ROB313 Assignment 2

1. Assignment Objectives

The objectives of Assignment 2 are as follows:

1) Derive an expression for the weights of a generalized linear model (GLM) with least-squares loss and Tikhonov regularization.
2) Given a GLM, derive how **α** can be estimated by minimizing a given objective function, comparing to a dual representation example from class.
3) Given a grid of shape and regularization parameters, implement a radial basis function (RBF) model that minimizes least-squares loss using a Gaussian kernel.
4) With a designed dictionary of basis functions, implement a greedy regression algorithm using orthogonal matching pursuit metric.

2. Question 1 – Tikhonov Regularization

2.1 Expression for the weights of the given generalized linear model (Q1)

Q1. $\hat{f}(\underline{x}, \underline{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\underline{x})$ ($y - \hat{y}$)

$$\hat{\underline{w}} = \underset{\underline{w} \in \mathbb{R}^M}{\operatorname{argmin}} \left[\sum_{i=1}^{N} \left(y^{(i)} - w_0 - \sum_{j=1}^{M-1} w_j \phi_j(\underline{x}^{(i)}) \right)^2 + \sum_{j=1}^{M} \sum_{j=1}^{M} \Gamma_{ij} w_{i-1} w_{j-1} \right]$$ ($\lambda \underline{w}$)

- Loss function: $L(\underline{w}) = \sum_{i=1}^{N} \left(y^{(i)} - w_0 - \sum_{j=1}^{M-1} w_j \phi_j(\underline{x}^{(i)}) \right)^2 + \sum_{j=1}^{M} \sum_{j=1}^{M} \Gamma_{ij} w_{i-1} w_{j-1}$

$$= \|\underline{y} - \underline{\Phi w}\|_2^2 + \underline{\Gamma} \|\underline{w}\|_2^2$$

$$= (\underline{y} - \underline{\Phi w})^T (\underline{y} - \underline{\Phi w}) + \underline{w}^T \underline{\Gamma} \underline{w}$$

$$= (\underline{y}^T - \underline{w}^T \underline{\Phi}^T)(\underline{y} - \underline{\Phi w}) + \underline{w}^T \underline{\Gamma} \underline{w}$$

- Set $\frac{\partial L}{\partial \underline{w}} = 0 = 2(\underline{y} - \underline{\Phi w}) \cdot (-\underline{\Phi}^T) + 2\underline{\Gamma w}$

$$0 = -2\underline{\Phi}^T(\underline{y} - \underline{\Phi w}) + 2\underline{\Gamma w}$$

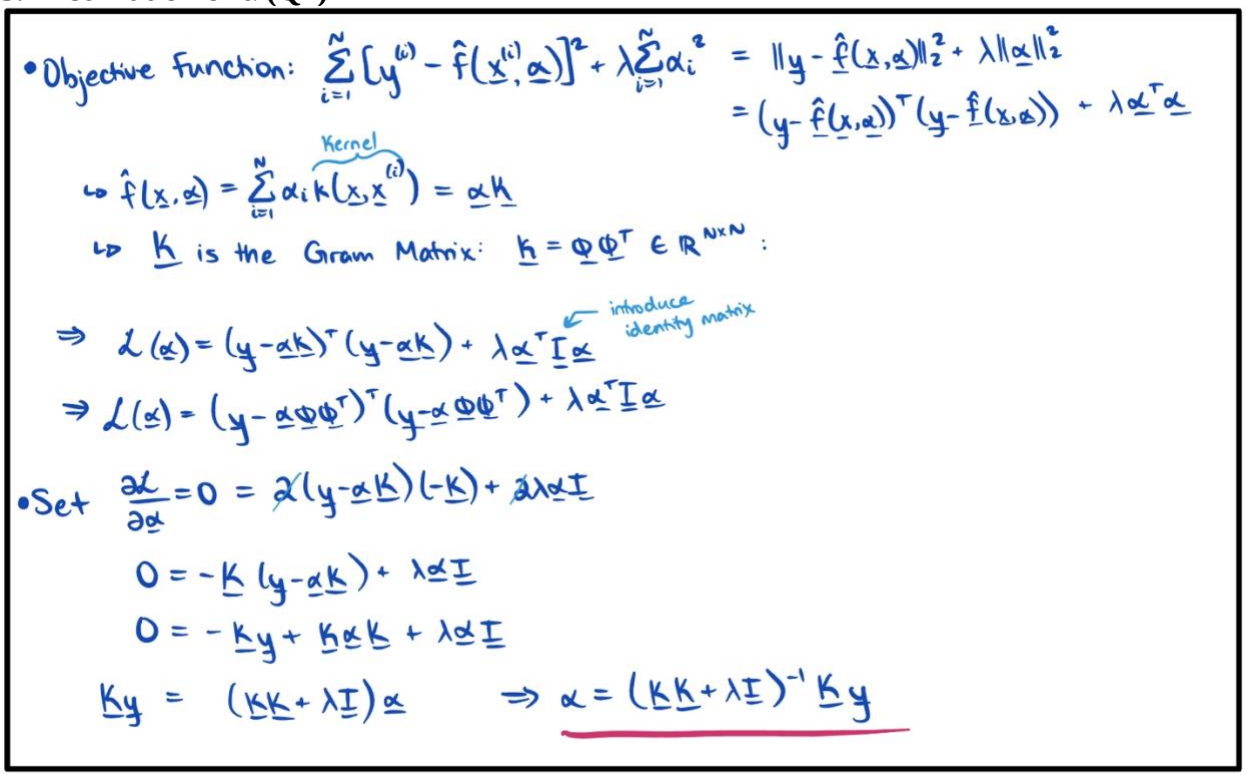
$$\underline{\Phi}^T(\underline{y} - \underline{\Phi w}) = \underline{\Gamma w}$$

