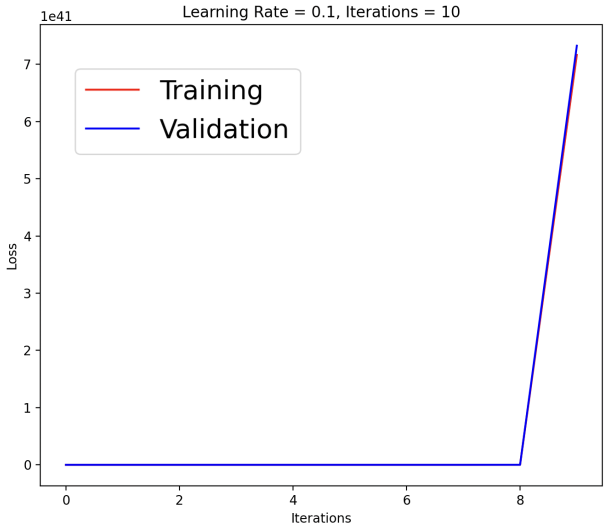
COL341 Assignment-1 Report

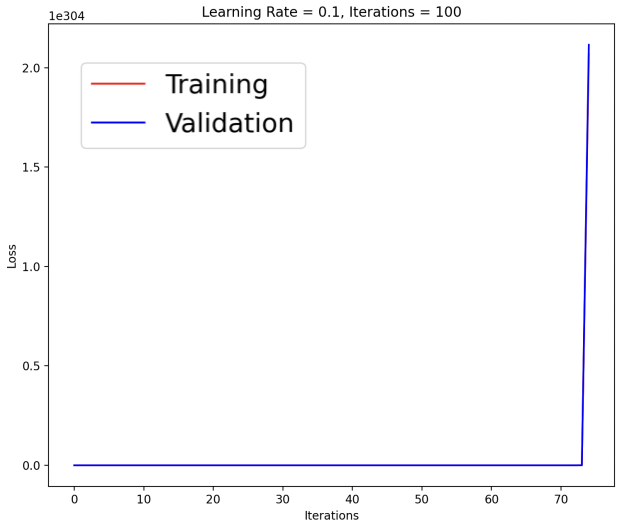
Amaiya Singhal (2021CS50598)

3.1 Linear Regression

Learning Rate = 0.1



(a) 10 Iterations

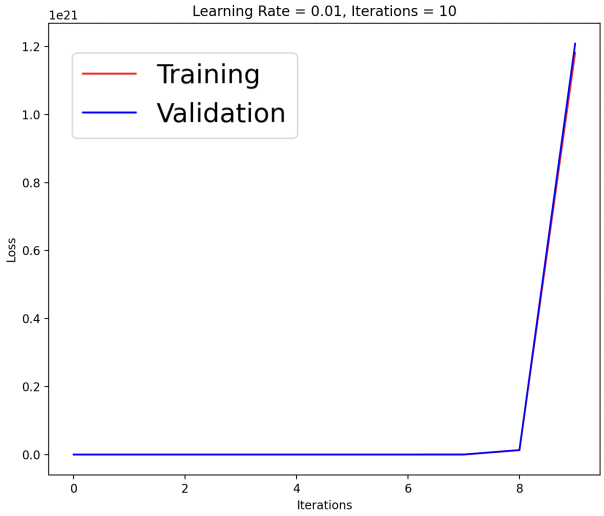


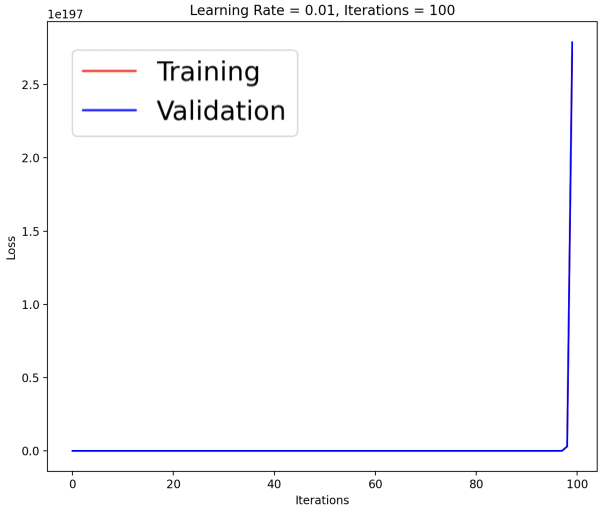
(b) 100 Iterations

Observation

- The MLE loss for both the train (RED) and validation (BLUE) coincide and shoot up very quickly and diverge.
- We can observe that the loss diverges in this case because of the learning rate being too high. So we need to reduce the learning rate to be able to train our model.

Learning Rate = 0.01

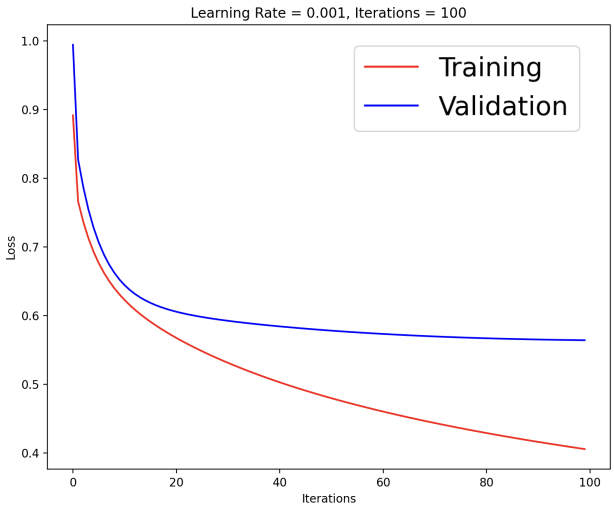




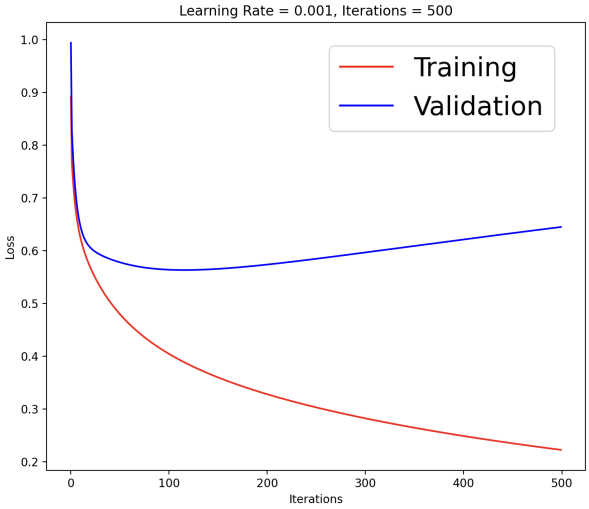
Observation

- The MLE loss for both the train (RED) and validation (BLUE) coincide and shoot up very quickly and diverge.
- We can observe that the loss diverges in this case (similar to the case of learning rate = 0.1) because of the learning rate still being too high. So we need to reduce the learning rate even further to be able to train our model.

