Statistical Pattern Recognition
Assignment-2

Kapil Kumar Bhardwaj – CS24MT012
Manish Bisht – CS24MT022
Mridul Chandrawanshi –CS24MT002

TABLE OF CONTENTS:

CHAPTER 1
Polynomial Curve Fitting Dataset 1

Introduction

Polynomial curve fitting is a fundamental technique in statistical pattern recognition, used to model the relationship between a dependent variable and one or more independent variables. By fitting a polynomial of varying degrees to the data, we can capture complex patterns and relationships, improving our predictive accuracy. This chapter explores polynomial curve fitting applied to a univariate dataset, examining model complexity, regularization, and their impact on performance.

(1) ABOUT THE DATASET:

The dataset 1 consists of two numerical columns, the first column is assumed to be **X** i.e., independent value and second column **Y** i.e., dependent values. The data is Univariate input i.e., single variable input, and the goal is finding the relationship between **X** and **Y**.
Dataset consist 1001 (X_n, Y_n) pairs of values.
Sample dataset is:

X	Y
0.925	5.9258
0.511	1.8809
0.238	0.55075
0.59	2.4825

Sample Dataset 1

(2) STEPS AND PROCEDURE:

Step 1: Prepare the Data.
- (a) Split the data in train (10, 50,100 and 70%) and test (30%).

Step 2: Creating Polynomial feature.
- (a) $X_{poly} = [1, x, x^2, \dots, x^d]$

Step 3: Train the model.
- (a) Train the model for different complexities i.e., M = {2,3,4,5,6,7,8,9}
- (b) Training with Regularization

$$J(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{d} w_j^2, \quad \lambda \text{ is regularization parameter.}$$

Step 4: Choosing the best model.
- (a) Cross-Validation: Splitting the training data into further multiple parts and validating the model several times. This will help in selecting the best polynomial degree and regularization coefficients.
- (b) Calculate MSE: For each polynomial degree and regularization value, compute the MSE for both training and test sets.

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

Step 5: Visualizing the results.
- (a) Create plots showing the polynomial approximation for various training sizes and polynomial degrees.
- (b) Computing weight values before and after applying regularization for each polynomial degree
- (c) Generate plots comparing model outputs to actual target outputs for both training and test dataset.

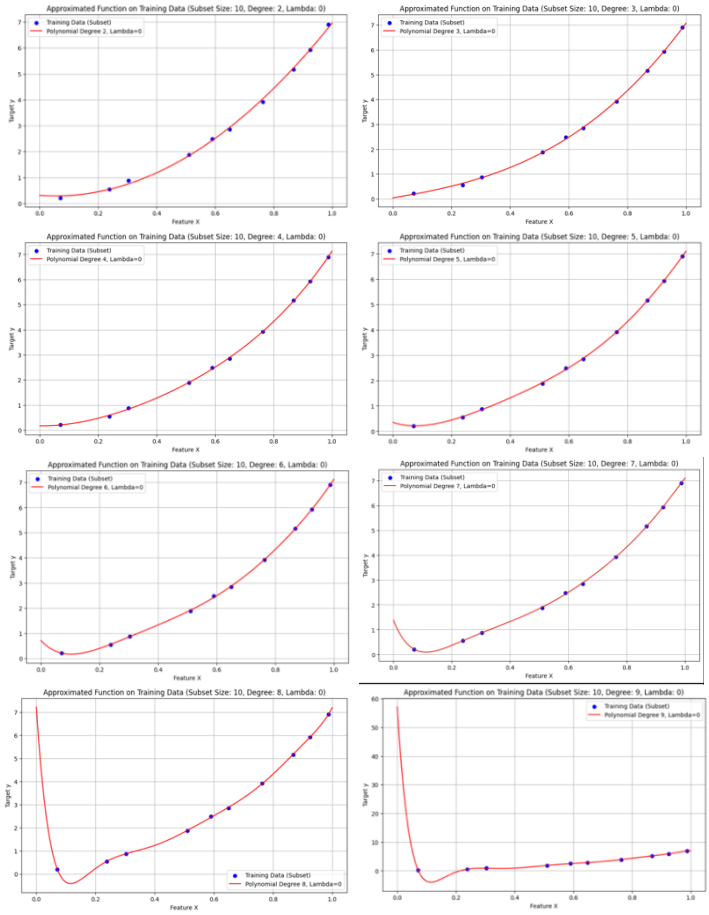
Step 6: Finalize the model.
- (a) Choose the model with the best performance (lowest MSE) and appropriate complexity based on cross-validation results.
- (b) Ensure the selected model generalizes well by evaluating it on the test dataset.

(3) MODLE COMPLEXITY AND REGULARIZATION VALUES USED:

(a) Model Complexity m = {2,3,4,5,6,7,8,9}
(b) Regularization values $\lambda = \{0, 0.000001, 0.01, 0.1, 1\}$

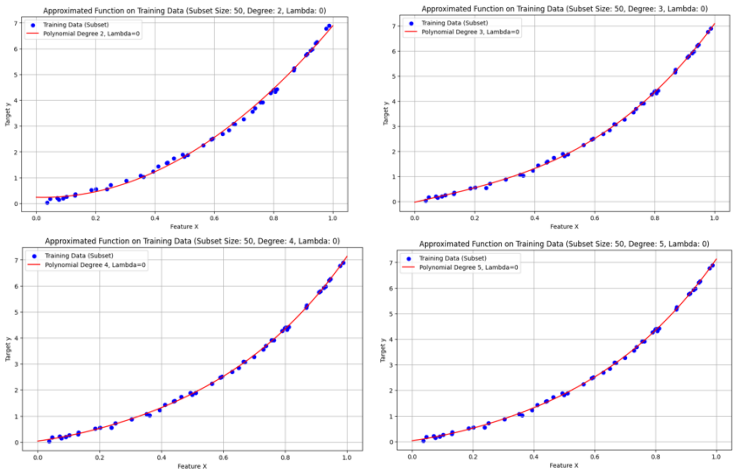
(4) APPROXIMATING FUNCTION FOR DIFFERENT TRAINING SIZES:

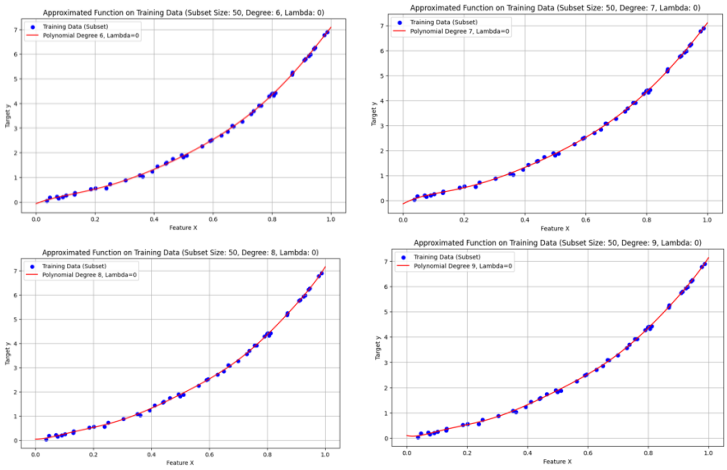
(a) Plots For dataset size = 10, and Model Complexities = {2-9}



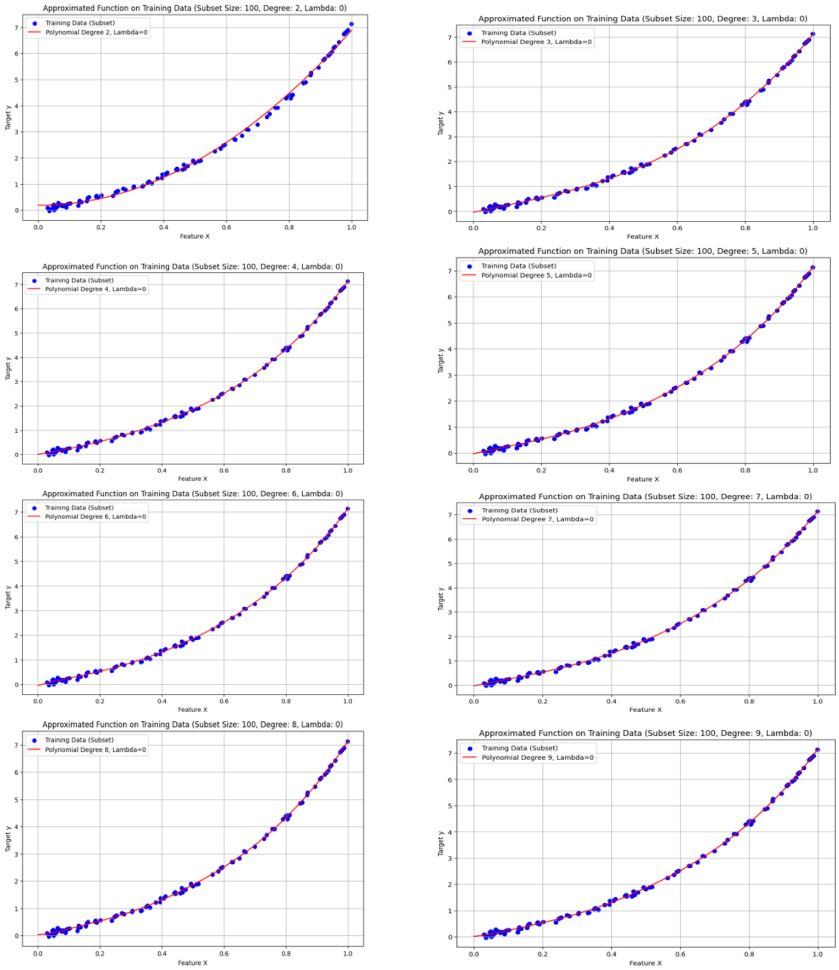
OBSERVATION: According to the graphs plotted model is fine a

