COL341
Assignment 1

Kushagra Rode

February 12, 2023

3.1

Some Key Points :

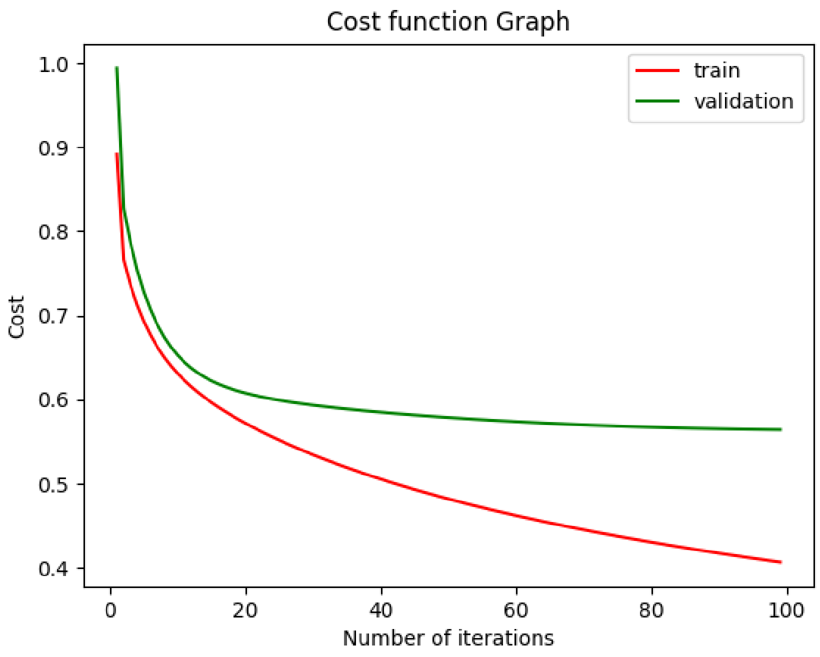
1. The initial weights for my model are set as zeros. I tried with random weights but they were taking a very long time to converge whereas when initialised by zeros it converges fairly quickly.
2. I have not taken the norm of the gradient as mentioned by the Teaching Assistant

Let us first start with the variation of learning rates
For learning rates 0.1 and 0.01, the error shoots up to really large values(of the order of 10^{128}) within few iterations only. So, these learning rates result in divergence of the cost function very quickly and hence are not appropriate for training our model.
Now, as far as learning rate 0.001 is concerned we can talk about the two stopping criterion and the mse and mae on train and validation data.

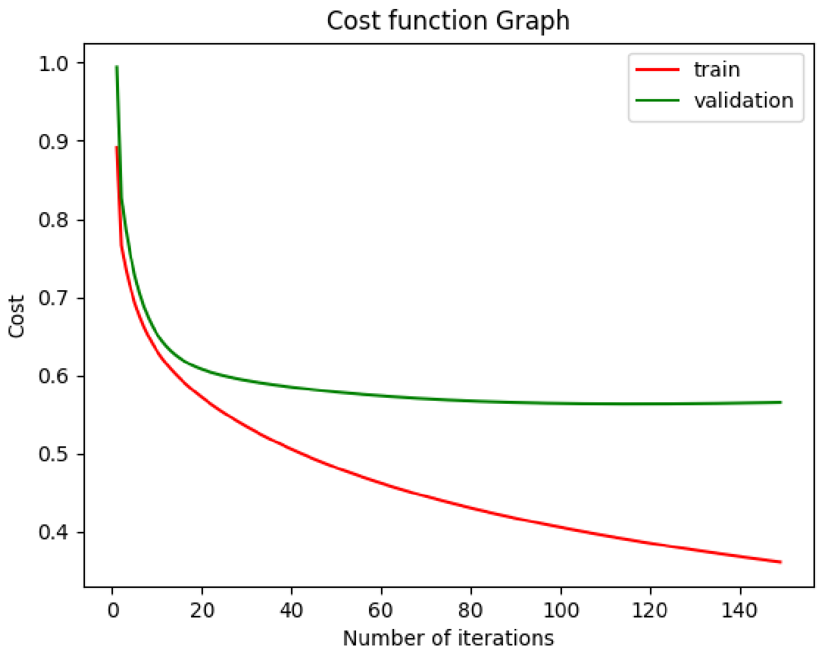
Stopping Criteria - maxit

The learning rate is quite good as the model converges pretty quickly to its optimal value. Following the are the mse and mae errors at various iterations I ran my code on.
100 iterations -
MSE train - 0.40563860533241736
MAE train - 0.5098124729409207
MSE validation - 0.5641227732034114

MAE validation - 0.5664336599762227

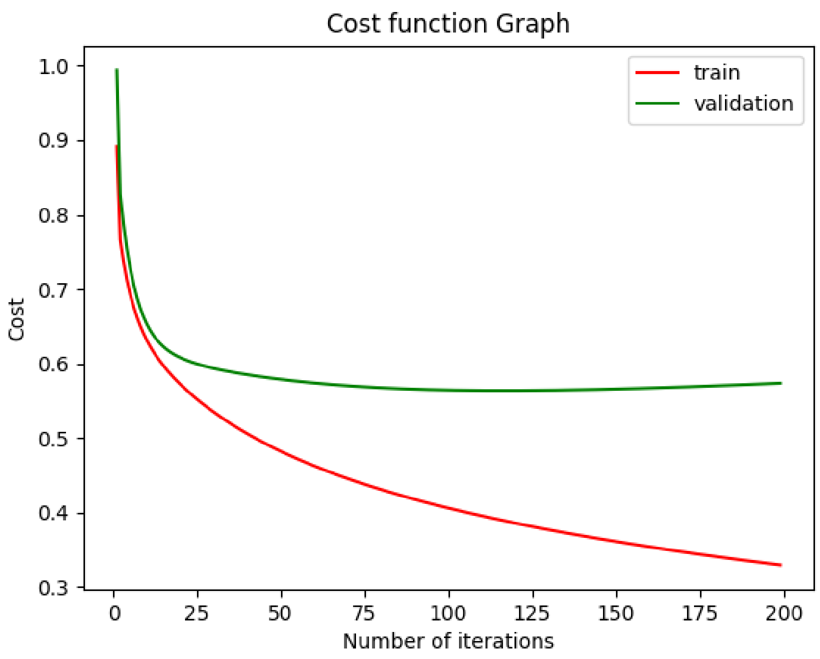


150 iterations -
MSE train - 0.3606670874273817
MAE train - 0.485734103408233364
MSE validation - 0.5655592449355096
MAE validation - 0.5588133416800689



