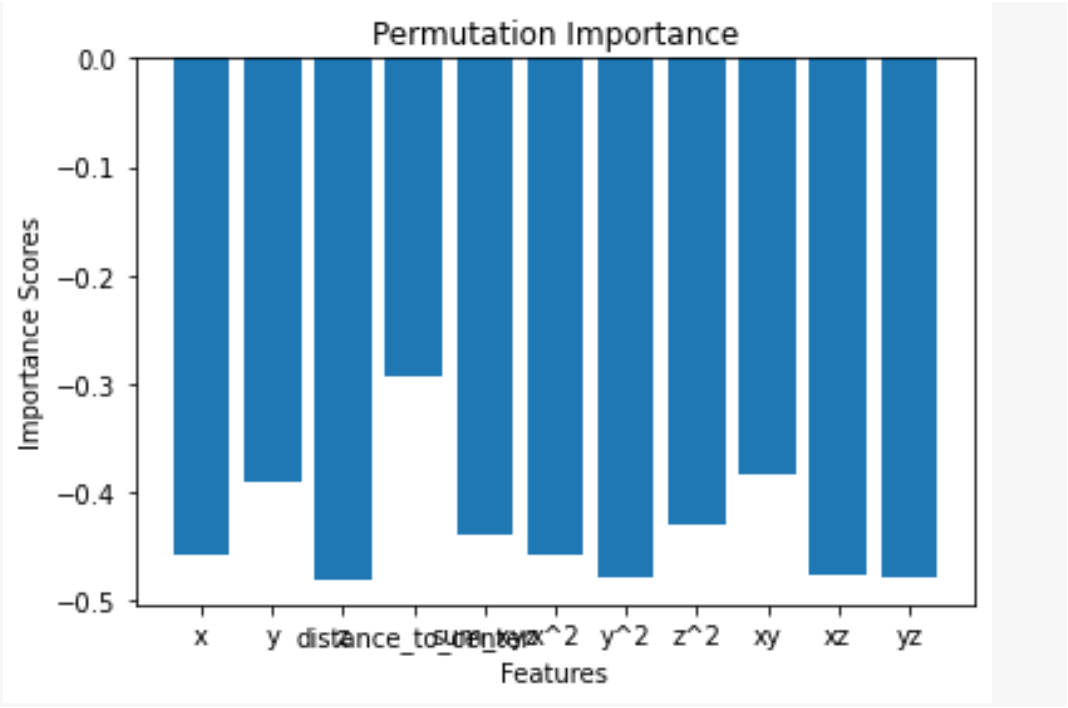
2.Feature attribution [code: #%%Feature attribution]

When bringing many features, especially highly correlated ones, to the model, there is a risk of increasing overfitting. Therefore, it is crucial to select the most useful features. We employ the following methods. It should be notice that these analyses are for model:

[kernel =Matern, length_scale=1.29, nu=1.54)]

1.SHAP (SHapley Additive explanations) is an attribution method that quantifies the impact of changing individual input variables while keeping other variables constant. It captures interactions and nonlinear relationships between variables, providing insights into each feature's contribution to the model's output.

2.Permutation Importance measures the change in performance when a feature's values are randomly shuffled. Features with high importance scores significantly affect the model's performance when shuffled.



Permutation importance for feature

Permutation Importance Scores: [-0.4583315 -0.39092659 -0.47975965 -0.29357902 -0.43980696 -0.4583022 -0.47801272 -0.42930856 -0.38413916 -0.4756331 -0.47917125]

The score demonstrates how the model's performance decrease with the change of the parameter. The least negative score, which is for 'distance to the center', indicate the least importance of the features. The importance of other features are about the same.

3. Feature selection [code: #%%Feature selection]

1. Correlation-based Feature Selection:

   Evaluates the importance of features by calculating their Pearson

correlation to assess the strength of the linear relationship between each feature and the target variable.

result:

x','y','z','distance_to_center', 'sum_xyz',x^2, y^2, z^2, xy, xz, yz

Selected Features: [0 5 8 1 9 3 10 4 2 6 7]

Feature Importance Scores: [-0.11912385 0.10935046 0.04823055 0.08458194 0.05217221 -0.11892911 -0.00713153 -0.00304945 0.11775419 0.10011666 0.05342933]

According to this, ‘x’, 'x^2',’ xy,’y’,’xz’ is more correlated with the target.