(b) Plots For dataset size = 50, and Model Complexities = {2-9}

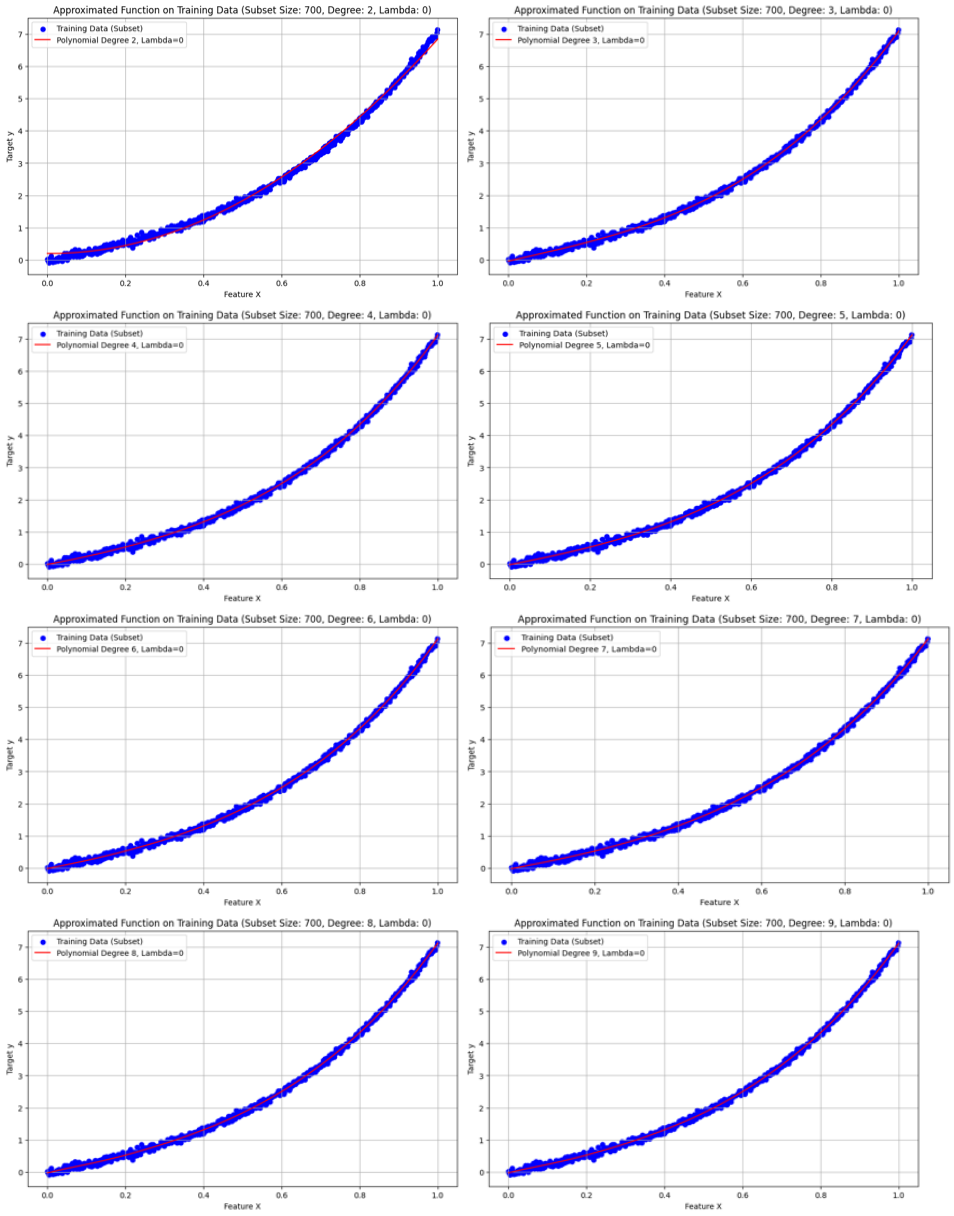




(c) Plots For dataset size = 100, and Model Complexities = {2-9}

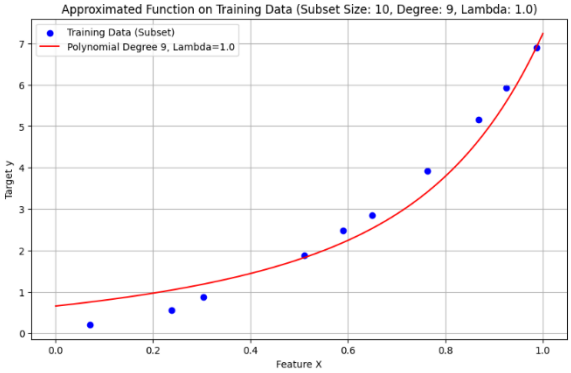


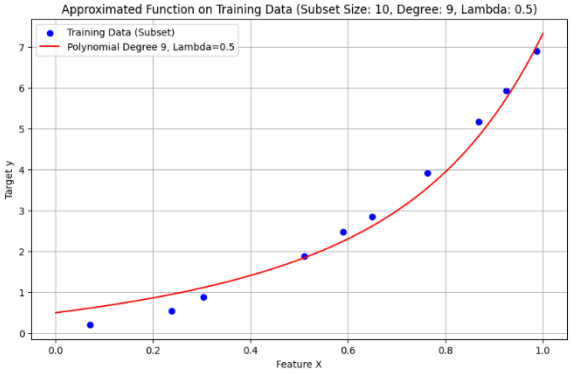
(d) Plots For dataset size = 700, and Model Complexities = {2-9}

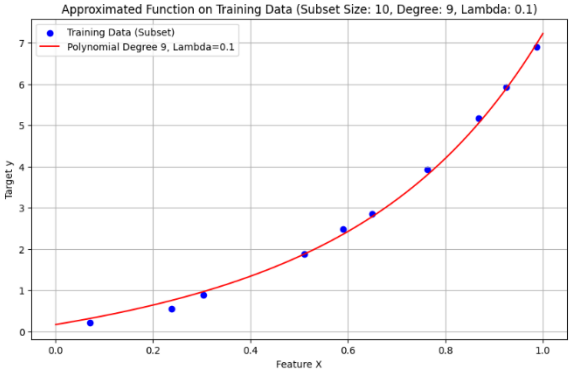


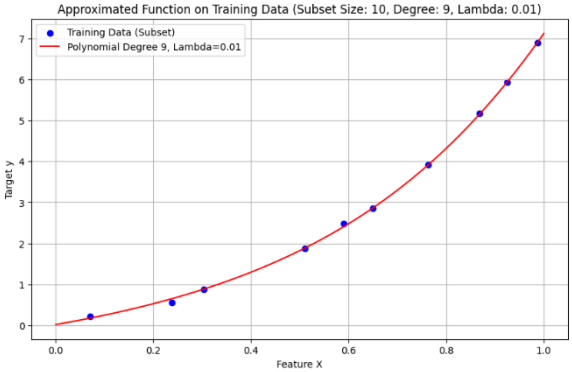
(5) APPROXIMATING FUNCTION WITH REGULARIZATION:

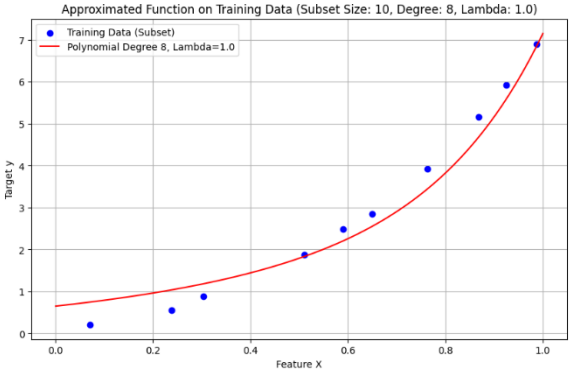
(a) Plot for data size = 10, polynomial degree 8 and 9, Lambda = {0.01, 0.1, 0.5, 1}