$$\underline{\Phi}^T \underline{y} - \underline{\Phi}^T \underline{\Phi w} = \underline{\Gamma w}$$

$$\underline{\Phi}^T \underline{y} = (\underline{\Gamma} + \underline{\Phi}^T \underline{\Phi}) \underline{w} \Rightarrow \underline{w} = (\underline{\Gamma} + \underline{\Phi}^T \underline{\Phi})^{-1} \underline{\Phi}^T \underline{y}$$

3. Question 2 – Generalized Linear Model

3.1 Estimation of α (Q2)



3.2 Comparison to dual representation in class (Q2)

The dual representation found in class obtains the result $\boldsymbol{\alpha} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$. This is different from the result for α obtained above because here using a kernel function, each kernel evaluation is treated as a feature with a weight α and λ as a regularization parameter, and it deals directly with the given data. With the dual representation in class, we find the weights of the actual features, and the model can be expressed as terms of kernels centered at the training inputs.

4. Question 3 – Radial Basis Function (RBF) Model

4.1 Code Structure Overview & Implementation Strategies (Q3)

Outside of any functions, I entered the given question values for the different parameters to test. The grid of shape parameters consisted of θ = {0.05, 0.1, 0.5, 1, 2}, and the grid of regularization parameters consisted of λ = {0.001, 0.01, 0.1, 1}.

4.1.1 Gaussian kernel function

The first function implemented was drawn from the class-provided notes for a multi-dimensional gaussian kernel. Given points in two arrays with a lengthscale parameter defaulted to θ = 1, the function returns the Gram matrix. The kernel is evaluated using the squared Euclidian distances between the points in the arrays.

4.1.2 Radial basis function model

An RBF function was implemented to minimize the root mean square error (RMSE) loss. The optimal parameters θ and λ were found by iterating through every combination of θ and λ from the provided values and choosing that combination where the RMSE was minimum.

In this function, I added in parameters into the function to improve the usability of this model, which minimized the number of lines needed to comment/uncomment to run different parts. The first parameter is the dataset that is desired to run, which is either mauna_loa or rosenbrock, and either of the dataset is automatically loaded depending on the input. Furthermore, I added a parameter, which was a boolean operator; the default value, validation=True, caused the RBF model to operate on the validation set, which found the minimum RMSE loss, which determined the optimal θ and λ parameter values. If the validation parameter is set to False, the RBF model operates on the testing data, allowing one to obtain the RMSE loss for every parameter combination as well. The testing loss was found by finding the loss corresponding to the optimal parameter values found earlier with the validation set. The other two parameters to this function are the list of possible shape and regularization parameters, which were already defined earlier.

