Infrared Oral Temperature Estimation

EE 559 Course Project
Data Set: Infrared Thermography

Yuming Li, liyuming@usc.edu

August 19, 2024

1 Abstract

This project focuses on predicting oral temperature using infrared thermography data through various machine learning models. The dataset comprises 1,020 data points and 33 features, including demographic and environmental variables. Key preprocessing steps—such as imputation, synthesis, normalization, and feature selection—were implemented to prepare the data. The study employed multiple regression models, including Lasso, Ridge, K-Nearest Neighbors, Support Vector Regression, Bayesian Ridge, Random Forest, RBF Network, and Multi-Layer Perceptron. After fine-tuning the hyperparameters, these models were evaluated using Mean Absolute Error (MAE), Mean Square Error (MSE), and Root Mean Squared Error (RMSE). The Random Forest model, with 200 estimators and a maximum depth of 15, delivered the best performance, achieving an MAE of 0.1877. This result underscores the model's capability to capture complex relationships within the data, highlighting its potential for non-invasive temperature prediction.

2 Introduction

2.1 Problem Assessment and Goals

The Infrared Thermography Temperature Dataset[1] is composed of temperature readings taken from various points on patients' infrared images, supplemented by additional environmental and demographic information, such as gender, age, ethnicity, ambient temperature, humidity, and distance. This dataset includes a total of 33 features and 1,020 data points, all of which are intended to be used in a regression task to predict the oral temperature of patients.

The primary goal of this study is to develop a predictive model that can accurately estimate oral temperature based on the combination of infrared thermography data and additional features. By leveraging advanced machine learning techniques, this project aims to identify the most relevant features and optimize the model to minimize prediction errors.

To evaluate the performance of the predictive models, we will utilize key metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The model's performance will also be compared against baseline systems, including a trivial system that always outputs the mean value from the training set and more standard models like 1-Nearest Neighbor (1NN) and Linear Regression.

In addition to building and evaluating the predictive model, this project will involve comprehensive data preprocessing, including handling missing data, feature selection, and dimensionality reduction, all of which are crucial steps in enhancing the model's performance and generalization capabilities.

The successful completion of this project will not only result in a reliable model for predicting oral temperature but will also provide insights into how different features contribute to the prediction, potentially aiding in the development of more effective diagnostic tools using infrared thermography.

2.2 Literature Review

Recent advancements in diagnostic tools have underscored the importance of integrating machine learning algorithms with infrared thermography (IRT) to improve temperature measurement accuracy, especially in the context of the COVID-19 pandemic. Razmara et al. (2024) systematically evaluated various regression models using heuristic feature engineering techniques to enhance the reliability of IRT-based temperature measurements. Their study demonstrated that a Convolutional Neural Network (CNN) model achieved the lowest Root Mean Square Error (RMSE) of 0.2223, outperforming other models, including non-neural network methods such as Binning, which achieved an RMSE of 0.2296. These findings highlight the potential of combining advanced feature engineering with machine learning to improve the effectiveness of non-contact diagnostic tools, providing a foundation for future research in non-invasive medical diagnostics [2].

3 Approach and Implementation

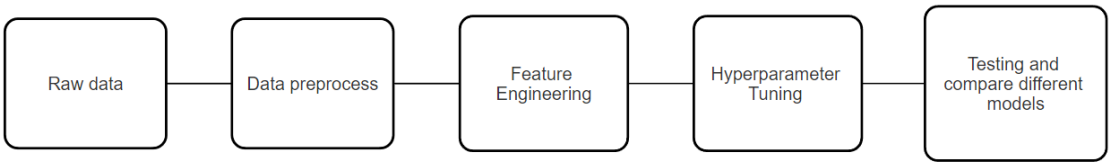

Figure 1: overview of the system work flow

3.1 Dataset Usage

The dataset was strategically divided into a training set and a test set to maintain clear boundaries throughout the model development process. This division ensured that model evaluation remained unbiased, with the test set reserved solely for final performance assessment.

3.1.1 Data Partitioning

The dataset, consisting of 1,020 data points, was split into a training set of 710 points and a test set of 310 points. This partitioning was designed to provide a robust training set for model development while preserving the test set for unbiased evaluation. The training set was exclusively used for all preprocessing steps, feature selection, and model training to prevent data leakage.

3.1.2 Preprocessing and Feature Engineering

Preprocessing and feature engineering were conducted entirely on the training set. This included imputing missing values, normalizing the features, and selecting relevant features based on their importance. By limiting these steps to the training data, the integrity of the test set was maintained, ensuring that model evaluation was not influenced by any prior exposure to the test data.

3.1.3 Cross-Validation and Model Tuning

During model training, 5-fold cross-validation was employed to optimize hyperparameters and evaluate model performance. The training set was divided into five folds, where each fold served as a validation set once, while the remaining four folds were used for training. This approach provided a robust framework for hyperparameter tuning, enabling the selection of the best model configuration based on validation performance.

3.1.4 Final Model Evaluation

The test set was used exclusively for the final evaluation after model development and tuning were completed. This final step was critical to assessing the model's generalization ability to new, unseen data. By reserving the test set for this purpose, the risk of overfitting was minimized, ensuring that the reported performance metrics accurately reflected the model's true predictive power.

3.2 Preprocessing

Preprocessing is a crucial step in preparing the dataset for machine learning models, ensuring the data is in a suitable format and mitigating any potential biases or inconsistencies. The preprocessing steps applied in this project included one-hot encoding, imputation, synthesis of data from multiple testing rounds, and normalization. Each technique was chosen based on its ability to enhance model performance and accuracy, as explained in detail below.

3.2.1 One-hot Encoding

One-hot encoding was used to convert categorical features into a binary format that is suitable for machine learning models. Categorical variables such as "Gender," "Age," and "Ethnicity" were transformed into a set of binary features, allowing the model to process these non-numeric attributes effectively. For instance, if the "Gender" feature has two categories, Male and Female, the encoding process generates two binary features, one for each category. Male is encoded as [1, 0], while Female is encoded as [0, 1].

This approach prevents the model from assuming any ordinal relationship between the categories, which could occur if numerical labels were used instead. By applying one-hot encoding, we eliminate the risk of the model misinterpreting categorical variables, thereby improving the robustness of the predictions.

3.2.2 Imputation

Imputation is a crucial step in handling missing data within a dataset. Missing values can introduce bias and negatively impact the performance of machine learning models. To address this, mean imputation was employed, which involves replacing missing values with the mean of the observed values for that feature. The process can be mathematically expressed as:

$$x_i = \begin{cases} x_i & \text{if } x_i \text{ is not missing,} \\ \bar{x} & \text{if } x_i \text{ is missing,} \end{cases} \tag{1}$$

where x_i is the feature value for the i-th data point, and $\bar{x}$ is the mean of the non-missing values for that feature.

Mean imputation was chosen because it preserves the overall distribution of the data while being simple and computationally efficient. This method is particularly effective when the proportion of missing data is small, as it maintains the dataset's statistical properties without significantly altering its variance.

3.2.3 Synthesis of Data from Multiple Rounds

In the dataset, each individual has temperature measurements recorded over four distinct rounds. To reduce the dimensionality and complexity of the dataset, these multiple rounds of data were synthesized by averaging the values across the four rounds for each feature.

Mathematically, the synthesis of data from multiple rounds can be expressed as:

$$\bar{x} = \frac{1}{4} \sum_{i=1}^{4} x_i, \tag{2}$$

where x_i represents the value of the feature in the i-th round, and $\bar{x}$ is the averaged value across all four rounds.

This approach of averaging not only simplifies the dataset by reducing the number of features but also helps in mitigating the potential variability or noise that could be present in any single round of data collection. By aggregating the data in this manner, we retain the essential information while ensuring that each feature represents a more stable and representative measure of the underlying physiological state.

Moreover, this method reduces the dimensionality of the dataset, which is beneficial for model training, as it helps to avoid overfitting and enhances the generalization ability of the model. The synthesized features, being averaged, are less prone to the impact of outliers or irregularities that might occur in individual rounds, thus contributing to the robustness of the subsequent analysis and modeling process.

3.2.4 Normalization

Normalization is a key preprocessing step that involves scaling the features of the dataset to ensure they have a consistent range and contribute equally to the model training process. This is particularly important when the features in a dataset have different units or scales, as it prevents features with larger magnitudes from dominating those with smaller magnitudes.

The standardization method, also known as Z-score normalization, was employed in this study. This technique transforms the data such that each feature has a mean of zero and a standard deviation of one. The mathematical expression for standardization is given by:

$$Z = \frac{X - \mu}{\sigma}, \tag{3}$$

where X is the original feature value, μ is the mean of the feature values, and σ is the standard deviation of the feature values.

Standardization was chosen because it maintains the distribution shape of the feature values while ensuring that all features contribute equally to the model. This is especially beneficial for algorithms like gradient descent, which converge faster when the input data is standardized. By applying normalization, the model becomes more robust and less sensitive to the scale of individual features.

3.3 Feature engineering and dimensionality adjustment

In this section, three feature engineering techniques are discussed: Pearson Correlation-based Feature Selection, Principal Component Analysis (PCA), and Kernel Principal Component Analysis (KPCA). Each of these methods contributes to reducing dimensionality and selecting relevant features for the model.

3.3.1 Pearson Correlation-based Feature Selection

Pearson correlation is used to measure the linear relationship between each feature and the target variable[3]. The Pearson correlation coefficient for a feature x and the target variable y is calculated using the following formula:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{4}$$

where x_i and y_i are the individual sample points for the feature and target variable, respectively, and $\bar{x}$ and $\bar{y}$ are their respective means.

Implementation The process begins by calculating the Pearson correlation coefficient for each numerical feature in the dataset. Features with an absolute correlation coefficient exceeding a predetermined threshold are selected as relevant for the model. Boolean features are also retained and added back to the selected feature set to ensure no loss of potentially important binary information. This approach effectively reduces the dimensionality of the dataset by focusing only on features that show a strong linear relationship with the target variable.

