CS F320 - FOUNDATIONS OF DATA SCIENCE

Semester I, 2021-2022

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI

HYDERABAD CAMPUS

ASSIGNMENT-1

Vineet Venkatesh - 2019A7PS0043H

Pavan Kumar Reddy Yannam - 2019A7PS0038H

Jagath Kaparthi - 2019A4PS0547H

Table of Contents

1. Introduction

In this assignment, we perform Polynomial Regression, using

a. Polynomials of degrees 0 to 9 without regularization, and
b. Polynomials of degree 9 with regularization using
    i. Ridge regression, and
    ii. Lasso regression.

The general polynomial of order M is of the form,

$$y(x,w) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_Mx_M$$

The polynomial coefficients $w_0, \ldots, w_M$ are collectively denoted by the vector ***w***.The values of the coefficients will be determined by fitting the polynomial to the training data. This can be done by minimizing an error function that measures the misfit between the function y(x,w), for any given value of **w**, and the training set data points. We use the Sum of Squares error function which is given by,

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

Where, t_n is the target value of the n^{th} datapoint.

The polynomial y(x,w) can also be represented in a vectorized form as, $w^T\varphi(x_n)$. Using this, the regularized error equation is shown below,

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

Where q = 1 for Lasso Regression and q=2 for Ridge Regression.

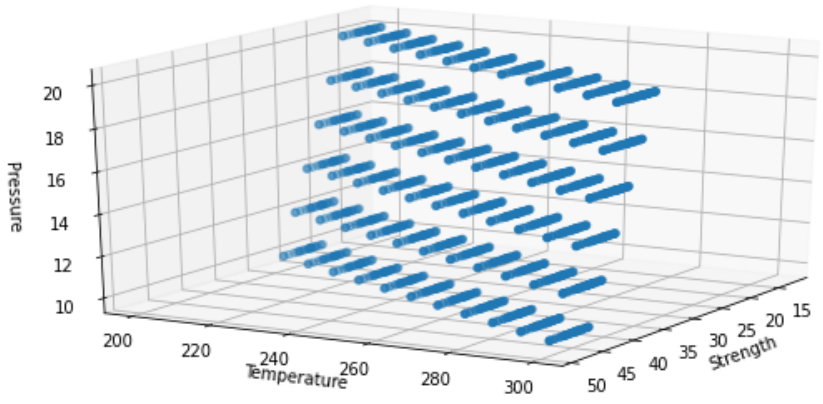
2. Experimental Setup

2.1 Dataset

The dataset used for this regression task consists of 2 feature attributes, ‘Strength’ and ‘Temperature’ used to predict ‘Pressure’, which is the target attribute. The dataset consists of 1650 different data points. A sample of which is shown below.

	Strength	Temperature	Pressure
0	30.7	240	16
1	24.7	250	18
2	30.6	260	16
3	32.8	240	10
4	20.7	240	20
...	...	...	...

The raw data is plotted as shown below.

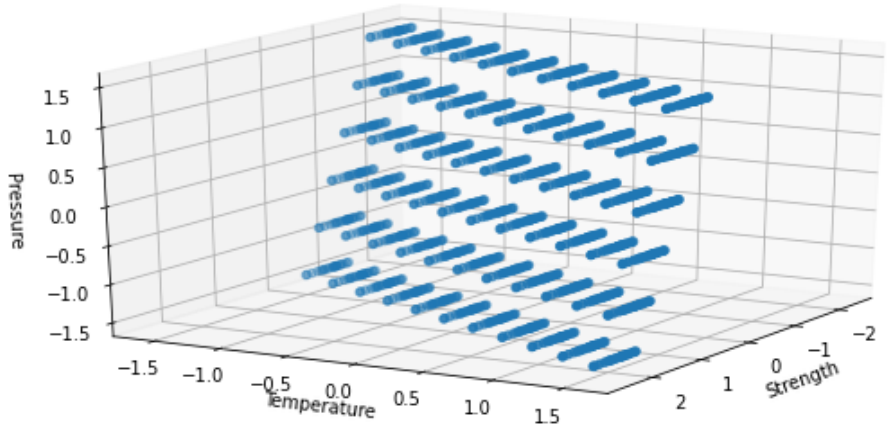


2.2 Data Preprocessing

Before implementing the algorithms, we pre-process the data which includes,

- Shuffling the data,
- Standardizing the values, and
- Creating a random 70-30 split to aid in training and testing respectively.

Dataset after preprocessing is plotted below.



2.3 Model

A polynomial regression is performed to predict the Pressure values from strength and temperature. In this assignment, the performance of polynomials of degrees ranging from 0 to 9 are accessed. As more than one feature is present in the given dataset, multivariate polynomials are used. They are in the following form:

$$p(x,y) = \sum_{k=0}^{4} \sum_{j=0}^{k} p_{j,k} x^j y^{k-j}$$

p (x, y) = y^m + x . y^m−1 + x^2 . y^m−1 + · · · + x^m−1 . y + x^m

Gradient Descent and Stochastic Gradient Descent are used to optimize the model to best fit the data given and learn the relationship between features. In order to overcome overfitting observed in higher degree polynomials, regularization was used. Lasso and Ridge Regression were performed.

2.4 Implementation of the model in Python

All computations of the Regression algorithm are done in a class named Regression.

1. The first step is to transform the features to Polynomial features of the corresponding degree. The number of polynomial features will be C(n+d, d) where n is the number of input features, and d is the degree.
2. Next, the fit method of the class, which trains the model on the training, is called with parameters - the learning rate, the number of epochs the gradient descent should run for and the type of gradient descent - ‘Stochastic’ or ‘Batch’. A variable w, representing the weights, is initialized randomly.

3. The model is then trained, and the weights updated using the Mean Square Error between the predicted and true output value. The weight update can be done using either,
    a. Gradient Descent: For each datapoint in the training set, the error is calculated, multiplied by the learning rate and subtracted from the weights.
    b. Stochastic Gradient Descent: A random datapoint from the training set is chosen, and the weights are updated using this single error.
4. Once the dataset is trained and the error saturates, the evaluate method is called with the testing data as the parameters. The testing Loss/Error is calculated and reported.

Cost function varies for no regularization and regularization (ridge and lasso regression).

3. Results

The summary of all the results collected are shown in Table 1. Detailed description of the results is given in the following subsections.

Table 1. Polynomial Regression

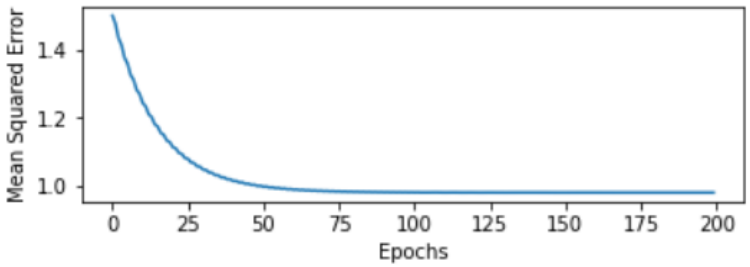
Degree of Polynomial	Training (GD/SGD)	Testing Error	Training Error
0	GD	1.0614	0.9812
0	SGD	1.0720	0.9761
1	**GD**	**0.1932**	**0.2029**
1	SGD	0.2076	0.2132
2	**GD**	**0.1936**	**0.2006**
2	SGD	0.2044	0.2149
3	GD	0.2142	0.2298
3	SGD	0.2060	0.2215
4	GD	0.2067	0.2115
4	SGD	0.2623	0.2772
5	GD	0.2115	0.2147
5	SGD	0.3642	0.3403
6	GD	0.2472	0.2776
6	SGD	0.3779	0.4818
7	GD	0.2534	0.2596
7	SGD	0.6372	0.7207

8	GD	0.3487	0.3354
8	SGD	0.8251	0.9991
9	GD	0.4258	0.4022
9	**SGD**	**2.5263**	3.1931

3.1 Analysis based on Degree of Polynomial and Optimization algorithms

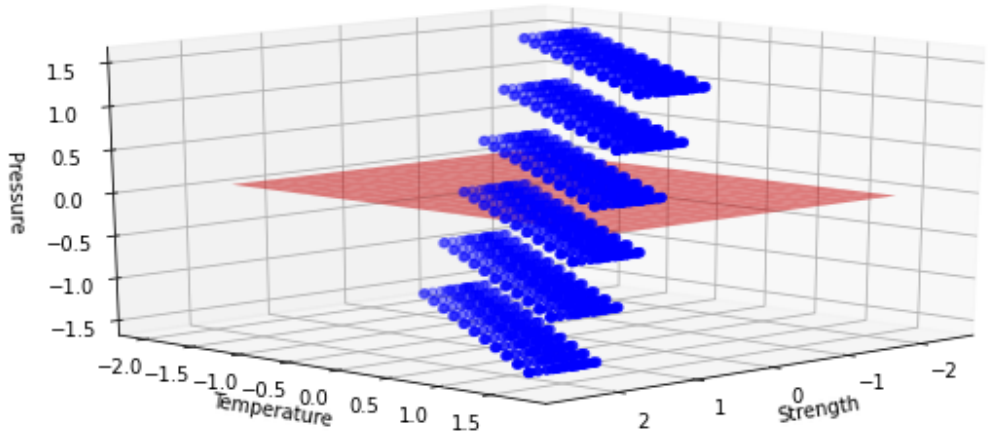
3.1.1 Degree 0 Polynomial

i. Gradient Descent

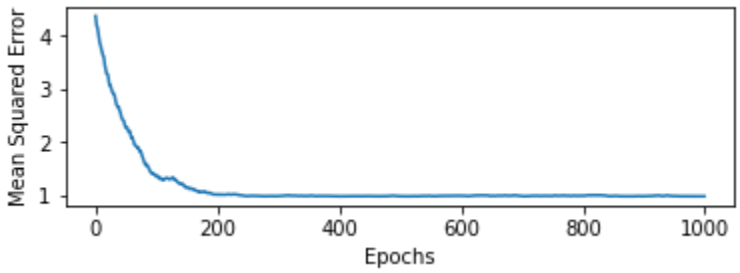


Minimum Training Error = 0.9812
Testing Error = 1.0614

Surface Plot:

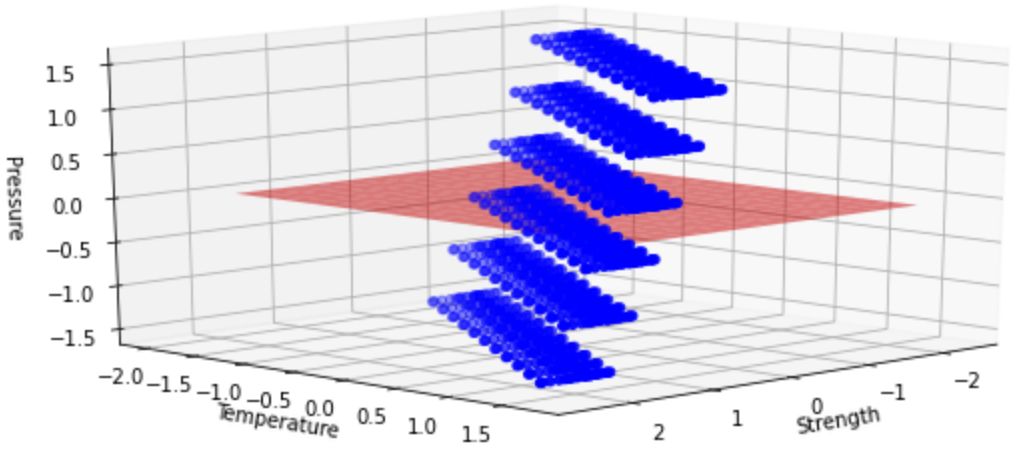


ii. Stochastic Gradient Descent



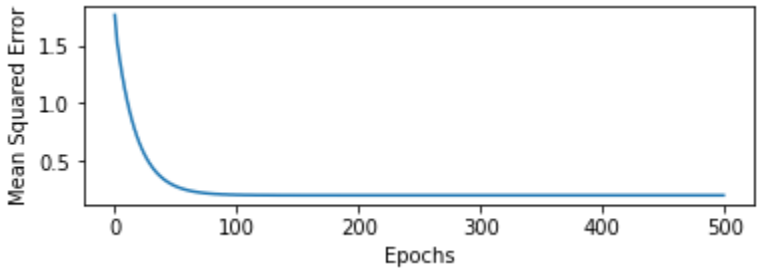
Minimum Training Error = 0.9761
Testing Error = 1.0720

Surface Plot:



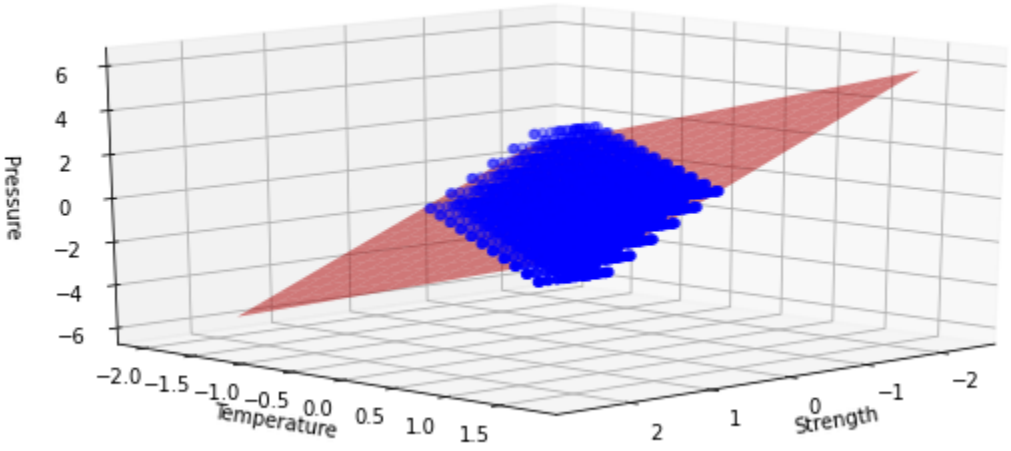
3.1.2 Degree 1 Polynomial

i. Gradient Descent:

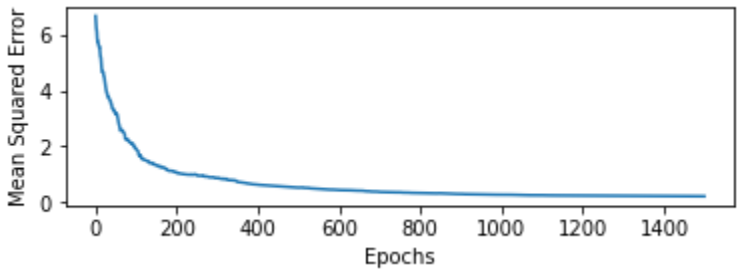


Minimum Training Error = 0.2029
Testing Error = 0.1932

Surface Plot:

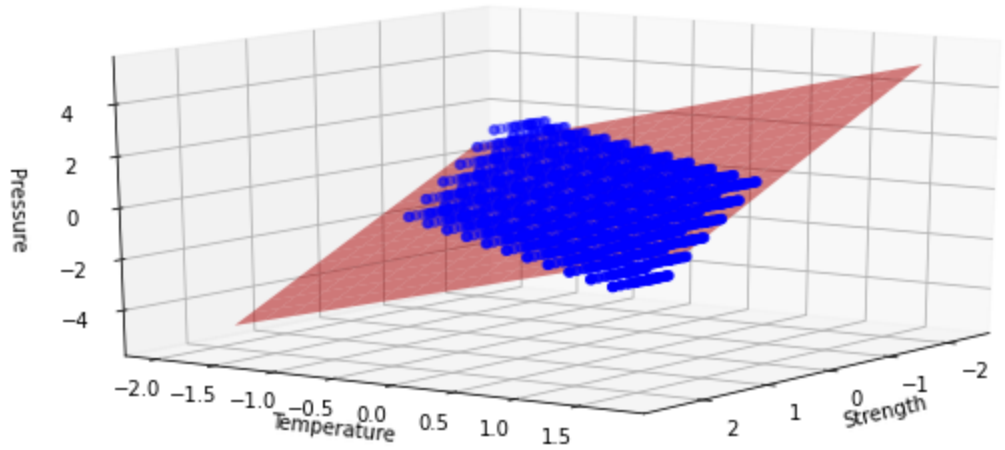


ii. Stochastic Gradient Descent:



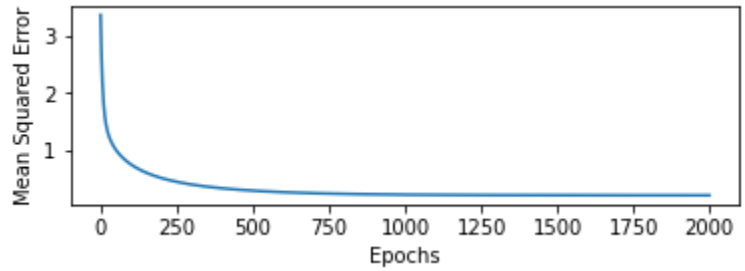
Minimum Training Error = 0.2132
Testing Error = 0.2076

Surface Plot:



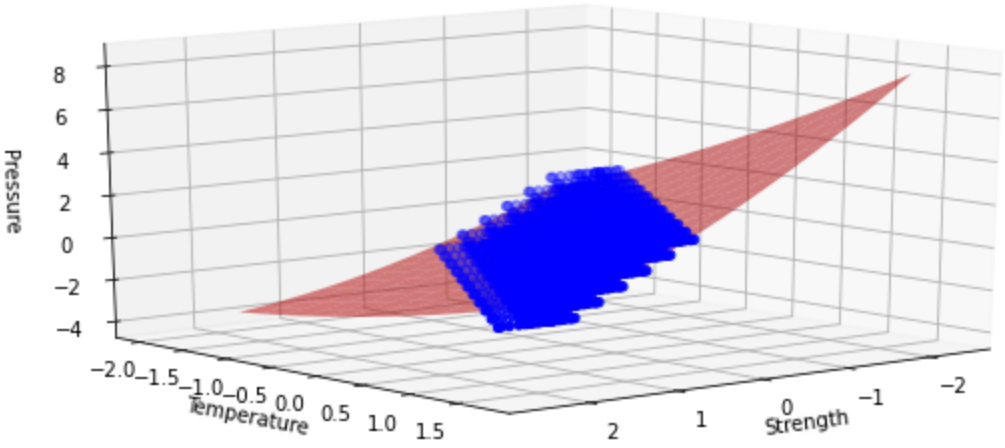
3.1.3 Degree 2 Polynomial

i. Gradient Descent:

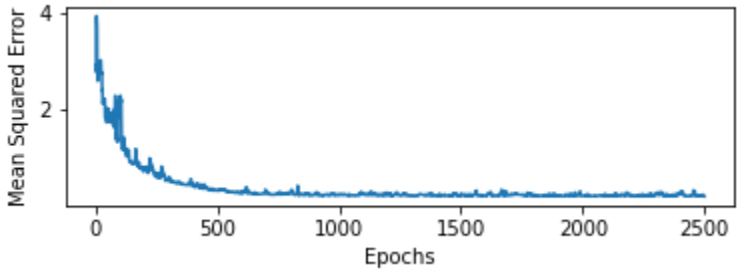


Minimum Training Error = 0.2006
Testing Error = 0.1936

Surface Plot:

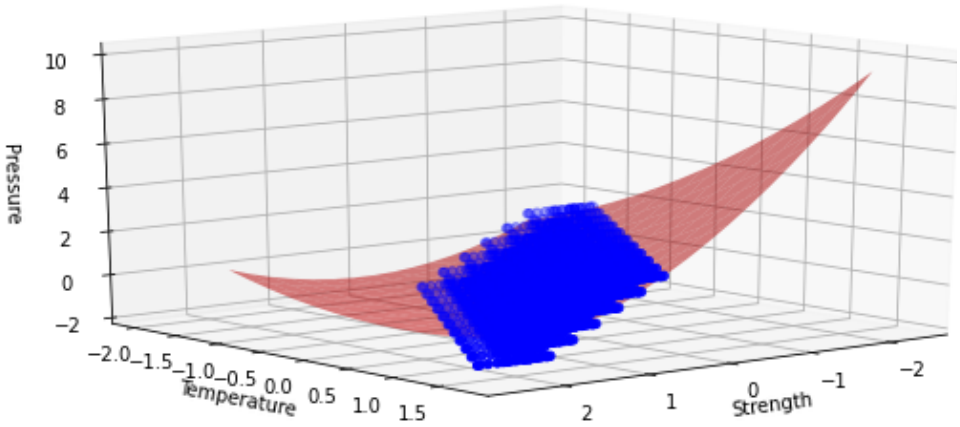


ii. Stochastic Gradient Descent:



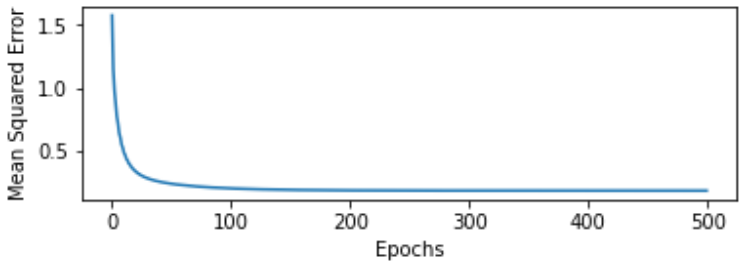
Minimum Training Error = 0.2149
Testing Error = 0.2044

Surface Plot:



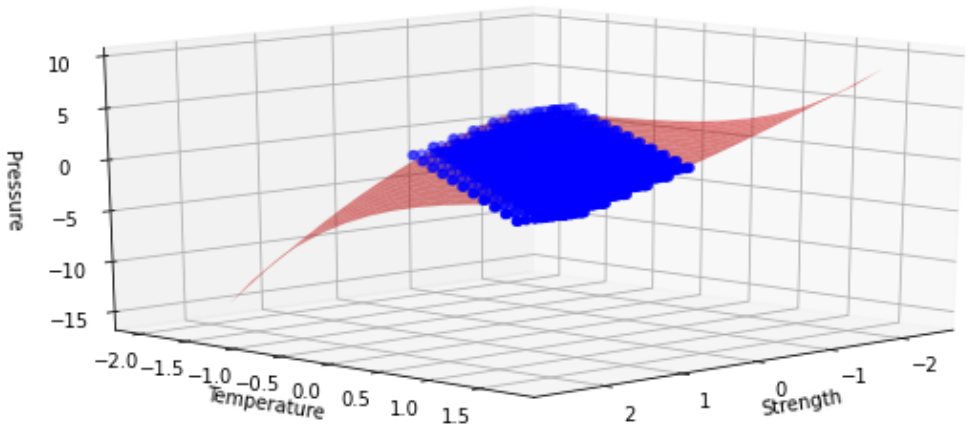
3.1.3 Degree 3 Polynomial

i. Gradient Descent:

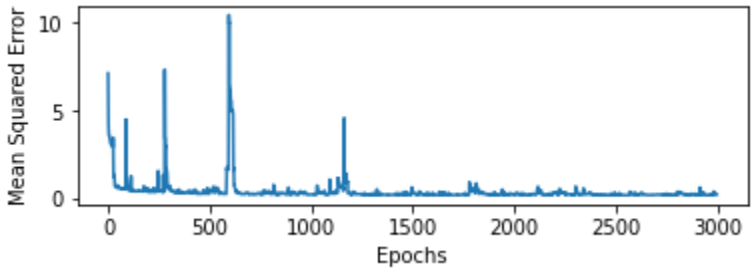


Minimum Training Error = 0.2298
Testing Error = 0.2142

Surface Plot:

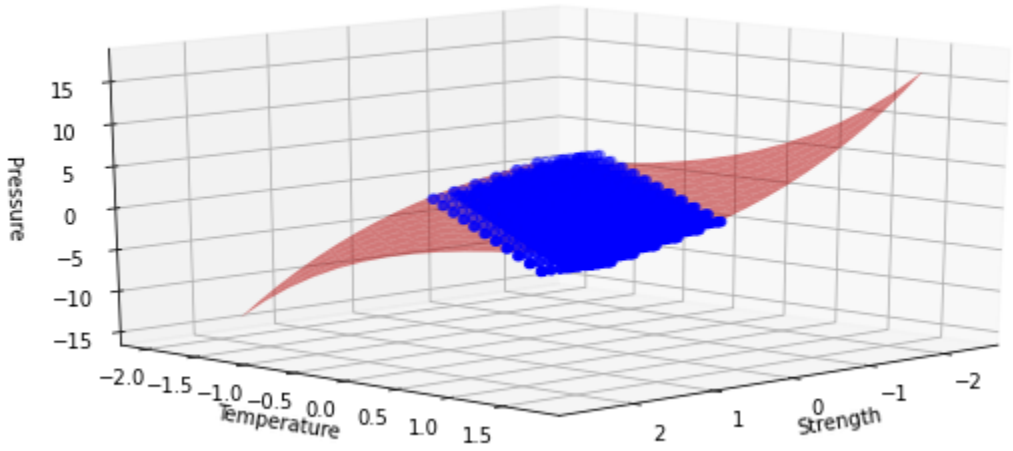


ii. <u>Stochastic Gradient Descent:</u>



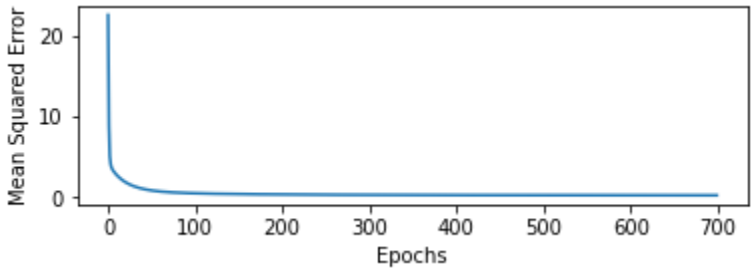
Minimum Training Error = 0.2215
Testing Error = 0.2060

Surface Plot:



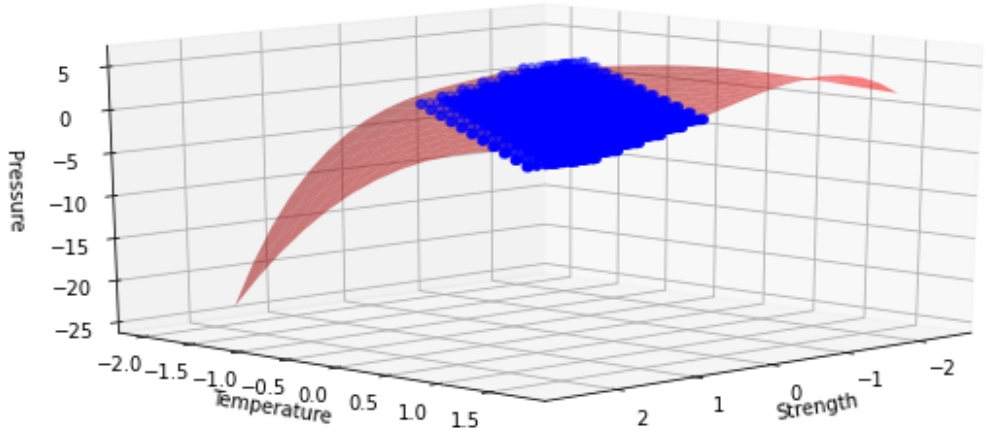
3.1.5 Degree 4 Polynomial

i. Gradient Descent:

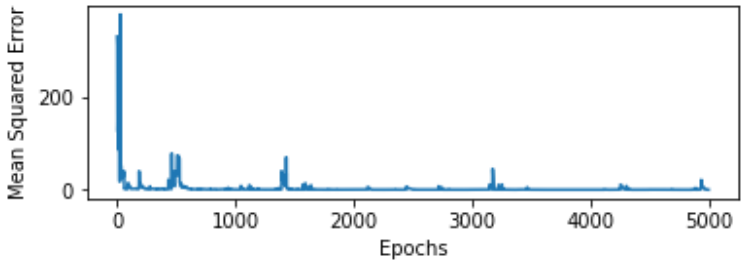


Minimum Training Error = 0.2115
Testing Error = 0.2067

Surface Plot:

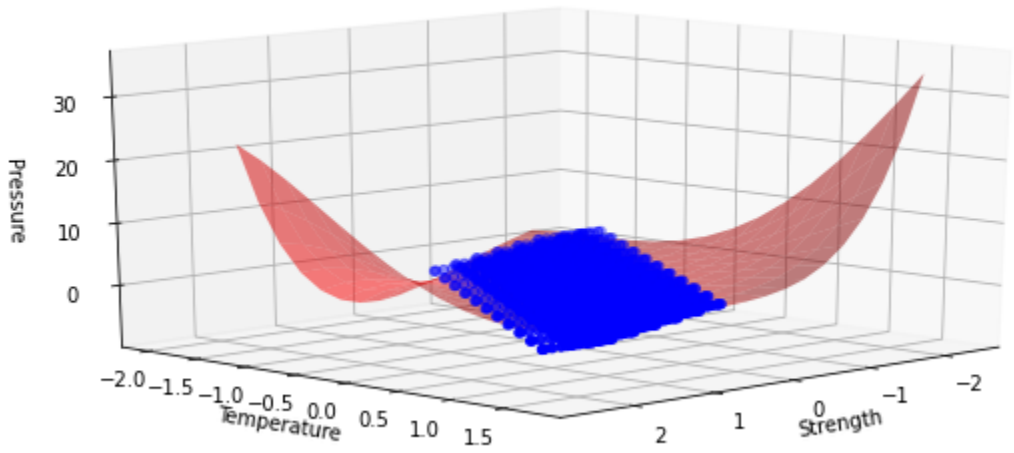


ii. Stochastic Gradient Descent:



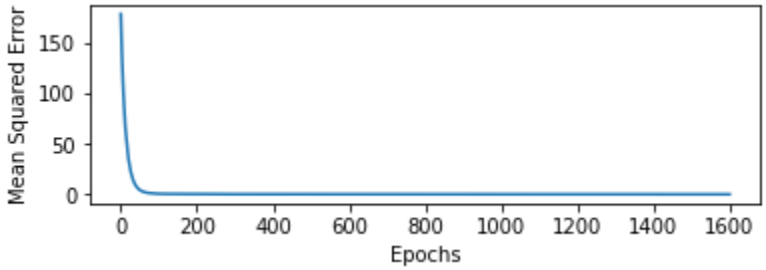
Minimum Training Error = 0.2772
Testing Error = 0.2623

Surface Plot:



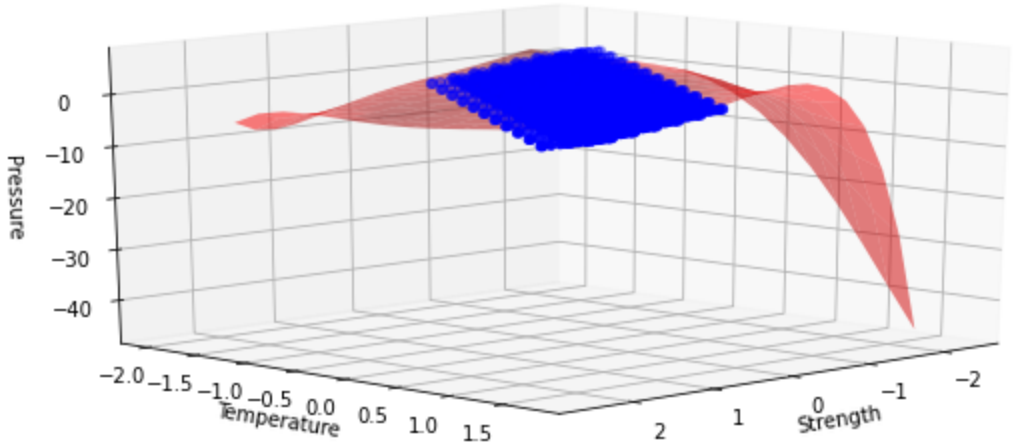
3.1.6 Degree 5 Polynomial

i. Gradient Descent:

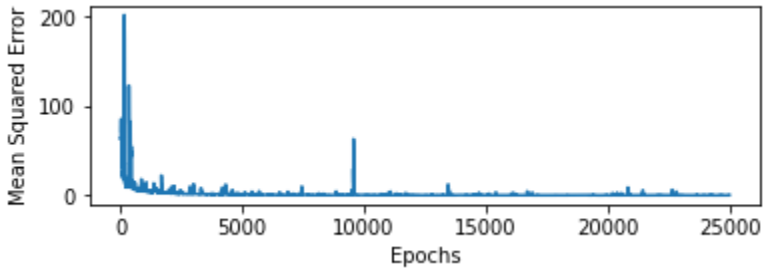


Minimum Training Error = 0.2147
Testing Error = 0.2115

Surface Plot:

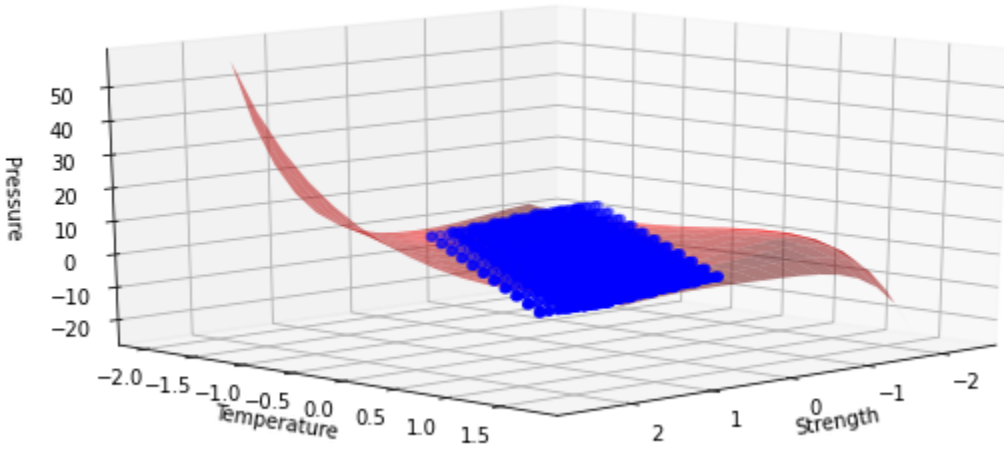


ii. Stochastic Gradient Descent:



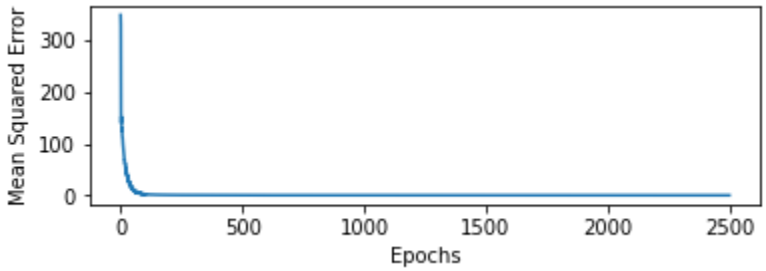
Minimum Training Error = 0.3403
Testing Error = 0.3642

Surface Plot:



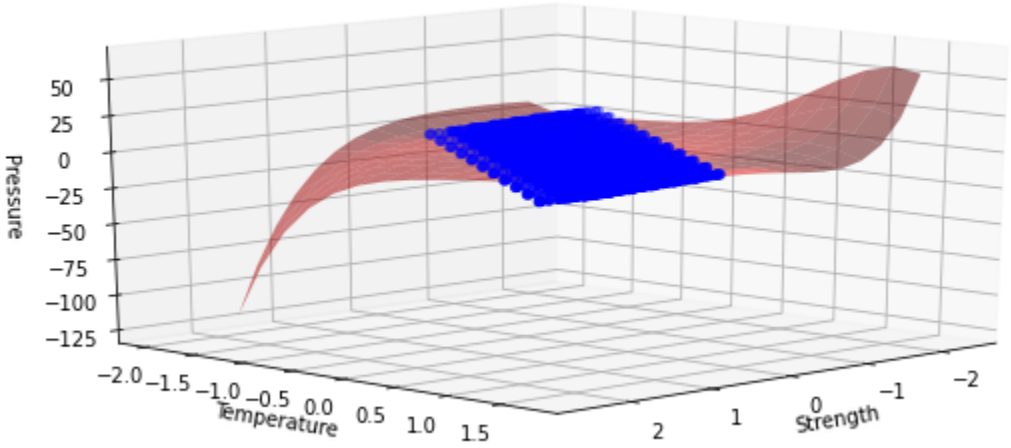
3.1.7 Degree 6 Polynomial

i. Gradient Descent:

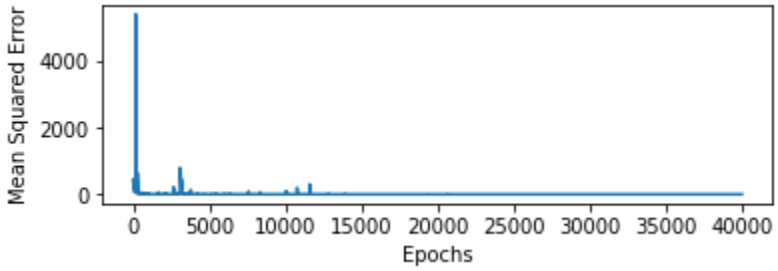


Minimum Training Error = 0.2776
Testing Error = 0.2472

Surface Plot:

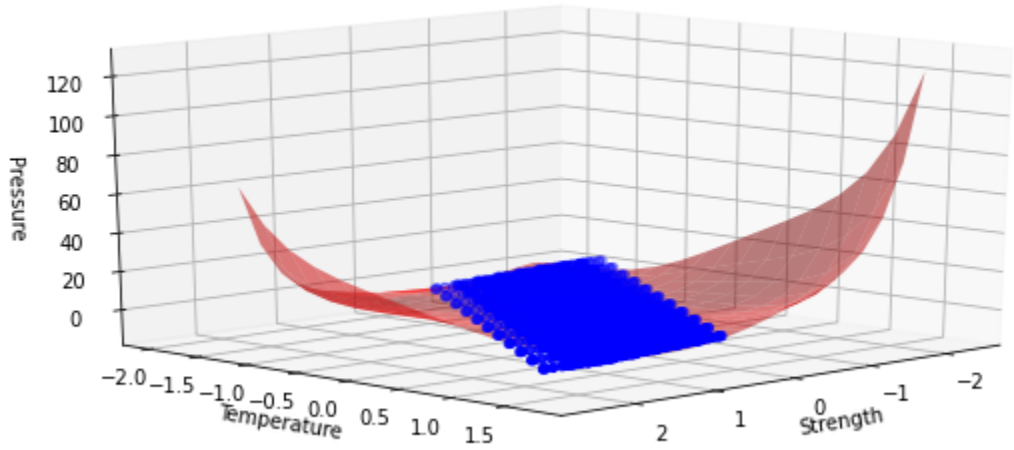


ii. Stochastic Gradient Descent:



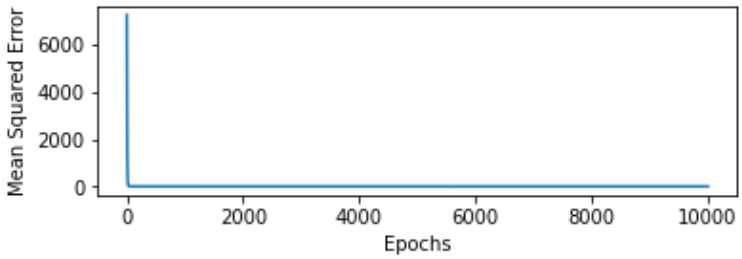
Minimum Training Error = 0.4818
Testing Error = 0.3779

Surface Plot:



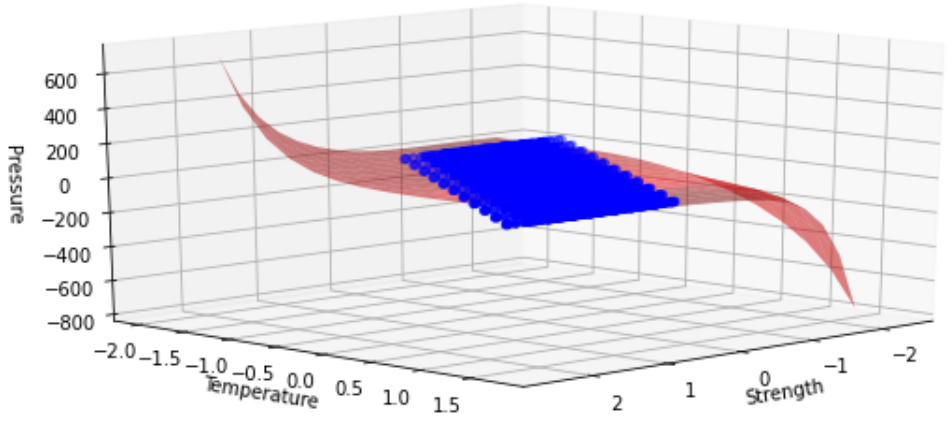
3.1.8 Degree 7 Polynomial

i. Gradient Descent:

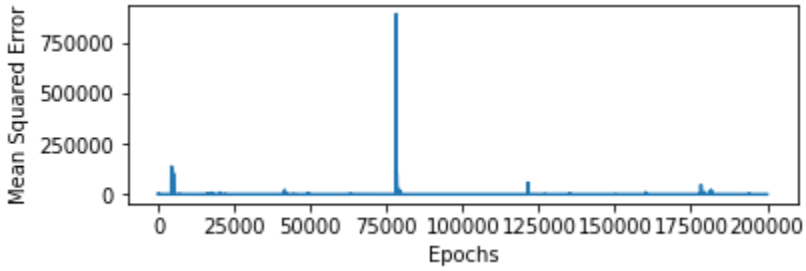


Minimum Training Error = 0.2596
Testing Error = 0.2534

Surface Plot:

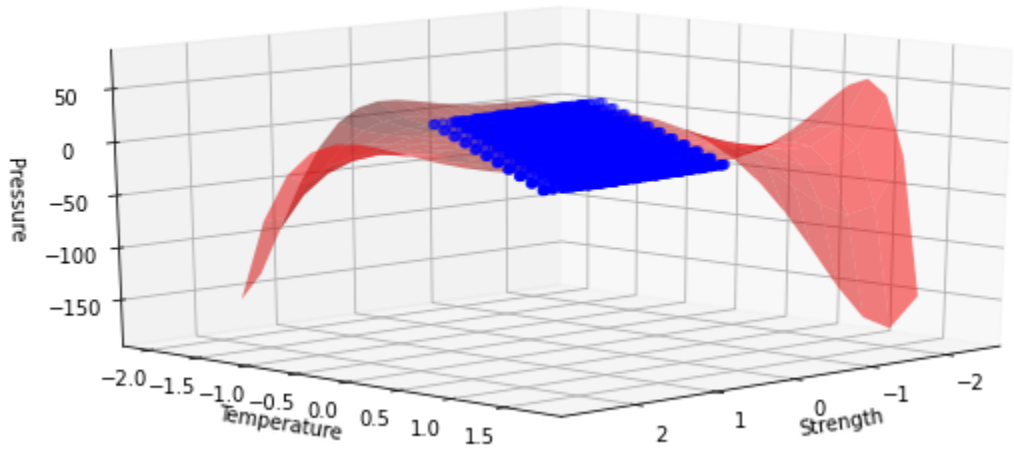


ii. Stochastic Gradient Descent:



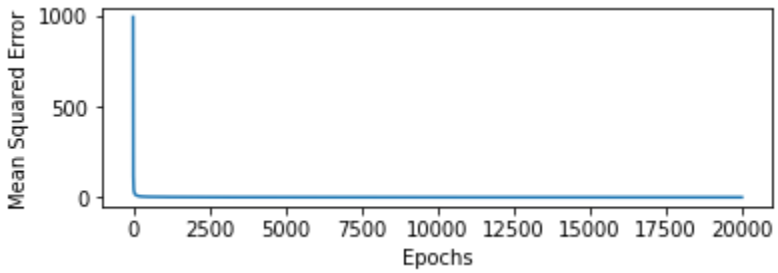
Minimum Training Error = 0.7207
Testing Error = 0.6372

Surface Plot:



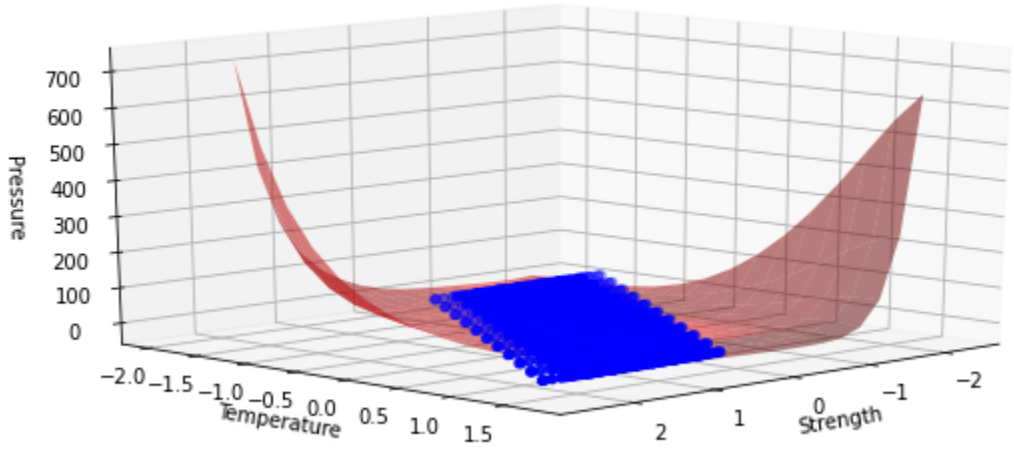
3.1.9 Degree 8 Polynomial

i. Gradient Descent:

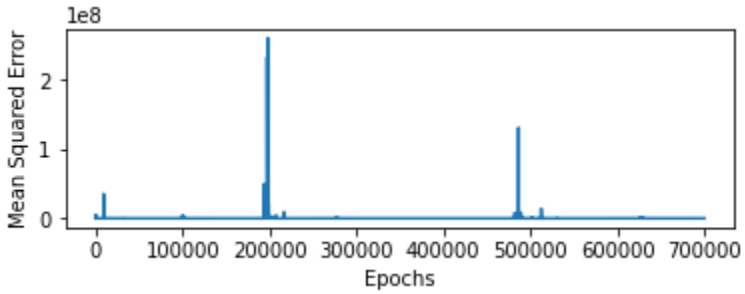


Minimum Training Error = 0.3354
Testing Error = 0.3487

Surface Plot:

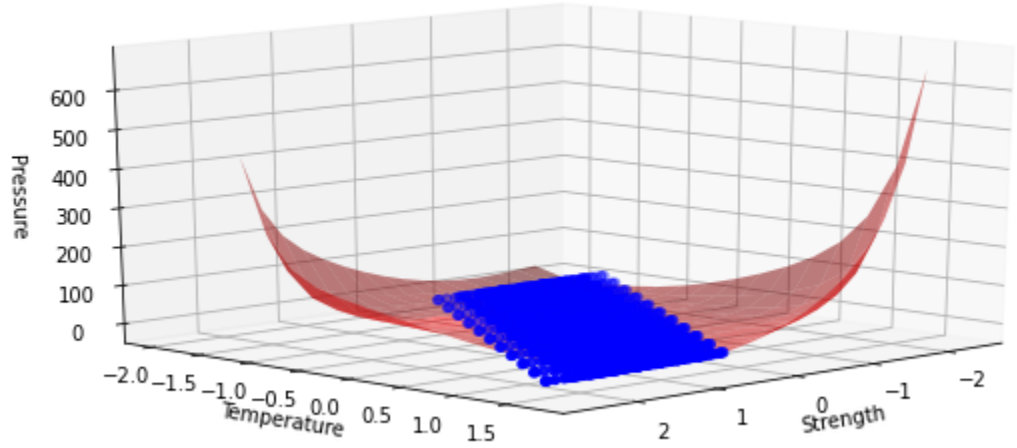


ii. Stochastic Gradient Descent:



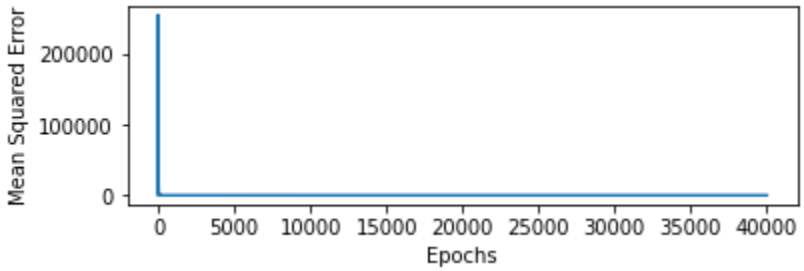
Minimum Training Error = 0.9991
Testing Error = 0.8251

Surface Plot:



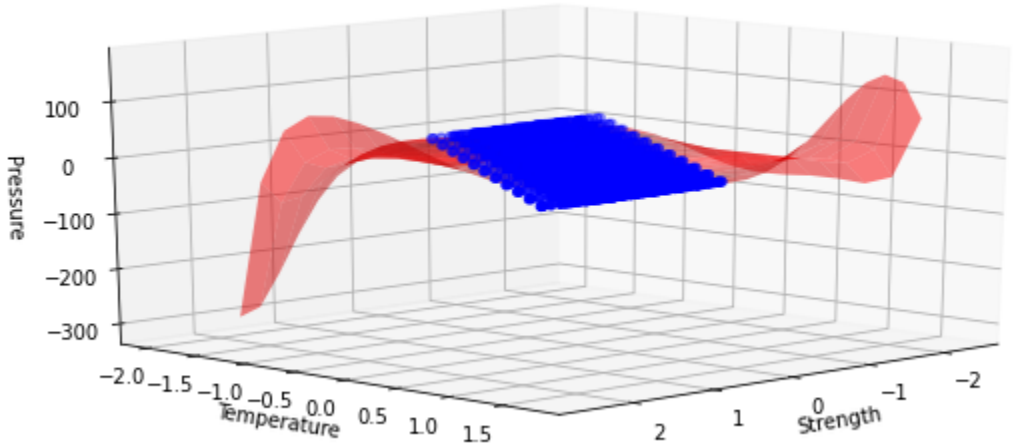
3.1.10 Degree 9 Polynomial

i. Gradient Descent:

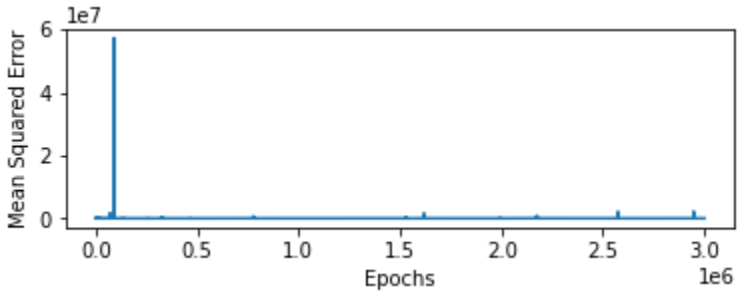


Minimum Training Error = 0.4022
Testing Error = 0.4258

Surface Plot:



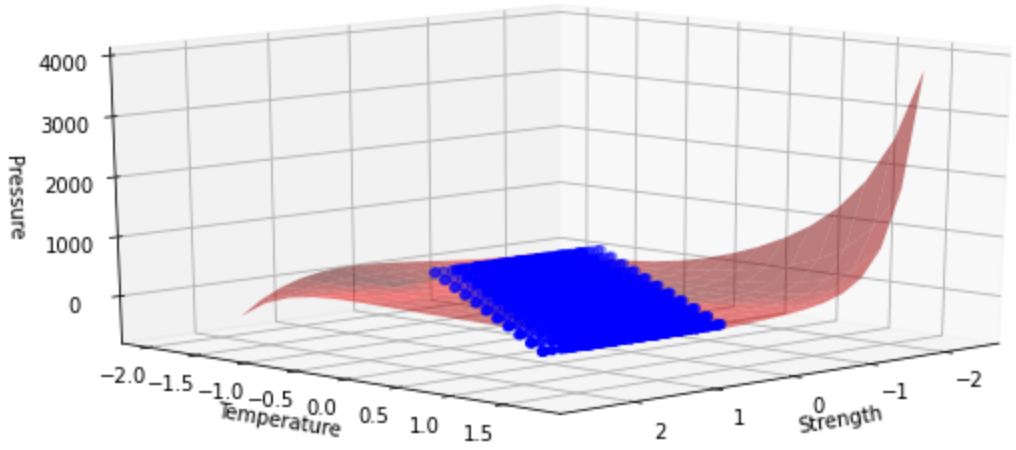
ii. Stochastic Gradient Descent:



Minimum Training Error = 3.1931

Testing Error = 2.5263

Surface Plot:



3.1.11 Best Model

As shown in Table 1, the 1-degree and 2-degree polynomial regression models give the best testing errors, thus the best performance.

3.2 Overfitting

It can be observed that overfitting is taking place in higher degree polynomial models.This occurs because a large number of features modelled by coefficients of the polynomials, tend to describe the error in the data rather than relationship between the original features. Therefore increasing the complexity is not the solution to get the optimal statistical model to fit the data. We make use of various techniques such as regularization in order to avoid overfitting.

3.3 Analysis based on Ridge and Lasso Regression

As discussed earlier regularization techniques such as Lasso and Ridge Regression are performed to overcome overfitting in Higher degree polynomial models. In this assignment regularization is implemented along with Gradient Descent and Stochastic Gradient Descent Optimization Algorithms. The results are shown in the following subsections.

Figure 1.Geometrical Interpretation of Lasso and Ridge Regression

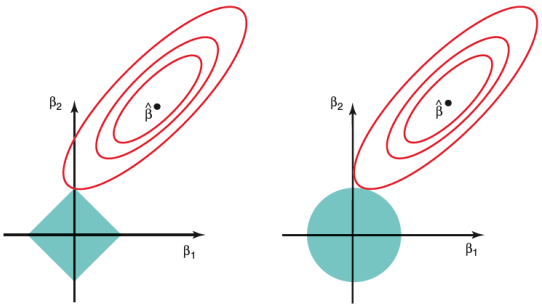


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

3.3.1 Lasso Regression

Lasso (Least Absolute Shrinkage and Selection Operator) is a type of regression where the model is regularized by shrinking the values. In case of polynomial regression, only some of the weights/coefficients of terms in the polynomial are chosen thus decreasing the complexity of the algorithm which results in removing overfitting to some extent. Therefore, here we get a sparse solution. The objective function for Lasso Regression (for gradient descent) is:

$$RSS_{LASSO}(w,b) = \sum_{(i=1)}^{N} (y_i - (w \cdot x_i + b))^2 + \alpha \sum_{(j=1)}^{p} |w_j|$$

Gradient to minimize the above objective function is:

1. Gradient Descent :
```
elif regularizer==1:
    return (1/m)*x.T.dot(y_pred - y) + ( (lam/2*m) * ( np.sign(self.w) )
```
2. Stochastic Gradient Descent :
```
elif regularizer==1:
    self.w -= learning_rate * (dw + (lam/2*m) * ( np.sum( np.sign(self.w) )
```

The geometric representation of the same is shown in figure 1.