For the radial basis function model, a kernel matrix was constructed using the gaussian kernel helper function explained earlier in section 4.1.1. Regularization was also implemented to prevent overfitting of the data using the built in numpy "np.eye" function. Then, the Cholesky factorization was completed using the also built in cho_factor function, which enabled calculating an estimate for the alpha values. Finally, the error was calculated for each prediction, and the optimal θ and λ corresponded to those that resulted in the smallest error.

4.2 Results (Q3)

The following table summarizes the RMSE losses on the validation and testing sets of the mauna_loa and rosenbrock datasets. The validation RMSE was found by running the RBF model with every possible combination of shape parameters (θ's) and regularization parameters (λ's); the reported validation RMSE's were the lowest RMSE found. As well, the optimal θ and optimal λ were those parameters that resulted in the lowest RMSE. The full results for every θ and λ combination are provided in the Appendix for the validation sets as an example (for Q3).

Using the optimal parameters found (having the smallest RMSE loss), the RBF model was used on the test data. For each dataset, the resulting test RMSE is shown using their optimal θ and λ.

Table 1: RMSE with selected hyperparameters

	Optimal θ	**Optimal λ**	**Validation RMSE**	**Test RMSE**
Mauna_loa	θ = 1	λ = 0.001	0.124479	0.154843
Rosenbrock	θ = 2	λ = 0.001	0.193240	0.084572

From also printing out the test RMSE values for the other parameter combinations explored during validation, it can be seen that the minimum test RMSE still corresponded to the same optimal parameters chosen from the validation set. Thus, the parameters chosen based on validation RMSE were accurate.

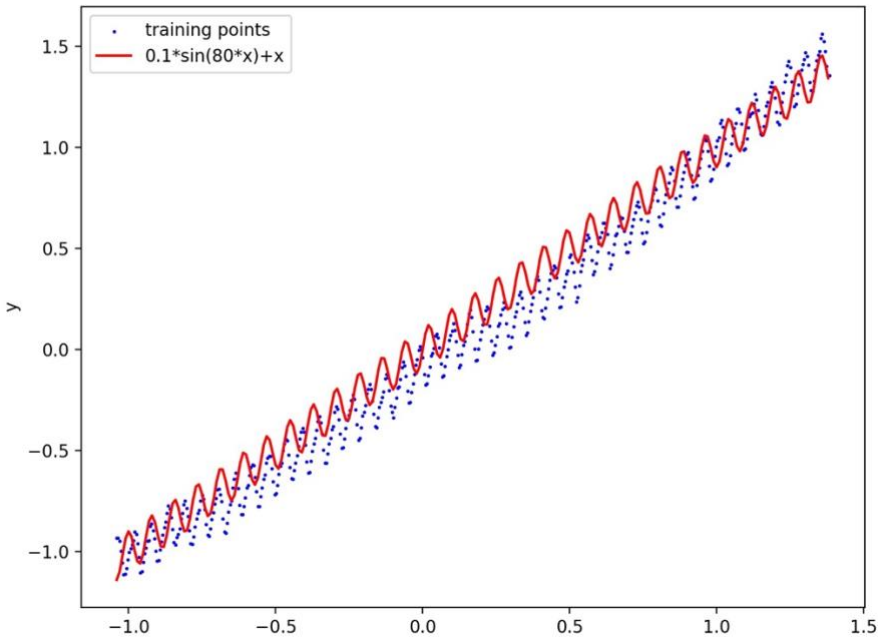
5. Question 4 – Greedy Regression Algorithm

5.1 Code Structure Overview & Implementation Strategies (Q4)

5.1.1 Basis helper function structure

A greedy regression algorithm was implemented using over 400 basis functions. First, in order to design a dictionary of candidate basis functions, I observed the mauna_loa dataset structure by plotting it. The data seemed to be a combination of sinusoidal and polynomial elements, so I tested out a few different implementations on a graphing calculator and then on top of the mauna_loa data. The different parameters were adjusted and played with to see which ranges may fit the dataset the best; this helped in determining a ball park range of parameters to loop over. For example, the graph below shows a random function found from trying out different parameters on a 0.1*sin(80*x) + x function. The code for this plot can be found in the main function, and there are comments throughout the code to describe the function of each line.

Graph 1: Playing with different parameters to understand how to design the basis functions.



As a result, I created three helper functions that would construct the basis functions: sinusoidal, exponential, and polynomial. The parameters were different inputs, such as phase shifts and amplitudes to adjust the function outputs.