Learning Rate = 0.001

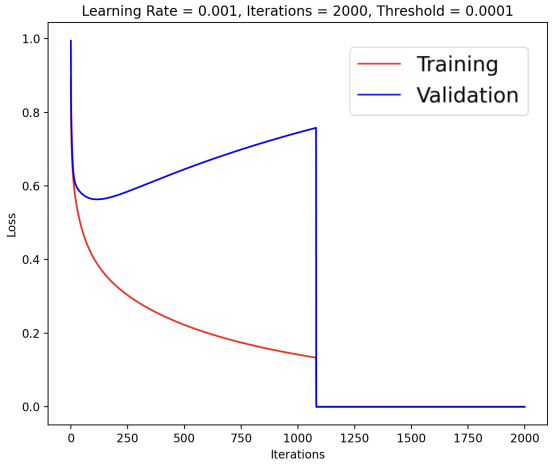


(a) 100 Iterations

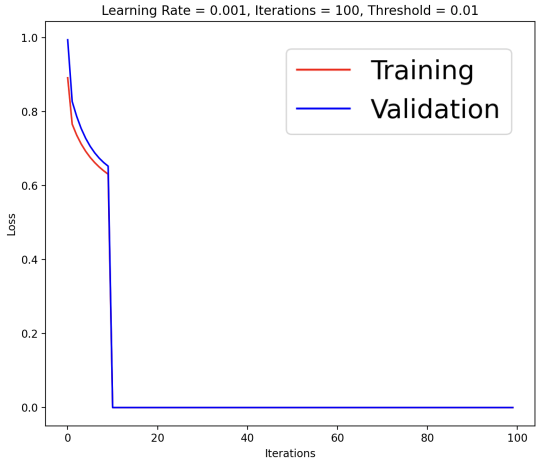


(b) 500 Iterations

- We are able to successfully train our model at this learning rate and from the plots as the loss in the train set is converging towards 0.
- We observe that the optimum point where the loss on the **validation** set is the least (neither underfitting nor overfitting on the train set) is slightly ahead of 100 iterations.



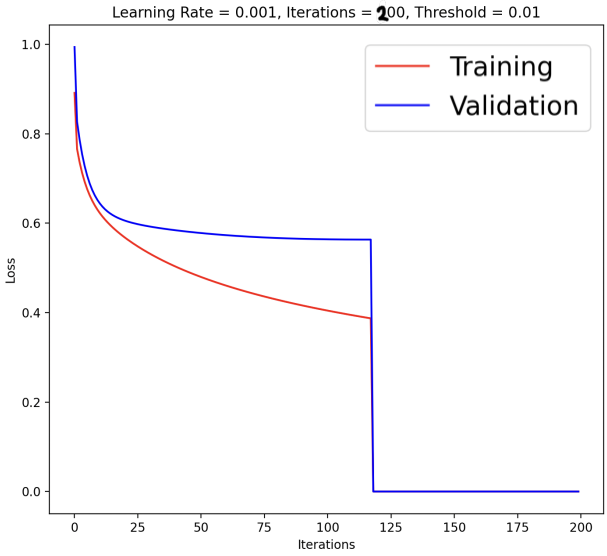
(a) High Threshold



(b) Low Threshold

(The vertical line represents the point where the error went below the threshold)

- On calculation the threshold to achieve optimum results on validation is achieved at approx. 0.00095 at 116 iterations.



Threshold for optimum fitting

MSE and MAE Calculations

The results are only for the learning rate = 0.001 since we are not able to train our model on the learning rates of 0.1 and 0.01.

100 Iterations (Underfitting)

Train

- **MSE:** 0.4056386053324175
- **MAE:** 0.5098124729409206

Validation

- **MSE:** 0.5641227732034119
- **MAE:** 0.5664336599762227

500 Iterations(Overfitting)

Train

- **MSE:** 0.2227964415614262
- **MAE:** 0.37599741382068186

Validation

- **MSE:** 0.644952584870146
- **MAE:** 0.6234786909528429

116 Iterations (Best Fit)

Train

- **MSE:** 0.38916776862168007
- **MAE:** 0.5014068042425861

Validation

- **MSE:** 0.5634600567431233
- **MAE:** 0.5629263256519684

0.01 (Low Threshold)

Train

- **MSE:** 0.6307818721829046
- **MAE:** 0.61869875428434

Validation

- **MSE:** 0.6525025959643062
- **MAE:** 0.6407273025947152

0.0001 (High Threshold)

Train

- **MSE:** 0.13347244782146084
- **MAE:** 0.28484740628008287

Validation

- **MSE:** 0.7577856859704185
- **MAE:** 0.6807120561155299

0.00095 (Optimum)

Train

- **MSE:** 0.38916776862168007
- **MAE:** 0.5014068042425861

Validation

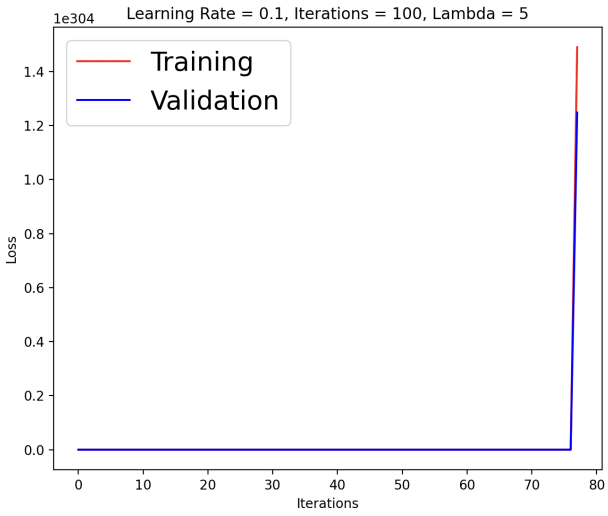
- **MSE:** 0.5634600567431233
- **MAE:** 0.5629263256519684

The best results on the Validation set is obtained at 116 iterations (or a threshold of 0.00095) for the learning rate = 0.001.