2.SelectKBest Feature Selection:

Ranks features based on their F-test scores with the target variable, choose the relevant ones to the target variable.

results:

Selected Features: [0 1 5 8 9]

Feature Importance Scores: [1.39629169e+01 1.17391699e+01 2.26166112e+00 6.98948418e+00 2.64748726e+00 1.39166487e+01 4.93354582e-02 9.02024610e-031.36391906e+01 9.82108608e+00 2.77697991e+00]

According to this, ‘x’,’y’ ,'x^2',’ xy,,’xz’ is more correlated with the target.

We choose ‘x’,’y’ ,'x^2',’ xy,,’xz’ and ‘‘distance to the center’’ as our final features.

4.3 Bayesian optimization for Parameters

[code: #%%Bayesian method for optimizing parameters]

Bayesian method: Uses a surrogate model, typically a Gaussian process, to approximate the unknown objective function. It guides the search by iteratively selecting points to evaluate based on an acquisition function. This balances exploration and exploitation, focusing on promising regions of the search space. The surrogate model is continuously updated with new observations. Bayesian optimization efficiently explores the search space by leveraging the surrogate model and uncertainty estimates.

Bayesian optimization is preferred over random search for hyperparameter optimization because it efficiently explores the search space, balances exploration and exploitation, adapts sampling, requires fewer trials, and has automatic stopping criteria.

The results:

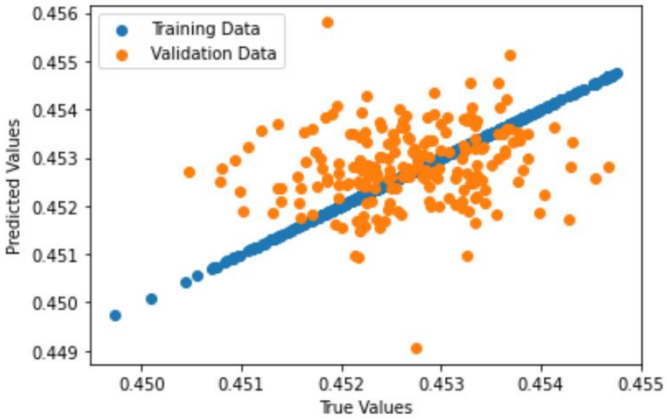
parameters: {'kernel': 'Matern', 'alpha': 1.561524032632997, 'nu': 0.5055167072438539, 'length_scale': 1.9428844101505567}

It should be noted that all the method below is based on this model.

4.4 Resolve overfitting:

Test the overfitting.

[code: #%% test overfitting]



The scatter plot of True value vs scatter value before adding alpha (previous model)

MSE on training set: 3.804285459272239e-29

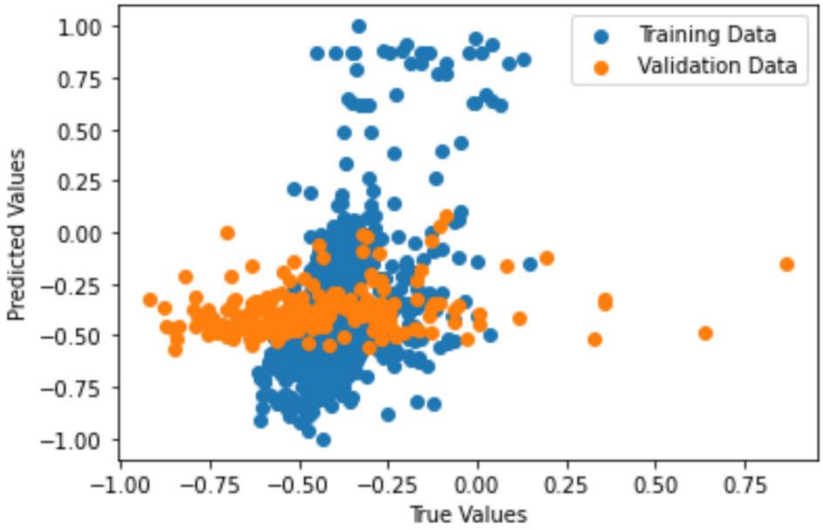
MSE on validation set: 0.15166325180372806

If the model performs much better on the training set than on the test

set, it may indicate that the model has memorized the training data and failed to generalize to new data.

