# COL341
# Assignment 1 - Report
# Eshan Jain
# 2020CS50424

**3.1 Basic Implementation**

i) Learning rate = 0.1

For the learning rate of 0.1, the gradient descent algorithm does not converge with increasing iterations. For the given data, this is too large a learning rate to effectively take small enough steps in the direction of the steepest descent in the cost function. Therefore for both the training and validation data, the MSE and MAE scores increase with the number of iterations of the gradient descent algorithm. Changing the relative tolerance parameter does not really make sense for this, as the difference in increasing with the number of iterations.

After just 10 iterations of the gradient descent algorithm, the values of the MSE and MAE scores for the training and validation data are:

```
Linear Regression Training MAE:  1.4289700304930604e+16
Linear Regression Training MSE:  2.1069183181566595e+32
Linear Regression Validation MAE:  7.520094794824873e+17
Linear Regression Validation MSE:  5.764661649009649e+35
```

And for 200 iterations, the MAE and MSE scores for both training and validation data overflow the float limit and are undefined.

ii) Learning Rate = 0.01

Again for the learning rate of 0.01, the gradient descent algorithm does not converge. For the given data, this is still too large a learning rate to effectively take small enough steps in the direction of the steepest descent in the cost function. Therefore for both the training and validation data, the MSE and MAE scores increase with the number of iterations of the gradient descent algorithm. But this time, the values are not as large as when the learning rate was 0.1.

After 10 iterations, we have the following:

```
Linear Regression Training MAE:  2557949.5798920966
Linear Regression Training MSE:  6751269111006.135
Linear Regression Validation MAE:  11119304.065800318
Linear Regression Validation MSE:  126032457599460.77
```

After 200 iterations, overflow has not occurred yet, and the values are:

```
Linear Regression Training MAE:  1.744605461532271e+126
Linear Regression Training MSE:  3.1404791427653817e+252
Linear Regression Validation MAE:  7.583729801570534e+126
Linear Regression Validation MSE:  5.86263550915762e+253
```

However, 500 iterations do cause an overflow in both the MAE and MSE scores.

iii) Learning Rate =  0.001

This is a good enough learning rate for the given data, and the gradient descent algorithm converges in this case.

In comparison to the above two learning rates, we have:

After 10 iterations, we have the following:

```
Linear Regression Training MAE:  0.6598852183034914
Linear Regression Training MSE:  0.7043224286023175
Linear Regression Validation MAE:  0.6889660970549375
Linear Regression Validation MSE:  0.7314702265878003
```

After 200 iterations, we have the following:

```
Linear Regression Training MAE:  0.5101726130974069
Linear Regression Training MSE:  0.40633483214109956
Linear Regression Validation MAE:  0.5664843867152056
Linear Regression Validation MSE:  0.5641457987102589
```

After setting the max iterations to 1000 and the relative tolerance to 0.001, we have the following scores:

```
Linear Regression Training MAE:  0.4831120120021878
Linear Regression Training MSE:  0.35640642214944795
Linear Regression Validation MAE:  0.5589828050663396
Linear Regression Validation MSE:  0.5663776806392639
```

Now keeping the max iterations constant and changing the relative tolerance to 0.0001, we have:

```
Linear Regression Training MAE:  0.3761420631427545
Linear Regression Training MSE:  0.22294502987972967
Linear Regression Validation MAE:  0.6234446184856276
Linear Regression Validation MSE:  0.6449158474671899
```

We observe that the model is now going towards overfitting as on the training set, we achieve a better performance, but it does not generalize well, and we see a worse performance on the validation set.

Now keeping the max iterations constant and changing the relative tolerance to 0.01, we have:

```
Linear Regression Training MAE:  0.6333912305340261
Linear Regression Training MSE:  0.6577928864063601
Linear Regression Validation MAE:  0.6559258831892438
Linear Regression Validation MSE:  0.6773315133825653
```

Here we observe underfitting, as the model gives worse performance for both the training and validation set since the threshold is quite weak.

**3.2 Ridge Regression**

We have the cost function for ridge regression as:-

$$E_{in}(w) = \frac{1}{M}(\sum_{i=1}^{M}(y_i - \sum_{j=1}^{N}\theta_j x_{ij})^2 + \lambda \sum_{i=1}^{N} w_j^2)$$

Now since the first term is same as that in linear regression, only the second term would cause a change in the parameter update equation.

The gradient of the second term would be :-

$$\frac{1}{M}\lambda\theta$$

Therefore the weight update equation for ridge regression would be:-

$$w = w - \alpha(\frac{1}{M}[X^T(X\theta - y) + \lambda\theta])$$

Where the first term comes from the linear regression part and the second term from the regularisation part.

i) Lambda = 5

Again for the learning rates of 0.1 and 0.01, the gradient descent is diverging and, therefore, the MAE and MSE scores are undefined.

However, in the case when the learning rate is 0.001, we have the following:

After 10 iterations;

```
Ridge Regression Training MAE:  0.6598957872458188
Ridge Regression Training MSE:  0.7043420244013787
Ridge Regression Validation MAE:  0.6889752577115832
Ridge Regression Validation MSE:  0.7314937923262808
```

After 200 iterations:

```
Ridge Regression Training MAE:  0.5103790558737039
Ridge Regression Training MSE:  0.40680298019138417
Ridge Regression Validation MAE:  0.5665709367679435
Ridge Regression Validation MSE:  0.5641384220562985
```

After 1000 iterations and the relative tolerance of 0.001 :

```
Ridge Regression Training MAE:  0.48434238917139893
Ridge Regression Training MSE:  0.3585660265747396
Ridge Regression Validation MAE:  0.5588074665117294
Ridge Regression Validation MSE:  0.5659436777520362
```

Therefore we observe slightly better performance in ridge regression as compared to the linear regression, although a significant change is not observed due to the small value of teh regularisation parameter lambda.

ii) Lambda = 25

Now again, for the values of learning rate 0.1 and 0.01, we don't see the gradient descent converging.

In the case when the learning rate is 0.001, we observe the following:

After 10 iterations:-

```
Ridge Regression Training MAE:  0.6599380463195423
Ridge Regression Training MSE:  0.7044229194245482
Ridge Regression Validation MAE:  0.6890118808769022
Ridge Regression Validation MSE:  0.7315905471335976
```

After 200 iterations

```
Ridge Regression Training MAE:  0.5111945681132539
Ridge Regression Training MSE:  0.40867517139613574
Ridge Regression Validation MAE:  0.566913262021991
Ridge Regression Validation MSE:  0.5641320240341328
```

After 1000 iterations:

```
Ridge Regression Training MAE:  0.48871660015167134
Ridge Regression Training MSE:  0.366471793778875
Ridge Regression Validation MAE:  0.5586344377318001
Ridge Regression Validation MSE:  0.5645936705730505
```

Therefore we observe a much better performance with lambda = 25, as compared to lambda = 5. This is because as we penalize the model complexity more, it prevents overfitting, and although we achieve a slightly worse performance on the training set, we are able to generalize better and so achieve a greater performance on the validation set.

## 3.3 Using Scikit-Learn Library

The performance of sklearn models on the validation dataset are:

```
Scikitlearn Linear Regression MAE:  0.8322387027130557
Scikitlearn Linear Regression MSE:  1.025165437413566
Scikitlearn Ridge Regression MAE:  0.7803008143298173
Scikitlearn Ridge Regression MSE:  0.9319081640199326
```

Which is quite poor compared to our model’s performance.