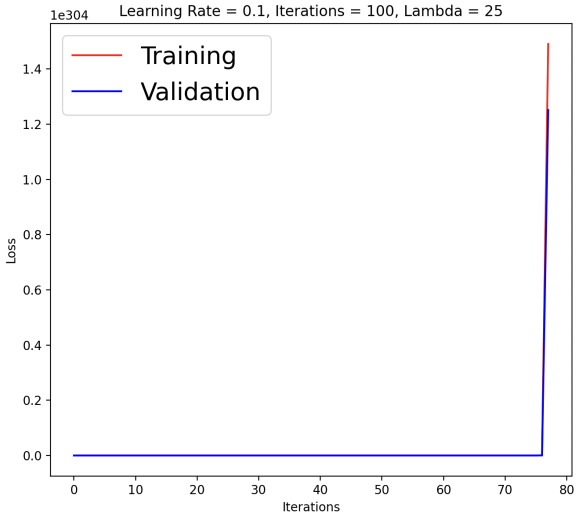
3.2 Ridge Regression

The data has been normalized for this part as Ridge Regression requires normalised data.

Learning Rate = 0.1

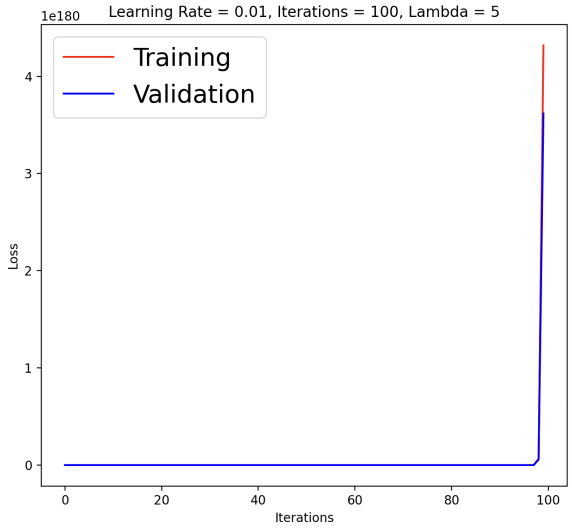


(a) $\lambda = 5$, Iterations = 100

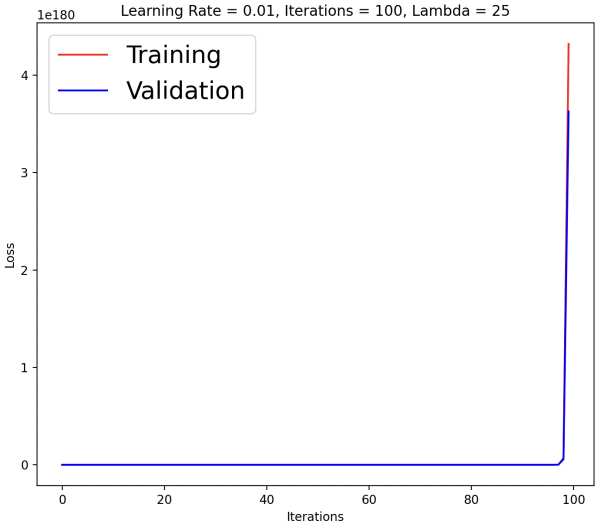


(b) $\lambda = 25$, Iterations = 100

Learning Rate = 0.01



(a) $\lambda = 5$, Iterations = 100

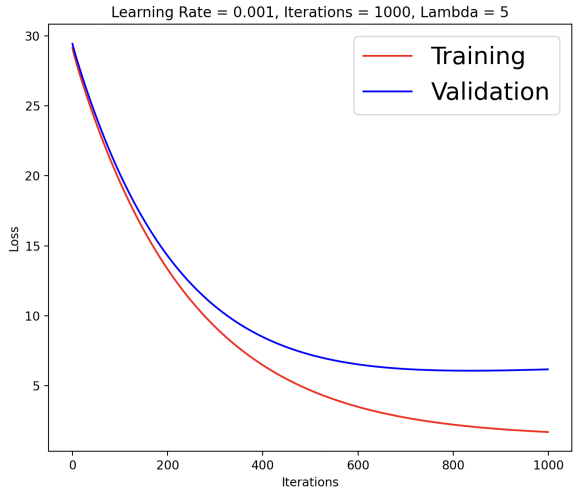


(b) $\lambda = 25$, Iterations = 100

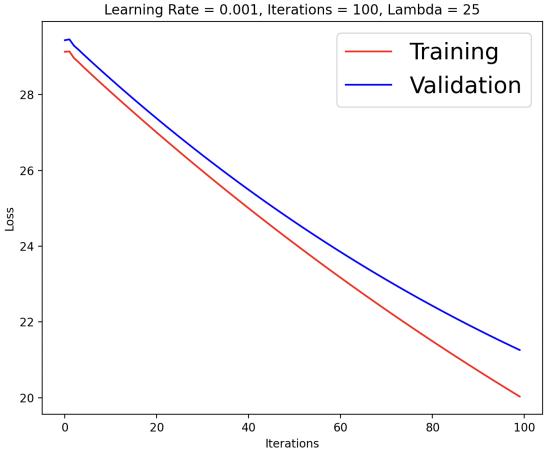
In both these cases, we are unable to train the data as the cost diverges very quickly because the learning rate is too high. So we need to reduce the learning rate.

Learning Rate = 0.001

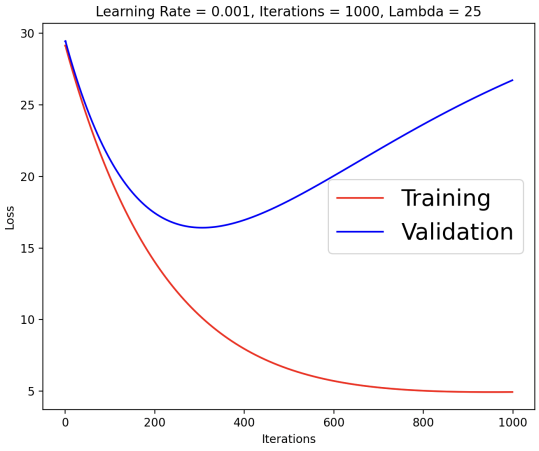
(a) $\lambda = 5$, Iterations = 100



(b) $\lambda = 5$, Iterations = 1000



(a) $\lambda = 25$, Iterations = 100



(b) $\lambda = 25$, Iterations = 1000

- For $\lambda = 5$, we obtain the optimum fit in 835 iterations.
- For $\lambda = 25$, we obtain the optimum fit in 305 iterations which is earlier than the case for $\lambda = 5$.