200 iterations -

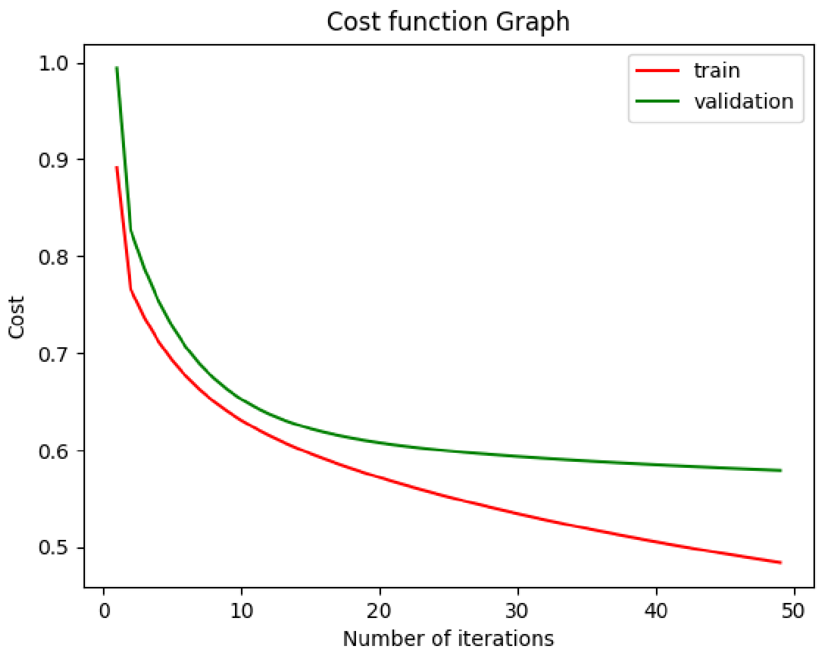
MSE Train- 0.328694914437971
MAE Train- 0.46503323964724724
MSE Validation - 0.5736557588637644
MAE Validation - 0.5629369557931773



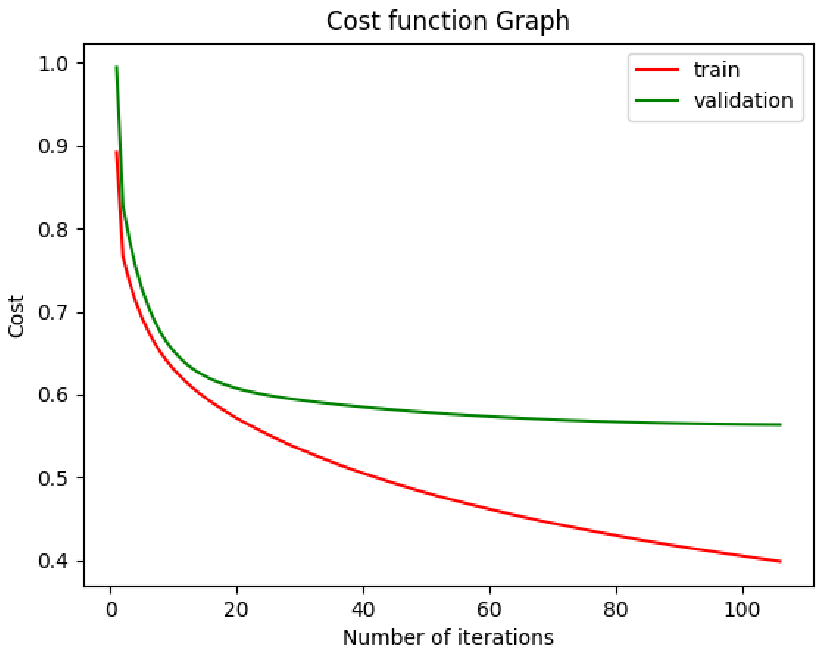
Increasing iterations leads to a decrease in the mse error of train data and validation data although at some point, further increase can lead to an increase in the mse of validation(as has happened in the case of 200 iterations).
So as we can see the model converges at about 150 iterations, moving to 200 iterations results in the validation mse to increase and hence 150 iterations is the most suitable for this data and learning rate.
Although, this stopping criteria is not quite accurate in terms of measuring the convergence of the model, and hence we'll make use of the relative tolerance.

Stopping Criteria - reltol

Again, the learning rate seems to be quite good for this stopping criteria as it converges pretty quickly. I tested for various thresholds,
At threshold = 0.001
MSE train - 0.4818391921991217
MAE train - 0.5514699636128566
MSE validation - 0.5786046694285445
MAE validation - 0.5791812581969676

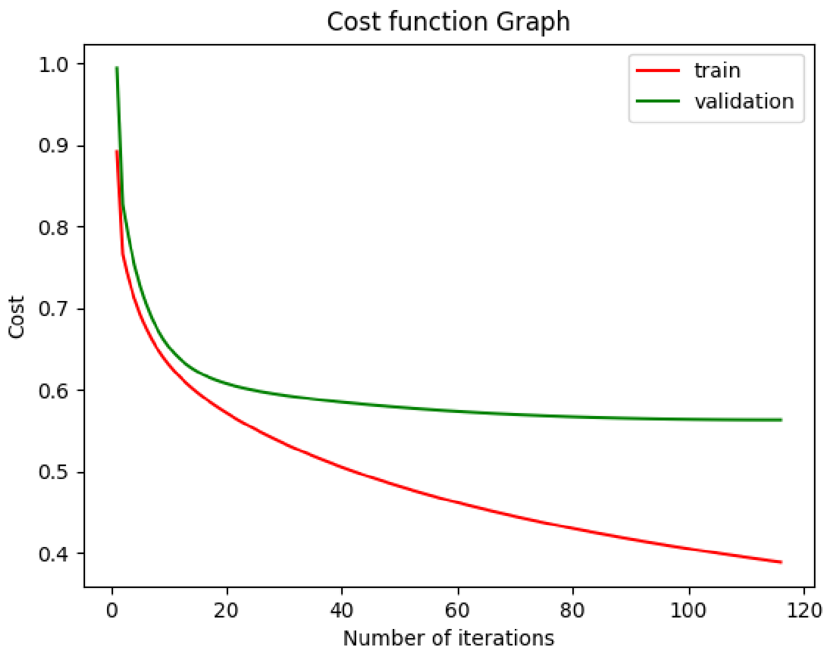


At threshold = 0.0001
MSE train - 0.3981291473680975
MAE train - 0.5059550200325378
MSE validation - 0.5636678772414668
MAE validation - 0.5648283061239373

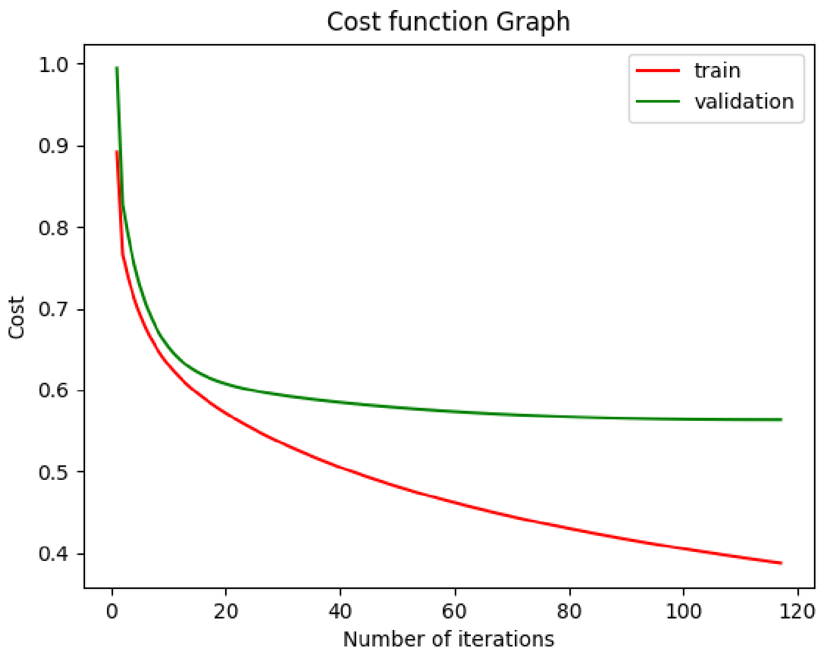


At threshold = 0.00001
MSE train - 0.3882155248869356
MAE train - 0.500919129092031

MSE validation - 0.5634606109343059
MAE validation - 0.5627266791099029



At threshold = 0.000001
MSE train - 0.3872714309887104
MAE train - 0.5004320924816416
MSE validation - 0.563465592871015
MAE validation - 0.5625294008571453



It takes about 120 iterations with threshold set as 0.00001 which comes out to

be the most optimal in terms of the MSE error. Further reducing the threshold does not help much.
Hence for me the threshold is 0.00001.