Parameter Selection The threshold for the Pearson correlation coefficient was set to 0.7. This value was chosen to strike a balance between retaining features that are significantly correlated with the target variable while excluding those that might add noise or irrelevant information.

3.3.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a technique used to reduce the dimensionality of a dataset by transforming the original features into a new set of orthogonal components called principal components[4]. These components capture the maximum variance in the data, with the first few components typically accounting for most of the variance. The mathematical formulation for PCA is:

$$\mathbf{Z} = \mathbf{X}\mathbf{W}, \tag{5}$$

where $\mathbf{X}$ is the original feature matrix, $\mathbf{W}$ is the matrix of eigenvectors (principal components), and $\mathbf{Z}$ represents the transformed feature matrix.

Implementation In practice, PCA is applied to the numerical features of the dataset. The algorithm identifies the directions (principal components) in which the data varies the most. The dataset is then projected onto these principal components, effectively reducing the number of dimensions while retaining the most important information. After the dimensionality reduction, the resulting principal components replace the original numerical features, while boolean features are retained and combined with these components.

Parameter Selection The number of principal components to retain is determined by the parameter *n_components*, which was set to explain 95% of the variance in the data. This choice ensures that the majority of the information in the dataset is preserved while reducing the complexity of the model.

3.3.3 Kernel Principal Component Analysis (KPCA)

Kernel Principal Component Analysis (KPCA) extends the traditional PCA method by allowing it to operate in a high-dimensional feature space using a kernel function[5]. KPCA is particularly useful for capturing non-linear relationships in the data. The mathematical formulation for KPCA is:

$$\mathbf{Z} = \kappa(\mathbf{X}, \mathbf{X})\mathbf{W}, \tag{6}$$

where $\kappa(\mathbf{X}, \mathbf{X})$ is the kernel matrix derived from the original feature matrix $\mathbf{X}$, and $\mathbf{W}$ represents the eigenvectors in the high-dimensional space.

Implementation The RBF (Radial Basis Function) kernel is commonly used in KPCA to map the original data into a higher-dimensional space where linear separations might be easier to find. Once the data is transformed using the kernel, PCA is applied in this new space to reduce dimensionality. The transformed features, known as kernel principal components, are then used in place of the original numerical features. Boolean features are again retained and merged with the kernel principal components.

Parameter Selection For KPCA, the parameter *n_components* was set to 20, indicating the number of principal components to retain after the transformation. The γ parameter, which controls the spread of the RBF kernel, was set to 0.1. These parameters were chosen based on experimental validation to ensure that the model captures non-linear patterns while maintaining manageable computational complexity.

3.4 Training, Classification or Regression, and Model Selection

In this section, hyperparameter tuning was performed on several models to identify the optimal configurations. The models evaluated include Lasso, Ridge, K-Nearest Neighbors (KNN), Support Vector Regression (SVR), Bayesian Ridge Regression, Random Forest Regressor, and Radial Basis Function (RBF) networks. Five-fold cross-validation was utilized for hyperparameter tuning of these models, allowing for an examination of the impact of three different feature engineering techniques—Pearson correlation, Principal Component Analysis (PCA), and Kernel PCA—on the optimal hyperparameters. For the Multi-Layer Perceptron (MLP), a traditional validation approach was used, where the dataset was split into a training set and a validation set in a 4:1 ratio. Given the computational complexity, cross-validation was not employed for the MLP. In the case of deep learning neural networks, feature engineering is generally unnecessary, as these models are capable of automatically learning relevant features from the data.

The following sections detail the process of identifying the optimal hyperparameters across various feature engineering techniques. Model performance was evaluated under different hyperparameter settings using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) as metrics. MAE served as the primary evaluation criterion, with the hyperparameters yielding the lowest MAE identified as optimal and highlighted in red within the tables(Due to space limitations, only a subset of the data in some tables is presented.). The corresponding figures illustrate the relationship between the hyperparameters and MAE across different feature engineering techniques.

3.4.1 Lasso Regression

Lasso Regression (Least Absolute Shrinkage and Selection Operator)[6] is a linear regression method that enhances the regular linear regression model by adding an L_1 penalty to the cost function. This penalty term encourages sparsity in the coefficient estimates, effectively performing feature selection by driving some coefficients to exactly zero.

The objective function for Lasso regression is defined as:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^T \beta\right)^2 + \alpha \|\beta\|_1 \right\}, \tag{7}$$

where:

- n is the number of observations.
- y_i represents the observed values.
- $\mathbf{x}_i$ denotes the vector of predictors for the i-th observation.
- β is the vector of coefficients.
- $\|\beta\|_1$ is the L_1 norm of the coefficient vector.
- α is the regularization parameter controlling the strength of the L_1 penalty.

The parameter α plays a crucial role in the Lasso model. It controls the trade-off between fitting the data well and keeping the model simple. A larger α value increases the penalty, which leads to more coefficients being shrunk towards zero, potentially resulting in a sparser model. Conversely, a smaller α value allows the model to fit the data more closely, at the risk of including more

features.Tuning the α parameter is essential to balance bias and variance in the model, often achieved through cross-validation techniques.

The range $[10^{-4}, 10^{2}]$ was explored, divided into 20 logarithmically spaced points. This logarithmic scale provides a detailed examination of values across several orders of magnitude, facilitating the identification of the optimal hyperparameter settings.

alpha	mae_avg	mse_avg	rmse_avg
0.0001	0.199769	0.064717	0.253877
0.000207	0.199308	0.064476	0.253396
0.000428	0.198847	0.064477	0.253400
0.000886	0.198680	0.064851	0.254144
0.001833	**0.198227**	**0.064714**	**0.253838**
0.003793	0.198791	0.065541	0.255401
0.007848	0.199905	0.066624	0.257554
0.016238	0.202653	0.068811	0.261784
0.033598	0.204679	0.071799	0.267292
0.069519	0.209485	0.078124	0.278497
0.143845	0.218937	0.094526	0.305395
0.297635	0.265459	0.164232	0.400639
0.615848	0.316093	0.238456	0.483924
1.274275	0.316093	0.238456	0.483924
2.636651	0.316093	0.238456	0.483924
5.455595	0.316093	0.238456	0.483924
11.288379	0.316093	0.238456	0.483924
23.357215	0.316093	0.238456	0.483924
48.329302	0.316093	0.238456	0.483924
100	0.316093	0.238456	0.483924

Table 1: None

alpha	mae_avg	mse_avg	rmse_avg
0.0001	0.210282	0.073194	0.270052
0.000207	0.209906	0.073095	0.269866
0.000428	0.209729	0.073138	0.269933
0.000886	0.209771	0.073413	0.270425
0.001833	0.209562	0.073254	0.270150
0.003793	0.209459	0.072944	0.269579
0.007848	0.209769	0.073238	0.270148
0.016238	0.209263	0.073322	0.270288
0.033598	**0.208724**	**0.074176**	**0.271715**
0.069519	0.209485	0.078124	0.278497
0.143845	0.218937	0.094526	0.305395
0.297635	0.265459	0.164232	0.400639
0.615848	0.316093	0.238456	0.483924
1.274275	0.316093	0.238456	0.483924
2.636651	0.316093	0.238456	0.483924
5.455595	0.316093	0.238456	0.483924
11.288379	0.316093	0.238456	0.483924
23.357215	0.316093	0.238456	0.483924
48.329302	0.316093	0.238456	0.483924
100	0.316093	0.238456	0.483924

Table 2: Pearson

alpha	mae_avg	mse_avg	rmse_avg
0.0001	0.222447	0.214749	0.391026
0.000207	0.222279	0.213069	0.389978
0.000428	0.222109	0.215358	0.391124
0.000886	**0.222086**	**0.222874**	**0.395223**
0.001833	0.222719	0.243337	0.406038
0.003793	0.223137	0.251438	0.410567
0.007848	0.224111	0.256797	0.414484
0.016238	0.230901	0.323507	0.449063
0.033598	0.244328	0.459818	0.509100
0.069519	0.244181	0.163805	0.376877
0.143845	0.243281	0.103931	0.320894
0.297635	0.249674	0.123508	0.347368
0.615848	0.251598	0.132886	0.360280
1.274275	0.284899	0.188077	0.428194
2.636651	0.316093	0.238456	0.483924
5.455595	0.316093	0.238456	0.483924
11.288379	0.316093	0.238456	0.483924
23.357215	0.316093	0.238456	0.483924
48.329302	0.316093	0.238456	0.483924
100	0.316093	0.238456	0.483924

Table 3: PCA

alpha	mae_avg	mse_avg	rmse_avg
0.0001	0.264260	0.156320	0.391330
0.000207	0.263741	0.156261	0.391218
0.000428	0.262787	0.156266	0.391145
0.000886	0.261167	0.156813	0.391651
0.001833	**0.260094**	**0.159734**	**0.394978**
0.003793	0.262604	0.167518	0.404174
0.007848	0.272176	0.184651	0.424203
0.016238	0.297430	0.215027	0.458857
0.033598	0.316093	0.238456	0.483924
0.069519	0.316093	0.238456	0.483924
0.143845	0.316093	0.238456	0.483924
0.297635	0.316093	0.238456	0.483924
0.615848	0.316093	0.238456	0.483924
1.274275	0.316093	0.238456	0.483924
2.636651	0.316093	0.238456	0.483924
5.455595	0.316093	0.238456	0.483924
11.288379	0.316093	0.238456	0.483924
23.357215	0.316093	0.238456	0.483924
48.329302	0.316093	0.238456	0.483924
100	0.316093	0.238456	0.483924

Table 4: Kernel PCA

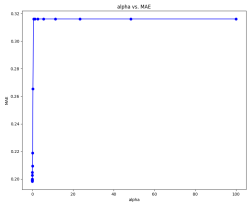

Figure 2: None

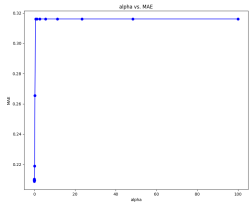

Figure 3: Pearson

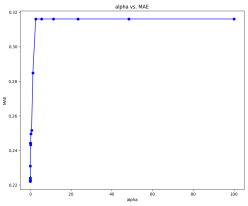

Figure 4: PCA

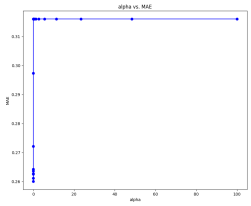

Figure 5: Kernel PCA

3.4.2 Ridge Regression

Ridge Regression, also known as Tikhonov regularization, is a linear regression method that includes an L_2 penalty in the cost function[7]. This penalty term discourages large coefficients by adding the squared magnitude of the coefficients as a penalty to the loss function, thereby reducing model complexity and mitigating multicollinearity.