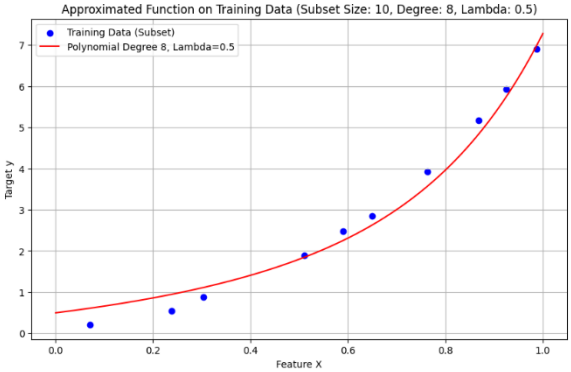


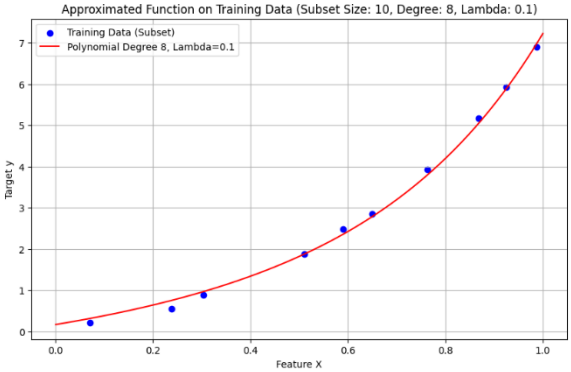


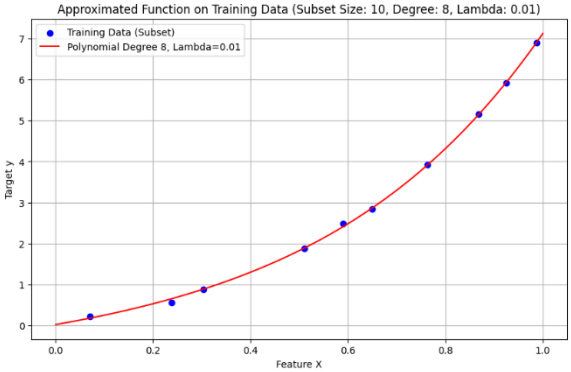




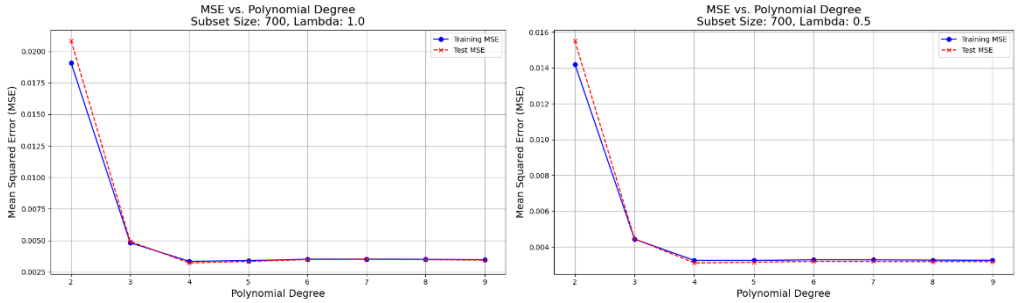


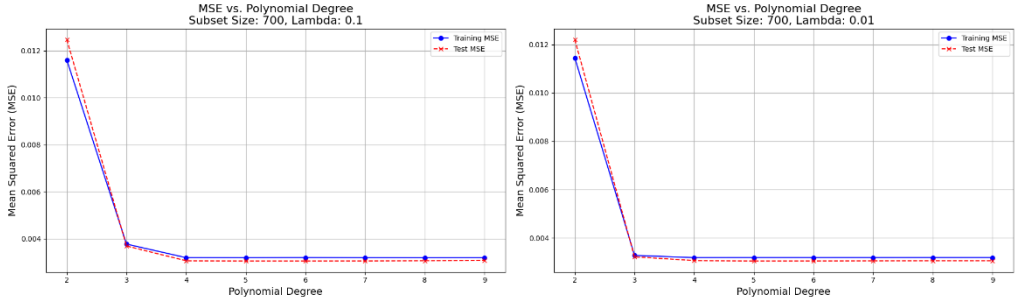


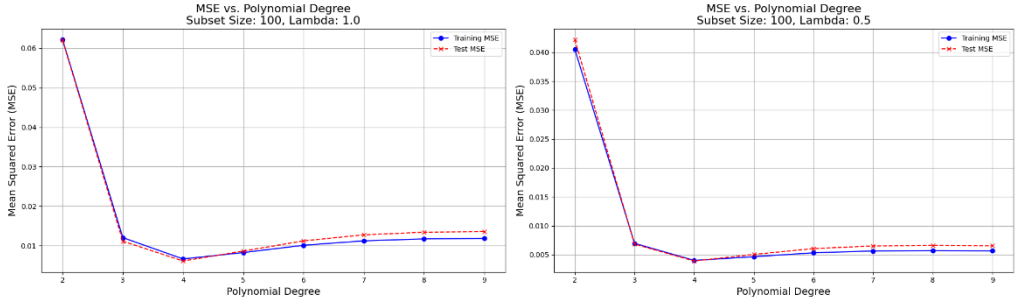


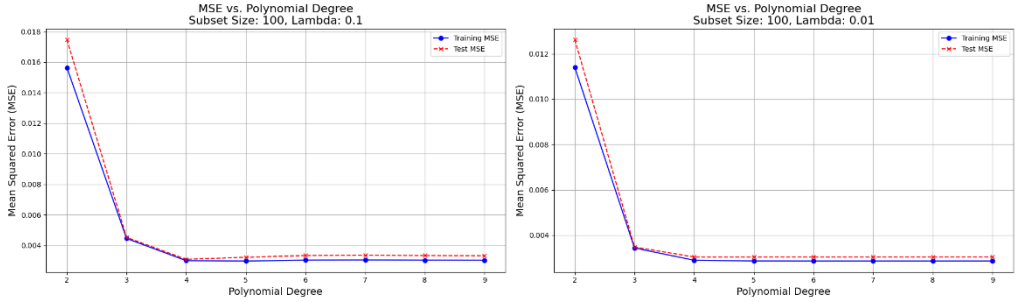


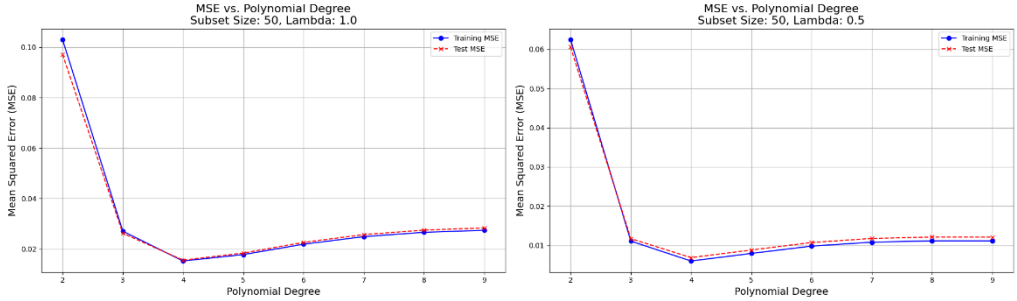
(b) MSE vs Polynomial Degree for subset size = {700, 100, 50, and 10}and lambda = {0.01, 0.1, 0.5, 1.0}