Cost Function: $\frac{1}{N}(\sum_{i=1}^{m}(y_i - \sum_{j=1}^{n}\theta_j x_{ij})^2 + \lambda\sum_{j=1}^{n}\theta_j^2)$
Update Function: $w = w - \frac{2\alpha}{N}(\sum_{i=1}^{m}(y_i - \sum_{j=1}^{n}\theta_j x_{ij})x + \lambda\sum_{j=1}^{n}\theta_j)$

MAE and MSE Values

$\lambda = \mathbf{5}$

- **Cost:** 6.168611576993586
- **MSE:** 1.0742232849206237
- **MAE:** 0.8815520726540608

$\lambda = \mathbf{25}$

- **Cost:** 26.71882524067378
- **MSE:** 1.2513428840594125
- **MAE:** 0.924525040984169

Observations

- The errors for the case of $\lambda = 25$ are more than that for $\lambda = 5$, this is because the higher the values of λ, the more we are restricting the variation in the weight values and hence increasing the errors.
- If we increase the value even further to a value such as 100, we won't even be able to train our model properly.
- The errors in this case are more than that obtained for the case of Simple Linear Regression because.
- Because we are regularising the rates, we make a tradeoff by increasing the errors while training but which will lead to better results on the test set.

3.3 Using `sklearn` Linear Regression

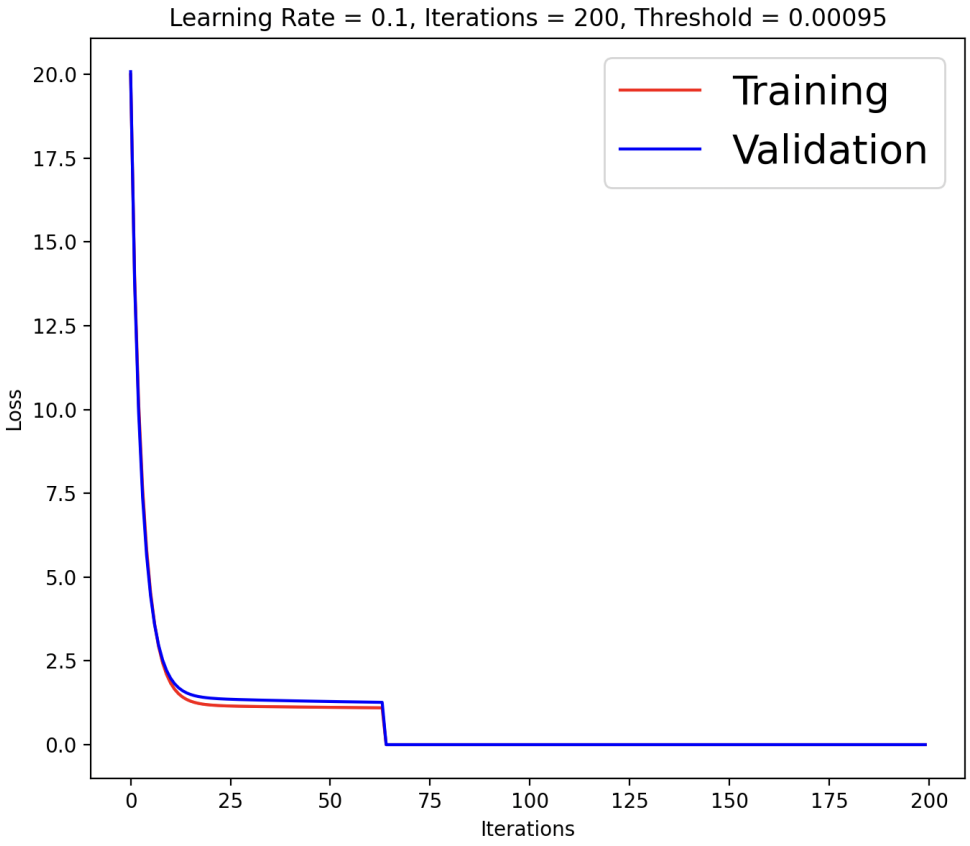
Using the `LinearRegression` from the `sklearn` library and training it on the `train` dataset we obtain the following results for the `validation` set:

- **Accuracy:** 49.89%
- **MSE:** 1.0249664536361625
- **MAE:** 0.8321230736880548

The values of errors obtained in this case are better than that obtained in the case of Ridge Regression (3.2) but worse off than in the case of Linear Regression (3.1).

3.4 Feature Selection

`SelectKBest`



Performing Linear Regression using SelectKBest

Observations

Validation

- **MSE:** 1.2625788156888162
- **MAE:** 0.9489668722843563

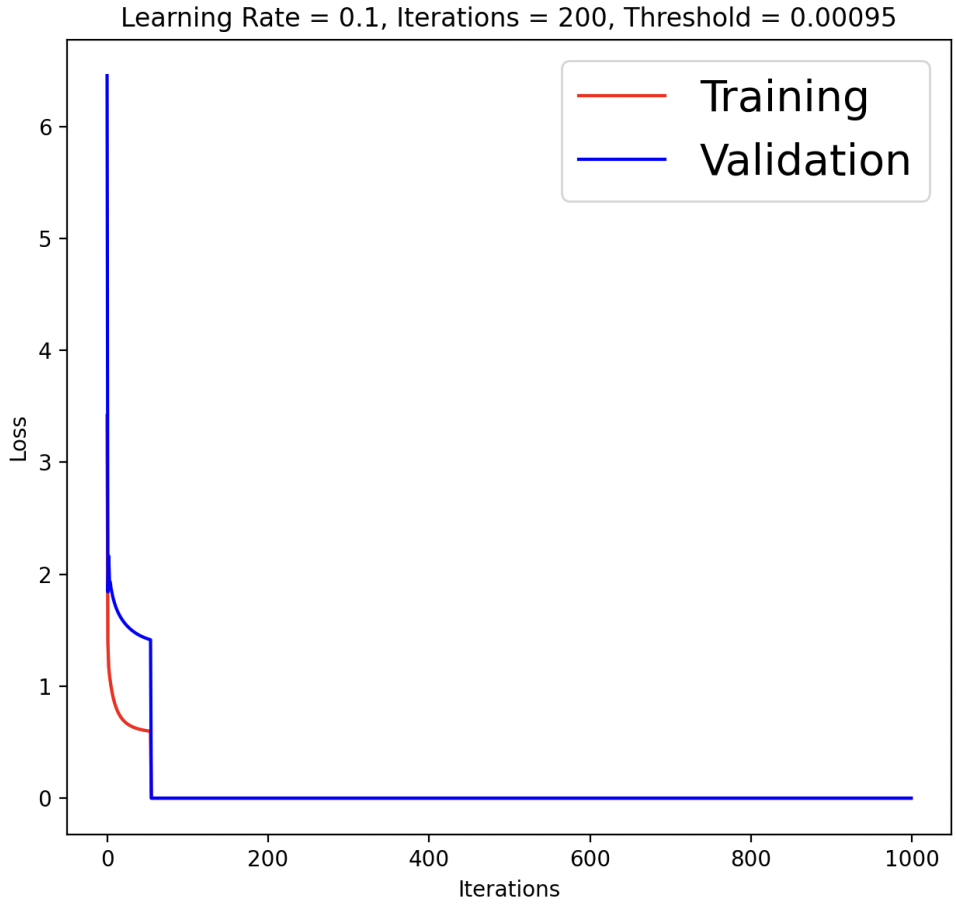
Train

- **MSE:** 1.098797009565467
- **MAE:** 0.8670448423420092

We were unable to train our model at this training rate or even at a tenth of this training rate with all the features. (the drop to 0 in values is the point of reltol stopping)

When using all the features we were able to train at a learning rate of 0.001. We had better results in that case (MSE = 0.56...) compared to the current model (1.26...) based on features obtained from `SelectKBest` but there was a tradeoff on the number of computations to be performed due to the large number of features.