The objective function for Ridge regression is defined as:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^T \beta\right)^2 + \alpha \|\beta\|_2^2 \right\}, \tag{8}$$

where:

- n is the number of observations.
- y_i represents the observed values.
- $\mathbf{x}_i$ denotes the vector of predictors for the i-th observation.
- β is the vector of coefficients.
- $\|\beta\|_2^2$ is the L_2 norm (squared) of the coefficient vector.
- α is the regularization parameter controlling the strength of the L_2 penalty.

The parameter α plays a crucial role in Ridge regression, as it governs the balance between minimizing the residual sum of squares and shrinking the magnitude of the regression coefficients. A larger value of α amplifies the penalty term, leading to smaller coefficients, which can help reduce model variance but at the cost of introducing some bias. Conversely, a smaller value of α diminishes the penalty, allowing the model to fit the training data more closely, potentially resulting in higher variance.

To fine-tune this parameter, a range of α values was carefully selected, spanning from 10^{-4} to 10^{2}. This range was divided into 20 logarithmically spaced values, providing a comprehensive exploration across several orders of magnitude. Such a selection ensures that the optimal regularization strength for the Ridge regression model can be identified, balancing the trade-off between bias and variance effectively.

alpha	mae_avg	mse_avg	rmse_avg
0.0001	0.202522	0.073250	0.268721
0.000207	0.202508	0.073171	0.268599
0.000428	0.202483	0.073019	0.268367
0.000886	0.202439	0.072749	0.267953
0.001833	0.202374	0.072326	0.267303
0.003793	0.202302	0.071772	0.266444
0.007848	0.202231	0.071133	0.265443
0.016238	0.202149	0.070328	0.264146
0.033598	0.201970	0.069183	0.262213
0.069519	0.201627	0.067757	0.259687
0.143845	0.201151	0.066425	0.257209
0.297635	0.200619	0.065492	0.255402
0.615848	0.200043	0.064950	0.254329
1.274275	0.199487	0.064671	0.253768
2.636651	**0.199186**	**0.064617**	**0.253654**
5.455595	0.199442	0.065097	0.254588
11.288379	0.200707	0.067649	0.259319
23.357215	0.203089	0.076747	0.273745
48.329302	0.206800	0.100050	0.301838
100	0.211770	0.141718	0.339161

Table 5: None

alpha	mae_avg	mse_avg	rmse_avg
0.0001	0.210635	0.073530	0.270658
0.000207	0.210631	0.073527	0.270653
0.000428	0.210622	0.073522	0.270645
0.000886	0.210607	0.073514	0.270629
0.001833	0.210582	0.073500	0.270605
0.003793	0.210551	0.073483	0.270575
0.007848	0.210528	0.073465	0.270545
0.016238	0.210519	0.073446	0.270513
0.033598	0.210527	0.073417	0.270463
0.069519	0.210518	0.073373	0.270384
0.143845	0.210482	0.073317	0.270282
0.297635	0.210397	0.073250	0.270157
0.615848	0.210237	0.073170	0.270005
1.274275	0.210004	0.073085	0.269841
2.636651	0.209713	0.073032	0.269736
5.455595	**0.209665**	**0.073075**	**0.269815**
11.288379	0.210192	0.073318	0.270270
23.357215	0.211223	0.073903	0.271363
48.329302	0.212771	0.074955	0.273308
100	0.214788	0.076492	0.276112

Table 6: Pearson

alpha	mae_avg	mse_avg	rmse_avg
0.0001	0.223001	0.222718	0.395610
0.000207	0.223001	0.222718	0.395610
0.000428	0.223000	0.222717	0.395609
0.000886	0.223000	0.222716	0.395609
0.001833	0.223000	0.222715	0.395608
0.003793	0.223000	0.222712	0.395605
0.007848	0.222998	0.222705	0.395600
0.016238	0.222995	0.222691	0.395590
0.033598	0.222990	0.222665	0.395570
0.069519	0.222979	0.222616	0.395532
0.143845	0.222960	0.222538	0.395467
0.297635	0.222923	0.222457	0.395380
0.615848	0.222857	0.222518	0.395336
1.274275	0.222744	0.223117	0.395535
2.636651	0.222574	0.224897	0.396326
5.455595	0.222408	0.228490	0.398063
11.288379	**0.222356**	**0.234341**	**0.400993**
23.357215	0.222370	0.243174	0.405483
48.329302	0.222785	0.256334	0.412250
100	0.224204	0.273108	0.421147

Table 7: PCA

alpha	mae_avg	mse_avg	rmse_avg
0.0001	0.264824	0.156519	0.391617
0.000207	0.264823	0.156519	0.391617
0.000428	0.264822	0.156519	0.391617
0.000886	0.264821	0.156518	0.391616
0.001833	0.264817	0.156518	0.391616
0.003793	0.264810	0.156518	0.391615
0.007848	0.264794	0.156517	0.391613
0.016238	0.264762	0.156516	0.391609
0.033598	0.264697	0.156513	0.391603
0.069519	0.264568	0.156511	0.391592
0.143845	0.264307	0.156518	0.391585
0.297635	0.263848	0.156575	0.391627
0.615848	0.263205	0.156846	0.391909
1.274275	0.262339	0.157852	0.393068
2.636651	**0.262105**	**0.160823**	**0.396620**
5.455595	0.264378	0.167564	0.404748
11.288379	0.271384	0.179004	0.418381
23.357215	0.282024	0.193521	0.435240
48.329302	0.292693	0.207779	0.451281
100	0.301499	0.219238	0.463794

Table 8: Kernel PCA

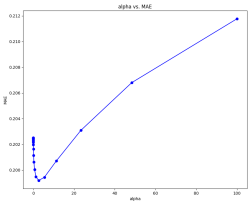

Figure 6: None

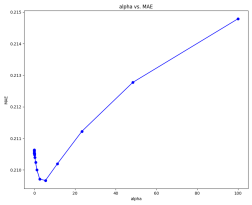

Figure 7: Pearson

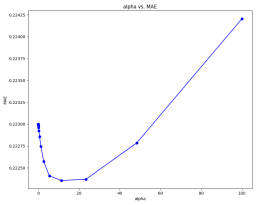

Figure 8: PCA

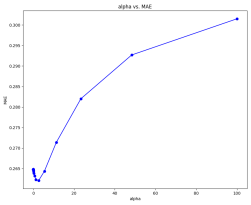

Figure 9: Kernel PCA

3.4.3 KNN Regression

K-Nearest Neighbors (KNN) Regression is a non-parametric method used for regression tasks[8]. Unlike linear models, KNN does not assume any specific form for the underlying data distribution. Instead, it predicts the target value for a given query point by averaging the target values of the k nearest neighbors in the feature space.

The KNN regression prediction for a query point $\mathbf{x}$ can be formulated as:

$$\hat{y}(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{k} y_i, \tag{9}$$

where:

- $\mathbf{x}$ is the query point for which the prediction is made.
- y_i represents the observed target values of the k nearest neighbors.
- k is the number of nearest neighbors considered.

The parameter k plays a critical role in KNN regression. A smaller value of k results in a model that is more sensitive to noise in the training data, leading to lower bias but potentially higher variance. Conversely, a larger k provides a smoother prediction by averaging over more neighbors, which reduces variance but can increase bias.

To systematically explore this trade-off between bias and variance, a range of k values was selected for tuning, spanning from $k = 1$ to $k = 30$. This range includes 30 consecutive integer values, allowing for a comprehensive evaluation of the K-Nearest Neighbors (KNN) model's performance across different neighborhood sizes. Such an approach facilitates the identification of the optimal k value that effectively balances the trade-off between underfitting and overfitting.

n_neighbors	mae_avg	mse_avg	rmse_avg
1	0.262113	0.119415	0.345403
2	0.231514	0.093272	0.304796
3	0.220516	0.083647	0.287997
4	0.217500	0.081313	0.283403
5	0.214014	0.079189	0.279521
6	0.211103	0.076943	0.275029
7	0.208501	0.075903	0.273242
8	0.205273	0.075520	0.272246
9	0.204930	0.075060	0.271367
10	**0.203049**	**0.075265**	**0.271587**
11	0.203515	0.074827	0.271034
12	0.204425	0.075598	0.272352
13	0.205488	0.076587	0.274047
14	0.204361	0.076783	0.274240
15	0.204629	0.077443	0.275204
16	0.205665	0.078401	0.276983
17	0.205866	0.078678	0.277428
18	0.205571	0.078564	0.277074
19	0.206597	0.079569	0.278740
20	0.207479	0.080618	0.280545
21	0.207743	0.081431	0.282098
22	0.208867	0.082054	0.283181
23	0.209238	0.082836	0.284509
24	0.209498	0.083719	0.286069
25	0.210346	0.084736	0.287804
26	0.210720	0.085327	0.288615
27	0.211210	0.085727	0.289237
28	0.212445	0.086798	0.291066
29	0.212768	0.087683	0.292559
30	0.213627	0.088733	0.294290