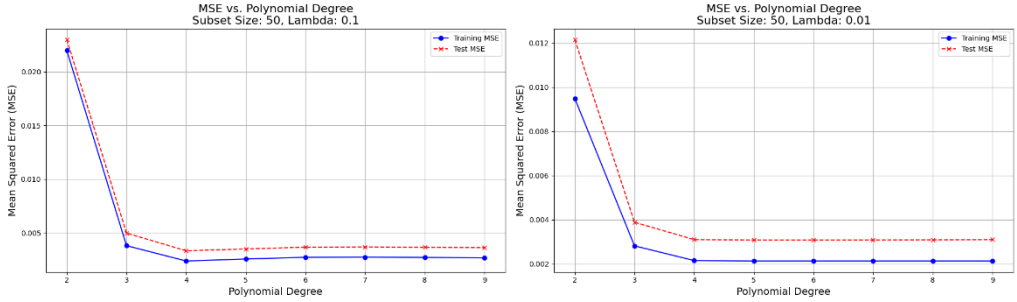


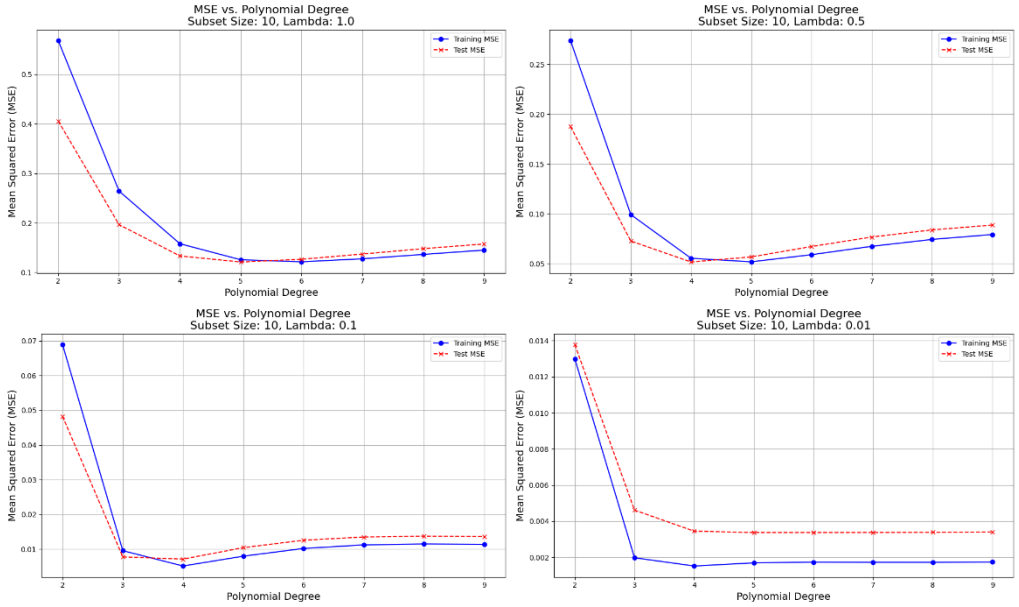




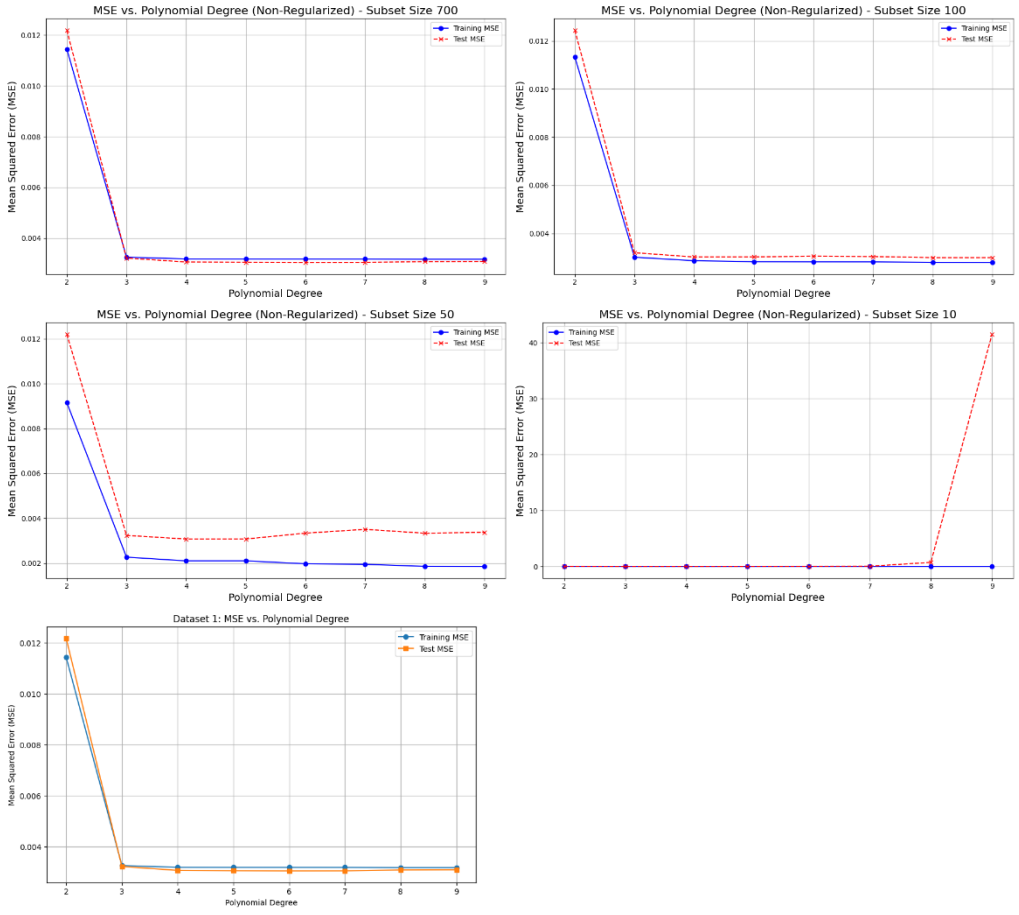




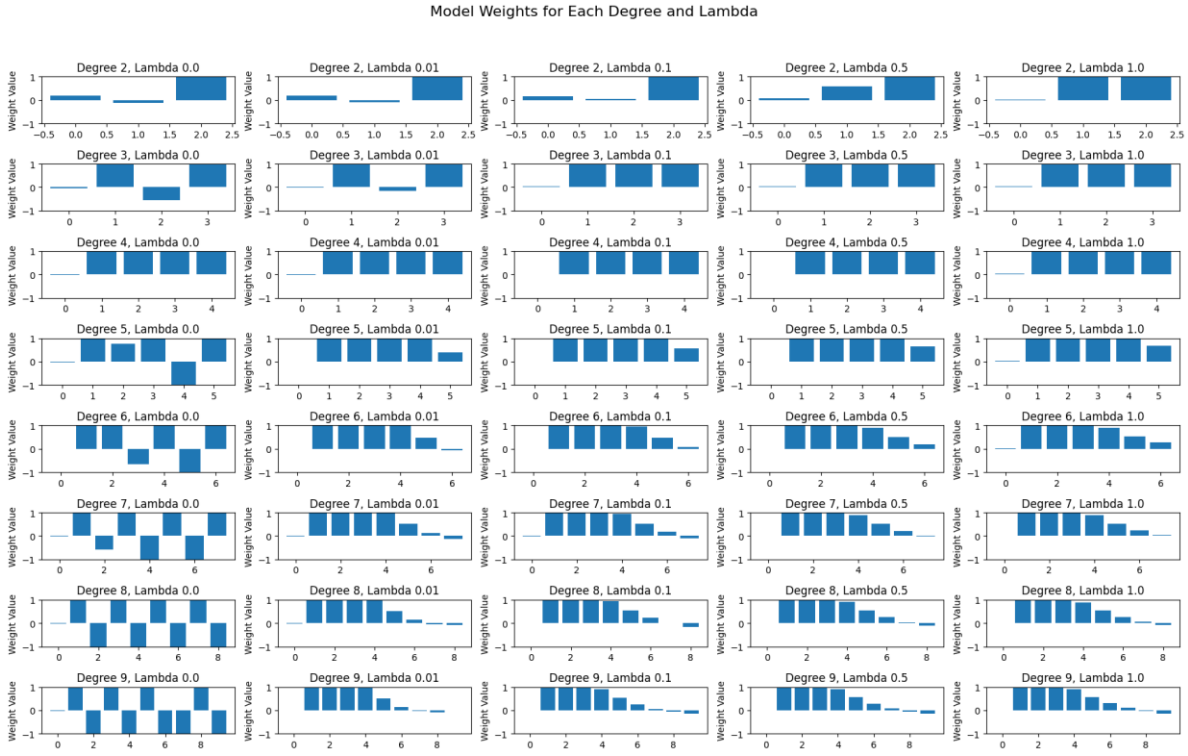




(c) MSE vs Polynomial Degree (non – regularized) for subset size = {700, 100, 50, and 10}

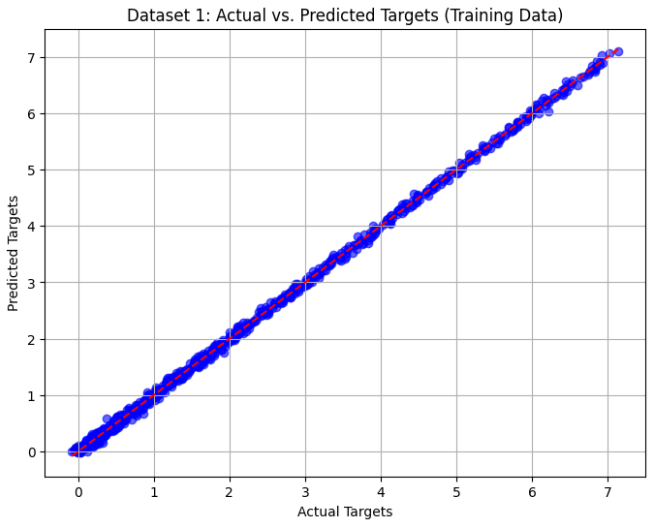


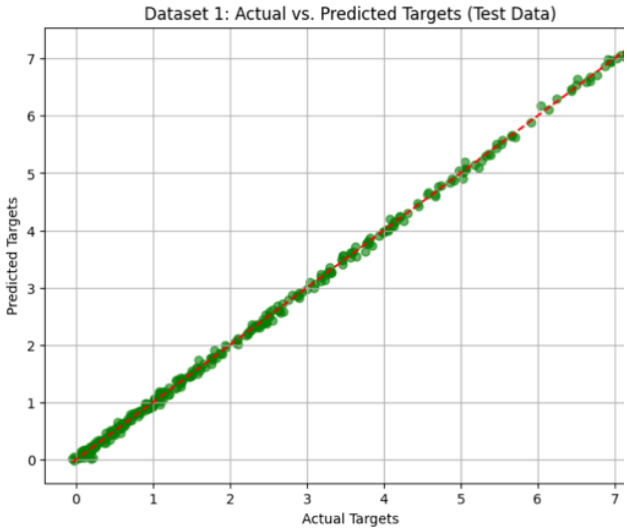
A. weight values for each case (before and after the regularization).