5.1.2 Basis dictionary maker

Another helper function created was called basis_maker(), which returns a list of possible basis functions. For the sinusoidal function, there were two for loops to iterate over around 20 angular frequency values and phase shifts; in total, this created at least 400 basis functions. In addition, there were for loops to iterate over the different parameters (degree) for the exponential function and polynomial function. All of these basis functions were appended to the returned list.

5.1.3 Greedy regression algorithm

Finally, the greedy regression algorithm was implemented in a greedy() function which took in defaulted inputs, which were a list of basis functions and the dataset to analyze (mauna_loa). Similar to the previous question, the data was automatically loaded within the function to improve usability of the code. The test and validation sets of data were combined since it was required to do so to predict on the test set, but they were still named as x_train and y_train.

Following the pseudocode provided from class, I implemented a while loop with two stop conditions; the first was to ensure that the norm of the residual vector (comparing to an epsilon value of 0.1) and the provided minimum description length (MDL) did not grow by continuously updating it and comparing it to its previous value.

The basis_maker function was then called to create the list of basis functions to explore. The format of each element was a tuple; the first part being the type of function (sinusoidal, exponential, or polynomial), and the second part being a dictionary with the described parameters for the respective function. Iterating through each of these functions, they were evaluated at the x training points to create a possible column for the phi (ϕ) matrix. J(ϕ_i) was calculated for the column found, and as long as it returned a real number (in case there was division by 0), I compared J with the best_reduction value found so far. If it was better, i.e., greater, then the best_reduction would be set to that J value; I would also keep track of the corresponding best_basis function and the best_col column for future reference.

After iterating through the list of all the basis functions but still in the while loop, the best basis so far was added to a list called "chosen" and removed from the overall list of basis functions. The corresponding best column was then added to the phi matrix using the built in np.hstack feature. Also, in the case of the first iteration, the phi matrix would be empty. In this case, instead of using np.hstack, the phi matrix was just set equal to the phi column to be added.

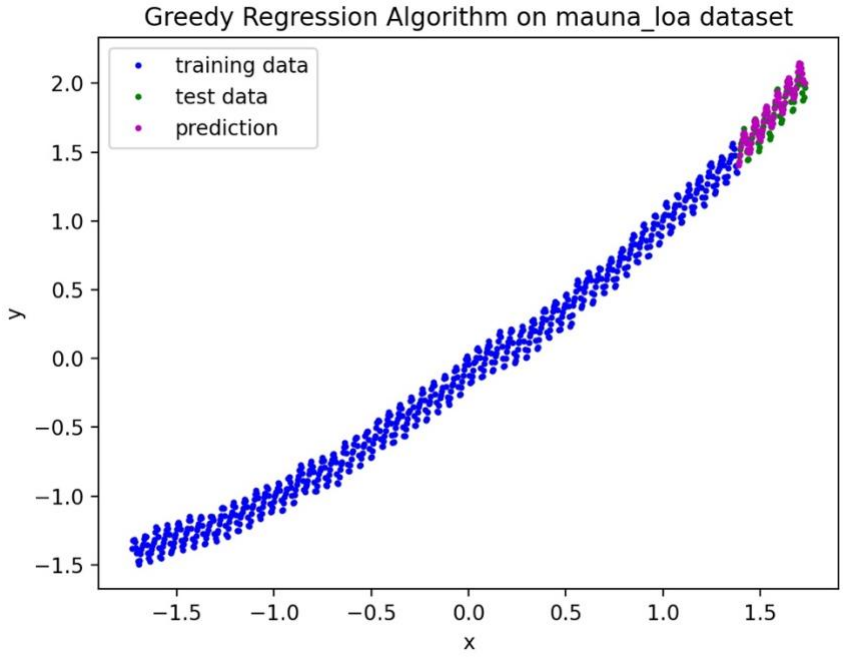
Still in the while loop, the weights were then solved for by calculating the pseudoinverse of the phi matrix by using the np.linalg.pinv() function, and then the weights were calculated as the dot product of that pseudoinverse matrix with the y training points. Finally, the residual vector and the MDL were updated. There is also some commented code after this section which was used to validate my choice of free parameters to iterate through for my basis functions; when uncommented, it plots how the basis prediction model gradually gets more accurate on the training data as the while loop continues to iterate.