## 3.4 Feature Selection

i) SelectKBest

Trained the linear regression model by selecting the best 10 features using the SelectKBest, and here are the results:-

For learning rate = 0.1:

After 10 iterations:

```
Linear Regression Training MAE:  2.3669448302135216
Linear Regression Training MSE:  7.020847634794508
```

After 1000 iterations:

```
Linear Regression Training MAE:  0.8875548211429107
Linear Regression Training MSE:  1.1421340055410378
```

A significant difference is that now the model converges even for learning rate = 0.1. This is because now, for the reduced number of features, this value of the learning rate is not too large, and in each iteration, the gradient descent algorithm can take small enough steps to ensure that we are moving in the right direction and toward convergence. So these MSE and MAE scores are much better than the undefined values in the case of the linear regression model trained on all features.

For the learning rate = 0.01:

After 10 iterations:

```
Linear Regression Training MAE:  4.819840559488586
Linear Regression Training MSE:  25.765714158705947
```

After 1000 iterations:

```
Linear Regression Training MAE:  0.9173913833914438
Linear Regression Training MSE:  1.2350872636970207
```

For the learning rate = 0.001:

After 10 iterations:

Linear Regression Training MAE:  5.290724008794868
Linear Regression Training MSE:  30.25872745594136

After 1000 iterations:

Linear Regression Training MAE:  2.266571104658656
Linear Regression Training MSE:  6.500151949323961

Therefore we observe that with the reduced number of features, the learning rate of 0.1 produces the best results, followed by 0.01 and 0.001, which is in stark contrast to the model trained on all features, which followed the opposite order.

ii) SelectFromModel

Trained the ridge regression model by selecting the best 10 features using the SelectFromModel, and here are the results:-

For lambda = 5:

For learning rate = 0.1,

After 10 iterations;

```
Ridge Regression Training MAE:  0.774502673098934
Ridge Regression Training MSE:  1.0795864757287257
```

After 1000 iterations:

```
Ridge Regression Training MAE:  0.6204450830121448
Ridge Regression Training MSE:  0.6263751375381954
```

For learning rate = 0.001,

After 10 iterations:

```
Ridge Regression Training MAE:  5.065274028807056
Ridge Regression Training MSE:  27.639600534427284
```

After 1000 iterations:

```
Ridge Regression Training MAE:  0.8666754758268737
Ridge Regression Training MSE:  1.2963249086288018
```

For lambda = 25:

For learning rate = 0.1,

After 10 iterations;

Ridge Regression Training MAE:  0.7931300211820306
Ridge Regression Training MSE:  1.115196739057996

After 1000 iterations:

Ridge Regression Training MAE:  0.6722545032500185
Ridge Regression Training MSE:  0.7692863063211384

For learning rate = 0.001,

After 10 iterations:

Ridge Regression Training MAE:  5.0654299434679775
Ridge Regression Training MSE:  27.641308799926264

After 1000 iterations:

Ridge Regression Training MAE:  0.8960875825019998
Ridge Regression Training MSE:  1.3458784844253513

Similar to what we observed in the linear regression model, with a reduced number of features, we get better results for the learning rate of 0.1 than 0.001. Also, as expected, we see better results with lambda = 25 than with lambda = 5, as was the case in the ridge regression model trained on all the features.

**3.5 Classification**

The cost function for each class is:

$$J(\theta) = \frac{-1}{M} \sum_{i=1}^{M} (y_i log(h_\theta(x_i)) + (1 - y_i) log(1 - h_\theta(x_i)))$$

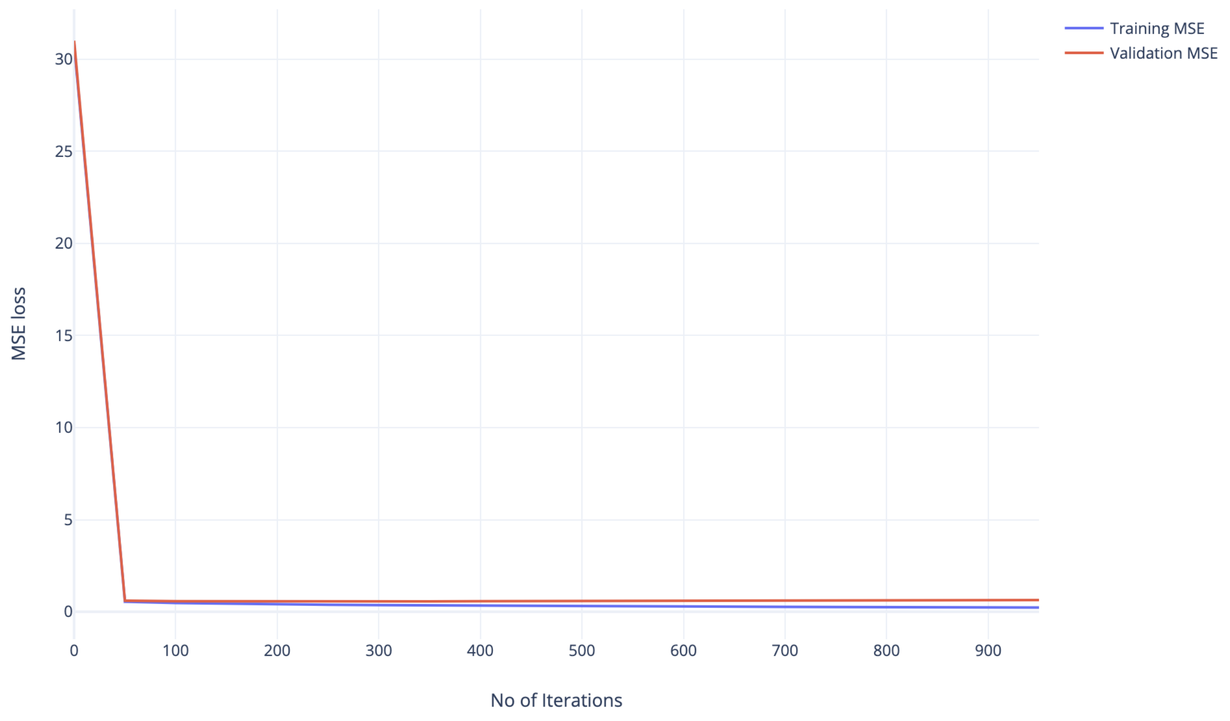
For each class, we create a binary output from the given output, which reflects whether the output is that class or not for all data celements. We update the weight vector based on this modified output and repeat this for each class separately. Therefore the weight update equation is:-

$$\theta_j = \theta_j - \alpha(\frac{1}{M} \sum_{i=1}^{M} ((h_\theta(x_i) - y_i) x_{ji}))$$