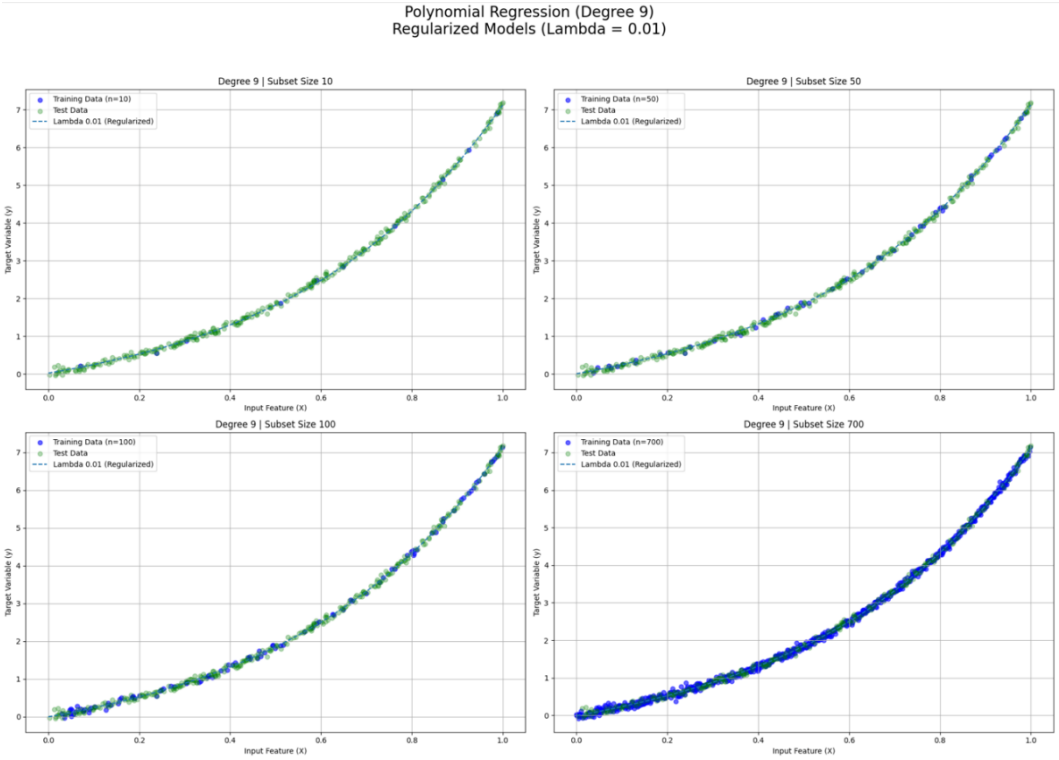
(6) RESULTS:

A. Plots of model output and target output for training data, and test data





B. plots of model output and target output for training data, and test data for the best regularization parameter on model complexity equal to 9 and lambda = 0.01



Observation:

The results indicate that as the polynomial degree increases from 2 to 9, the Mean Squared Error (MSE) for both training and testing datasets generally decreases, with models of higher degrees (3 to 9) demonstrating significantly lower MSEs compared to a quadratic model, suggesting better fit and complexity capture. Regularization (lambda) has a noticeable impact; while low lambda values (0.0 and 0.01) maintain low MSEs, higher values (0.1 to 1.0) slightly increase MSEs, particularly for lower-degree models, indicating over-regularization. The best performance is observed with a degree of 6 and a lambda of 0.0, achieving the lowest test MSE of 0.0030, highlighting the importance of balancing model complexity and regularization for optimal performance.

Conclusion

The polynomial curve fitting model demonstrated that higher polynomial degrees generally yield better fits with lower MSEs, particularly when combined with appropriate regularization. However, higher regularization values can lead to underfitting, indicating a need for balance between model complexity and regularization.

CHAPTER 2
Gaussian Basis Function Regression using K-means Clustering for Dataset 2

Introduction

Gaussian Basis Function Regression is a powerful technique that utilizes radial basis functions to model complex, non-linear relationships. When combined with K-means clustering, this method allows for effective partitioning of data and enhanced model accuracy. This chapter explores the application of Gaussian Basis Function Regression to a bivariate dataset, focusing on model complexity, regularization, and the benefits of clustering.

(1) ABOUT THE DATASET

Dataset 2 consists of three numerical columns: $X1$, $X2$ (independent variables), and Y (dependent variable). This dataset falls under the Bivariate input data category with 10,201 (X1, X2, Y) pairs. Sample data is:

X1	X2	**Y**
0.05	0.78	**8.7289**
0.59	0.68	**5.9745**
0.32	0.87	**8.2404**
0.14	0.69	**7.4628**
0.54	0.42	**3.1576**

Sample Dataset 2

(2) STEPS AND PROCEDURE

Step 1: Prepare the Data

(a) Split the data into train set (70%) and test set (30%)

(b) K-means clustering – Apply the k-means on training data to get the centers of clusters and these centroids will serve as center of Gaussian basis function. $\{c_1, c_2, \ldots, c_k\}$

(c) Number of clusters will be {2,4,8,16,32,64,128,256}

Step 2: Create Gaussian Basis Functions

(a) Identify the Cluster Centres : After performing k-means we get cluster centre as

$$C = \{c_1, c_2, \ldots, c_k\}$$

(b) Define the Gaussian Basis function: for each cluster c_i, Gaussian basis function will be-

$$\phi_j(x) = \exp\left(-\frac{(x - c_j)^2}{2\sigma^2}\right)$$

(c) Feature Matrix construction will be :

$$X_{\text{gauss}} = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_k(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_k(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_k(x_n) \end{bmatrix}$$

Step 3: Train the Model:

(a) Fit the model - Train the model using Ridge Regression for different complexities (number of basic functions)

k={2,4,8,16,32,128,256}

λ = {0 , 0.00001 , 0.001, 0.01 , 0.1, 1}

$J(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{k} w_j^2$

Here y and ŷ are the actual values and predicted values respectively, w is weights and λ regularization parameter.

Step 4: Choose the Best Model

(a) Cross-validation – In this context involves evaluating different model configurations to find the best performing one based on training MSE

(b) Calculate Mean Squared error : MSE $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$

Step 5: Visualize the Results

(a) Create plots showing the Gaussian basis function approximations for various training sizes and number of basic functions.

(b) Generate plots comparing model outputs to actual target outputs for both training and test datasets.

(c) Compute and plot weight values before and after applying regularization for each model complexity.

(d) Plot MSE for different model complexities and regularization parameters.

Step 6: Finalize the Model

(a) Based on cross-validation results, choose the model with the best performance (lowest MSE) and appropriate complexity.

(b) Ensure the selected model generalizes well by evaluating it on the test dataset.

(3) MODEL COMPLEXITY AND REGULARZATION

Model complexity – {2, 4, 8, 16, 32, 64, 128, 256}

Regression Coefficients- {0, 0.000001, 0.001, 0.01, 0.1, 1}

(4) CLUSTERING WITH K-MEANS

(a) Initialize K-means**:**

Select the number of clusters (`n clusters`) as the number of Gaussian basis functions. The number of clusters to consider are 2, 4, 8, 16, 32, 128, and 256.

(b) Run the K-means:

- Initialize centroids randomly from the data points.
- Assign each data point to the nearest centroid.
- Update the centroids by calculating the mean of the points assigned to each centroid.
- Repeat the assignment and update steps until convergence (i.e., the centroids no longer change significantly).