Here, the MSE is much larger on validation set, with nearly no error on the train set. The scatter plot also shows the same thing, indicating a severe overfitting problem.

To address overfitting, we can use techniques like cross-validation, regularization, dropout, data augmentation and early stopping. We will perform regularization and data augmentation here and data augmentation need to be carefully discussed below:



The scatter plot of True value vs scatter value adding alpha (model after Bayesian optimization)

MSE on training set: 0.09125576365462267

MSE on validation set: 0.0764201065459382

4.5 Data augmentation:

[code: #%% Data augmentation for train set]

[code: #%% Data augmentation for all set]

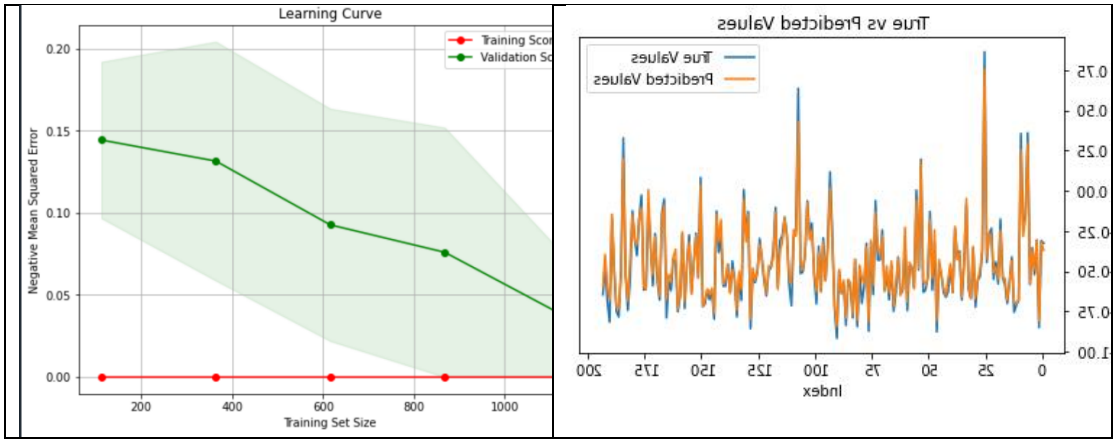
Data augmentation can help improve the model performance and prevent overfitting, especially when the initial data is scarce or imbalanced, which is exactly what we encounter here.

We have three choices in general: 1. No augmentation at all; 2. Augmentation on training set; 3. Augmentation on entire set. The conclusion is that if we want to address the overfitting problem, we can use both second and third approach by bringing noises in the data, which lead the model learn to be less sensitive to small variations in the input data, potentially making it more resilient to noise or minor fluctuations in real-world scenarios. However, only the third method can improve the regression performance on test set effectively. We will show the reason below.

Comparison of data augmentation on train set and entire set:

Under the default split method, train set get 80% of the entire data, which is 777 rows. To analyze and understand how the model's performance improves or stabilizes as more data is provided for

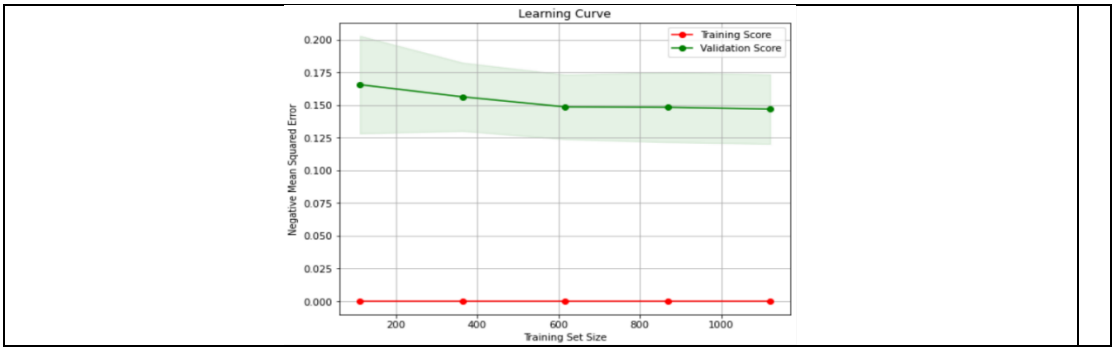
training, we draw the learning curve for the default model. First, we choose the data point to augment in the 40% entire set randomly randomly and bring noise standard error of 0.1.