## 3.6 Visualization

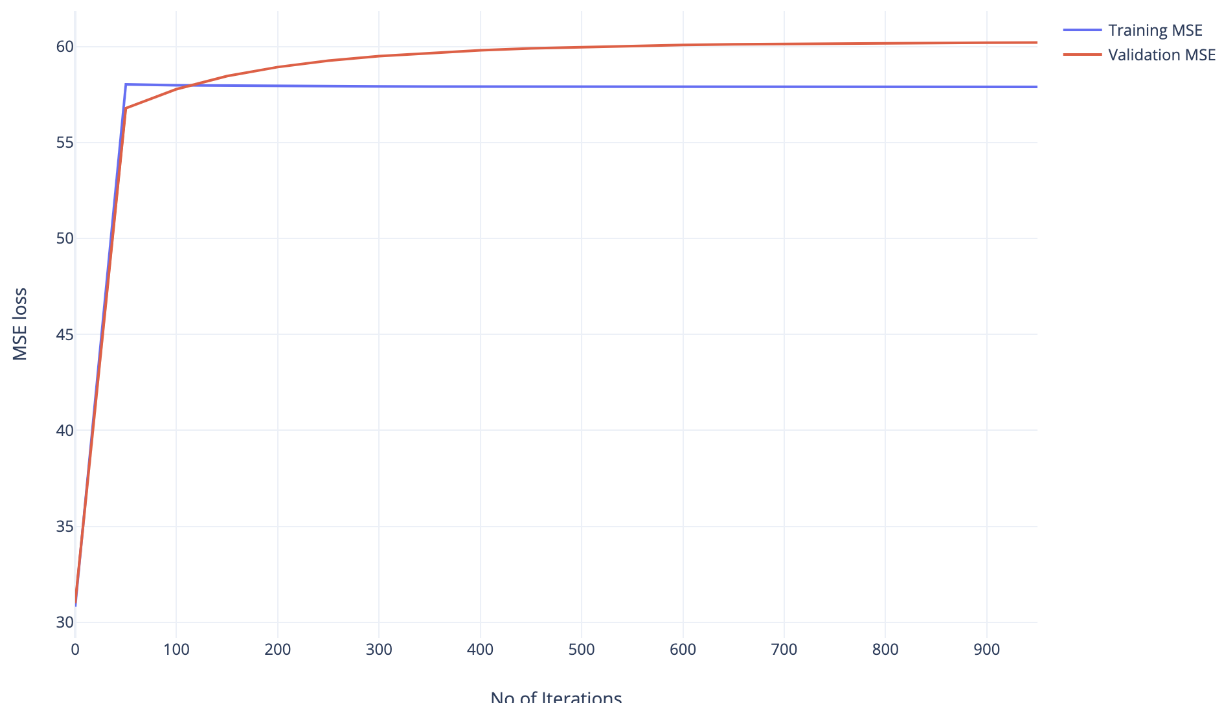
3.1 Without normalising, only the learning rate of 0.001 converges.

For learning rate = 0.001:

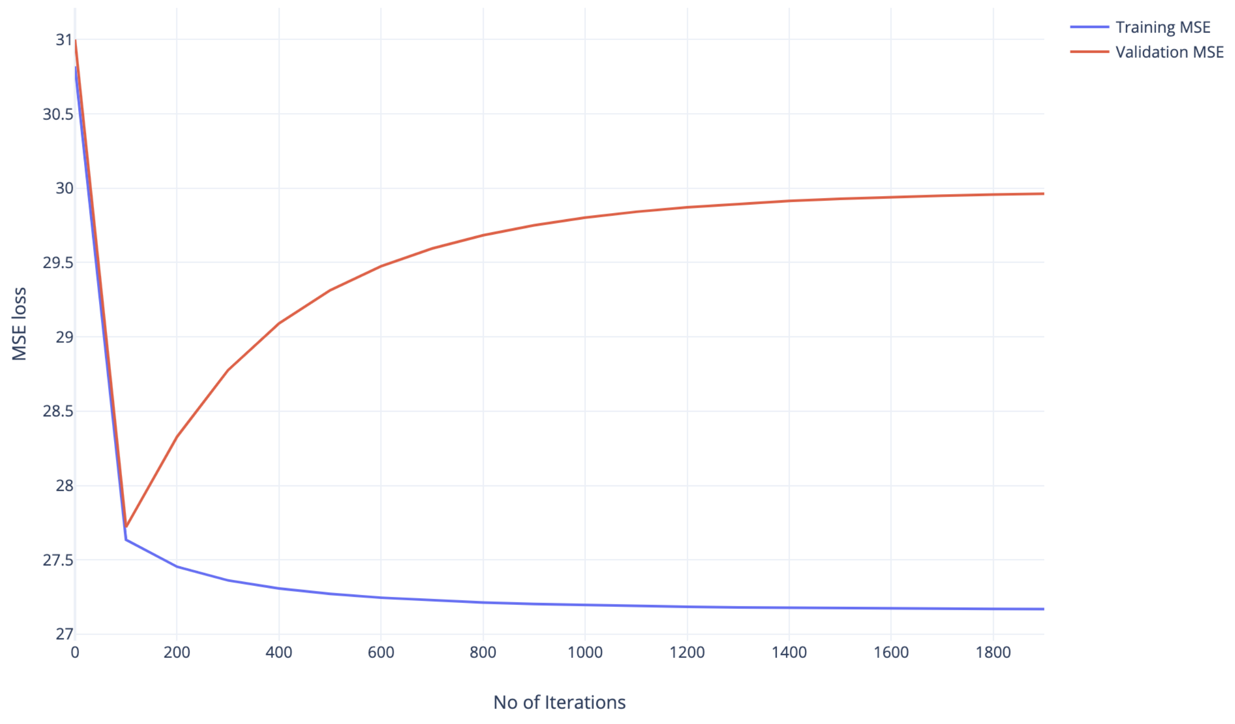


## 3.1 Using Normalised Gradient

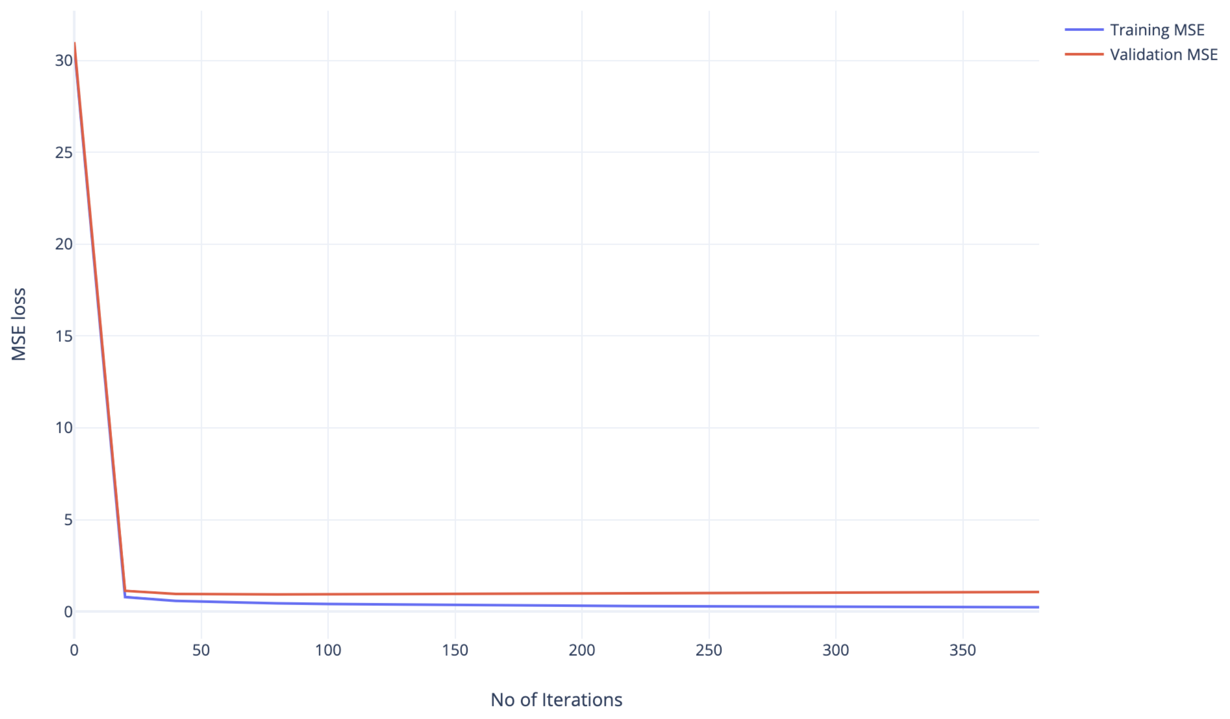
For learning rate = 0.1:



For learning rate = 0.01:



For learning rate = 0.001:

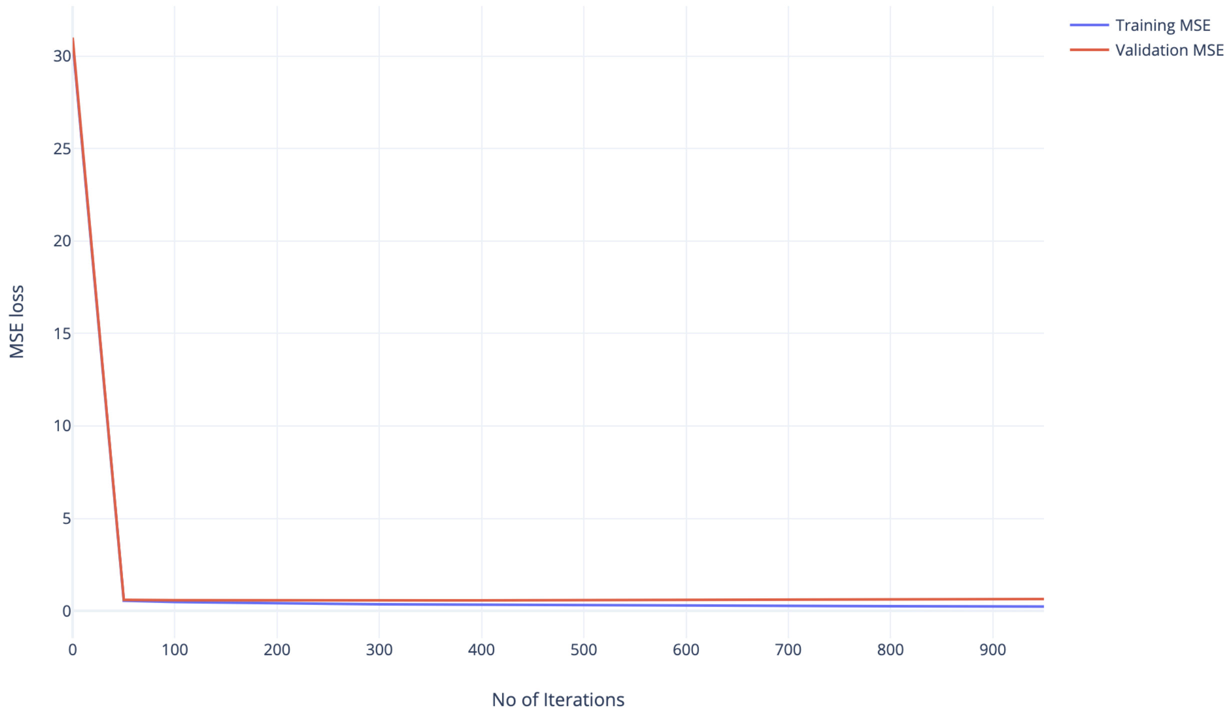


## 3.2

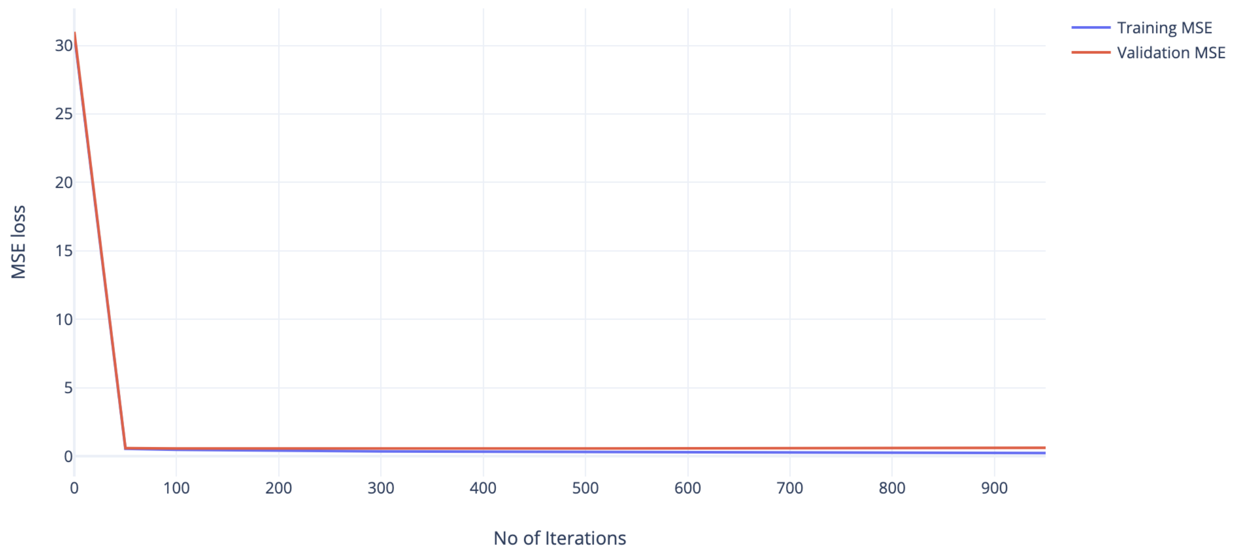
As mentioned, the gradient descent converges only when the learning rate = 0.001.

So for the learning rate = 0.001:

Lambda = 5



Lamda = 25

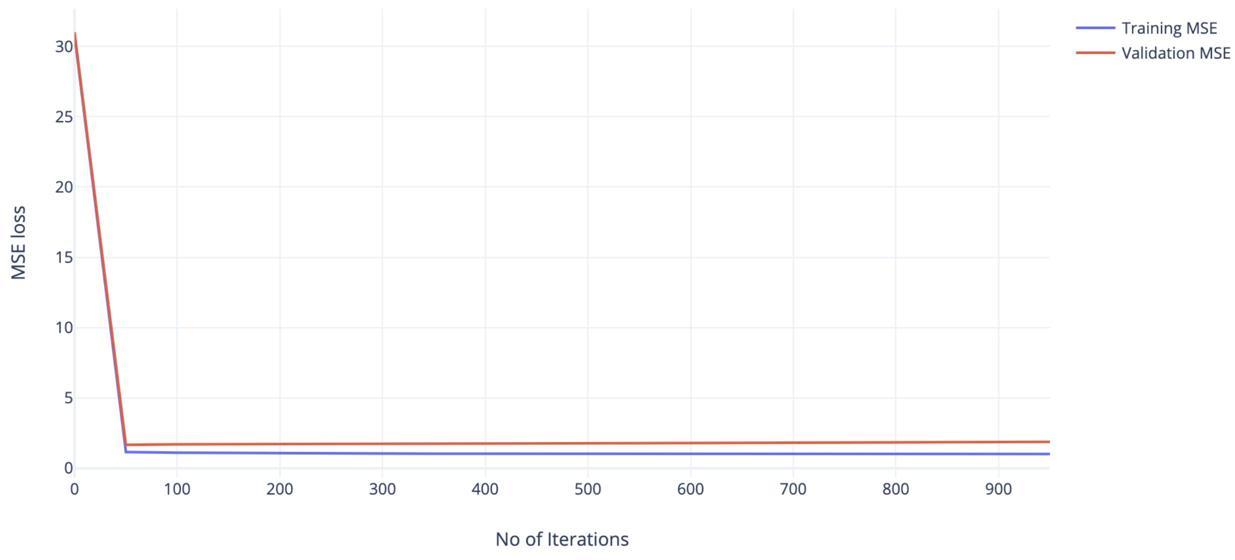


As expected, we observe better results with lambda = 25, since a higher penalty for the weights prevents overfitting, as can be seen from the plots (better performance on the validation set for lamda = 25).