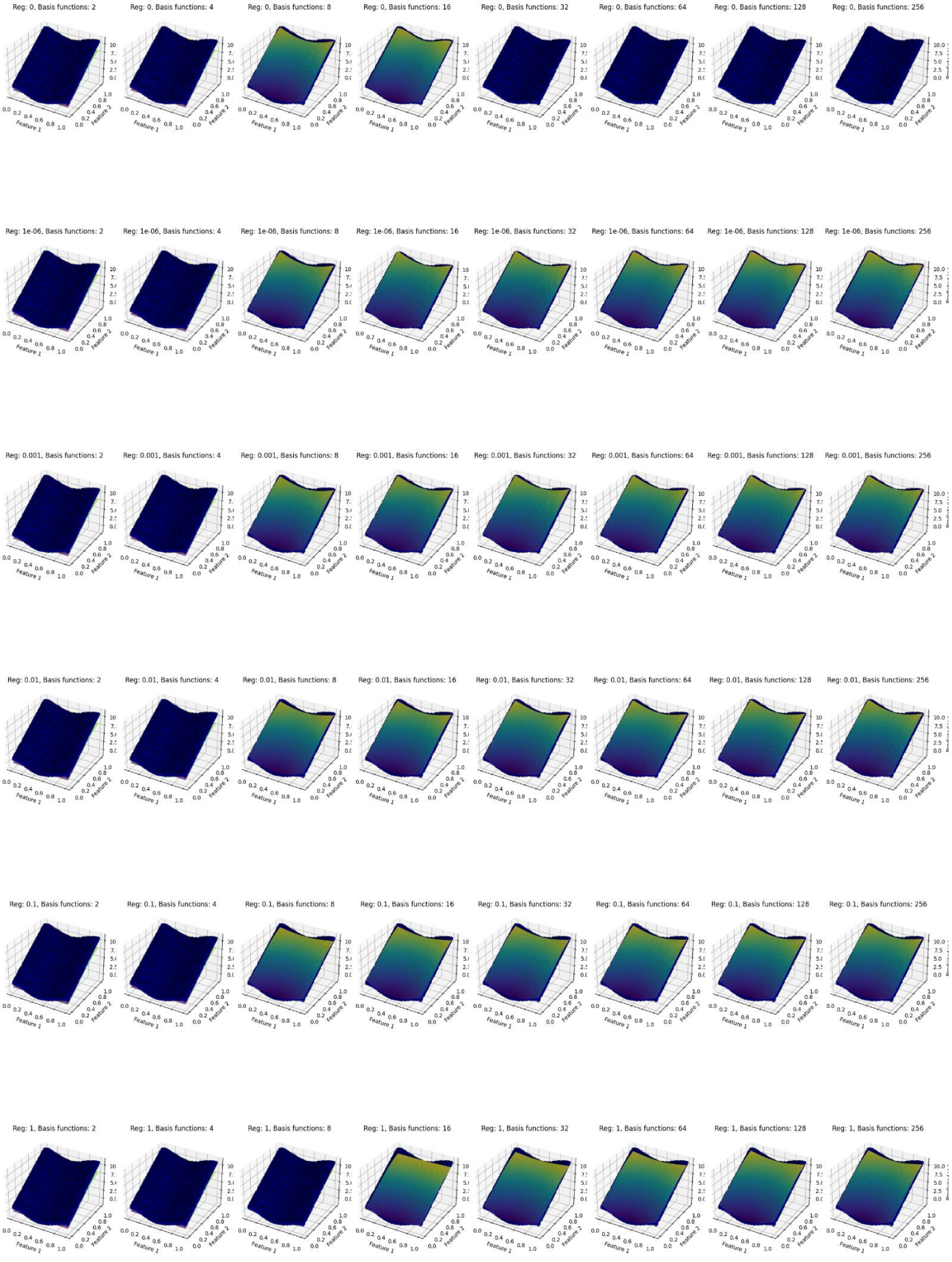
Convergence Criteria -

$$J = \sum_{i=1}^{n}\sum_{j=1}^{k} |x_i^{(j)} - \mu_j|^2$$

where $x_i^{(j)}$ is the data point assigned to cluster j and μ_j is the centroid of cluster j.

(5) RESULTS, OBSERVATION AND INFRENCES

(a) Plot for all regression value and basis function:


Reg: 0, Basis functions: 2
Reg: 0, Basis functions: 4
Reg: 0, Basis functions: 8
Reg: 0, Basis functions: 16
Reg: 0, Basis functions: 32
Reg: 0, Basis functions: 64
Reg: 0, Basis functions: 128
Reg: 0, Basis functions: 256
Reg: 1e-06, Basis functions: 2
Reg: 1e-06, Basis functions: 4
Reg: 1e-06, Basis functions: 8
Reg: 1e-06, Basis functions: 16
Reg: 1e-06, Basis functions: 32
Reg: 1e-06, Basis functions: 64
Reg: 1e-06, Basis functions: 128
Reg: 1e-06, Basis functions: 256
Reg: 0.001, Basis functions: 2
Reg: 0.001, Basis functions: 4
Reg: 0.001, Basis functions: 8
Reg: 0.001, Basis functions: 16
Reg: 0.001, Basis functions: 32
Reg: 0.001, Basis functions: 64
Reg: 0.001, Basis functions: 128
Reg: 0.001, Basis functions: 256
Reg: 0.01, Basis functions: 2
Reg: 0.01, Basis functions: 4
Reg: 0.01, Basis functions: 8
Reg: 0.01, Basis functions: 16
Reg: 0.01, Basis functions: 32
Reg: 0.01, Basis functions: 64
Reg: 0.01, Basis functions: 128
Reg: 0.01, Basis functions: 256
Reg: 0.1, Basis functions: 2
Reg: 0.1, Basis functions: 4
Reg: 0.1, Basis functions: 8
Reg: 0.1, Basis functions: 16
Reg: 0.1, Basis functions: 32
Reg: 0.1, Basis functions: 64
Reg: 0.1, Basis functions: 128
Reg: 0.1, Basis functions: 256
Reg: 1, Basis functions: 2
Reg: 1, Basis functions: 4
Reg: 1, Basis functions: 8
Reg: 1, Basis functions: 16
Reg: 1, Basis functions: 32
Reg: 1, Basis functions: 64
Reg: 1, Basis functions: 128
Reg: 1, Basis functions: 256
Feature 1
Feature 2

Observation:

Impact of Complexity on Fit:
Increasing the complexity (number of basis functions) generally improves the model's fit to the data, as seen by the reduction in both training and test MSE values. At higher complexities (e.g., 128 and 256 basis functions), the model captures more details in the data, achieving the lowest test MSE without any regularization.

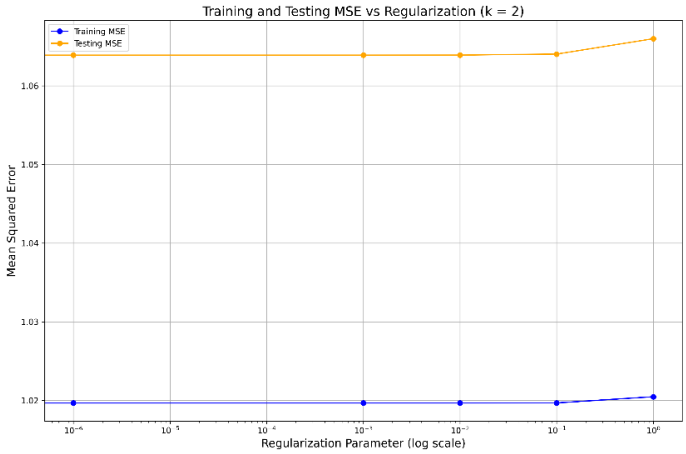
Effect of Regularization:
Regularization tends to smooth the model's predictions. While minimal regularization (1e-06) has a negligible impact, higher values (e.g., 0.01, 0.1, and 1) cause the model to underfit, as evidenced by higher MSE values and smoother, less detailed surfaces. This is particularly noticeable for complexities above 64, where regularization starts to suppress the model's flexibility.

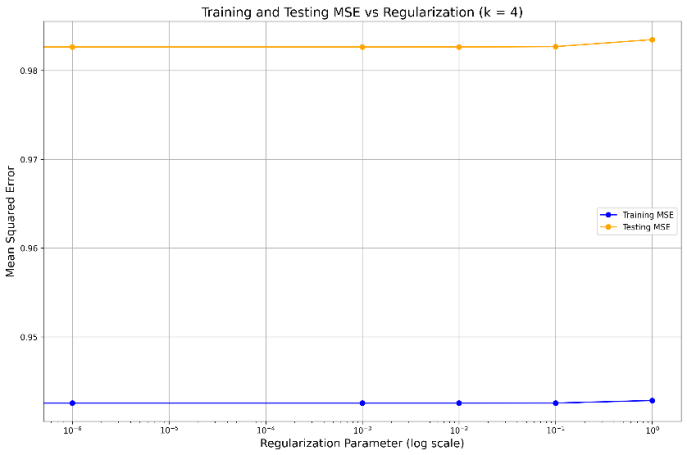
Overfitting and Underfitting:
Without regularization, the model benefits from high complexity without showing signs of overfitting, as the test MSE remains close to the training MSE. However, with increasing regularization, the model begins to underfit, especially at high regularization levels (0.1 and 1), where both training and test MSEs are substantially higher, and the model fails to capture data details.