3.2

N = number of input samples, M = number of features in an input The value of cost function ,

$$y_{pred} = X * (\theta)$$

, this y_{pred} would be $144 * 1$ vector

$$J(\theta) = \frac{1}{N}\sum_{n=1}^{N}(y_{pred}[n] - y[n])^2 + \lambda * \sum_{n=1}^{M}\theta_n^2$$

Hence computing the gradient it comes out to be

$$\nabla_\theta(J(\theta)) = \frac{1}{N}(2 * (X^T(y_{pred} - y)) + (2 * \lambda * theta))$$

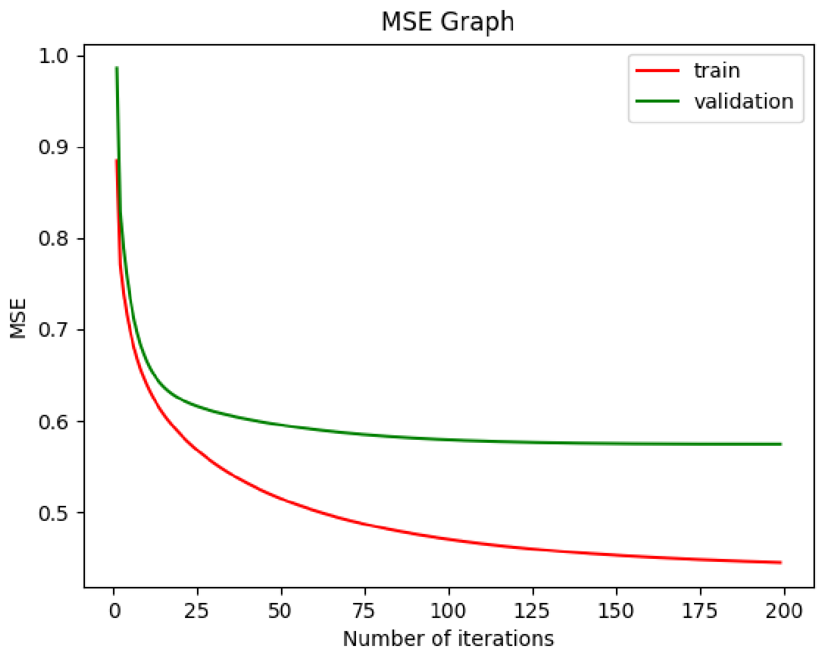
Hence the update function becomes,

$$\theta = \theta - \alpha * \nabla_\theta(J(\theta))$$

α is the learning rate
Now if we talk about the learning rate, again here I was getting really high cost values for learning rates 0.01 and 0.1. Further, it diverged pretty quickly at around 5 iterations or so. Hence, I have made use of the learning rate 0.001.

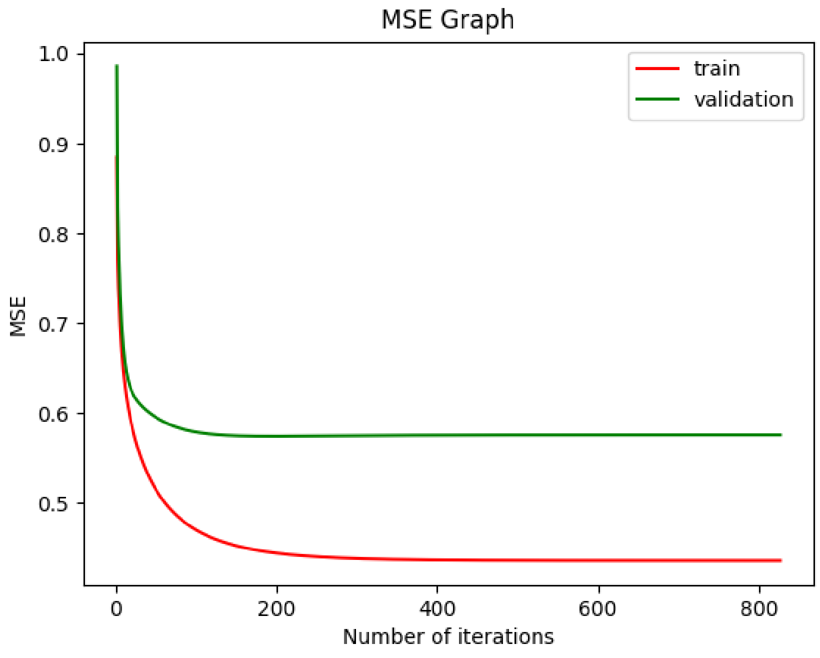
$\lambda = 5$

I tried out various iterations for lambda = 5, it came out to be best at 200 iterations.
Stopping Criteria - maxit
MSE train - 0.44456784681222317
MAE train - 0.5243649640440673
MSE validation - 0.5744746404368206
MAE validation - 0.572488495824359



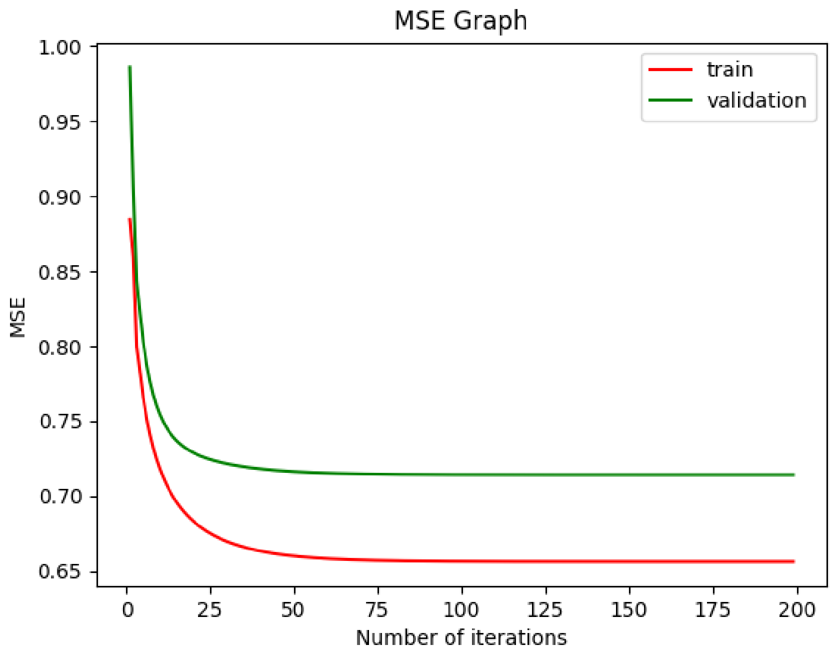
Stopping Criteria - reltol

For me, the best results came at reltol 0.000001
MSE train - 0.43626698165837624
MAE train - 0.5193632878611492
MSE validation - 0.5758967091799841
MAE validation - 0.5728512381409188