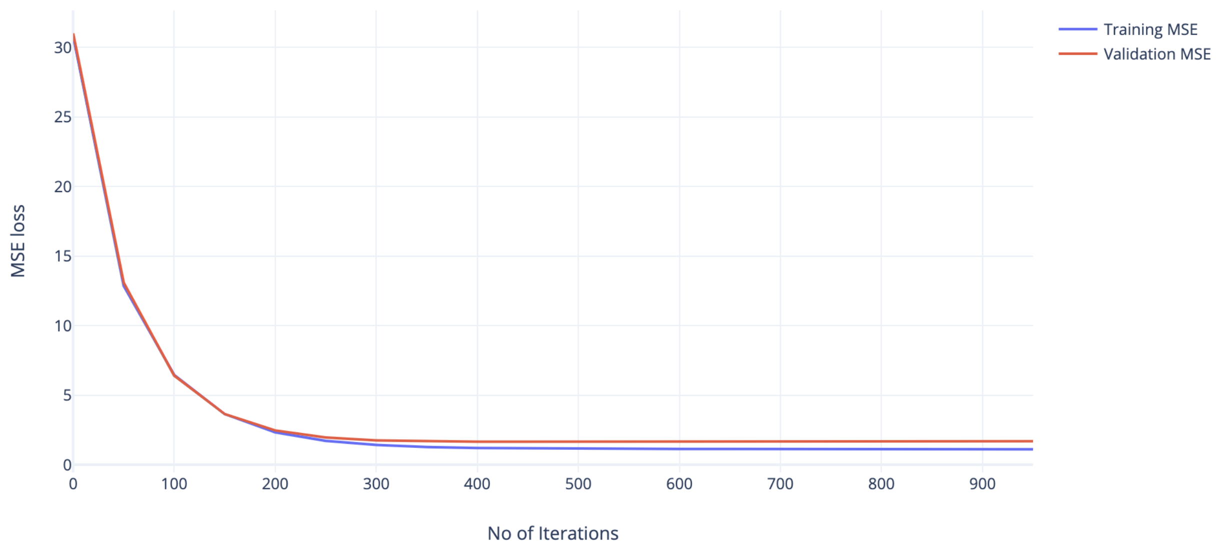
3.4

Linear Regression with top 10 features:

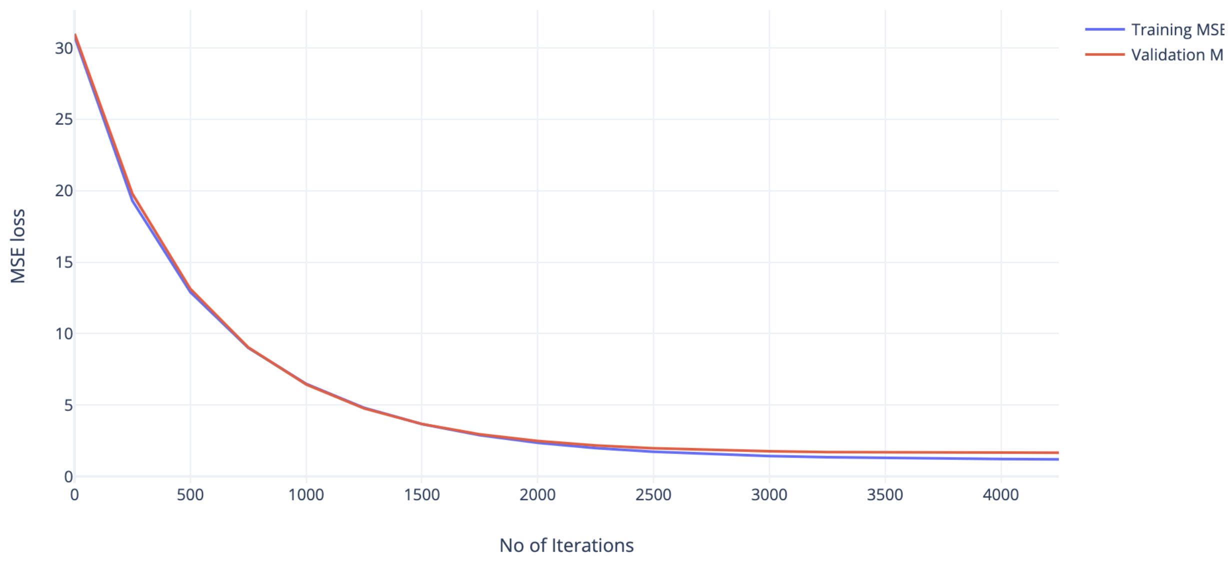
Learning Rate = 0.1



Learning Rate = 0.01



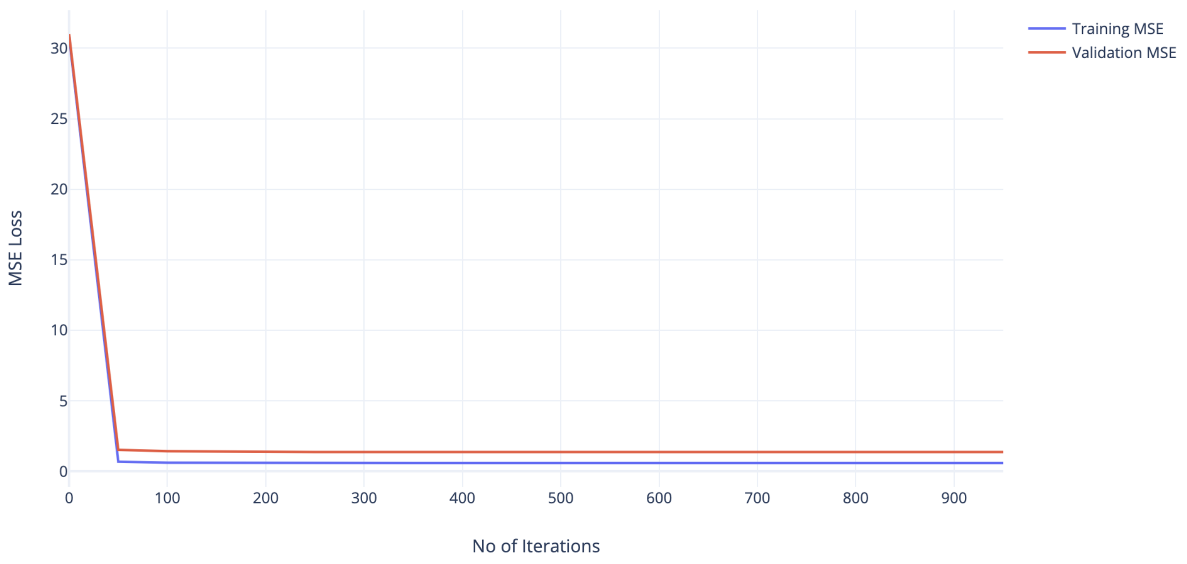
## Learning Rate = 0.001



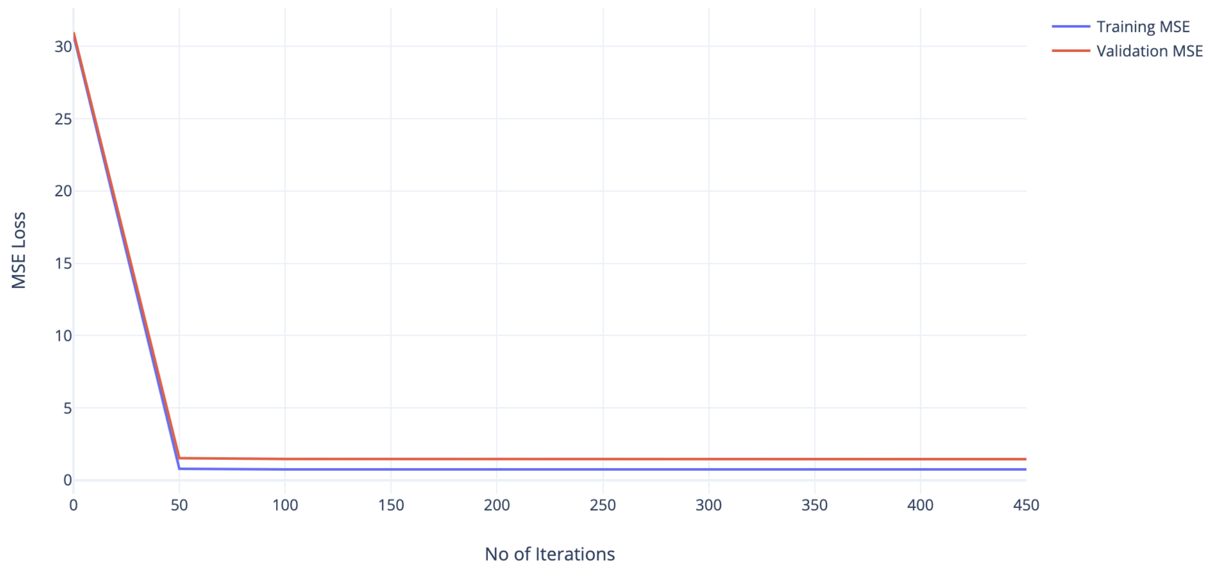
Ridge Regression using top 10 features:

Learning Rate = 0.1

Lambda = 5

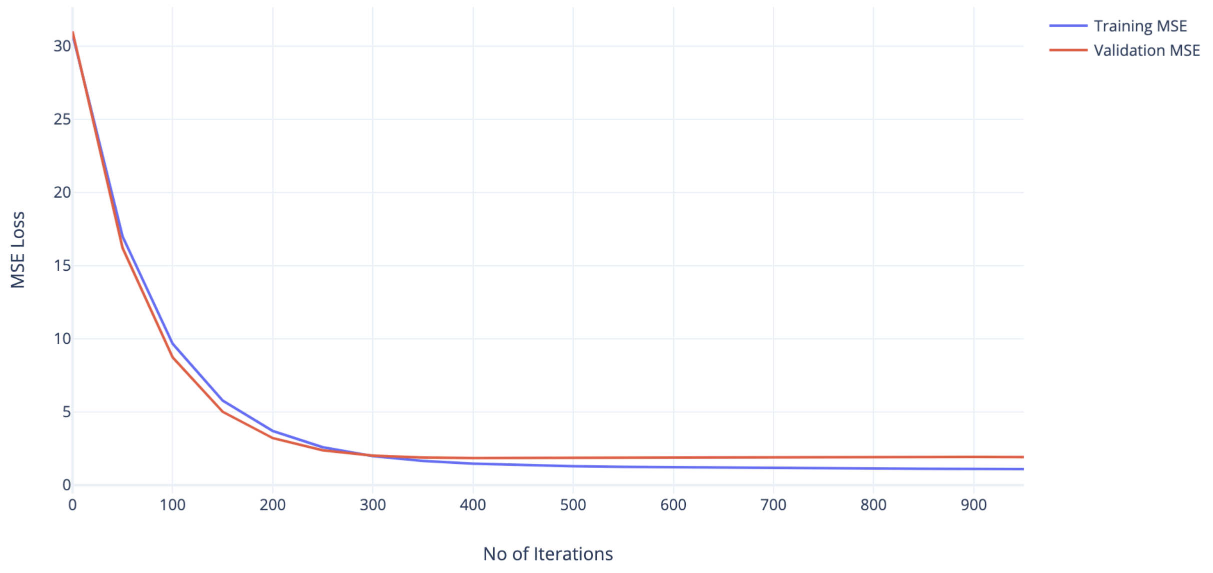


Lambda = 25

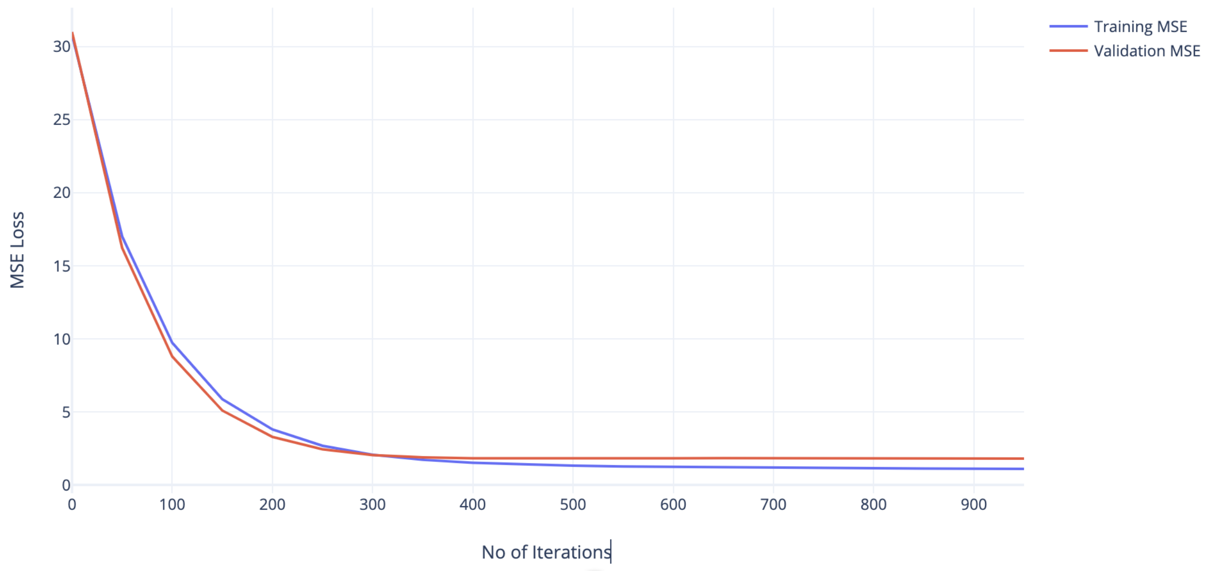


Learning Rate = 0.001

Lamda = 5



Lambda = 25



Again, the plots show that ridge regression prevents overfitting as the validation error is reduced as compared to the plots of linear regression.