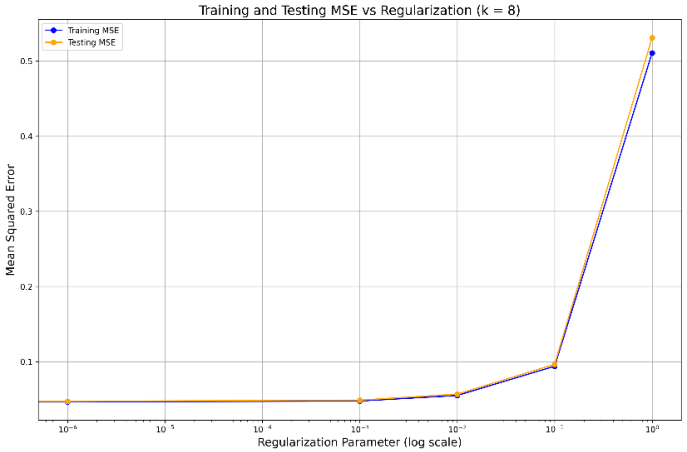
Optimal Setting:
The best performance is observed with no regularization (Reg. = 0) and the highest complexity (256 basis functions), achieving the lowest test MSE of 0.003247. This setting allows the model to generalize well on the test data without the need for regularization, highlighting that the dataset structure benefits from a high-complexity model.

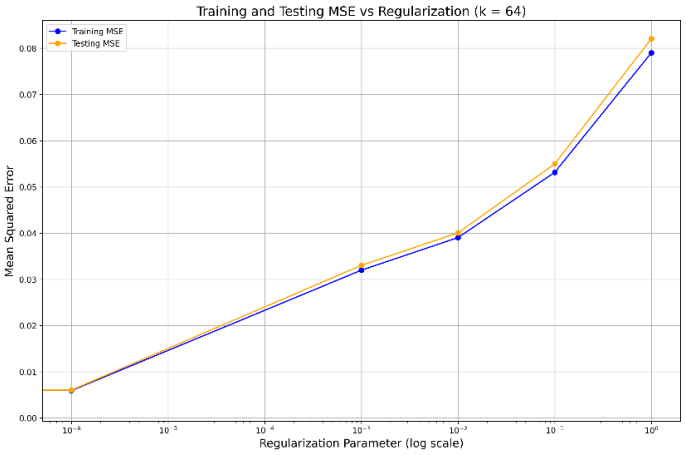
Regularization Trade-offs:
While regularization typically helps in managing overfitting, in this case, it restricts the model unnecessarily. The results suggest that for this specific dataset, the model does not suffer from overfitting even at high complexity, making regularization counterproductive beyond minimal levels.

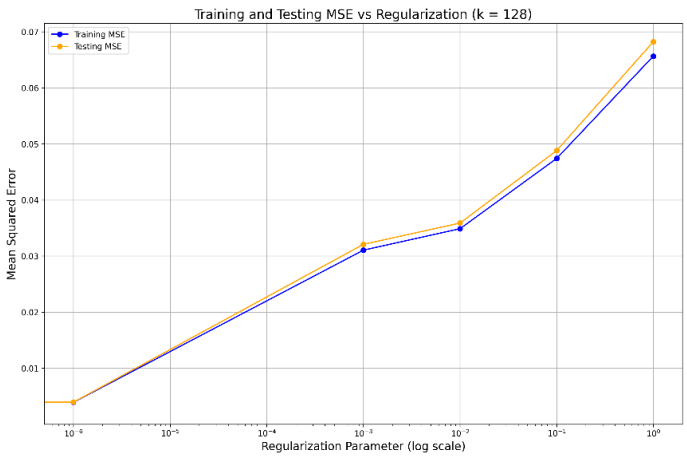
(b) Plot for training and testing MSE vs Regularization for all values of K

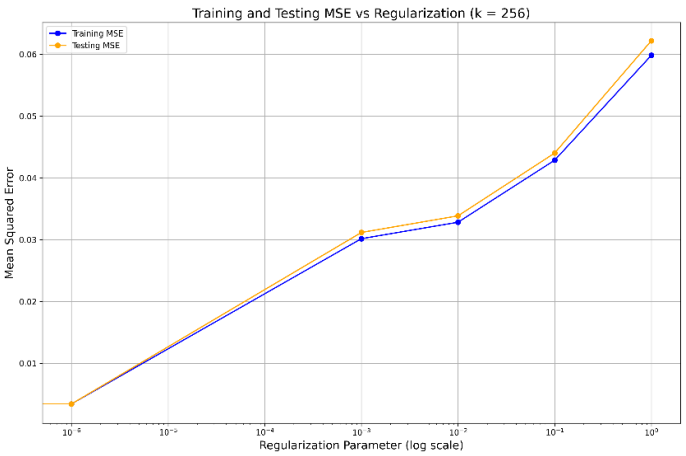












Observation:

The plots demonstrate that increasing regularization generally leads to higher Mean Squared Error (MSE) for both training and testing data across all model complexities, indicating that excessive regularization causes underfitting by restricting the model's flexibility. For simpler models (e.g., k=2 and k=4), the training and testing MSE values remain close and increase steadily with regularization, showing that these models are already generalizing well without much regularization. As complexity increases (e.g., k=8,16,32), a minimal level of regularization (around 10^{-6}) achieves the lowest MSE, suggesting that a slight constraint stabilizes these models effectively. In high-complexity models (e.g., k=128 and k=256), even small amounts of regularization significantly impact performance, causing MSE to rise rapidly, which underscores the need for balance between model flexibility and regularization. Ultimately, minimal regularization paired with higher model complexity captures the data patterns accurately while maintaining generalization, whereas higher regularization consistently leads to underfitting, emphasizing the importance of tuning regularization based on model complexity

(c) Plot for Training and testing MSE vs Number of Clusters

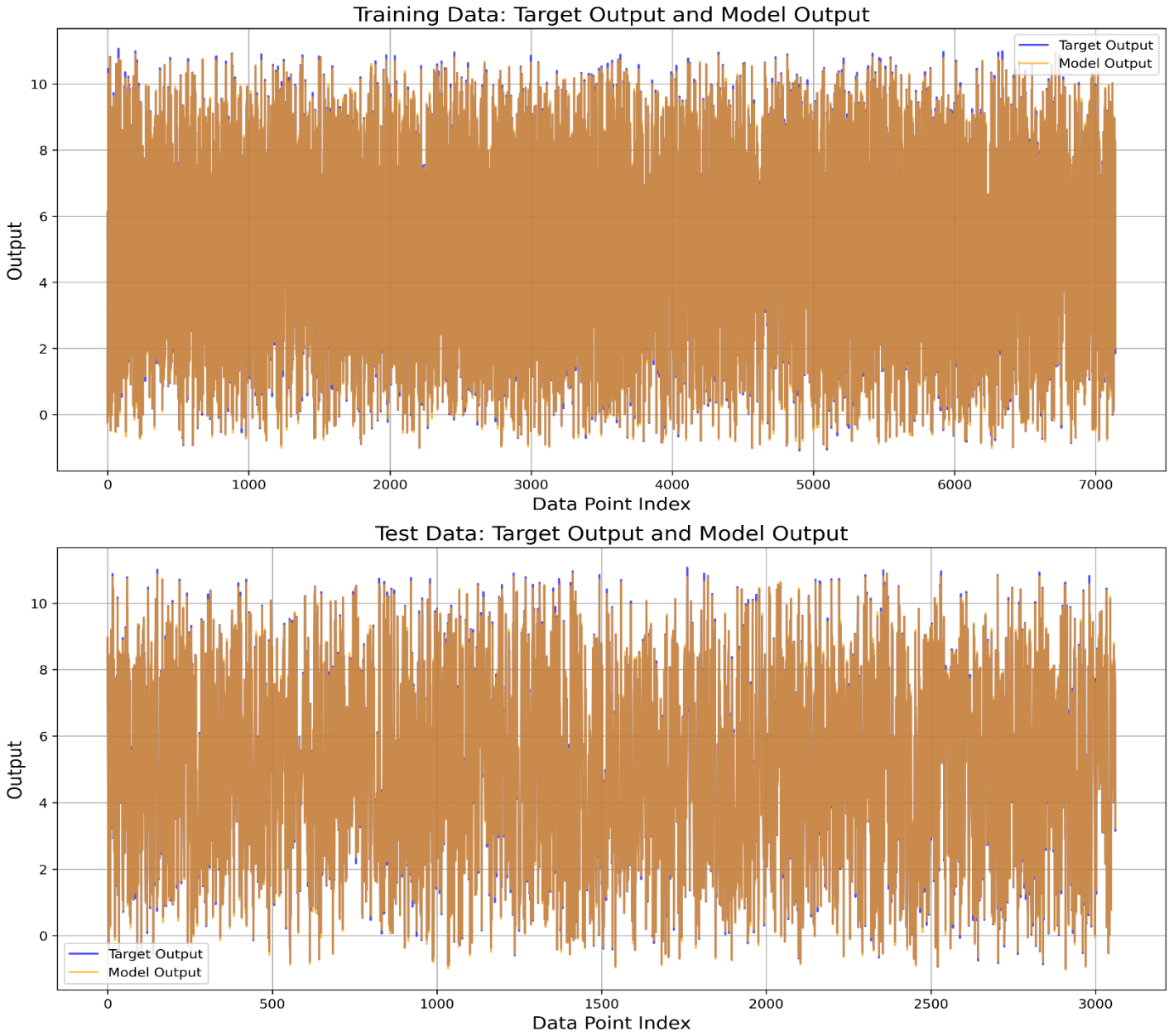
Observation:

The plot shows the relationship between the number of clusters (k) and the Mean Squared Error (MSE) for both training and testing data, with no regularization applied. As the number of clusters increases, both training and testing MSE decrease significantly. Initially, with a small number of clusters (e.g., k=4), the model exhibits high MSE, indicating poor fit and underfitting. However, as k reaches 8, there is a sharp drop in MSE, and the model achieves near-zero error for both training and testing. Beyond k=8, MSE remains consistently close to zero, suggesting that additional clusters do not further improve model performance. This indicates that a sufficient number of clusters is crucial for accurately capturing the data's structure, but excessive clustering provides no additional benefit when the model is already well-fitted.

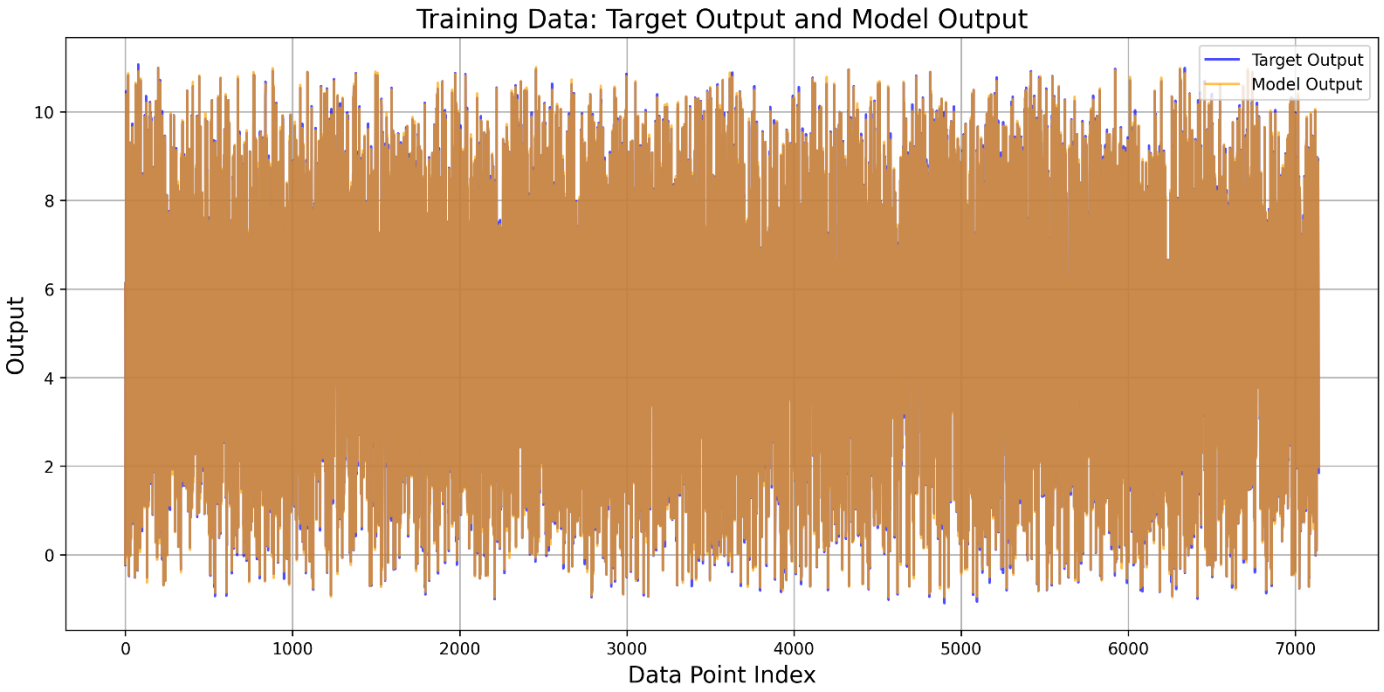
(d) Scatter Plot for target output vs model output (or training and testing dataset)

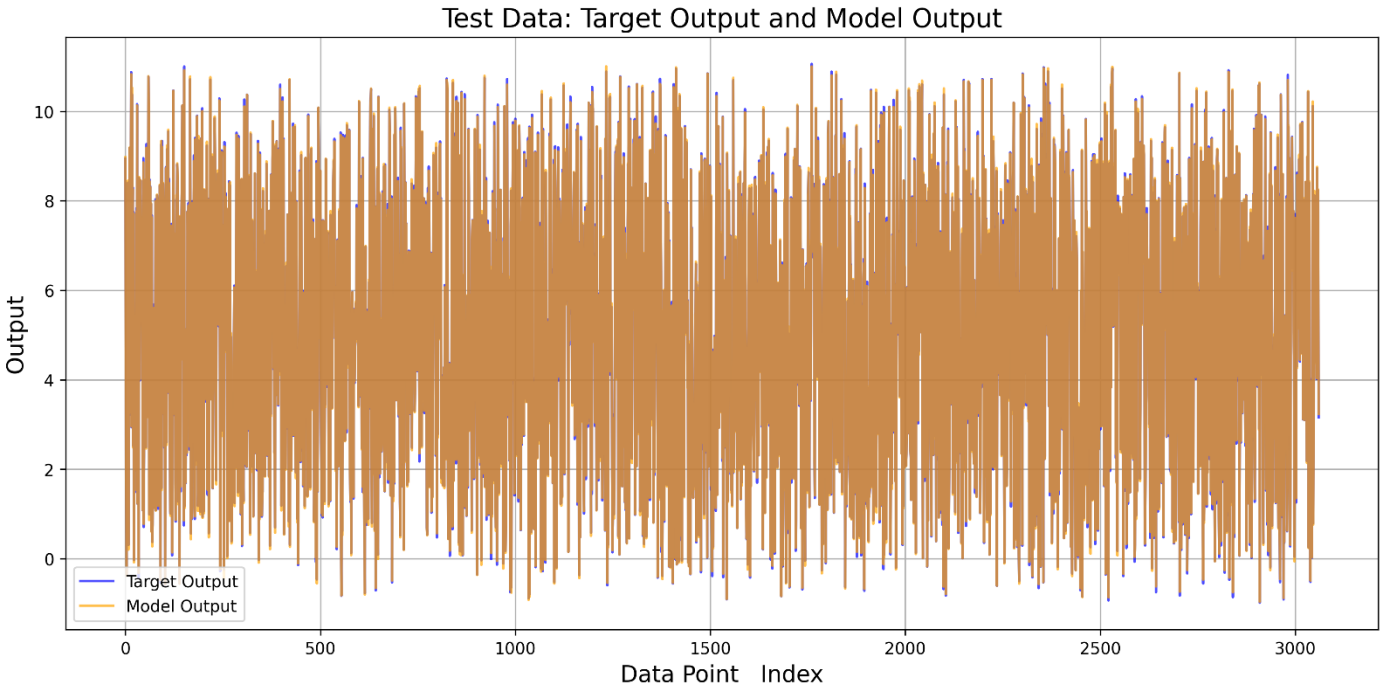
(i) plot for regularization = 0 and basis function = 16

(ii) plot for regularization = 0 and basis function = 16



(iii) plot for regularization = 0 and basis function = 256





Observation:

The plots show that with no regularization, both configurations—16 and 256 basis functions—result in a close alignment between the model output and target output, indicating a strong fit to the data. With 16 basis functions, the model captures the overall data pattern effectively, displaying minimal deviations from the target output across both training and test sets. Increasing the complexity to 256 basis functions further enhances the model's ability to match the target values, as seen in the nearly perfect overlay of model output on target output in the time series plot. This high complexity allows the model to capture finer details, achieving minimal error across the dataset without signs of overfitting. In both cases, the model generalizes well, though 256 basis functions provide a more precise fit to the data's intricate patterns