Table 9: None

n_neighbors	mae_avg	mse_avg	rmse_avg
1	0.267465	0.121599	0.348112
2	0.235986	0.093967	0.305637
3	0.223779	0.084888	0.290528
4	0.219489	0.082482	0.286097
5	0.214070	0.078626	0.279307
6	0.211843	0.077254	0.276852
7	0.209356	0.075962	0.274350
8	0.207139	0.074675	0.271777
9	0.205712	0.074540	0.271368
10	0.205585	0.075209	0.272378
11	0.203342	0.074281	0.270618
12	0.203228	0.074680	0.271116
13	0.203126	0.074750	0.271121
14	0.203385	0.075087	0.271755
15	**0.202822**	**0.074538**	**0.270648**
16	0.202843	0.074581	0.270652
17	0.203003	0.074711	0.270824
18	0.203009	0.074857	0.271080
19	0.203529	0.075533	0.272313
20	0.204415	0.076332	0.273676
21	0.204135	0.076734	0.274361
22	0.204526	0.077358	0.275405
23	0.204923	0.078120	0.276651
24	0.205358	0.078832	0.277952
25	0.205166	0.078920	0.278123
26	0.205068	0.079492	0.279087
27	0.205629	0.079912	0.279807
28	0.206175	0.080631	0.281082
29	0.205906	0.081021	0.281603
30	0.206242	0.081512	0.282532

Table 10: Pearson

n_neighbors	mae_avg	mse_avg	rmse_avg
1	0.277042	0.131845	0.362473
2	0.239577	0.095979	0.308959
3	0.222324	0.085098	0.290376
4	0.212817	0.079106	0.279585
5	0.212507	0.078565	0.278435
6	0.213087	0.078307	0.277930
7	0.209628	0.076634	0.274614
8	0.210361	0.077738	0.276456
9	0.209124	0.077359	0.275632
10	0.205465	0.075898	0.272834
11	0.204161	0.075666	0.272500
12	**0.203627**	**0.075805**	**0.272453**
13	0.204837	0.077350	0.275061
14	0.204396	0.077014	0.274393
15	0.205577	0.077927	0.275967
16	0.205163	0.078074	0.276202
17	0.205543	0.078588	0.277063
18	0.206037	0.078938	0.277754
19	0.206030	0.078929	0.277595
20	0.206313	0.079686	0.278876
21	0.206244	0.080286	0.279904
22	0.207161	0.081600	0.282199
23	0.208108	0.082411	0.283474
24	0.208348	0.083124	0.284765
25	0.210186	0.084389	0.287069
26	0.210599	0.084992	0.288105
27	0.210921	0.085704	0.289363
28	0.211615	0.086589	0.290771
29	0.212363	0.087907	0.292946
30	0.212040	0.088370	0.293718

Table 11: PCA

n_neighbors	mae_avg	mse_avg	rmse_avg
1	0.364014	0.348067	0.585928
2	0.327923	0.270045	0.514213
3	0.321174	0.236419	0.482267
4	0.308732	0.219253	0.464755
5	**0.301859**	**0.212775**	**0.457522**
6	0.311080	0.219384	0.465380
7	0.304638	0.214511	0.459886
8	0.305599	0.214194	0.459326
9	0.303552	0.212113	0.457559
10	0.302000	0.209852	0.455460
11	0.302401	0.210090	0.455767
12	0.303521	0.210755	0.456780
13	0.303657	0.209820	0.455618
14	0.304925	0.208634	0.454565
15	0.304962	0.206899	0.452821
16	0.306558	0.208736	0.454795
17	0.306703	0.208343	0.454197
18	0.306948	0.207947	0.453719
19	0.305741	0.206996	0.452782
20	0.308021	0.207563	0.453403
21	0.309272	0.209565	0.455546
22	0.311168	0.210891	0.457042
23	0.312535	0.212359	0.458557
24	0.313093	0.213213	0.459466
25	0.313885	0.214453	0.460694
26	0.314970	0.215462	0.461648
27	0.315618	0.216229	0.462360
28	0.315679	0.215924	0.462005
29	0.315966	0.216107	0.462145
30	0.317007	0.216834	0.462914

Table 12: Kernel PCA

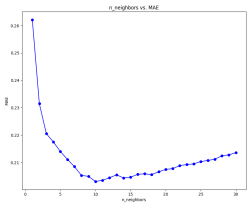

Figure 10: None

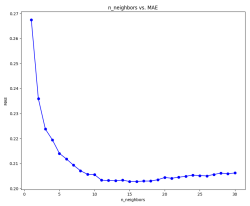

Figure 11: Pearson

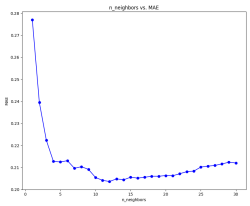

Figure 12: PCA

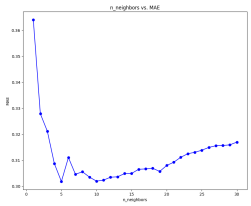

Figure 13: Kernel PCA

3.4.4 SVR

Support Vector Regression (SVR) extends the principles of Support Vector Machines (SVM) to regression problems[9]. The goal of SVR is to find a function that approximates the relationship between input features and the target variable within a specified margin of error, while also controlling model complexity. The optimization problem for SVR can be formulated as:

$$\min_{\mathbf{w},b,\xi,\xi^*} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{10}$$

subject to:

$$\begin{aligned} y_i - (\mathbf{w}\cdot\mathbf{x}_i + b) &\leq \epsilon + \xi_i \\ (\mathbf{w}\cdot\mathbf{x}_i + b) - y_i &\leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0 \end{aligned} \tag{11}$$

where:

- $\mathbf{w}$ is the weight vector.
- b is the bias term.
- ϵ defines the margin of tolerance within which no penalty is assigned to errors.
- C is the regularization parameter that controls the trade-off between model complexity (magnitude of $\mathbf{w}$) and the amount by which deviations larger than ϵ are tolerated.
- ξ_i and ξ_i^* are slack variables that allow some training points to be outside the ϵ-margin.

The parameter ϵ determines the width of the margin around the predicted values, within which errors are not penalized. A larger ϵ creates a wider margin, allowing for greater tolerance of deviations, but potentially at the cost of precision. The regularization parameter C controls the trade-off between minimizing the error on the training data and maintaining model simplicity.

For hyperparameter tuning, the range for C was set from 10^{-3} to 10^{3}, divided into 10 logarithmically spaced points. Similarly, the range for ϵ was chosen to span from 10^{-4} to 10^{1}, also divided into 10 logarithmic intervals. These ranges provide a comprehensive exploration of the hyperparameter space, facilitating the identification of the optimal settings for Support Vector Regression (SVR).

C	epsilon	mae_avg	mse_avg	rmse_avg
0.464158883	0.059948425	0.202008181	0.081348766	0.282266087
0.464158883	0.215443469	0.201378334	0.083339258	0.285831405
0.464158883	0.774263683	0.276517390	0.135252397	0.365625128
0.464158883	2.782559402	0.990140845	1.103647887	1.050313836
0.464158883	10	0.990140845	1.103647887	1.050313836
2.15443469	0.0001	0.203738467	0.074061365	0.270248740
2.15443469	0.000359381	0.203732457	0.074057612	0.270241354
2.15443469	0.00129155	0.203681224	0.074035371	0.270200972
2.15443469	0.004641589	0.203491104	0.073936853	0.270018376
2.15443469	0.016681005	0.203067607	0.073577590	0.269334361
2.15443469	**0.215443469**	**0.201166941**	**0.074322635**	**0.270406263**
2.15443469	0.774263683	0.270658744	0.125572152	0.351888322
2.15443469	2.782559402	0.990140845	1.103647887	1.050313836
2.15443469	10	0.990140845	1.103647887	1.050313836
10	0.0001	0.223120774	0.086998596	0.292607658
10	0.000359381	0.223089663	0.086977694	0.292571256
10	0.00129155	0.222998503	0.086895644	0.292428177
10	0.004641589	0.222681094	0.086617821	0.291947178
10	0.016681005	0.221394364	0.085561834	0.290120358

Table 13: None

C	epsilon	mae_avg	mse_avg	rmse_avg
0.464158883	0.215443469	0.207402114	0.084016259	0.286851454
0.464158883	0.774263683	0.274523898	0.127578834	0.355650329
0.464158883	2.782559402	0.990140845	1.103647887	1.050313836
0.464158883	10	0.990140845	1.103647887	1.050313836
2.15443469	0.0001	0.21517488	0.080479934	0.28137137
2.15443469	0.000359381	0.215167743	0.080480305	0.281372553
2.15443469	0.00129155	0.215162376	0.080488085	0.281388977
2.15443469	0.004641589	0.215081876	0.080449025	0.281320528
2.15443469	0.016681005	0.214851188	0.080451795	0.281290898
2.15443469	0.059948425	0.214692984	0.081084517	0.282379814
2.15443469	**0.215443469**	**0.207326991**	**0.079625043**	**0.279237962**
2.15443469	0.774263683	0.278078254	0.126969659	0.354770082
2.15443469	2.782559402	0.990140845	1.103647887	1.050313836
2.15443469	10	0.990140845	1.103647887	1.050313836
10	0.0001	0.226928012	0.092132604	0.301070015
10	0.000359381	0.22687911	0.092101635	0.301016318
10	0.00129155	0.226755825	0.09204988	0.300921794
10	0.004641589	0.226415174	0.091840301	0.300534507
10	0.016681005	0.225780002	0.091293655	0.299483248

Table 14: Pearson

C	epsilon	mae_avg	mse_avg	rmse_avg
0.1	0.016681005	0.213510883	0.105642437	0.321332938
0.1	0.059948425	0.213700548	0.106644862	0.322811934
0.1	0.215443469	0.215792937	0.109399511	0.326795422
0.1	0.774263683	0.303184323	0.168960721	0.408691657
0.1	2.782559402	0.990140845	1.103647887	1.050313836
0.1	10	0.990140845	1.103647887	1.050313836
0.464158883	0.0001	0.204243218	0.083703919	0.286335791
0.464158883	0.000359381	0.204219232	0.08369255	0.286316273
0.464158883	0.00129155	0.204165798	0.08368323	0.2863019
0.464158883	0.004641589	0.203995885	0.083639629	0.286244029
0.464158883	0.016681005	0.20352791	0.083566942	0.286076565
0.464158883	**0.059948425**	**0.202577276**	**0.083380958**	**0.28544175**
0.464158883	0.215443469	0.203432624	0.084795174	0.288183564
0.464158883	0.774263683	0.274953662	0.133474556	0.363352289
0.464158883	2.782559402	0.990140845	1.103647887	1.050313836
0.464158883	10	0.990140845	1.103647887	1.050313836
2.15443469	0.0001	0.203851636	0.075629479	0.272280358
2.15443469	0.000359381	0.203839532	0.075629987	0.272278606
2.15443469	0.00129155	0.203811805	0.075616171	0.272246776