`SelectFromModel`



Performing Linear Regression using SelectFromModel

Observations

Validation

- **MSE:** 0.597383501025248
- **MAE:** 0.6111143420209537

Train

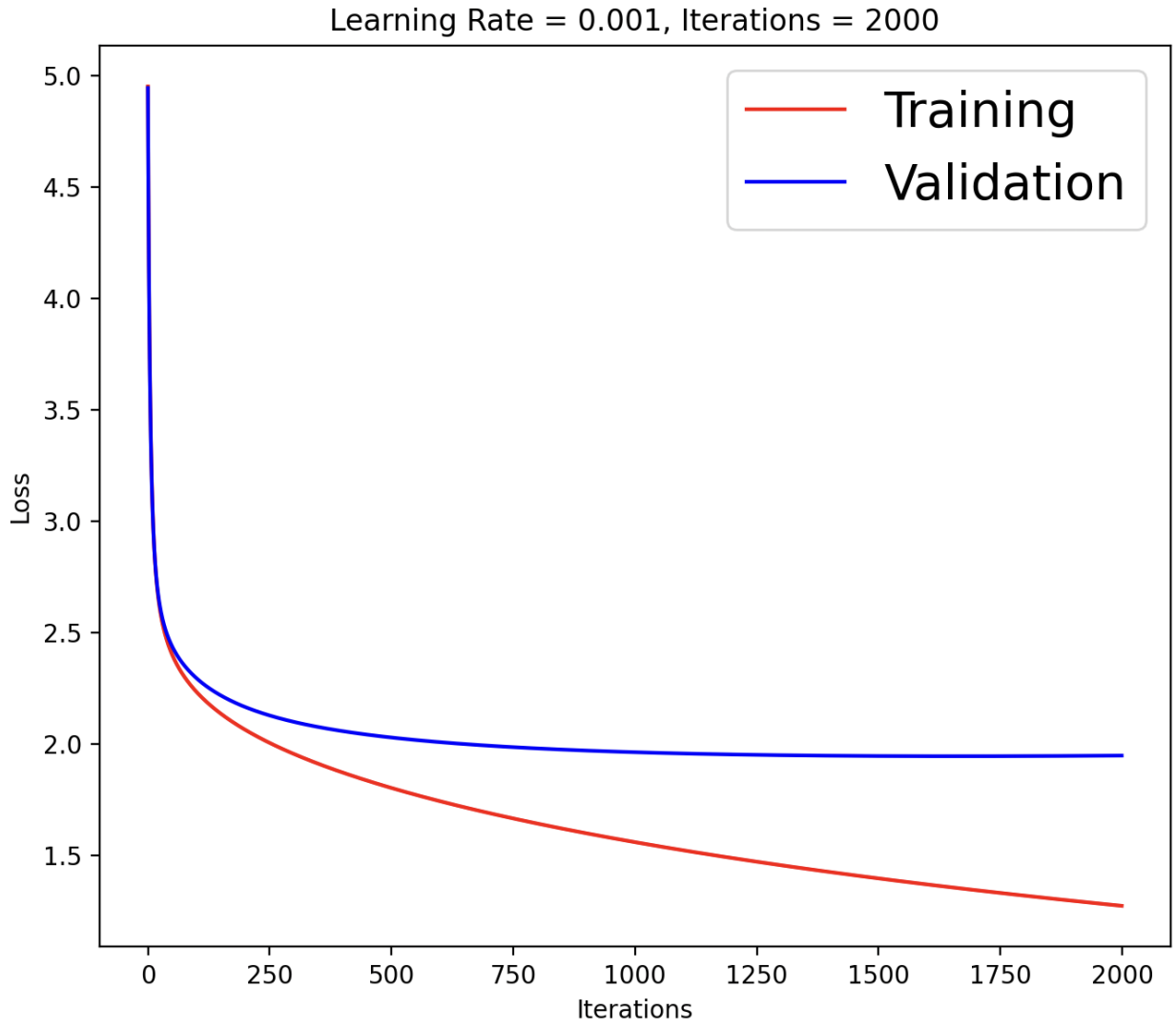
- **MSE:** 1.4147252524125067
- **MAE:** 0.8134249853427332

We were unable to train our model at this training rate or even at a tenth of this training rate with all the features but now we are able to (the drop to 0 in values is the point of reltol stopping).

When using all the features we were able to train at a learning rate of 0.001. We had better results in that case (MSE = 0.56...) compared to the current model (1.41...) based on features obtained from `SelectFromModel` but there was a tradeoff on the number of computations to be performed due to the large number of features.

3.5 Classification

Gradient : $-(y - h_\theta(x))x$
Update Function : $w = w - \frac{\alpha}{N} * \sum_{i=1}^{N}(h_\theta(x) - y)x$



The number of iterations to obtain the best fit comes out to be around 1665 iterations.

3.6 Visualisation

The training and validation loss plots for each of the parts 3.1, 3.2, 3.4 and 3.5 are in the respective sections. **The criteria for convergence for each of the plots is the difference in losses between consecutive iterations going below 0.00002**, this is a rough estimate obtained graphically.

For 3.1

Convergence achieved in 3100 iterations.
(Learning Rate = 0.001)

Iteration	Loss
155	0.356318161519581
310	0.2786725713397958
465	0.2311852979246087
620	0.19697176761237578
775	0.17075723480460178
930	0.14993733321418556
1085	0.13297429872000346
1240	0.11887678103960929
1395	0.10697084402748683
1550	0.09678094480302593
1705	0.08796166175323744
1860	0.0802559793630405
2015	0.07346865133993546
2170	0.06744861454288237
2325	0.06207706108897989
2480	0.057259161787102886
2635	0.05291820809202552
2790	0.048991391966254516
2945	0.0454267166941148
3100	0.04218070218053999

For 3.2

Convergence achieved in 1600 iterations.
(Learning Rate = 0.001)