Learning curve and True vs Prediction graph for model using entire data augmentation

It can be seen clearly that the performance of the model increases shapely and consistently even before bringing any augmented data to the training set. The negative mean squared error decreases continuously after the size of original training set (777), indicate the dramatic improvement of the model.

Second, we choose only augmented the train set by the same method.



Learning curve for model using train data augmentation

The result is dramatically different from the previous one, indicating augmented on train set does not improve the test score. This is not surprised, as it is often believed that only enhancing the training set can improve the performance and generalization ability of the model on the training set but has little effect on the performance of the verification set and the test set.

5.Result and analysis [code: 评价 cell 1-5]

We first show the result by basic criteria:

1. Mean Squared Error (MSE): the average squared difference between the predicted values and the true values.
2. Total Sum of Squares (TSS): the total variability in the true values of the target variable.
3. Residual Sum of Squares (RSS): the unexplained variability in the predicted values.

4. Mean Absolute Error (MAE): the average absolute difference between the predicted values and the true values.
5. Coefficient of Determination (R2): represents the proportion of the variance in the target variable that can be explained by the regression model.

We show results from following method:

TYPE:

1.Bayesian+70% data augmentation on entire set

2.Bayesian+40% data augmentation on entire set

3.Bayesian+70% data augmentation on train set

4.Bayesian+40% data augmentation on train set

5.Bayesian+ No augmentation

6.Random search+ No augmentation

Expect worse results from the top to bottom.

CRITERIA\TYPE	1	2	3	4	5	6
MSE	0.0024	0.00	0.25	0.24	0.076	0.15
TSS	13.99	13.99	13.99	13.99	13.99	13.99
RSS	4878.31	5454.45	9793.19	9511.85	3487.07	6535.90
MAE	0.037	3.04e-1	0.46	0.45	0.22	0.28
R2	0.96	0.72	-2.53	-2.46	-0.06	-1.11

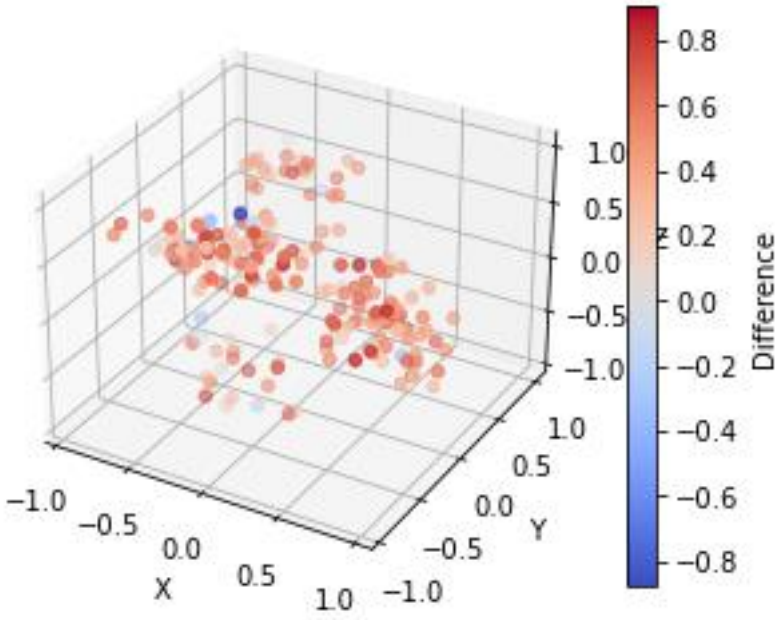
As GP model can give output with confidence interval (CI). We show the four criteria based on CI below:

1. Coverage: Measures the proportion of true values that fall within the prediction intervals. Higher coverage indicates more accurate prediction intervals.
2. Average Width: Calculates the average size of the prediction intervals. Smaller average width suggests more precise predictions.
3. Average Length: Measures the average width of prediction intervals relative to the range of true values. Smaller average length indicates less uncertainty compared to the range of true values.
4. Prediction Variance: Quantifies the variability or dispersion of the predicted values. Higher variance indicates greater variability in the predictions.