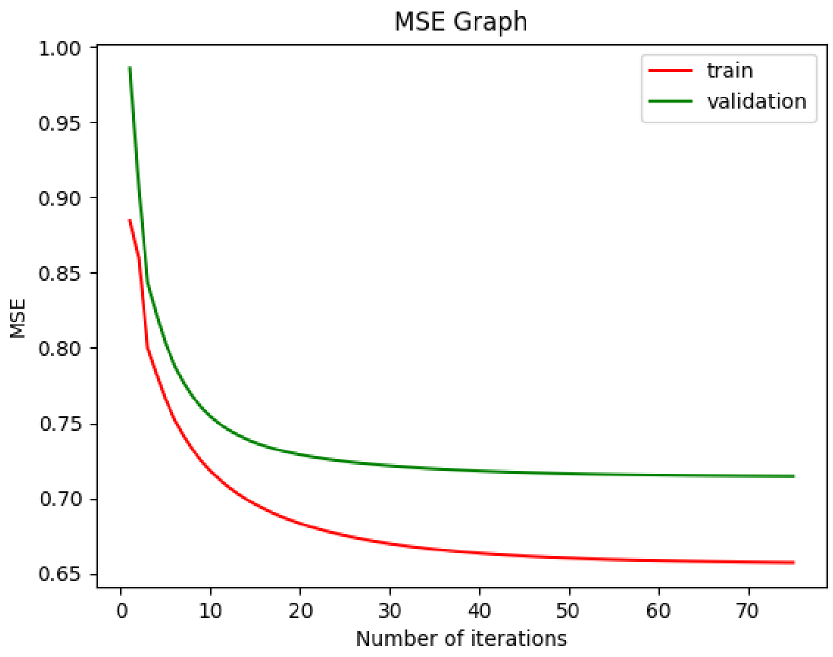


λ = 25

Again tried for various iterations.
maxit = 200
MSE train - 0.6563348729617011
MAE train - 0.6312580869889919
MSE validation - 0.714154381979633
MAE validation - 0.6597119828413315



threshold = 0.00001
MSE train - 0.6570993930199873
MAE train - 0.6316638327384285
MSE validation - 0.7145509378921098
MAE validation - 0.6598980521599307



Now, if we first compare the two lambda's we find that as we increase the lambda we are essentially avoiding the overfitting of the data and hence the validation MSE becomes stabilised at a higher value at lambda = 25 than at lambda = 5. Further it converges faster at lambda = 25.
Overall if we compare ridge regression with linear regression, I found out that the linear regression model gives a better mse error at validation data than the ridge one. This might be because we have very few samples as our data.
Also, another observation is the number of iterations required to converge, this is less in the case of linear regression but ridge regression with $\lambda = 25$ converges even faster.

3.3

Linear Regression
MSE validation(my model) - 0.5634606109343059
MAE validation(my model) - 0.5627266791099029
MSE validation - 1.0250630693359248
MAE validation- 0.8321564062320113
In this case, my linear regression's validation error comes out to be better than the scikit model's. Also the MAE also comes out to be less. Overall, my model fairs better than the scikit learn's model on the validation data.

Ridge Regression
Lambda = 5
MSE validation(my model) - 0.5758967091799841

MAE validation(my model) - 0.5728512381409188
MSE validation - 0.9319081640199341
MAE validation- 0.7803008143298183
Again, my validation MSE comes out to be lesser than validation MSE of scikit's model.
Lambda = 25
MSE validation(my model) - 0.7145509378921098
MAE validation(my model) - 0.6598980521599307
MSE validation- 0.771685564794975
MAE validation - 0.6815982994230616
Again, my validation MSE comes out to be lesser than validation MSE of scikit's model.

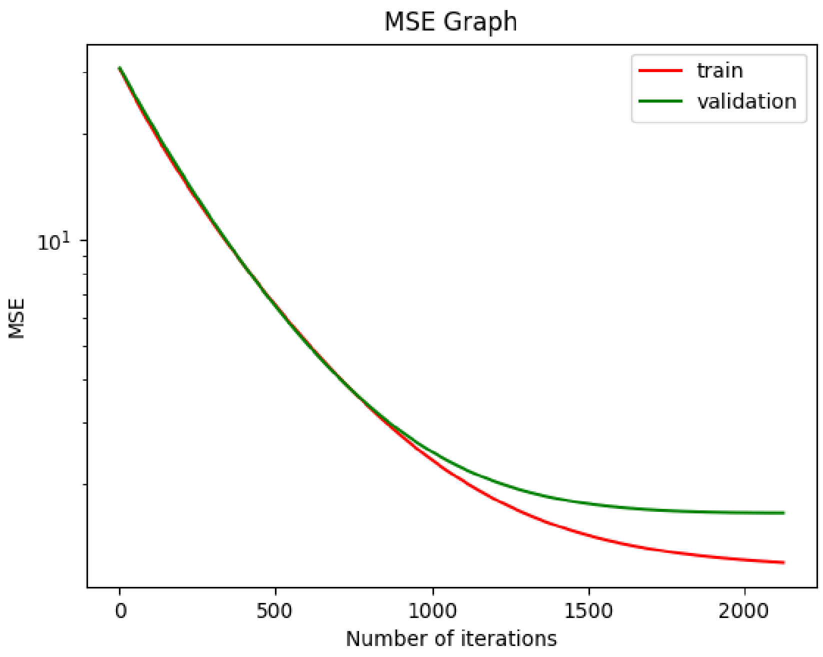
3.4

10 features can run with learning rate 0.1 but it cannot be compared with all features for the same learning rate as it overflows very quickly. learning rate - 0.001, threshold = 0.000001 for SelectKBest

SelectKBest

MSE train - 1.1966835658430395
MAE train - 0.9079524828361382
MSE validation - 1.6583852342715832
MAE validation - 1.0930917324866878

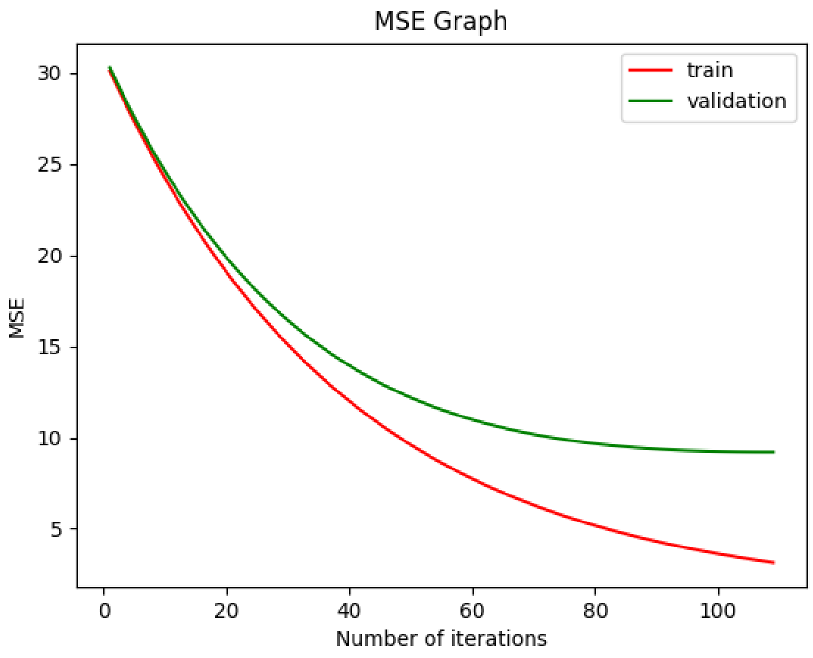
MSE train(all features) - 0.38821552488693567
MAE train(all features) - 0.500919129092031
MSE validation(all features) - 0.5634606109343059
MAE validation(all features) - 0.5627266791099029
If we see the above data, we find that with 10 features the validation error is higher. Although we might converge quickly because of less number of features in each input sample.