Iteration	Loss
80	21.12248511714897
160	15.509377544631226
240	11.466305009758903
320	8.558215723478051
400	6.470056676270222
480	4.973713743816905
560	3.904096052650521
640	3.141788209298821
720	1.9487348188504012
800	2.2177834141997916
880	1.9487348188504012
960	1.7608818981019219
1040	1.6308671309127485
1120	1.5418981632115323
1200	1.4819242429553359
1280	1.4423160399901471
1360	1.4169103520374497
1440	1.4013192366580662
1520	1.3924307102458058
1600	1.3880481813936478

For 3.4 `SelectKBest`

Convergence achieved in 1000 iterations.
(Learning Rate = 0.1)

Iteration	Loss
50	1.1123503786536448
100	1.0718388326195791
150	1.051203100243355
200	1.0392994043096075
250	1.0315745510562009
300	1.0260554502330013
350	1.0218202772915363
400	1.0183994077675986
450	1.015531858539691
500	1.0130610872818802
550	1.0108873396654352
600	1.0089440779000105
650	1.0071853407887004
700	1.0055784584467846
750	1.0040996067919286
800	1.002730980602136
850	1.0014589438241677
900	1.0002727951553259
950	0.9991639322879059
1000	0.9981252795138096

For 3.4 `SelectFromModel`

Convergence achieved in 200 iterations.
(Learning Rate = 0.1)

Iteration	Loss
10	0.8348939998444384
20	0.6969518618483095
30	0.6423547942444975
40	0.6159208109366516
50	0.6013870730064532
60	0.5926209077165008
70	0.5869534648174405
80	0.5830966865151274
90	0.5803710572299011
100	0.5783892490674373
110	0.5769159089345991
120	0.5758006432429179
130	0.5749435425660919
140	0.5742762027891727
150	0.5737506499791568
160	0.5733325628708573
170	0.5729969662450378
180	0.5727254118993125
190	0.5725040888150024
200	0.5723225309319561

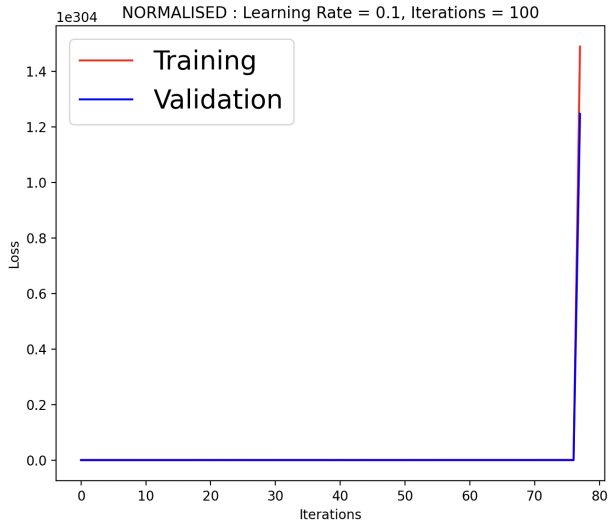
For 3.5

Convergence achieved in 8000 iterations.
(Learning Rate = 0.001)

Iteration	Loss
400	1.8703552773736536
800	1.6410210806131595
1200	1.4862335822614874
1600	1.3676719797167567
2000	1.2714862192324279
2400	1.1907677386560753
2800	1.1214761175902
3200	1.0609946767882228
3600	1.0075087986524618
4000	0.9597031494235763
4400	0.9165954019998017
4800	0.8774348384824471
5200	0.8416362672320539
5600	0.8087353420056963
6000	0.7783576290090015
6400	0.7501967769977573
6800	0.7239988401131385
7200	0.6995508329068281
7600	0.6766722437562598
8000	0.6552086475924069

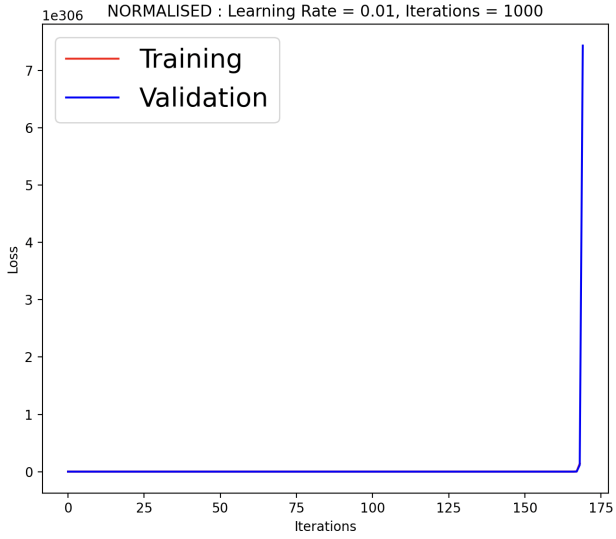
Performing 3.1 on Normalised Data

Learning Rate = 0.1



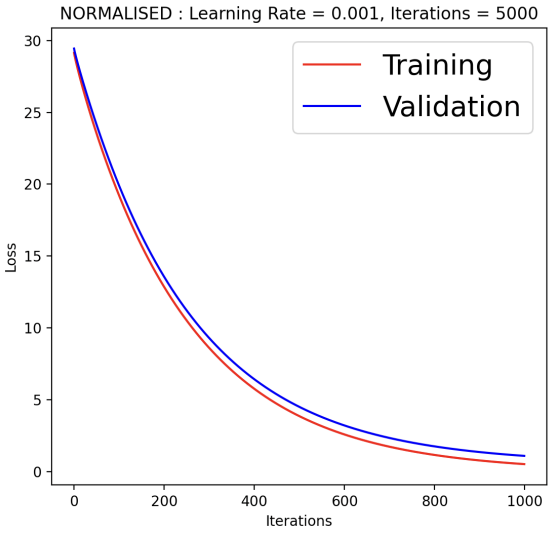
Observation: We are still not able to train the data at a learning rate of 0.1 after normalising.

Learning Rate = 0.01

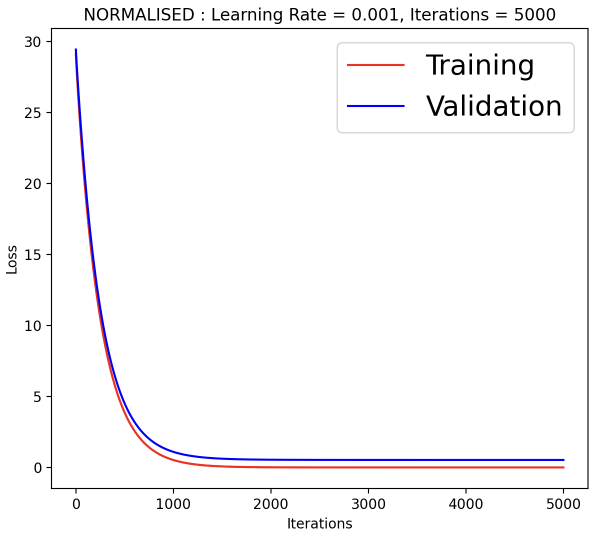


Observation: We are still not able to train the data at a learning rate of 0.01 after normalising.