Outside of the while loop, the rest of the code was to use this algorithm on the testing set. Every function from the chosen list of basis functions was iterated through; each function was then evaluated at the x test points, and this column was added to a final phi matrix. The y predictions were found with the dot product of this matrix with the weights found earlier, and then the RMSE was calculated. The test predictions were then plotted against the dataset, which are presented in the next section.

5.2 Results (Q4)

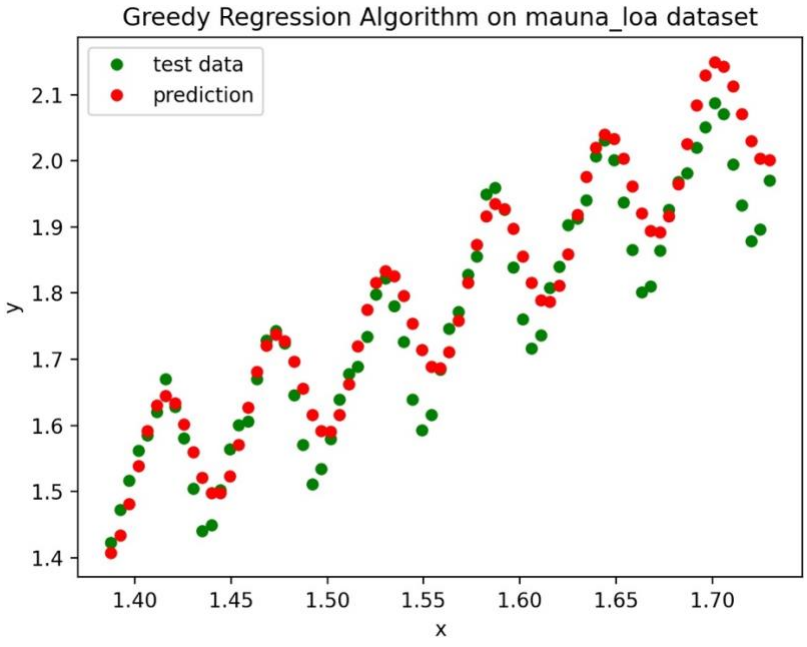
The following graph presents the training data in blue, the test data in green, and the prediction set on the testing data in magenta.

Graph 2: Greedy regression algorithm on mauna_loa with training, test, and prediction data.



From the graph, we can see how the prediction data in magenta is close to the actual testing data. Graph 3 shows a close-up version of how the predictions compared to the testing dataset.

Graph 3: Greedy regression algorithm on mauna_loa with test and prediction data.



The test RMSE from this greedy model was 0.05819999, showing that the performance of this model was moderately accurate. The accuracy of this model could be improved even more by having a larger dictionary of basis functions to iterate over and a wider range of parameters to test. As can be seen in the graph, the predictions in red are fairly close to the testing data in green due to the many parameter values tested and the combinations of sinusoidal, exponential, and polynomial functions. The actual chosen basis functions with their specified parameters are included in the Appendix under Q4 for reference.

From observing the graph, the model looks not very sparse because the predictions very gradually start to vary from the testing data, indicating that the algorithm may slightly overfit the data. The last wave of red points is farther from the green test points when compared to the first wave. From a more quantitative perspective, all seven coefficients in the weights were nonzero, but they were close to zero. Since there are no coefficients exactly zero, the model is not very sparse. To prevent overfitting the data and increase the sparsity of the model, more techniques can be implemented, such as principal component analysis and being more selective of the relevant features to use when training the model.