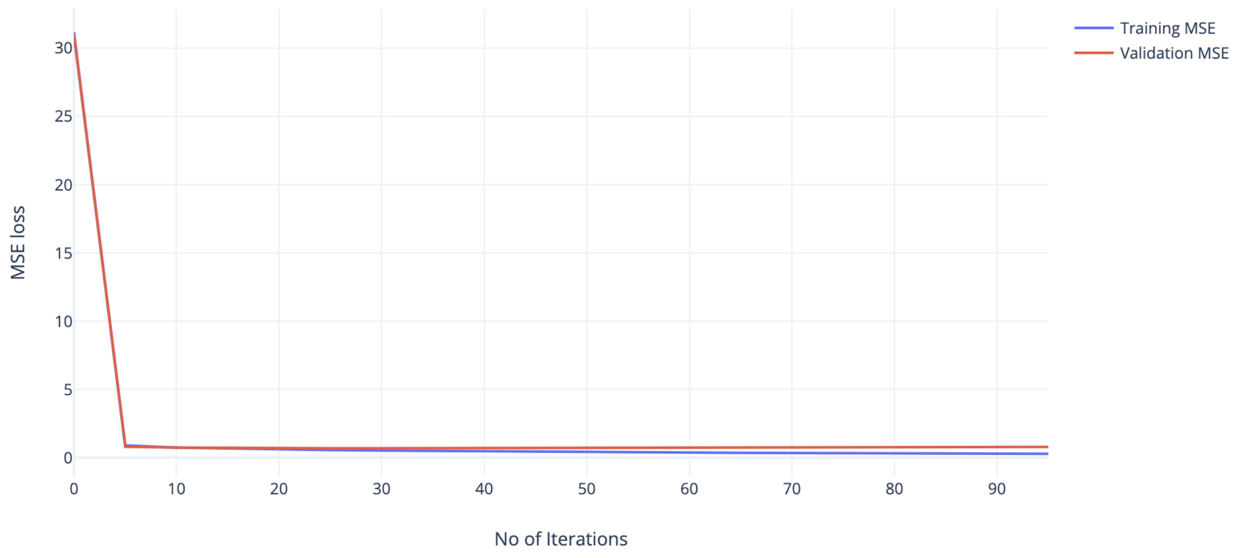
3.5

## Linear regression trained on parts of training dataset

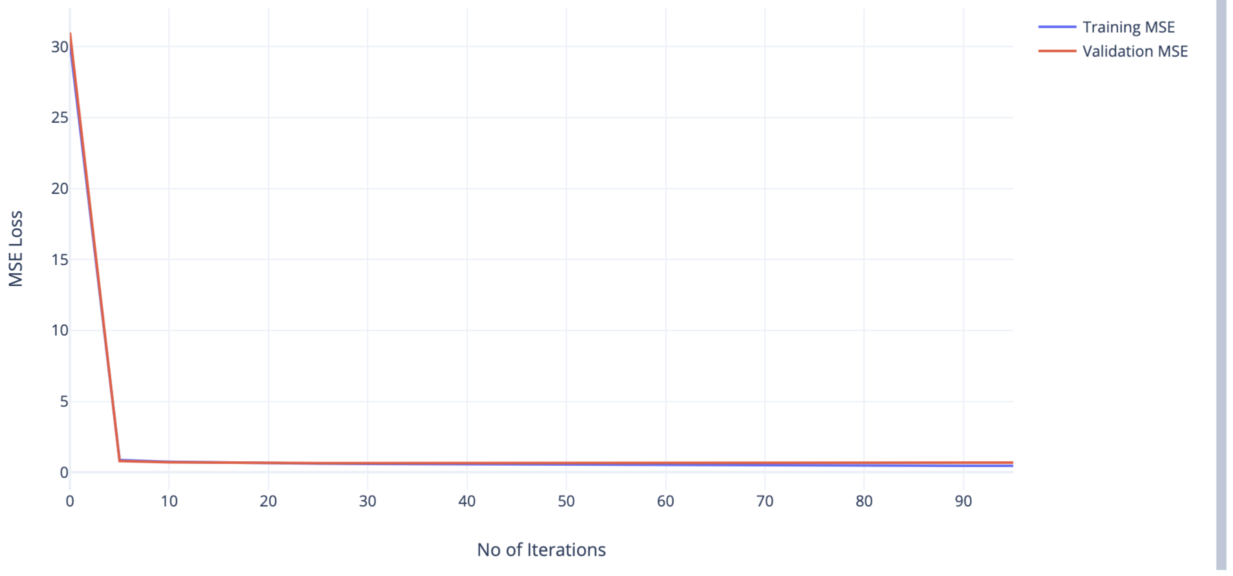
For 3.1:

Learning rate = 0.001 (since it only converges for this)

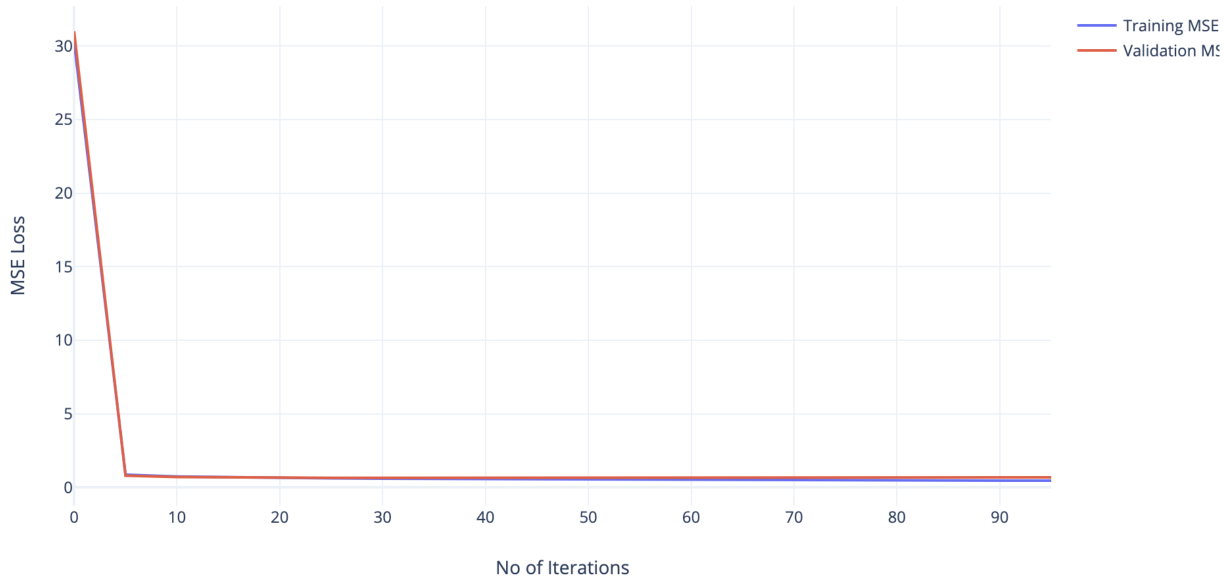
One-fourth of training data:-



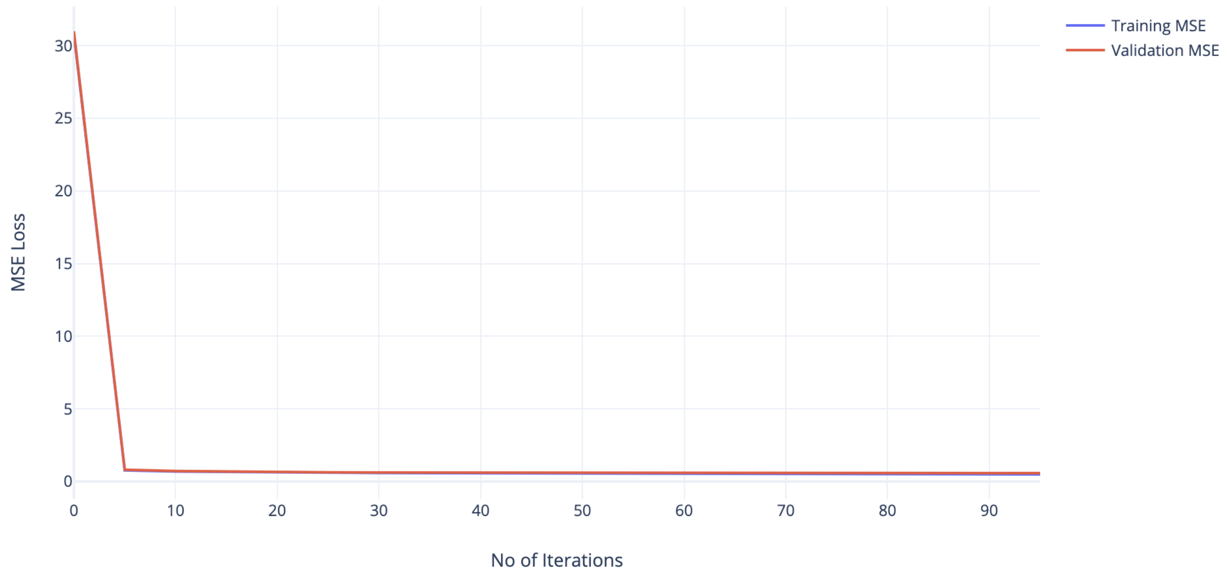
One half of training data:



Three Fourth of training data:


Training MSE
Validation MS
MSE Loss
No of Iterations
0
5
10
15
20
25
30
0
10
20
30
40
50
60
70
80
90

Full Data:


Training MSE
Validation MSE
MSE Loss
No of Iterations
0
5
10
15
20
25
30
0
10
20
30
40
50
60
70
80
90

**3.1 Linear Regression on two halves of training data:**

Mean Absolute Difference = 1.3851158525138192

**3.2 Ridge Regression on two halves of training data:**

For lambda = 5:

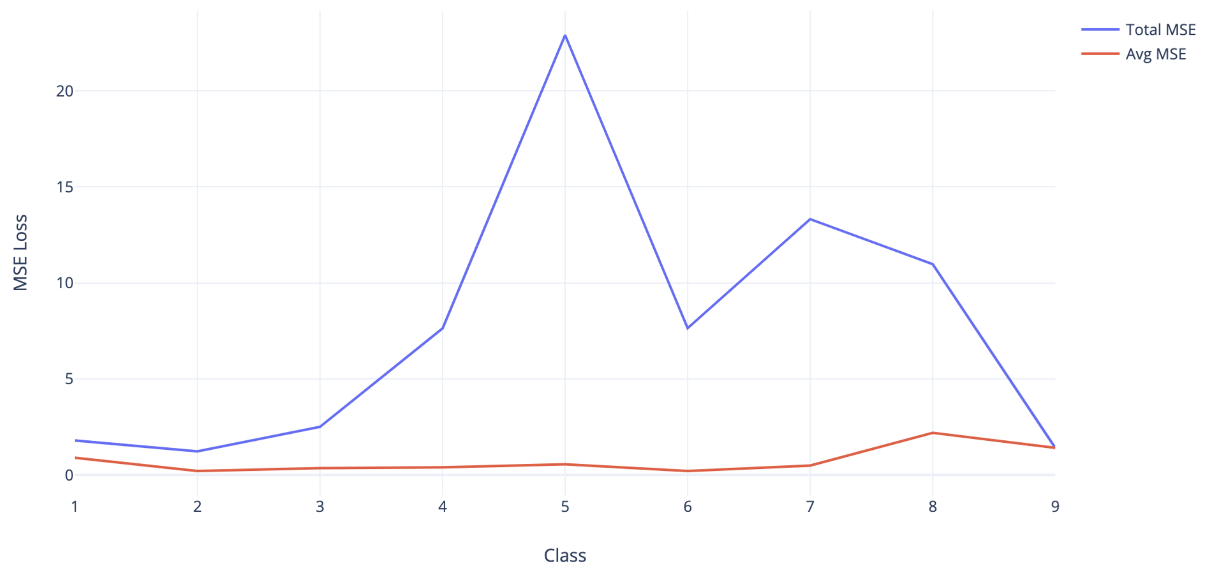
Mean Absolute Difference= 1.384426093447689

For lambda = 25

Mean Absolute Difference= 1.3816830903902242

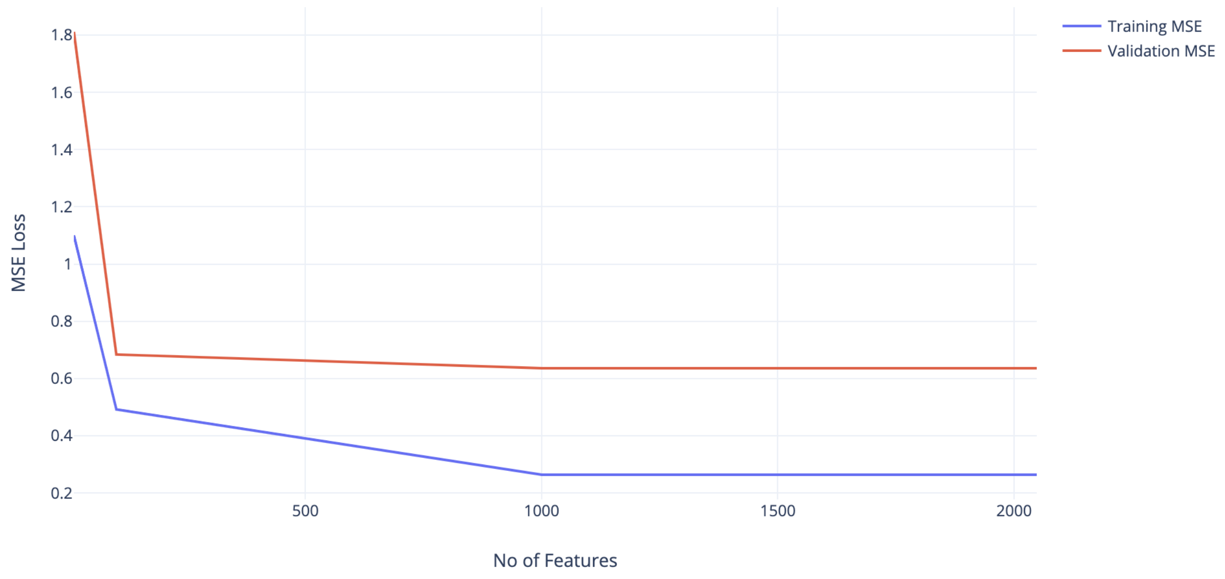
Ridge regression clearly shows better performance as expected.

**3.1 Total and Avg MSE Loss per class for linear regression**

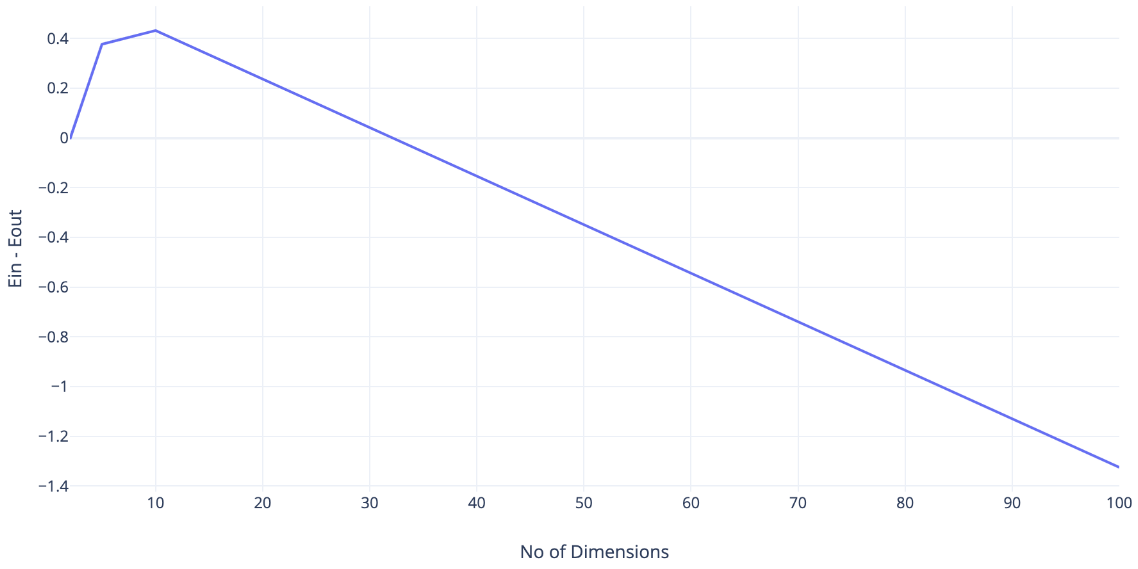


### 3.4 SelectFromModel Feature selection for 10,100, 1000, 2048 features

For learning rate = 0.001, we have:


Training MSE
Validation MSE
MSE Loss
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
500
1000
1500
2000
No of Features

## 3.7



This is the plot for E_in - E_out vs the number of dimensions (2, 5, 10, 100).