Table 15: PCA

C	epsilon	mae_avg	mse_avg	rmse_avg
0.464158883	0.215443469	0.254507978	0.163576518	0.400457343
0.464158883	0.774263683	0.351050915	0.228889781	0.475224945
0.464158883	2.782559402	0.990140845	1.103647887	1.050313836
0.464158883	10	0.990140845	1.103647887	1.050313836
2.15443469	0.0001	0.256041863	0.154693349	0.390535976
2.15443469	0.000359381	0.256016052	0.154678981	0.390514173
2.15443469	0.00129155	0.255948205	0.154614405	0.39042914
2.15443469	0.004641589	0.255717262	0.154406578	0.390150808
2.15443469	0.016681005	0.25456296	0.15371875	0.38920679
2.15443469	0.059948425	0.251571007	0.151835102	0.386618382
2.15443469	**0.215443469**	**0.244863993**	**0.14938641**	**0.382922309**
2.15443469	0.774263683	0.322854528	0.200710815	0.444693887
2.15443469	2.782559402	0.990140845	1.103647887	1.050313836
2.15443469	10	0.990140845	1.103647887	1.050313836
10	0.0001	0.27076925	0.162077264	0.399686851
10	0.000359381	0.270741148	0.162060988	0.399664267
10	0.00129155	0.270614539	0.161964403	0.399541846
10	0.004641589	0.27020985	0.161633697	0.399117872
10	0.016681005	0.268642168	0.160425965	0.397532169

Table 16: Kernel PCA

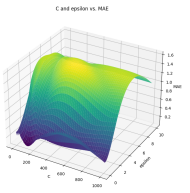

Figure 14: None

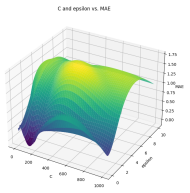

Figure 15: Pearson

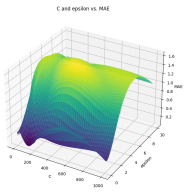

Figure 16: PCA

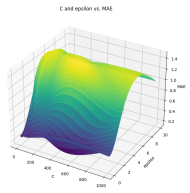

Figure 17: Kernel PCA

3.4.5 Bayesian Ridge Regression

Bayesian Ridge Regression[10] is an extension of Ridge Regression, where the regularization parameters are treated as random variables with associated priors, allowing for a fully probabilistic approach to regression. In this model, the weights $\mathbf{w}$ are assumed to have a Gaussian prior distribution, and the regularization parameters α and λ are assigned Gamma priors.

The objective in Bayesian Ridge Regression is to estimate the posterior distribution of the weights given the data. The prior for the weights is given by:

$$p(\mathbf{w}|\lambda) = \mathcal{N}(\mathbf{w}|0, \lambda^{-1}\mathbf{I}), \tag{12}$$

where λ controls the precision of the weights. The likelihood of the data given the weights is:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \alpha) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \alpha^{-1}\mathbf{I}), \tag{13}$$

where α is the precision of the noise.

In Bayesian Ridge Regression, hyperparameters α and λ are not fixed, but instead, they are modeled as random variables with their own priors:

$$p(\alpha) = \text{Gamma}(\alpha|\alpha_1, \alpha_2), \quad p(\lambda) = \text{Gamma}(\lambda|\lambda_1, \lambda_2). \tag{14}$$

Here, α_1 and λ_1 are the shape parameters of the Gamma distributions, which play a crucial role in determining the strength of the priors.

For hyperparameter tuning, the range for α_1 was defined to span from 10^{-6} to 10^{-2}, divided into 10 logarithmically spaced points. Similarly, λ_1 was explored over the same range, from 10^{-6} to 10^{-2}, also divided into 10 logarithmic intervals. These ranges allow for an extensive exploration of the prior distributions' influence on the regression model, facilitating the identification of optimal hyperparameters.

C	epsilon	mae_avg	mse_avg	rmse_avg
0.00129155	7.74E-06	0.201698048	0.070584914	0.26432886
0.00129155	2.15E-05	0.201698053	0.070584932	0.264328889
0.00129155	5.99E-05	0.201698067	0.070584983	0.264328972
0.00129155	0.00016681	0.201698106	0.070585124	0.264329203
0.00129155	0.000464159	0.201698215	0.070585516	0.264329846
0.00129155	0.00129155	0.201698517	0.070586607	0.264331635
0.00129155	0.003593814	0.201699357	0.070589645	0.264336613
0.00129155	0.01	0.201701695	0.070598103	0.264350472
0.003593814	1.00E-06	0.201698007	0.070584764	0.264328614
0.003593814	2.78E-06	0.201698008	0.070584766	0.264328618
0.003593814	7.74E-06	0.201698009	0.070584773	0.264328628
0.003593814	2.15E-05	0.201698015	0.070584791	0.264328658
0.003593814	5.99E-05	0.201698029	0.070584842	0.264328741
0.003593814	0.00016681	0.201698068	0.070584983	0.264328972
0.003593814	0.000464159	0.201698176	0.070585375	0.264329615
0.003593814	0.00129155	0.201698478	0.070586466	0.264331404
0.003593814	0.003593814	0.201699318	0.070589504	0.264336382
0.003593814	0.01	0.201701656	0.070597962	0.26435024
0.01	**1.00E-06**	**0.201697899**	**0.070584372**	**0.264327971**
0.01	2.78E-06	0.201697899	0.070584374	0.264327975

Table 17: None

C	epsilon	mae_avg	mse_avg	rmse_avg
0.00129155	2.78E-06	0.209960234	0.073203203	0.270055837
0.00129155	7.74E-06	0.209960235	0.073203203	0.270055838
0.00129155	2.15E-05	0.209960237	0.073203204	0.27005584
0.00129155	5.99E-05	0.209960243	0.073203206	0.270055844
0.00129155	0.00016681	0.20996026	0.073203213	0.270055857
0.00129155	0.000464159	0.209960306	0.073203232	0.270055893
0.00129155	0.00129155	0.209960435	0.073203285	0.270055993
0.00129155	0.003593814	0.209960793	0.073203432	0.27005627
0.00129155	0.01	0.209961799	0.073203842	0.270057043
0.003593814	1.00E-06	0.209960224	0.073203198	0.270055829
0.003593814	2.78E-06	0.209960224	0.073203199	0.27005583
0.003593814	7.74E-06	0.209960225	0.073203199	0.27005583
0.003593814	2.15E-05	0.209960227	0.0732032	0.270055832
0.003593814	5.99E-05	0.209960233	0.073203202	0.270055836
0.003593814	0.00016681	0.20996025	0.073203209	0.270055849
0.003593814	0.000464159	0.209960296	0.073203228	0.270055885
0.003593814	0.00129155	0.209960425	0.073203281	0.270055985
0.003593814	0.003593814	0.209960783	0.073203428	0.270056262
0.003593814	0.01	0.209961789	0.073203837	0.270057036
0.01	**1.00E-06**	**0.209960196**	**0.073203187**	**0.270055807**

Table 18: Pearson

C	epsilon	mae_avg	mse_avg	rmse_avg
0.00129155	2.78E-06	0.222432161	0.241348248	0.404578135
0.00129155	7.74E-06	0.222432161	0.241348258	0.404578139
0.00129155	2.15E-05	0.222432161	0.241348283	0.404578152
0.00129155	5.99E-05	0.222432162	0.241348355	0.404578189
0.00129155	0.00016681	0.222432163	0.241348556	0.404578291
0.00129155	0.000464159	0.222432167	0.241349113	0.404578574
0.00129155	0.00129155	0.222432178	0.241350664	0.404579362
0.00129155	0.003593814	0.222432208	0.241354978	0.404581555
0.00129155	0.01	0.222432292	0.241366984	0.404587656
0.003593814	1.00E-06	0.22243216	0.241348124	0.404578071
0.003593814	2.78E-06	0.22243216	0.241348127	0.404578073
0.003593814	7.74E-06	0.22243216	0.241348136	0.404578078
0.003593814	2.15E-05	0.222432161	0.241348162	0.404578091
0.003593814	5.99E-05	0.222432161	0.241348234	0.404578127
0.003593814	0.00016681	0.222432162	0.241348434	0.404578229
0.003593814	0.000464159	0.222432166	0.241348992	0.404578512
0.003593814	0.00129155	0.222432177	0.241350542	0.4045793
0.003593814	0.003593814	0.222432207	0.241354857	0.404581493
0.003593814	0.01	0.222432291	0.241366862	0.404587594
0.01	**1.00E-06**	**0.222432158**	**0.241347786**	**0.4045779**

Table 19: PCA

C	epsilon	mae_avg	mse_avg	rmse_avg
1.00E-06	0.000464159	0.263164552	0.157849242	0.393145023
1.00E-06	0.00129155	0.263164506	0.157849369	0.393145172
1.00E-06	0.003593814	0.263164377	0.157849721	0.393145587
1.00E-06	**0.01**	**0.263164019**	**0.157850702**	**0.393146742**
2.78E-06	1.00E-06	0.263164578	0.157849171	0.39314494
2.78E-06	2.78E-06	0.263164578	0.157849172	0.39314494
2.78E-06	7.74E-06	0.263164577	0.157849172	0.393144941
2.78E-06	2.15E-05	0.263164577	0.157849174	0.393144944
2.78E-06	5.99E-05	0.263164574	0.15784918	0.39314495
2.78E-06	0.00016681	0.263164569	0.157849197	0.39314497
2.78E-06	0.000464159	0.263164552	0.157849242	0.393145023
2.78E-06	0.00129155	0.263164506	0.157849369	0.393145172
2.78E-06	0.003593814	0.263164377	0.157849721	0.393145587
2.78E-06	0.01	0.263164019	0.157850702	0.393146742
7.74E-06	1.00E-06	0.263164578	0.157849171	0.39314494
7.74E-06	2.78E-06	0.263164578	0.157849171	0.39314494
7.74E-06	7.74E-06	0.263164577	0.157849172	0.393144941
7.74E-06	2.15E-05	0.263164577	0.157849174	0.393144944
7.74E-06	5.99E-05	0.263164575	0.15784918	0.39314495
7.74E-06	0.00016681	0.263164569	0.157849197	0.39314497