This plot is in log scale at y-axis for better visualisation.

SelectfromModel

learning rate - 0.001, threshold = 0.0001 MSE train - 3.1491694734887883
MAE train - 1.5601052365283994
MSE validation - 9.192144821709354
MAE validation - 2.3748924088864762



We can clearly see that the SelectKbest fairs better than the select from model case by a large extent. Hence in the visualisation part, I'll use select k best.

3.5

$$h_{\theta_r}(x_n) = \frac{e^{\theta_r^T x_n}}{1 + \sum_{p=1}^{8} e^{\theta_p^T x_n}}$$

Now say we want to find the gradients for some θ_r, we can assume other thetas as some constant with respect to θ_r. Hence,

$$h_{\theta_r}(x_n) = \frac{e^{\theta_r^T x_n}}{k + e^{\theta_r^T x_n}}$$

here k = 1 + sum of other exponents

$$J(\theta_r) = -\sum_{n=1}^{N} y_n \log(h_{\theta_r}(x_n)) + (1 - y_n) \log(1 - h_{\theta_r}(x_n))$$

On differentiating with respect to θ_r, we get the following

$$\nabla(J(\theta_r))_{\theta_r} = -\sum_{n=1}^{N} (\frac{y_n}{h_{\theta_r}(x_n)} - \frac{1 - y_n}{1 - h_{\theta_r}(x_n)}) * (\nabla_{\theta_r} h_{\theta_r}(x_n))$$

And if we solve the gradient, we get

$$\nabla_{\theta_r} h_{\theta_r}(x_n) = x_n * h_{\theta_r}(x_n) * (1 - h_{\theta_r}(x_n))$$

$$\nabla(J(\theta_r))_{\theta_r} = -\sum_{n=1}^{N} (y_n - h_{\theta_r}(x_n)) * x_n$$

In matrix form it can be written as ,

$$\nabla(J(\theta_r))_{\theta_r} = X^T(h_{\theta_r}(x_n)) - y))$$

So, we can update the θ_r by the following function

$$\theta_r = \theta_r - \alpha * \nabla(J(\theta_r))_{\theta_r}$$

MSE train - 0.027777777777777776
MAE train - 0.013888888888888888
MSE validation - 0.8095238095238095
MAE validation - 0.6190476190476191

Following are the predictions on validation data :
$[2, 5, 6, 3, 3, 4, 7, 6, 4, 5, 5, 5, 5, 6, 6, 6, 5, 5, 7, 6, 7]$

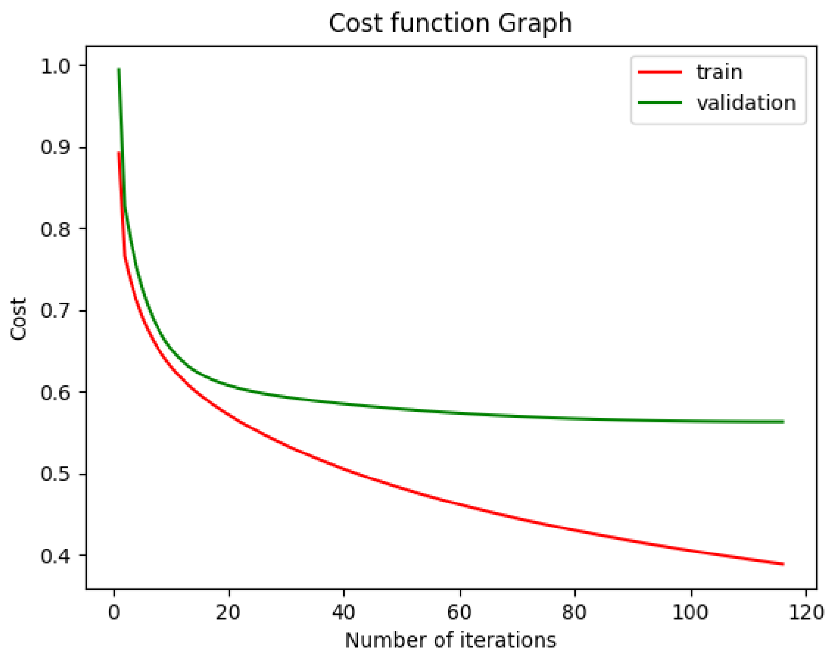
3.6

Plots of MSE for sections 3.1, 3.2, 3.4, 3.5

Here I have plotted the graphs for the best implementation in each section given.