Learning Rate = 0.001

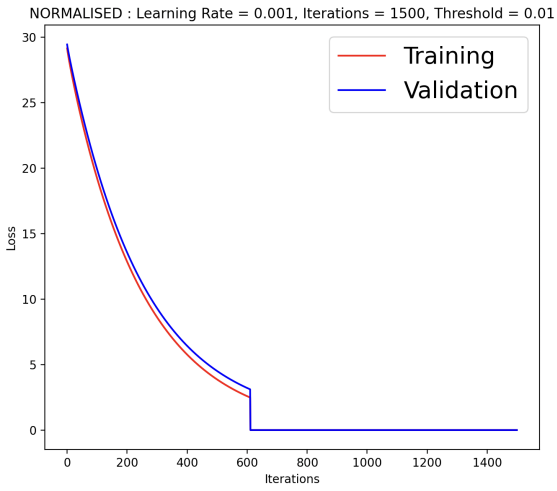


(a) Iterations = 1000

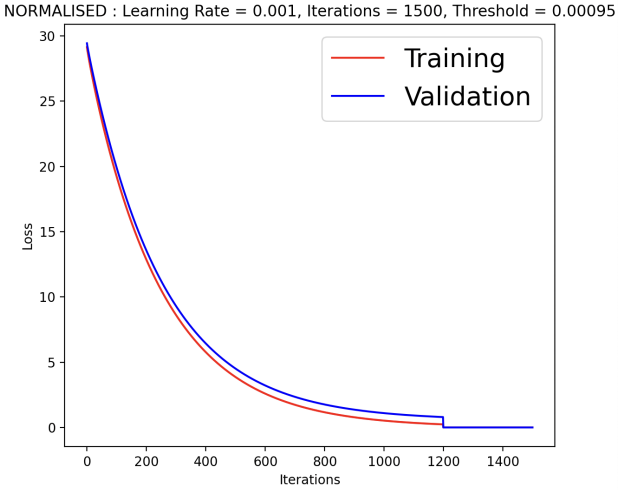


(b) Iterations = 5000

- The optimum value of the learning rate which was obtained at 117 iterations for the non normalised data is now obtained after 7600 iterations.



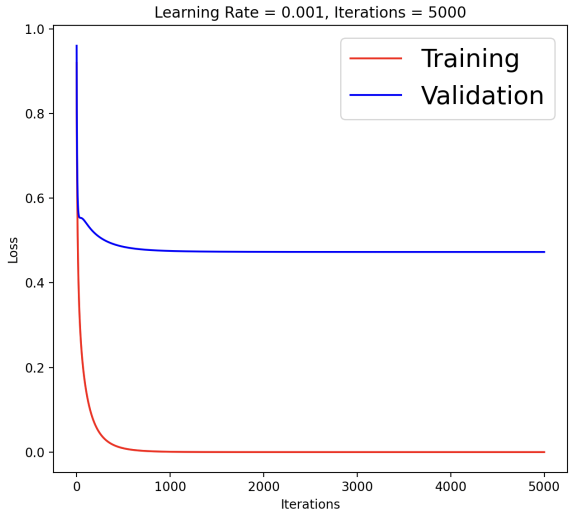
(a) Iterations = 1500, Low Threshold



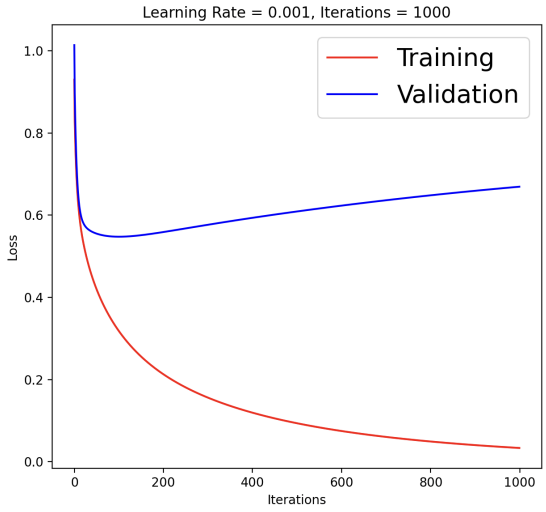
(b) Iterations = 1500, High Threshold

- The threshold obtained from 3.1 is achieved after 1198 iterations instead of 117 iterations for the non normalised data. Even low threshold cases are obtained much later than for non normalised data.
- The difference in number of iterations for best fit (7600) and for crossing the threshold (1198) shows that for the normalised data the same learning rates allows smaller steps.

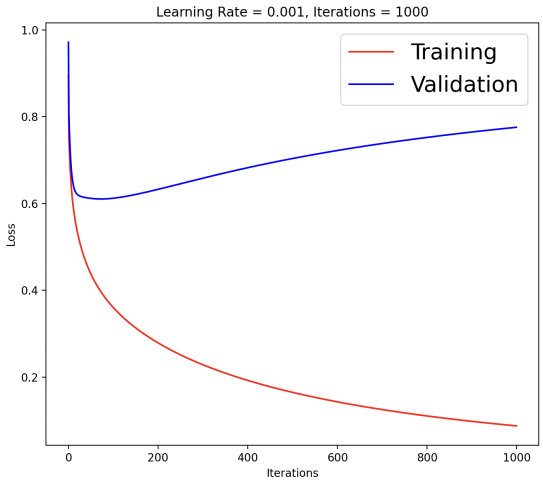
Repeating 3.1 on $(1/4),\ (1/2),\ (3/4),\ 1$ fraction of the train sample



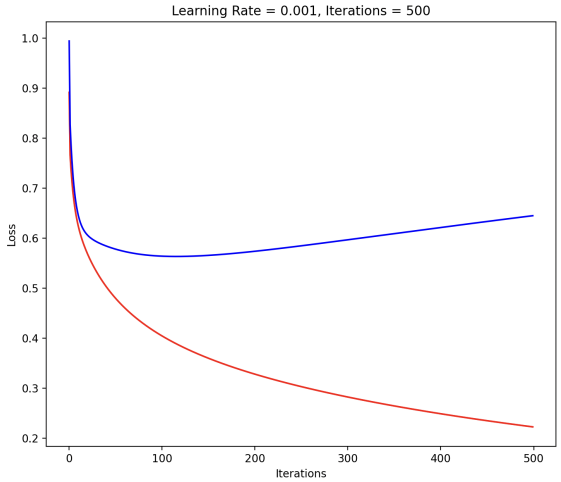
(a) On 1/4 of the train set



(b) On 1/2 of the train set



(c) On 3/4 of the train set



(d) On the full train set

- We can see that we are not able to obtain the point of best fit even after a lot of iterations when training using one fourth of the sample.
- For half, threee-fourth and the entire train sample cases the plots are very similar and we are able to obtain the best fit point after around 90-100 iterations in each of the cases.
- This shows that higher train sample size is important as the one-fourth sample size might not contain enough information for the model to efficiently train on.