Table 20: Kernel PCA

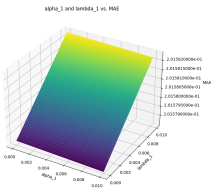

Figure 18: None

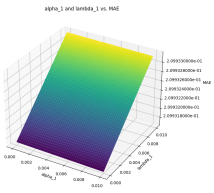

Figure 19: Pearson

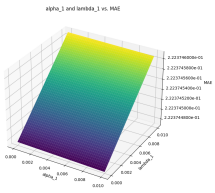

Figure 20: PCA

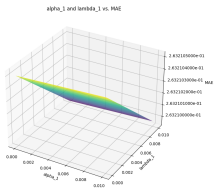

Figure 21: Kernel PCA

3.4.6 Random Forest Regression

Random Forest Regression is an ensemble learning method that constructs multiple decision trees during training and outputs the mean prediction of the individual trees[11]. This method combines the concepts of bagging (bootstrap aggregating) and feature randomness to create a model that is both robust to overfitting and effective in capturing complex relationships within the data.

In a Random Forest Regression model, the prediction for a given input $\mathbf{x}$ is computed by averaging the predictions from all the trees in the forest:

$$\hat{y} = \frac{1}{B} \sum_{i=1}^{B} T_i(\mathbf{x}), \tag{15}$$

where:

- $\hat{y}$ is the final predicted value for the input $\mathbf{x}$.
- $T_i(\mathbf{x})$ represents the prediction from the i-th decision tree.
- B is the total number of trees in the forest.

Key hyperparameters in the Random Forest Regression model include:
- **Number of trees** (*n_estimators*): This parameter specifies the number of trees in the forest. Increasing the number of trees typically enhances the model's performance by reducing variance, but it also increases computational complexity. - **Maximum depth of the trees** (*max_depth*): This parameter controls the maximum depth of each tree in the forest. A smaller *max_depth* can prevent overfitting by limiting the complexity of each tree, whereas a larger *max_depth* allows the trees to capture more detailed patterns in the data, which might increase the risk of overfitting.

For hyperparameter tuning, the number of estimators, *n_estimators*, was varied from 50 to 300, in increments of 50. This range allowed the model to be evaluated with 50, 100, 150, 200, 250, and 300 trees. Similarly, the maximum depth of the trees, *max_depth*, was adjusted between 5 and 20, with possible depths of 5, 10, 15, and 20. These ranges were chosen to provide a thorough exploration of the model's behavior under different configurations, aiding in the identification of the optimal settings for Random Forest Regression.

n_estimators	max_depth	mae_avg	mse_avg	rmse_avg
50	5	0.196435463	0.066507281	0.256747626
50	10	0.195915628	0.065466921	0.255034399
50	15	0.195861251	0.0652076	0.254518974
100	5	0.196973387	0.066943902	0.257757806
100	10	0.195947818	0.066212495	0.256497121
100	15	0.195332873	0.065082336	0.254193759
150	5	0.195313764	0.065601286	0.254990661
150	10	0.195106531	0.064719651	0.25346498
150	15	0.194821936	0.065202161	0.254559759
200	5	0.195585703	0.065795649	0.255347195
200	10	0.19445577	0.065158058	0.254389327
200	**15**	**0.19428571**	**0.064608745**	**0.25328501**
250	5	0.196151911	0.065961591	0.255753368
250	10	0.194951371	0.065297062	0.254645925
250	15	0.194815678	0.064868947	0.25386467

Table 21: None

n_estimators	max_depth	mae_avg	mse_avg	rmse_avg
50	5	0.198060198	0.06639875	0.256573155
50	10	0.201216293	0.06827339	0.260451302
50	15	0.202585877	0.068556307	0.260783381
100	5	0.197405984	0.065967992	0.255713305
100	10	0.200350415	0.0679414	0.259788316
100	15	0.20405013	0.0688461	0.261536067
150	5	0.197571682	0.066521436	0.256828847
150	10	0.199797203	0.067241378	0.258235136
150	15	0.202441427	0.068791048	0.261427988
200	**5**	**0.196040805**	**0.065343224**	**0.254516925**
200	10	0.199951134	0.067070934	0.258093148
200	15	0.20224896	0.068037481	0.259938246
250	5	0.196954807	0.065915733	0.255537522
250	10	0.201371333	0.067488539	0.258773243
250	15	0.200380881	0.067204958	0.258180591

Table 22: Pearson

n_estimators	max_depth	mae_avg	mse_avg	rmse_avg
50	5	0.207774802	0.079611008	0.279668088
50	10	0.201705985	0.073673604	0.269592793
50	15	0.203756335	0.076278704	0.274009538
100	5	0.206259688	0.077904664	0.27650061
100	10	0.202372754	0.07514589	0.272015629
100	**15**	**0.200252739**	**0.073066602**	**0.269767075**
150	5	0.206452575	0.078016579	0.276777563
150	10	0.200410681	0.073958161	0.269779199
150	15	0.201949618	0.074661186	0.270989542
200	5	0.206141673	0.078371779	0.277308519
200	10	0.201342119	0.074813083	0.271242711
200	15	0.200509828	0.073618861	0.269144508
250	5	0.205146846	0.077126769	0.275127688
250	10	0.201200892	0.073946021	0.269643725
250	15	0.201462597	0.074318922	0.270612342

Table 23: PCA

n_estimators	max_depth	mae_avg	mse_avg	rmse_avg
50	5	0.227351761	0.118909206	0.338503517
50	10	0.228605625	0.118909141	0.338547806
50	15	0.228350356	0.126392792	0.349794108
100	5	0.230932074	0.126860627	0.350315971
100	10	0.22874858	0.121426492	0.342227214
100	15	0.227665711	0.120622276	0.342368048
150	5	0.229735106	0.124026321	0.345895622
150	10	0.227811647	0.122240539	0.343131021
150	15	0.22729383	0.122036113	0.342787333
200	5	0.230229805	0.12553732	0.347313079
200	10	0.227282069	0.122105703	0.342983277
200	**15**	**0.226613828**	**0.121343327**	**0.34183039**
250	5	0.229759135	0.125208316	0.347182374
250	10	0.22747031	0.122821011	0.344154477
250	15	0.227548182	0.12330526	0.344608606

Table 24: Kernel PCA

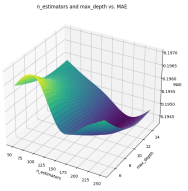

Figure 22: None

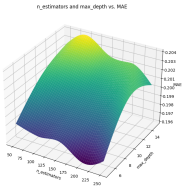

Figure 23: Pearson

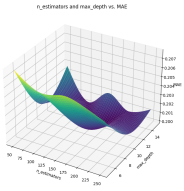

Figure 24: PCA

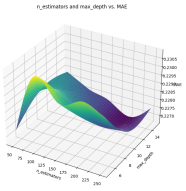

Figure 25: Kernel PCA

3.4.7 RBF Network

The Radial Basis Function (RBF) network is a type of artificial neural network that uses radial basis functions as activation functions[12]. In this approach, the centers of the RBF units are selected using the K-means clustering algorithm. The outputs of these RBF units are then used as inputs to a Ridge regression model, which produces the final output.

Selection of RBF Centers Using K-means The first step in constructing the RBF network is to determine the centers μ_m of the radial basis functions. These centers are selected using the K-means clustering algorithm. Given a set of training data points $\mathbf{x}_i$, the K-means algorithm partitions the data into K clusters, each represented by a cluster center μ_m. These centers μ_m serve as the RBF centers for the network.

RBF Kernel Transformation Once the RBF centers μ_m are determined, the next step is to apply the RBF kernel transformation to map the input data points into a higher-dimensional space. The Gaussian RBF kernel is commonly used and is defined as:

$$\phi_m(\mathbf{x}) = \exp\left(-\gamma\|\mathbf{x}-\mu_m\|^2\right) = \exp\left(-\frac{\|\mathbf{x}-\mu_m\|^2}{2\sigma^2}\right), \tag{16}$$

where γ is a parameter that controls the width of the RBF, $\mathbf{x}$ is the input data point, and μ_m is the center of the m-th RBF unit.

To select an appropriate value for γ, we first calculate the default value γ_d using the following approach:

1. Calculate the Average Spacing α:

The average spacing α between basis-function centers is given by:

$$\alpha = \frac{\Delta_x}{M^{1/D}}, \tag{17}$$

where Δ_x represents the extent (width) of the feature space in any one dimension, M is the number of basis-function centers, and D is the dimensionality of the feature space.

2. Determine σ and γ_d:

Choose the half-width of the RBF to cover a few average spacings, thus let $\sigma = 2\alpha$. The default value for γ_d is then calculated as:

$$\gamma_d = \frac{1}{2\sigma^2} = \frac{1}{8\alpha^2}. \tag{18}$$

Once γ_d is calculated, we use a set of scaled values for γ as $\gamma = \gamma' \cdot \gamma_d$.