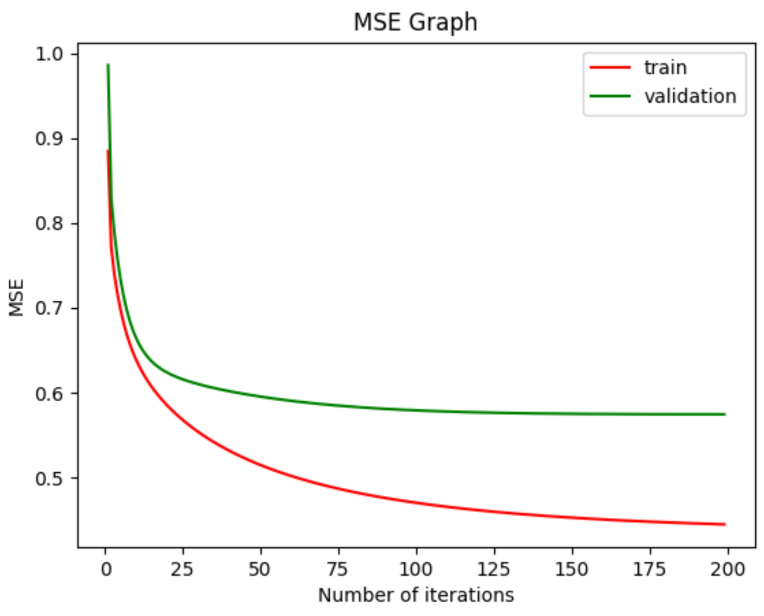
3.1

learning rate = 0.001, threshold/reltol = 0.00001



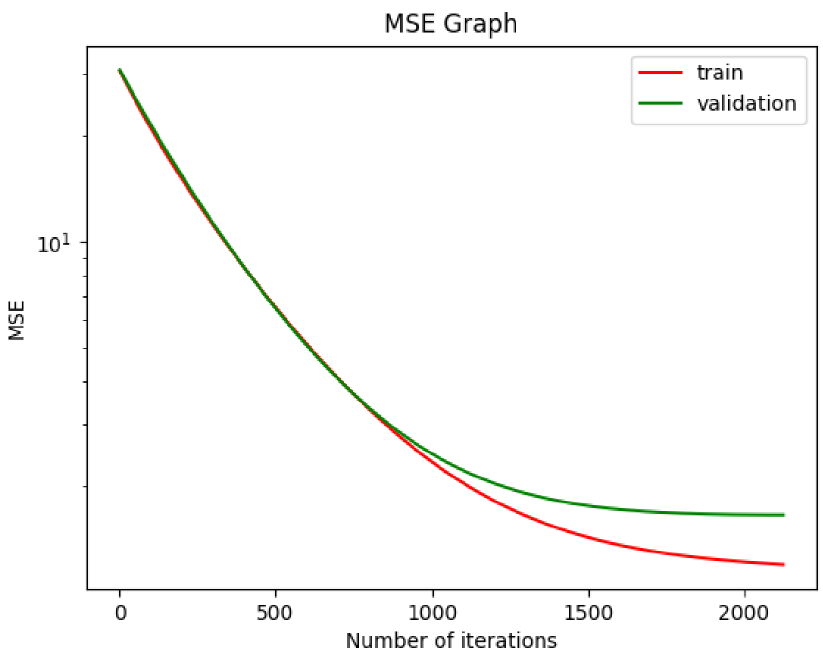
3.2

learning rate = 0.001, max iterations = 200



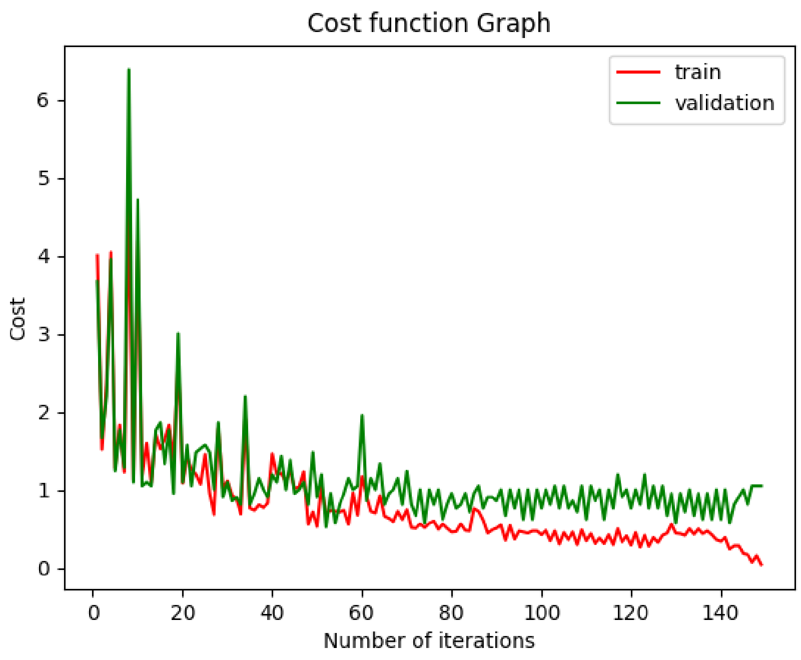
3.4

SelectKBest



Plot is on log-scale on y-axis for better visualisation

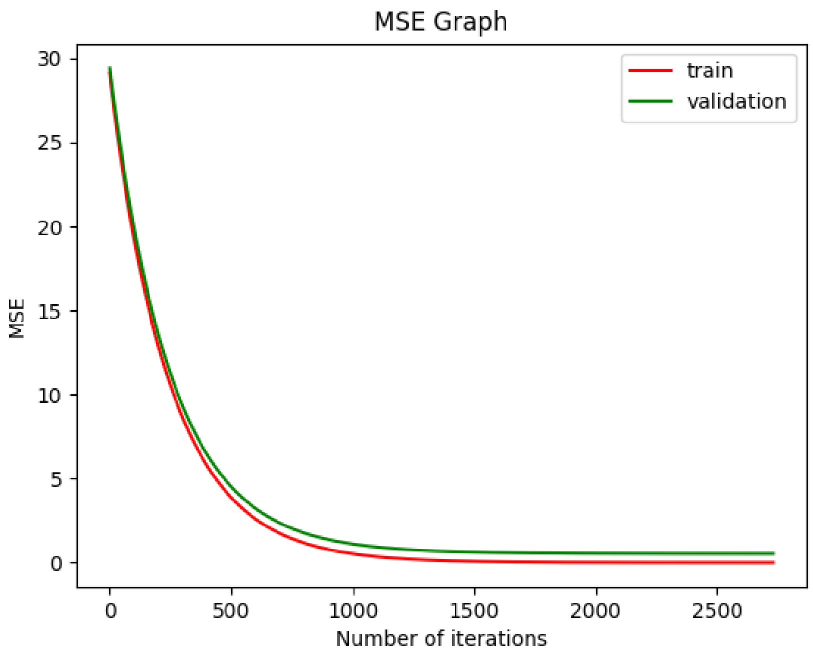
3.5



We can clearly see the usage of validation data in this. Validation data helps in preventing overfitting by stopping early rather than having fixed iterations.

Normalising the Data

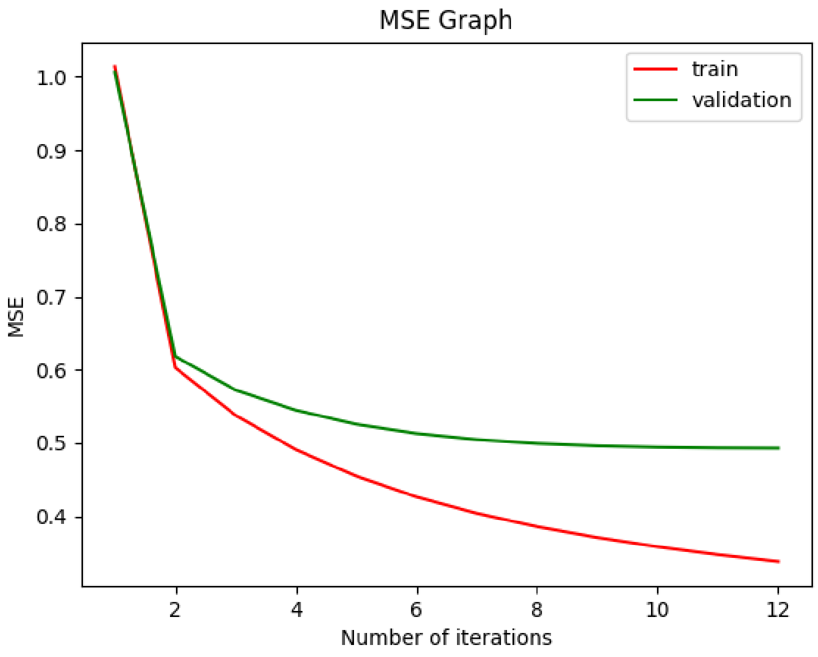
Learning Rate = 0.001, threshold/reltol = 0.00001
MSE train - 0.0005141779248934496
MAE train - 0.022530302884798 61
MSE validation - 0.5305893724247385
MAE validation - 0.5628438864092823