Table 2 shows the results of Lasso Regression with Gradient Descent. Initially the training was started with lambda value at 0.0185 and then it was reduced till 0.003 where the best performance was achieved. Further reducing the lambda worsened the performance. Here it can be observed that there is not overfitting in any of the models.

Table 2. Lasso Regression with Gradient Descent

Lambda	Train RMSE	Test RMSE
0.0005	1.2540	1.3116
0.0010	**1.4040**	**1.4250**

0.0015	1.1387	1.1709
0.0020	1.1567	1.1563
0.0025	0.9136	0.9252
0.0030	**0.8794**	**0.8749**
0.0035	0.9929	1.0282
0.0040	0.9776	0.9866
0.0045	**0.8629**	**0.8811**
0.0075	0.9589	0.9645
0.0085	0.9586	0.9792
0.0125	0.9646	0.9645
0.0185	0.9562	0.9635
0.0200	0.9671	0.9630
0.0225	0.9633	0.9627
0.0240	0.9714	0.9700
0.0245	0.9623	0.9602

To compare the test and train root mean square errors we trained some more models with lambda values ranging from 0 and 0.1 as shown in figure 2. We further explored till lambda=1, the resulting graph is shown in figure 3.

Figure 2. RMSE Test and Train vs Lambda [0,0.1] - Gradient Descent

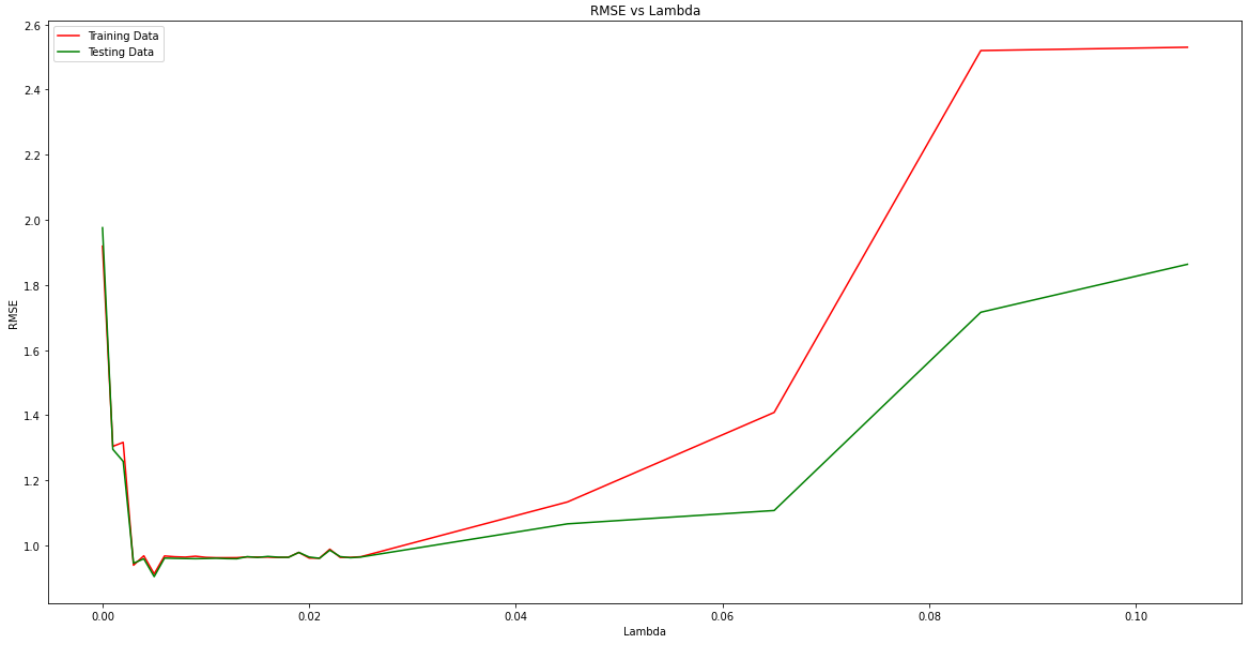
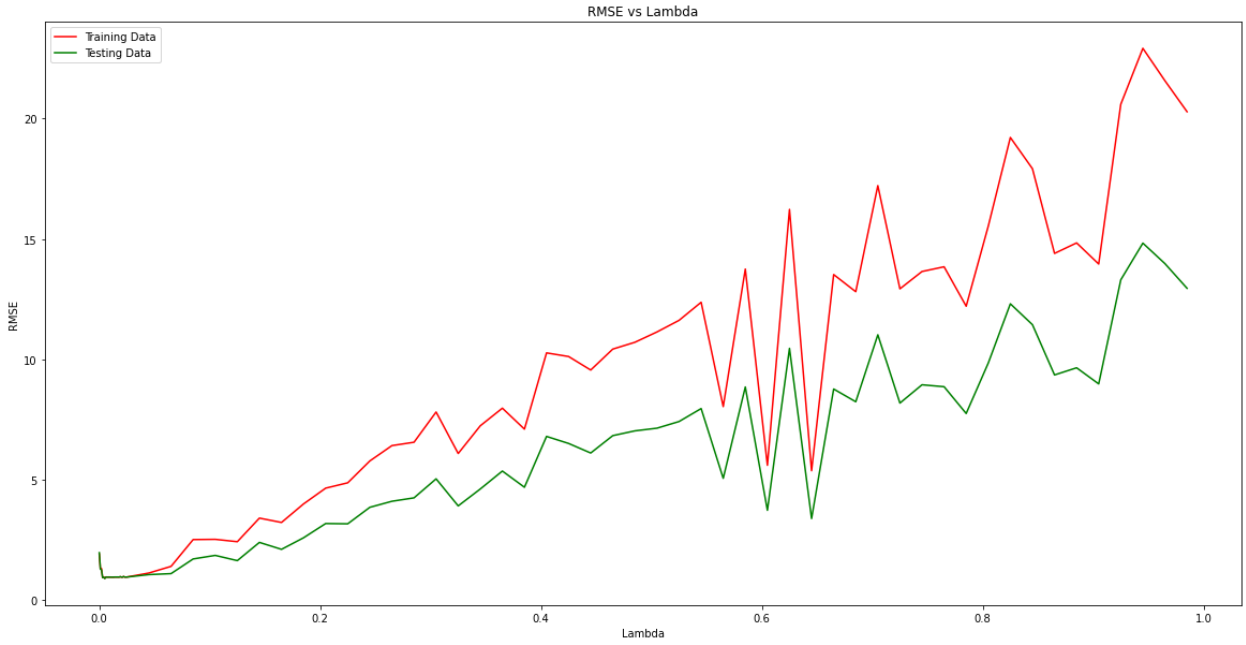


Figure 3. RMSE Test and Train vs Lambda [0,1] - Gradient Descent



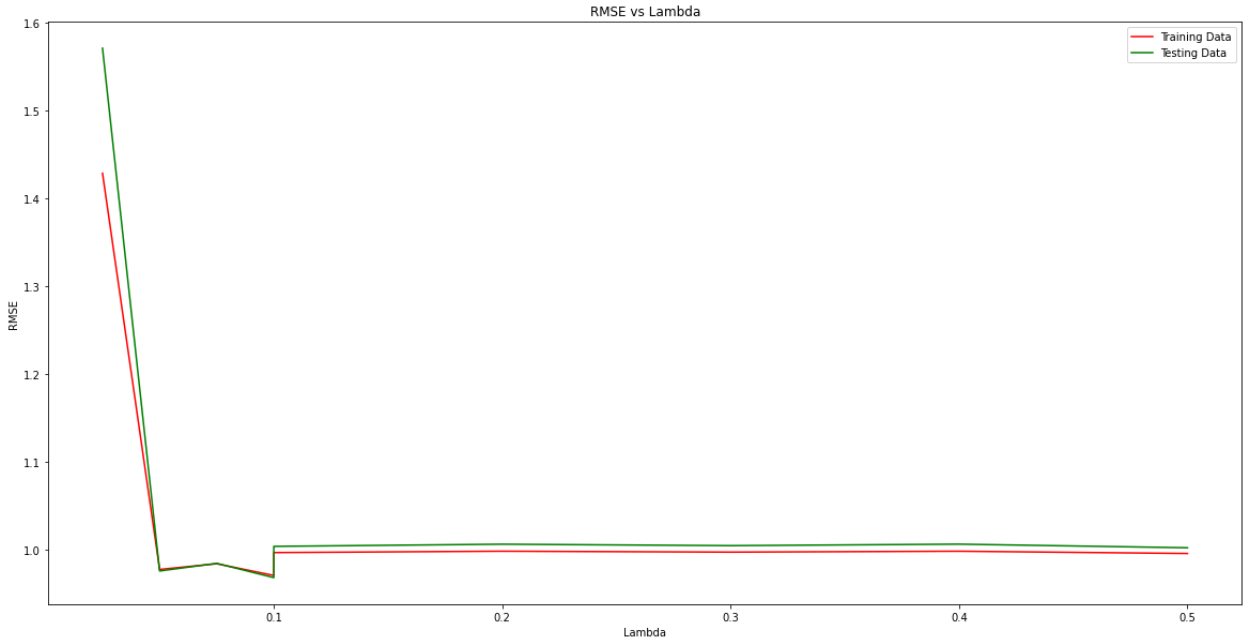
Similarly we conducted experiments with Stochastic Gradient Descent. The results were poorer compared to Gradient Descent. Results are shown in Table 3. Here too, we started out with 0.5 and kept reducing the lambda till it started increasing again.