Ridge Regression for Final Output The final step is to use the outputs of the RBF units as inputs to a Ridge regression model. The Ridge regression model is defined as:

$$\hat{y} = \sum_{m=1}^{M} w_m \phi_m(\mathbf{x}) + b, \tag{19}$$

where w_m are the weights associated with each RBF unit, b is the bias term, and M is the total number of RBF units (or centers). The objective of Ridge regression is to minimize the following loss function:

$$\min_{\mathbf{w},b} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \|\mathbf{w}\|_2^2 \right\}, \tag{20}$$

where λ is the regularization parameter, $\mathbf{w}$ is the weight vector, and $\hat{y}_i$ is the predicted output for the i-th training sample.

Parameter Selection and Model Training For this RBF network, the critical hyperparameters include the number of clusters K (used in K-means) and the RBF kernel width parameter γ. After calculating the default γ_d, the values for γ are explored over a range defined as $\gamma' \in \{0.01, 0.1, 1, 10, 100\}$, multiplying γ_d by each value in the list. The number of clusters K is tuned over the range $\{10, 20, \ldots, 100\}$, which includes values from 10 to 100 with a step of 10. Cross-validation is employed to select the optimal values of K and γ. After determining these parameters, the RBF network is trained on the training dataset, and the final model is evaluated using the Ridge regression framework.

gamma_pr	k	mae_avg	mse_avg	rmse_avg
0.01	10	0.312634	0.232468	0.477583
0.01	20	0.315219	0.282400	0.515480
0.01	30	0.311156	0.280943	0.511699
0.01	40	0.311467	0.351280	0.548387
0.01	50	0.312303	0.511034	0.612521
0.01	60	0.313022	0.648082	0.655404
0.01	70	0.309217	0.605324	0.639430
0.01	80	0.314030	0.868361	0.711976
0.01	90	0.312877	0.953310	0.730941
0.1	10	0.235754	0.128645	0.347116
0.1	20	0.234165	0.147642	0.360377
0.1	30	0.232923	0.136448	0.349700
0.1	40	0.232213	0.140826	0.352187
0.1	50	0.231640	0.141696	0.352198
0.1	60	0.231434	0.142061	0.352184
0.1	70	0.230223	0.132727	0.344119
0.1	80	0.230219	0.138727	0.348840
0.1	90	0.230210	0.133768	0.344887
1	10	0.232413	0.094715	0.306450
1	20	0.225533	0.089156	0.297474
1	30	0.217071	0.082517	0.285993
1	40	0.212206	0.079189	0.280155
1	50	0.207840	0.075752	0.274020
1	60	0.205244	0.074208	0.271162
1	70	0.203738	0.072278	0.268781
1	80	0.201610	0.072681	0.268192
1	**90**	**0.198720**	**0.069607**	**0.262639**
10	10	0.238154	0.115971	0.337634
10	20	0.227984	0.107205	0.324080
10	30	0.219113	0.099130	0.311348
10	40	0.218230	0.099081	0.312012
10	50	0.216846	0.095672	0.306306
10	60	0.215215	0.093774	0.303233
10	70	0.213201	0.093357	0.302476
10	80	0.211328	0.092334	0.300360
10	90	0.212405	0.091826	0.299855
100	10	0.311978	0.233282	0.478399
100	20	0.310415	0.231435	0.476526
100	30	0.308737	0.230328	0.475161
100	40	0.309346	0.230356	0.475203
100	50	0.308239	0.230012	0.474884
100	60	0.308192	0.229972	0.474766
100	70	0.306965	0.228792	0.473547
100	80	0.307412	0.228821	0.473593
100	90	0.307831	0.229732	0.474460

Table 25: without

gamma_pr	k	mae_avg	mse_avg	rmse_avg
0.01	10	0.307025	0.222012	0.466931
0.01	20	0.300659	0.209400	0.453189
0.01	30	0.294082	0.196164	0.438734
0.01	40	0.286638	0.183605	0.424702
0.01	50	0.280045	0.171685	0.411029
0.01	60	0.275048	0.162975	0.400598
0.01	70	0.272244	0.157936	0.394508
0.01	80	0.264074	0.145549	0.378644
0.01	90	0.263895	0.143662	0.376121
0.1	10	0.217361	0.082422	0.285751
0.1	20	0.214912	0.079032	0.279902
0.1	30	0.214790	0.078071	0.278234
0.1	40	0.213960	0.076996	0.276410
0.1	50	0.213518	0.076441	0.275472
0.1	60	0.213472	0.076341	0.275311
0.1	70	0.213055	0.075940	0.274584
0.1	80	0.212633	0.075568	0.273964
0.1	90	0.212116	0.075324	0.273489
1	10	0.218305	0.083375	0.287474
1	20	0.211236	0.078118	0.278232
1	30	0.207678	0.074914	0.273423
1	40	0.205544	0.073324	0.269460
1	50	0.205170	0.071905	0.267108
1	60	0.204281	0.071272	0.265911
1	70	0.204265	0.071395	0.266133
1	80	0.202963	0.070395	0.264192
1	**90**	**0.202661**	**0.070275**	**0.263921**
10	10	0.219762	0.099617	0.312566
10	20	0.218075	0.097549	0.309077
10	30	0.219227	0.098082	0.310083
10	40	0.214722	0.091965	0.300087
10	50	0.214224	0.090867	0.298830
10	60	0.214670	0.090922	0.298982
10	70	0.214441	0.090774	0.298515
10	80	0.211266	0.088189	0.294179
10	90	0.214251	0.090431	0.297792
100	10	0.293967	0.213060	0.457122
100	20	0.293091	0.212016	0.456227
100	30	0.294925	0.213744	0.458009
100	40	0.292764	0.209767	0.453787
100	50	0.293387	0.210272	0.454422
100	60	0.293137	0.208514	0.452363
100	70	0.291224	0.207496	0.451106
100	80	0.289495	0.205572	0.449190
100	90	0.293604	0.209843	0.453846

Table 26: Pearson

gamma_pr	k	mae_avg	mse_avg	rmse_avg
0.01	10	0.309759	0.235194	0.479487
0.01	20	0.309333	0.327595	0.538497
0.01	30	0.306959	0.448550	0.588300
0.01	40	0.301626	0.447945	0.581635
0.01	50	0.299630	0.574922	0.622408
0.01	60	0.297029	0.598147	0.625386
0.01	70	0.291059	0.656287	0.636468
0.01	80	0.291664	0.864246	0.687620
0.01	90	0.292148	0.941631	0.705103
0.1	10	0.231782	0.108015	0.323360
0.1	20	0.229971	0.107439	0.321223
0.1	30	0.230143	0.106377	0.319701
0.1	40	0.228171	0.101731	0.313637
0.1	50	0.227189	0.099100	0.310157
0.1	60	0.226640	0.098010	0.308774
0.1	70	0.225225	0.094437	0.303850
0.1	80	0.225191	0.097396	0.307458
0.1	90	0.224983	0.097142	0.307083
1	10	0.228564	0.092905	0.303262
1	20	0.218756	0.084592	0.289680
1	30	0.207946	0.077458	0.276677
1	40	0.204415	0.074178	0.270862
1	50	0.200902	0.071728	0.266356
1	60	0.198822	0.070993	0.264919
1	70	0.197849	0.069356	0.261915
1	80	0.197009	0.069017	0.261237
1	**90**	**0.195972**	**0.068694**	**0.260561**
10	10	0.245267	0.127056	0.353482
10	20	0.236159	0.119925	0.343145
10	30	0.229459	0.113917	0.334225
10	40	0.224601	0.109243	0.327588
10	50	0.223576	0.106752	0.323841
10	60	0.224322	0.109543	0.327983
10	70	0.220184	0.104820	0.320729
10	80	0.220214	0.102108	0.316556
10	90	0.222667	0.104406	0.320249
100	10	0.313182	0.234913	0.480158
100	20	0.312296	0.234045	0.479171
100	30	0.311936	0.233519	0.478655
100	40	0.311786	0.233241	0.478400
100	50	0.312211	0.233731	0.478948
100	60	0.312202	0.233845	0.478970
100	70	0.311469	0.232715	0.477926
100	80	0.311806	0.232838	0.478048
100	90	0.312208	0.233443	0.478655

Table 27: PCA

gamma_pr	k	mae_avg	mse_avg	rmse_avg
0.01	10	0.316069	0.238378	0.483849
0.01	20	0.316052	0.238303	0.483778
0.01	30	0.315924	0.238128	0.483609
0.01	40	0.315864	0.237992	0.483472
0.01	50	0.315691	0.237805	0.483293
0.01	60	0.315896	0.237764	0.483252
0.01	70	0.315568	0.237543	0.483044
0.01	80	0.315690	0.237398	0.482894
0.01	90	0.315574	0.237405	0.482906
0.1	10	0.316595	0.235871	0.481582
0.1	20	0.317143	0.234556	0.480335
0.1	30	0.316044	0.232633	0.478460
0.1	40	0.315454	0.231251	0.477001
0.1	50	0.315124	0.229995	0.475745
0.1	60	0.314542	0.228840	0.474534
0.1	70	0.313291	0.227254	0.472903
0.1	80	0.312962	0.226034	0.471598
0.1	90	0.312473	0.225458	0.471031
1	10	0.321499	0.230912	0.477298
1	20	0.319669	0.221849	0.467945
1	30	0.309017	0.209391	0.454442
1	40	0.303247	0.202101	0.446062
1	50	0.300166	0.197806	0.441215
1	60	0.295581	0.192702	0.435183
1	70	0.294748	0.190524	0.432755
1	80	0.290193	0.185660	0.427040
1	90	0.292805	0.187298	0.429297
10	10	0.316305	0.225444	0.472110
10	20	0.311749	0.215335	0.461316
10	30	0.300262	0.204099	0.449067
10	40	0.296440	0.198477	0.442878
10	50	0.292195	0.195207	0.439388
10	60	0.291203	0.195352	0.439536
10	70	0.287014	0.189970	0.433622
10	**80**	**0.284451**	**0.188622**	**0.431971**
10	90	0.284488	0.187401	0.430721
100	10	0.314455	0.235568	0.480924
100	20	0.312709	0.232800	0.478209
100	30	0.303413	0.217588	0.462594
100	40	0.302232	0.215422	0.460328
100	50	0.297468	0.206895	0.451744
100	60	0.297590	0.206480	0.451401
100	70	0.299642	0.211773	0.456440
100	80	0.291960	0.199300	0.443591
100	90	0.295905	0.205529	0.450592

Table 28: Kernel PCA

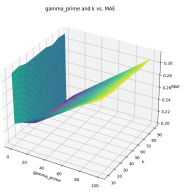

Figure 26: None

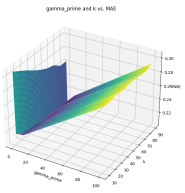

Figure 27: Pearson

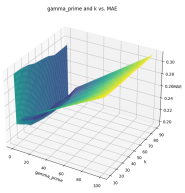

Figure 28: PCA

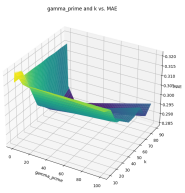

Figure 29: Kernel PCA

3.4.8 MLP

The Multi-Layer Perceptron (MLP) is a type of feedforward artificial neural network[13] that is widely used for various regression and classification tasks due to its ability to model complex, non-linear relationships. The MLP used in this study is designed with three hidden layers and includes mechanisms such as Batch Normalization and Dropout to improve model performance and generalization.