6. Appendix

Q3: All validation RMSE from every θ and λ parameter combination for mauna_loa

```
alyssa@MyLabMacBook ~/desktop/rob313-a2/data (main)
$ python3 rob313-a2.py
Q3: RBF model
Dataset being tested: mauna_loa

Using validation set to find best model

theta = 0.05, lambda = 0.001, RMSE = 1.2197089795507747
theta = 0.05, lambda = 0.01, RMSE = 1.1173089221572012
theta = 0.05, lambda = 0.1, RMSE = 1.0820180096816154
theta = 0.05, lambda = 1, RMSE = 1.0922111805750374
theta = 0.1, lambda = 0.001, RMSE = 1.4162863006803785
theta = 0.1, lambda = 0.01, RMSE = 1.0591366263503217
theta = 0.1, lambda = 0.1, RMSE = 0.9659075555689463
theta = 0.1, lambda = 1, RMSE = 0.9967263920688288
theta = 0.5, lambda = 0.001, RMSE = 0.347100897470012
theta = 0.5, lambda = 0.01, RMSE = 0.4277204982193483
theta = 0.5, lambda = 0.1, RMSE = 0.4737407089006155
theta = 0.5, lambda = 1, RMSE = 0.6063342412851929
theta = 1, lambda = 0.001, RMSE = 0.1244786703155218
theta = 1, lambda = 0.01, RMSE = 0.22949484239065696
theta = 1, lambda = 0.1, RMSE = 0.33911150518029426
theta = 1, lambda = 1, RMSE = 0.4436147618852462
theta = 2, lambda = 0.001, RMSE = 0.20170528415357084
theta = 2, lambda = 0.01, RMSE = 0.2524042653638991
theta = 2, lambda = 0.1, RMSE = 0.21714466927435222
theta = 2, lambda = 1, RMSE = 0.24921984409731715
min RMSE = 0.1244786703155218
```

Q3: All validation RMSE from every θ and λ parameter combination for rosenbrock

```
alyssa@MyLabMacBook ~/desktop/rob313-a2/data (main)
$ python3 rob313-a2.py
Q3: RBF model
Dataset being tested: rosenbrock

Using validation set to find best model

theta = 0.05, lambda = 0.001, RMSE = 0.7354631814717737
theta = 0.05, lambda = 0.01, RMSE = 0.7389126298691878
theta = 0.05, lambda = 0.1, RMSE = 0.7523067858566777
theta = 0.05, lambda = 1, RMSE = 0.8081025674515984
theta = 0.1, lambda = 0.001, RMSE = 0.6265852275649663
theta = 0.1, lambda = 0.01, RMSE = 0.6320278838901512
theta = 0.1, lambda = 0.1, RMSE = 0.6477347961602582
theta = 0.1, lambda = 1, RMSE = 0.7205205160899887
theta = 0.5, lambda = 0.001, RMSE = 0.35150688946956177
theta = 0.5, lambda = 0.01, RMSE = 0.38100713616119414
theta = 0.5, lambda = 0.1, RMSE = 0.4190932888434195
theta = 0.5, lambda = 1, RMSE = 0.5133129463151594
theta = 1, lambda = 0.001, RMSE = 0.25723582897453956
theta = 1, lambda = 0.01, RMSE = 0.29740722139373404
theta = 1, lambda = 0.1, RMSE = 0.3581845343504239
theta = 1, lambda = 1, RMSE = 0.4666354244201736
theta = 2, lambda = 0.001, RMSE = 0.1932395869738141
theta = 2, lambda = 0.01, RMSE = 0.24102735026434702
theta = 2, lambda = 0.1, RMSE = 0.31170686092422084
theta = 2, lambda = 1, RMSE = 0.4365470710069977
min RMSE = 0.1932395869738141
```

Q4: Chosen basis functions for the greedy regression algorithm on mauna_loa

```
$ python3 rob313-a2.py
Q4: Greedy regression algorithm

a chosen basis function is:  <function polynomial at 0x11ccfef80> {'degree': 1, 'c': 0}

a chosen basis function is:  <function sin at 0x11ccfee60> {'w': 110, 'phi': -0.1415926535897949}

a chosen basis function is:  <function polynomial at 0x11ccfef80> {'degree': 4, 'c': 0}

a chosen basis function is:  <function exponential at 0x11ccfeef0> {'a': 0.1}

a chosen basis function is:  <function polynomial at 0x11ccfef80> {'degree': 5, 'c': 0}

a chosen basis function is:  <function polynomial at 0x11ccfef80> {'degree': 2, 'c': 0}

a chosen basis function is:  <function sin at 0x11ccfee60> {'w': 120, 'phi': -1.9415926535897938}

testing RMSE:  0.058199994215026606

Number of non-zero coefficients: 7
Number of zero coefficients: 0
Coefficient 1: [0.99574813]
Coefficient 2: [-0.09942727]
Coefficient 3: [0.03078621]
Coefficient 4: [-0.12086177]
Coefficient 5: [0.00274589]
Coefficient 6: [0.06912736]
Coefficient 7: [0.00271485]
```