Variation with Amount of data used for training

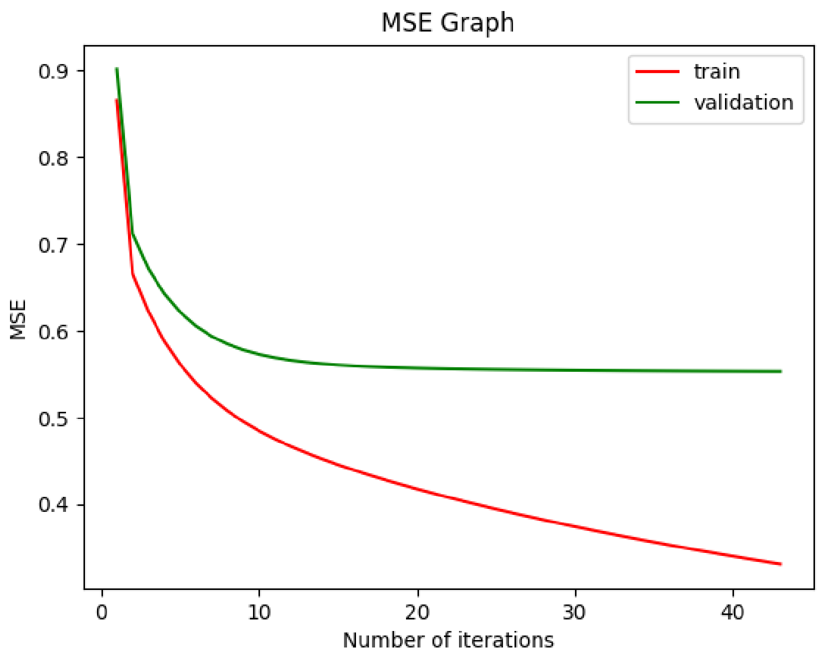
Using 1/4th data

Learning Rate = 0.001, reltol = 0.001
MSE train - 0.32983812288815284
MAE train - 0.4304959898762694
MSE validation - 0.4928804984929856
MAE validation - 0.5582287832590459



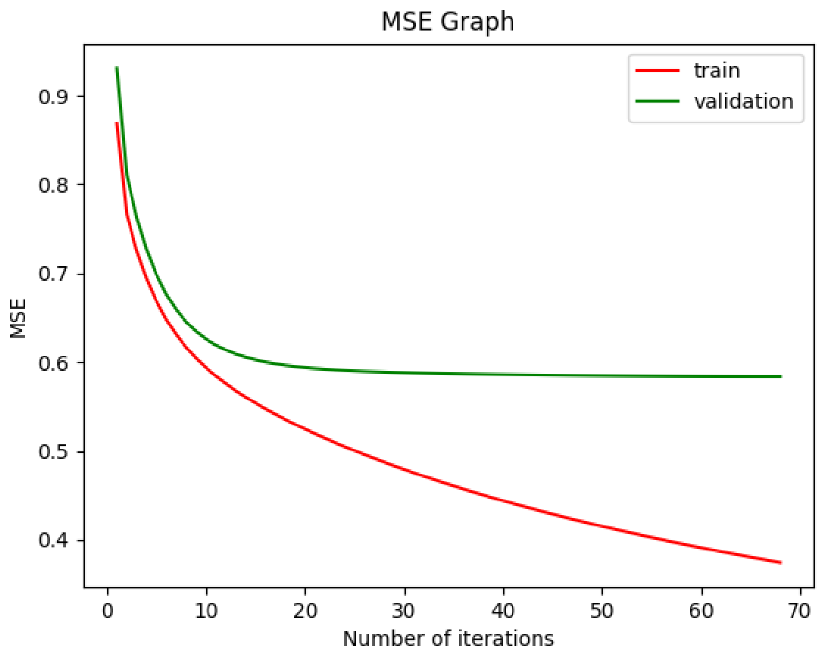
Using 1/2th data

Learning Rate = 0.001, reltol = 0.0001
MSE train - 0.3285381465482822
MAE train - 0.46390388638727664
MSE validation - 0.5530831924951342
MAE validation - 0.5749253035407624



Using 3/4th data

Learning Rate = 0.001, reltol = 0.00001
MSE train - 0.3724844010110938
MAE train - 0.4811839998428107
MSE validation - 0.5837330866077267
MAE validation - 0.5803509284507753



Using full data

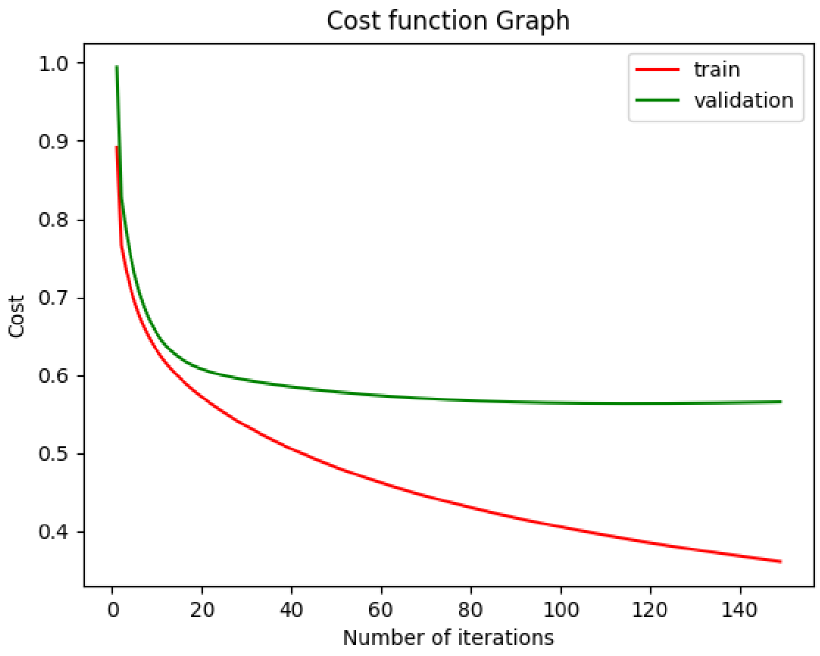
Learning Rate = 0.001, max iterations = 150

MSE train - 0.3606670874273817

MAE train - 0.48573410340823364

MSE validation - 0.5655592449355096

MAE validation - 0.5588133416800689



Breaking the input into two halves

Linear Regression

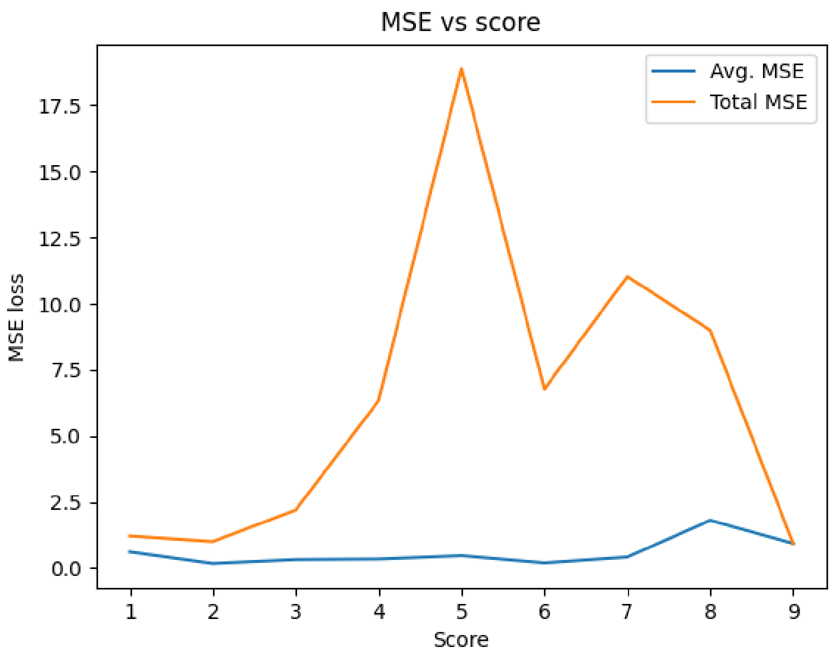
Learning Rate = 0.001, threshold = 0.0001
Mean Absolute Difference when predicting on Training Data = 0.4209801643706056
Mean Absolute Difference when predicting on Validation Data = 0.29167065677347725