The input layer's size is determined by the second dimension of the input matrix, reflecting the number of features in the dataset. The hidden layers then transform this input through a series of linear and non-linear operations.

The learning rate used in training the MLP is not static; instead, it follows a decay schedule to gradually reduce the learning rate as training progresses. This allows the model to make large adjustments in the early stages of training, while fine-tuning the weights as it approaches convergence. The learning rate at epoch t is given by:

$$\eta_t = \eta_0 \times \gamma^{\left\lfloor \frac{t}{\text{step size}} \right\rfloor}, \tag{21}$$

where η_t is the learning rate at epoch t, η_0 is the initial learning rate, γ is the decay factor, and the step size controls the frequency of decay.

The architecture of the MLP is as follows:

The first hidden layer comprises 128 neurons, applying a linear transformation followed by Batch Normalization and a Dropout layer with a dropout rate of 0.1. The output of this layer is computed as:

$$\mathbf{h}_1 = \text{Dropout}\left(\text{ReLU}\left(\text{BatchNorm}\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right)\right)\right), \tag{22}$$

where $\mathbf{x}$ is the input vector, $\mathbf{W}_1$ and $\mathbf{b}_1$ are the weights and biases of the first layer.

The second hidden layer, consisting of 64 neurons, similarly applies a linear transformation, Batch Normalization, and Dropout:

$$\mathbf{h}_2 = \text{Dropout}\left(\text{ReLU}\left(\text{BatchNorm}\left(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2\right)\right)\right), \tag{23}$$

where $\mathbf{W}_2$ and $\mathbf{b}_2$ are the weights and biases of the second layer.

The third hidden layer comprises 32 neurons, with its output computed as:

$$\mathbf{h}_3 = \text{Dropout}\left(\text{ReLU}\left(\text{BatchNorm}\left(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3\right)\right)\right), \tag{24}$$

where $\mathbf{W}_3$ and $\mathbf{b}_3$ are the weights and biases of the third layer.

Finally, the output layer, consisting of a single neuron, produces the model's prediction:

$$\hat{y} = \mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4, \tag{25}$$

where $\mathbf{W}_4$ and $\mathbf{b}_4$ are the weights and biases of the output layer.

Hyperparameters were carefully selected to optimize the MLP's performance. The number of neurons in each hidden layer was chosen to provide a balance between model capacity and overfitting risk. The dropout rate of 0.1 was applied consistently across all layers to introduce regularization.

Two optimizers were considered: Adam and Stochastic Gradient Descent (SGD). The Adam optimizer was used with a learning rate decay schedule defined by the equation above, with a step size of 200 epochs. The SGD optimizer, with momentum set to 0.9, also followed a learning rate decay schedule, but with a step size of 100 epochs.

The model was trained for 700 epochs, which provided sufficient time for convergence while preventing overfitting. This architecture and hyperparameter configuration were designed to ensure robust performance on the predictive task.

optimizer	learning_rate	momentum	mae_avg	mse_avg	rmse_avg
Adam	**0.01**	**None**	**0.217547427**	**0.104250311**	**0.3099278**
SGD	0.001	0.9	0.235978521	0.105812202	0.319346299

Table 29: MLP performance with different parameters

4 Results and Analysis: Comparison and Interpretation

	train_mae	train_mse	train_rmse	test_mae	test_mse	test_rmse
Trivial System	0.31588415	0.237727852	0.487573432	0.353235348	0.311589254	0.558201804
1NN	0	0	0	0.275967742	0.13808871	0.371602892
Linear Regression	0.186846861	0.056671198	0.238057131	0.220180457	0.080433906	0.28360872
Lasso	0.190381042	0.059607656	0.244146791	0.207656555	0.072647803	0.269532564
Ridge	0.188214815	0.057721913	0.240253851	0.214137362	0.076533131	0.27664622
KNN	0.187802817	0.06489036	0.254735863	0.190688172	0.064418351	0.253807705
SVR	0.150637444	0.031719209	0.178098874	0.211236663	0.07881704	0.280743726
Bayesian Ridge Regression	0.18992261	0.058778401	0.242442573	0.21069297	0.074542857	0.273025378
Random Forest	**0.073123069**	**0.009007136**	**0.094905934**	**0.187663949**	**0.06077273**	**0.246521258**
RBF Network	**0.191238849**	**0.062944209**	**0.250886845**	**0.18767581**	**0.058927629**	**0.242750136**
MLP	0.19843887	0.065419222	0.250659664	0.221239857	0.083306677	0.284184285

Table 30: Performance metrics for various models

The table above presents the training and testing losses for various models, each optimized with the best feature engineering techniques and hyperparameters. It is evident that the most effective models are the Random Forest Regressor (without feature selection, with `n_estimators=200` and `max_depth=15`) and the RBF Network (with PCA-based feature selection, `gamma_prime=1`, `k=90`).

The exceptional performance of these two models in regression tasks can be attributed to several factors:

Firstly, the Random Forest Regressor excels due to its ensemble nature, which reduces overfitting and improves generalization by averaging the predictions of multiple decision trees. The model's ability to handle complex, nonlinear relationships and interactions between features is further enhanced by the deep trees and a sufficiently large number of estimators.

Secondly, the RBF Network's strong performance stems from its architecture, which combines the Radial Basis Function as the hidden layer with a ridge regression output layer. This design allows the model to capture intricate, non-linear patterns in the data, while the ridge regression component helps in managing multicollinearity and preventing overfitting.

5 Libraries used and what you coded yourself

The following libraries were utilized in this project: `pandas` (primarily for handling tabular data), `sklearn` (mainly for imputing missing values and applying machine learning models), and `torch` (for implementing the MLP neural network).

In this project, both cross-validation and hyperparameter tuning were custom-implemented, using a brute-force search approach. Additionally, the RBF Network and neural network architectures were specifically designed for this application.

6 Summary and conclusions

In this project, we meticulously preprocessed the dataset, conducted feature selection, and performed hyperparameter tuning to evaluate the performance of multiple models in predicting oral temperature. Among the models tested, Random Forest and the Radial Basis Function Network (RBF Network) exhibited superior performance, demonstrating their effectiveness in this regression task.

Looking ahead, it would be highly intriguing to explore the use of different types of neural networks, such as ResNet[14] and Vision Transformers[15]—originally designed for image classification—in regression tasks. These models, known for their deep learning capabilities and sophisticated architectures, may offer significant improvements in predictive accuracy when applied to regression problems in medical diagnostics.

References

[1] Infrared Thermography Temperature Dataset, available at `https://archive.ics.uci.edu/dataset/925/infrared+thermography+temperature+dataset`, accessed August 19, 2024.

[2] Razmara, P., Khezresmaeilzadeh, T., & Jenkins, B. K. (2024). *Fever Detection with Infrared Thermography: Enhancing Accuracy through Machine Learning Techniques*. arXiv preprint arXiv:2407.15302. Available at: `https://doi.org/10.48550/arXiv.2407.15302`.

[3] Pearson, K. (1895). *Note on Regression and Inheritance in the Case of Two Parents*. Proceedings of the Royal Society of London, 58, 240-242.

[4] Hotelling, H. (1933). *Analysis of a complex of statistical variables into principal components*. Journal of Educational Psychology, 24(6), 417-441.

[5] Schölkopf, B., Smola, A., & Müller, K. R. (1997). *Kernel principal component analysis*. In *Proceedings of the 7th International Conference on Artificial Neural Networks (ICANN)*, 583-588.

[6] Tibshirani, R. (1996). *Regression shrinkage and selection via the lasso*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 58(1), 267-288.

[7] Hoerl, A. E., & Kennard, R. W. (1970). *Ridge regression: Biased estimation for nonorthogonal problems*. Technometrics, 12(1), 55-67.

[8] Altman, N. S. (1992). *An introduction to kernel and nearest-neighbor nonparametric regression*. The American Statistician, 46(3), 175-185.

[9] Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., & Vapnik, V. (1997). *Support vector regression machines*. In Advances in Neural Information Processing Systems (NIPS) (pp. 155-161).

[10] Michimae, H., Matsunami, M., & Emura, T. (2022). *Bayesian ridge estimators based on copula-based joint prior distributions for regression coefficients*. Computational Statistics, 37, 2741–2769. Available at: `https://doi.org/10.1007/s00180-022-01213-8`.

[11] Breiman, L. (2001). *Random forests*. Machine Learning, 45(1), 5-32.

[12] Broomhead, D. S., & Lowe, D. (1988). *Radial basis functions, multivariable functional interpolation and adaptive networks*. Royal Signals and Radar Establishment Malvern (United Kingdom).

[13] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning representations by back-propagating errors*. Nature, 323(6088), 533-536.

[14] He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[15] Dosovitskiy, A., et al. (2021). *An image is worth 16x16 words: Transformers for image recognition at scale*. In International Conference on Learning Representations (ICLR).