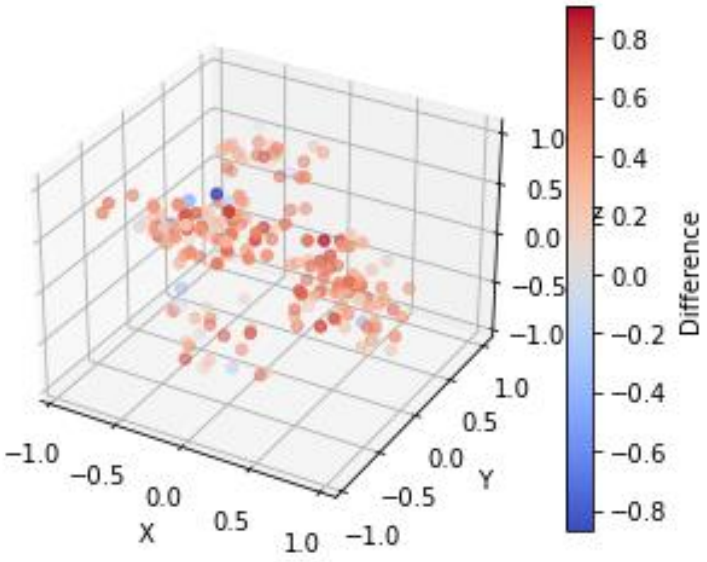
CRITERIA\TYPE	1	2	3	4	5	6
Coverage	0.99	0.98	1.0	1.0	1.0	0.548
Average width	0.275	0.286	0.294	0.296	0.293	0.309
Average length	0.155	0.160	0.165	0.170	0.164	0.174
Prediction variance	0.076	0.082	0.087	0.090	0.086	0.096

[code#%%画图 cell：residual in 3D]

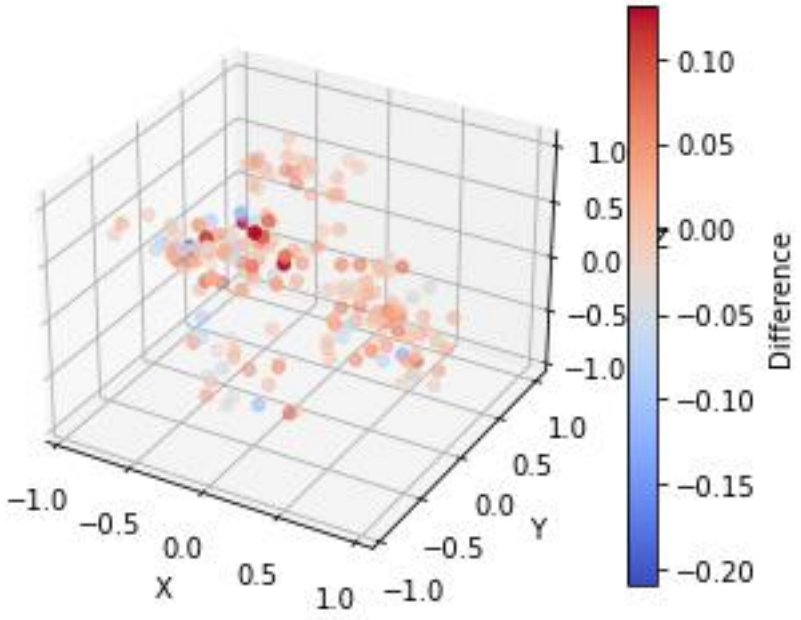
To clearly show the regression performance in 3D, we also show the plot the following graphs which use color to identify the magnitude of error. Redder places indicate bad performance while the bluer places indicate the good performance. It should be notice that only the points in test set are considered.



The 3D residual plot for the improved model without augmentation



The 3D residual plot for improved model + 40% augmentation on train set



The 3D residual plot for improved model + 40% augmentation on entire set