Ridge Regression

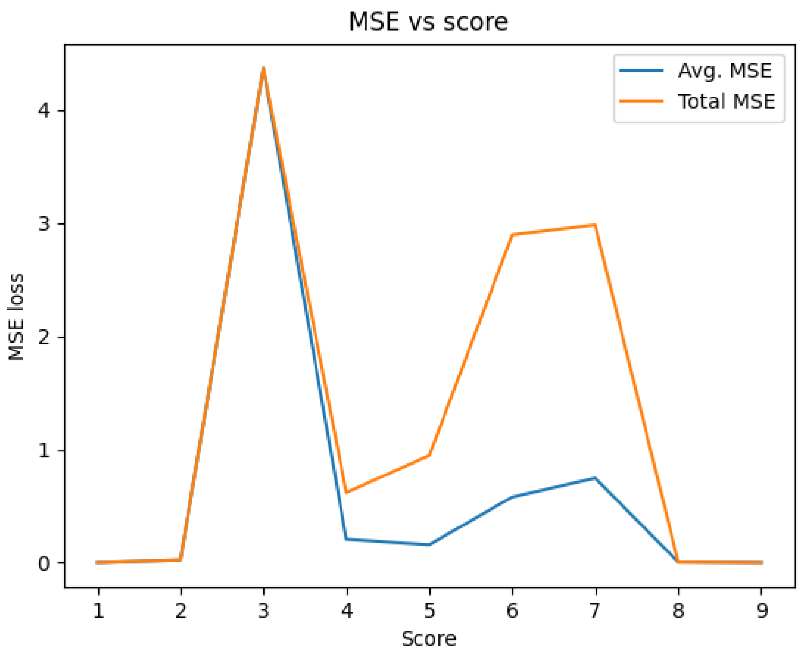
λ = 5
Learning Rate = 0.001, threshold = 0.0001
Mean Absolute Difference when predicting on Training Data = 0.2797362851344063
Mean Absolute Difference when predicting on Validation Data = 0.24841910557155794
λ = 25
Learning Rate = 0.001, threshold = 0.0001
Mean Absolute Difference when predicting on Training Data = 0.2296108550769005
Mean Absolute Difference when predicting on Validation Data = 0.266575384179481

MSE Loss vs Score

For Training Data

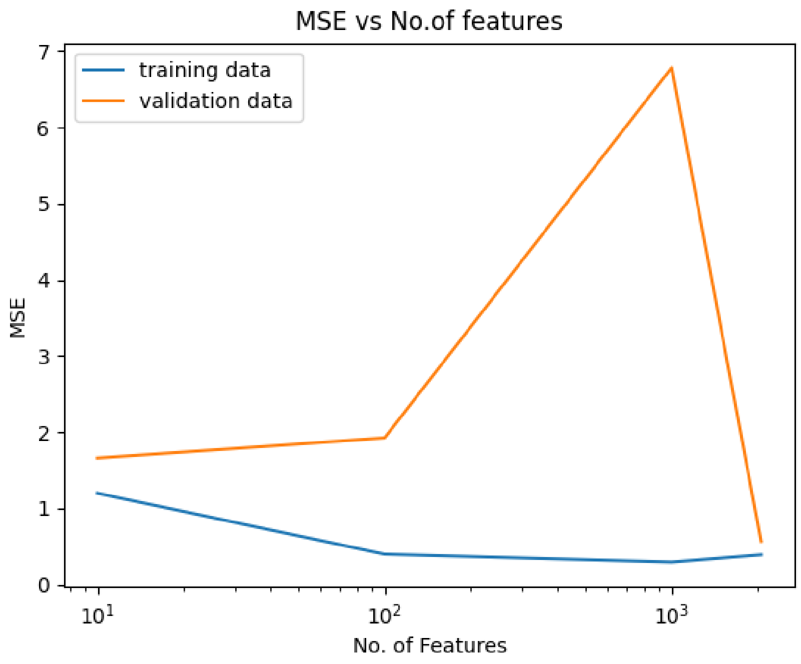


For Validation Data

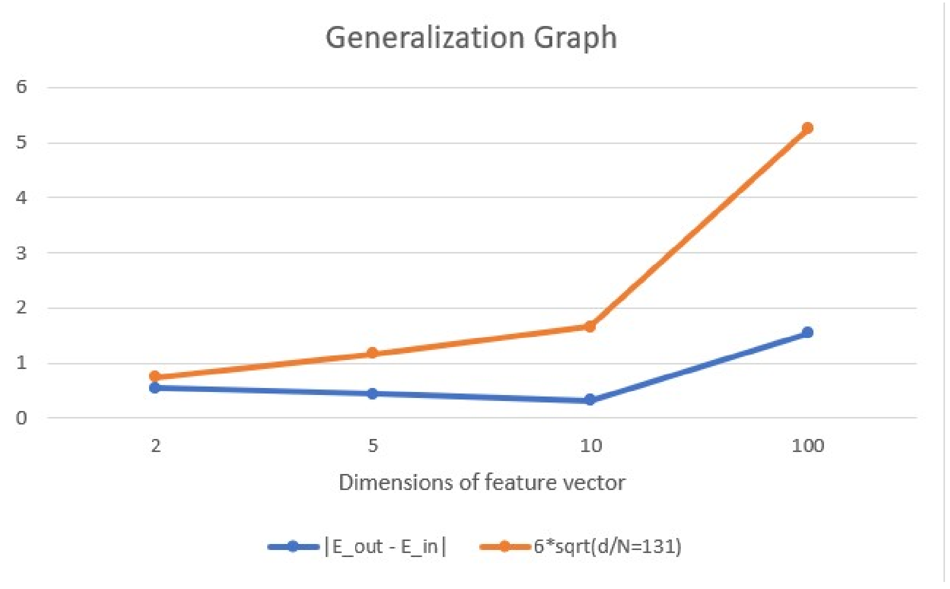


Feature Selection

I will be using SelectKBest for this part as it was the one which gave better results above. The graph has log scale on the X-axis for better clarity



3.7



On computing $E_{out} - E_{in}$ on the various datasets following graph is obtained. I have plotted this graph along with the function $6 * \sqrt{d/N}$ depicted in orange. This is to show that the asymptotic bound given holds as the function $6 * \sqrt{d/N}$ is always greater than $E_{out} - E_{in}$.