Table 2. Lasso Regression with Stochastic Gradient Descent

Lambda	Train RMSE	Test RMSE
0.0250	1.4288	1.5713
0.0500	0.9771	0.9753
0.0750	0.9836	0.9840
0.1000	**0.9702**	**0.9678**
0.1000	0.9963	1.0034
0.2000	0.9979	1.0059
0.3000	0.9968	1.0042
0.4000	0.9978	1.0059
0.5000	0.9953	1.0018

Best performance is achieved at lambda=0.01. It can be observed that error increases for lambda below 0.05 and error is stagnant for lambda between 0.075 and 0.5. Figure 4 shows the plot between Train Root Mean Squared Error and Test RMSE for different values of Lambda.

Figure 4. RMSE Test and Train vs Lambda [0.025, 0.5] - Stochastic Gradient Descent



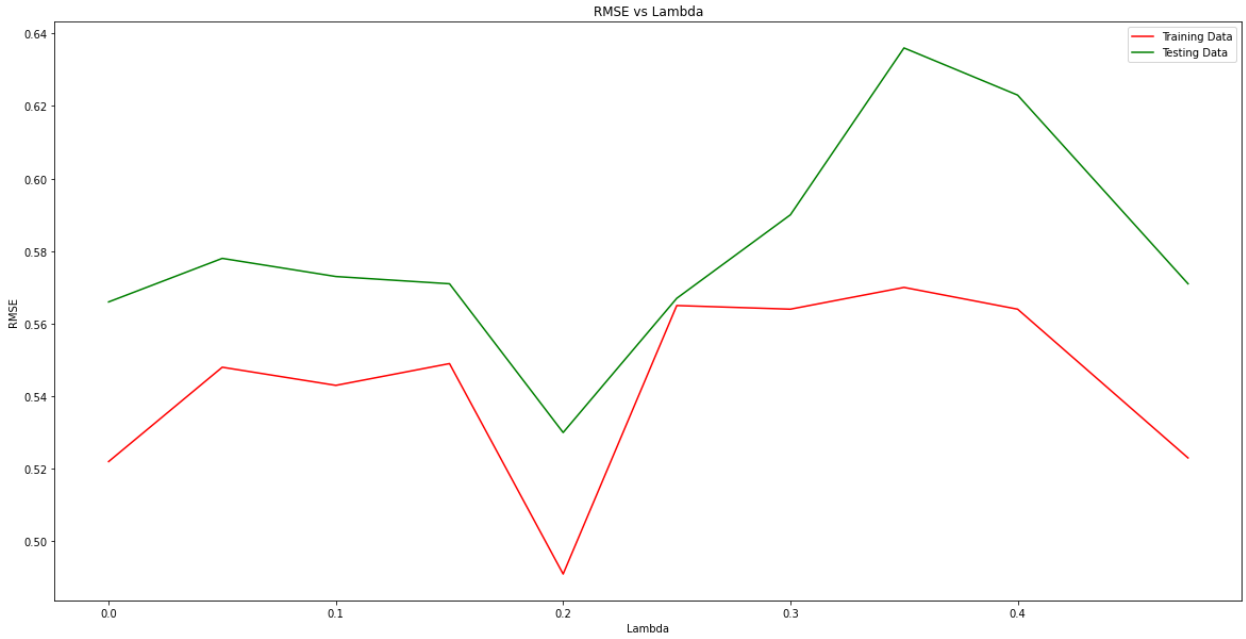
3.3.2 Ridge Regression

Table 3. Ridge Regression with Gradient Descent

Lambda	Train RMSE	Test RMSE
0.0000	0.5224	0.5655
0.0500	0.5476	0.5779
0.1000	0.5429	0.5725
0.1500	0.5489	0.5714
0.2000	**0.4909**	**0.5297**
0.2500	0.5653	0.5665
0.3000	0.5638	0.5900
0.3500	0.5698	0.6362
0.4000	0.5643	0.6231
0.4750	0.5230	0.5708

Best performance is achieved at lambda=0.2. It can be observed that error is stagnant for lambda between 0 and 0.475. Figure 4 shows the plot between Train Root Mean Squared Error and Test RMSE for different values of Lambda.

Figure 5. RMSE Test and Train vs Lambda [0, 0.475] - Gradient Descent



Similarly we conducted experiments with Stochastic Gradient Descent. The results were poorer compared to Gradient Descent. Results are shown in Table 4.

Table 4. Ridge Regression with Stochastic Gradient Descent

Lambda	Train RMSE	Test RMSE
0.0250	6.5645	9.4495
0.0500	3.3658	4.1941
0.0750	3.4015	3.4396
0.1000	3.3477	2.9501
0.2000	**2.4805**	**2.5189**
0.3000	3.0117	2.9959
0.4000	3.0476	3.8649
0.5000	2.6885	3.5205

Best performance is achieved at lambda=0.2. It can be observed that error increases as lambda decreases below 0.05 and error is stagnant for lambda between 0 and 0.475. Figure 4 shows the plot between Train Root Mean Squared Error and Test RMSE for different values of Lambda.

Figure 6. RMSE Test and Train vs Lambda [0.025, 0.5] - Stochastic Gradient Descent

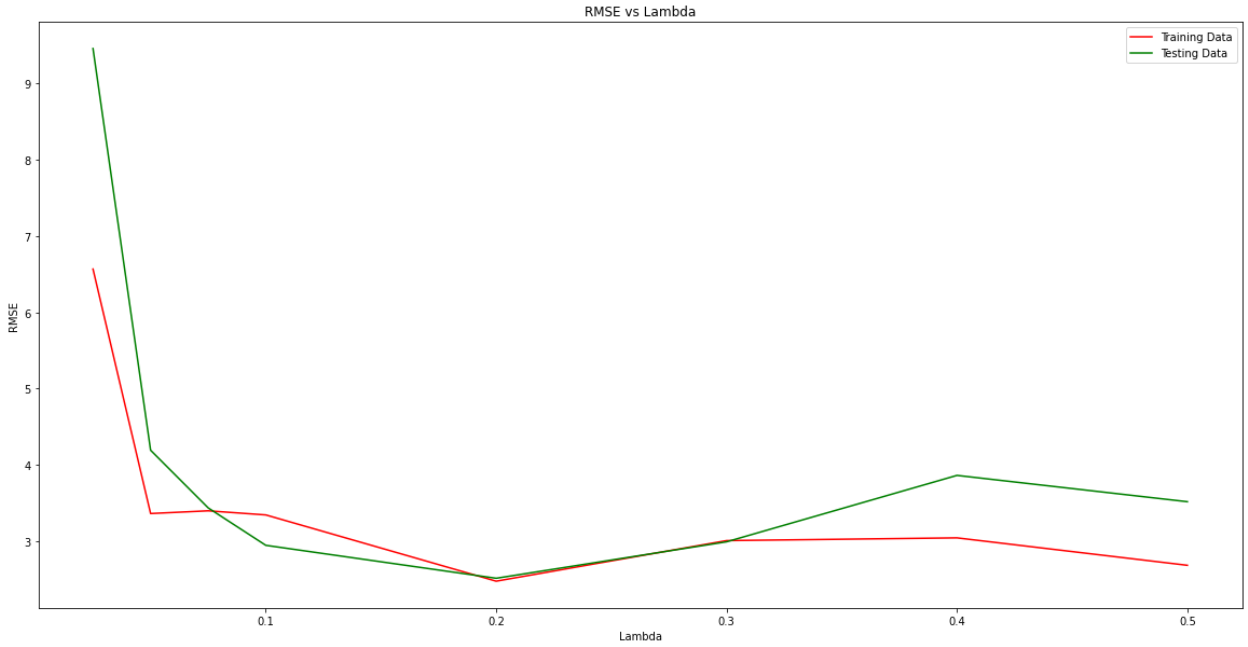


Table : Ridge and Lasso Regression Results with Stochastic Gradient Descent

Lambda	Ridge Regression		Lasso Regression	
	Train RMSE	Test RMSE	Train RMSE	Test RMSE
0.0250	6.5645	9.4495	1.4288	1.5713
0.0500	3.3658	4.1941	0.9771	0.9753
0.0750	3.4015	3.4396	0.9836	0.9840
0.1000	3.3477	2.9501	**0.9702**	**0.9678**
0.2000	**2.4805**	**2.5189**	0.9979	1.0059
0.3000	3.0117	2.9959	0.9968	1.0042
0.4000	3.0476	3.8649	0.9978	1.0059
0.5000	2.6885	3.5205	0.9953	1.0018

4. Comparative Analysis between performance with and without Regularization

Table 5. Best Models with least Test RMSE values

	Gradient Descent	Stochastic Gradient Descent
No Regularization	0.4395	0.4521
Ridge	0.5297	2.5189
Lasso	0.8749	0.9678

Lower degree polynomials without regularization perform far better than higher degree polynomials while testing, as shown in Table 1. However, when regularization is applied with an appropriate Lamda and regularization type, higher degree polynomials show similar performance to lower degree polynomials without regularization, as shown in Table 5.

Further, from Table 5 it can be inferred that models trained using gradient descent optimization usually perform better than stochastic gradient descent.