Training on 2 halves of the train set

I split the train set into 2 using the `sklearn train_test_split` module and trained using both these halves and found out the best fits for both the models.

Observations for 3.1 model

- **Mean Absolute Difference on Train set:** 0.2741055289860345
- **Mean Absolute Difference on Validation set:** 0.2593006357750594

Observations for 3.2 model ($\lambda = 5$)

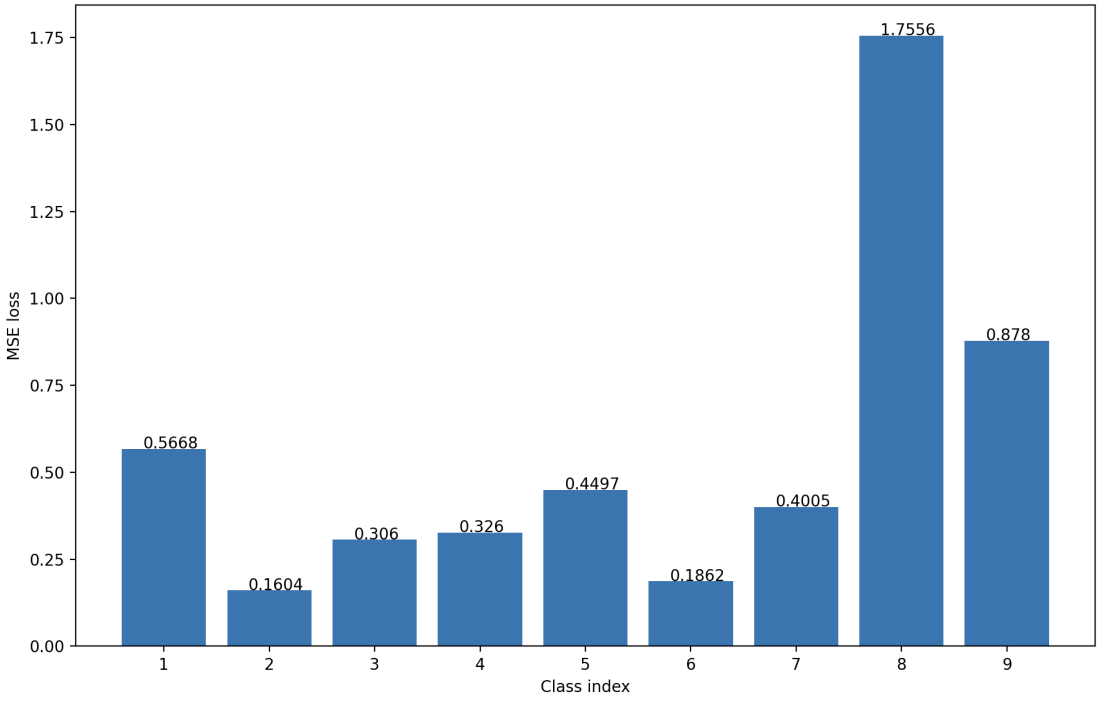
- **Mean Absolute Difference on Train set:** 0.20314796842974936
- **Mean Absolute Difference on Validation set:** 0.0867153870330457

Observations for 3.2 model ($\lambda = 25$)

- **Mean Absolute Difference on Train set:** 1.4500304783491502
- **Mean Absolute Difference on Validation set:** 1.4899738430125198

We observe that the Ridge Regression model with $\lambda = 5$ produces the best results on both the test and validation sets whereas the model with $\lambda = 25$ produces the worst results with maximum variation. This shows that having a lambda value that is too high is not a favourable but a good value such as 5 in this case can lead to significantly better results than the performing standard Linear Regression.

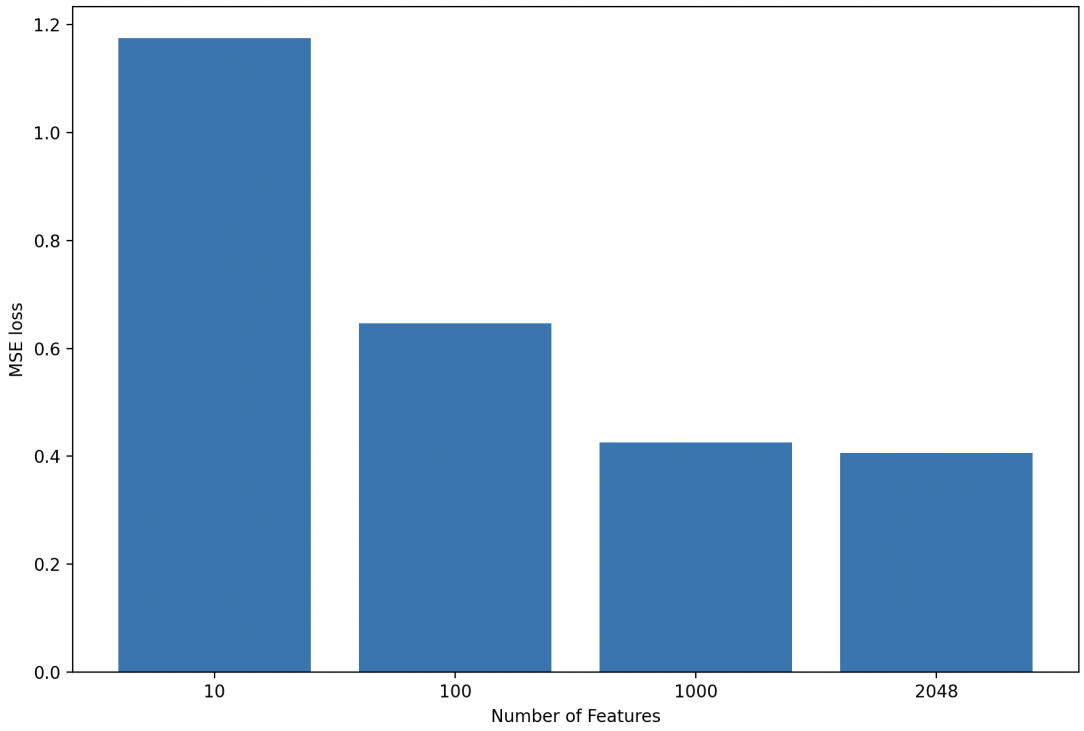
Plot of MSE loss against Correct Score Values for 3.1 on train



Errors for all classes are small and under around 0.5 except classes for 8 and 9.

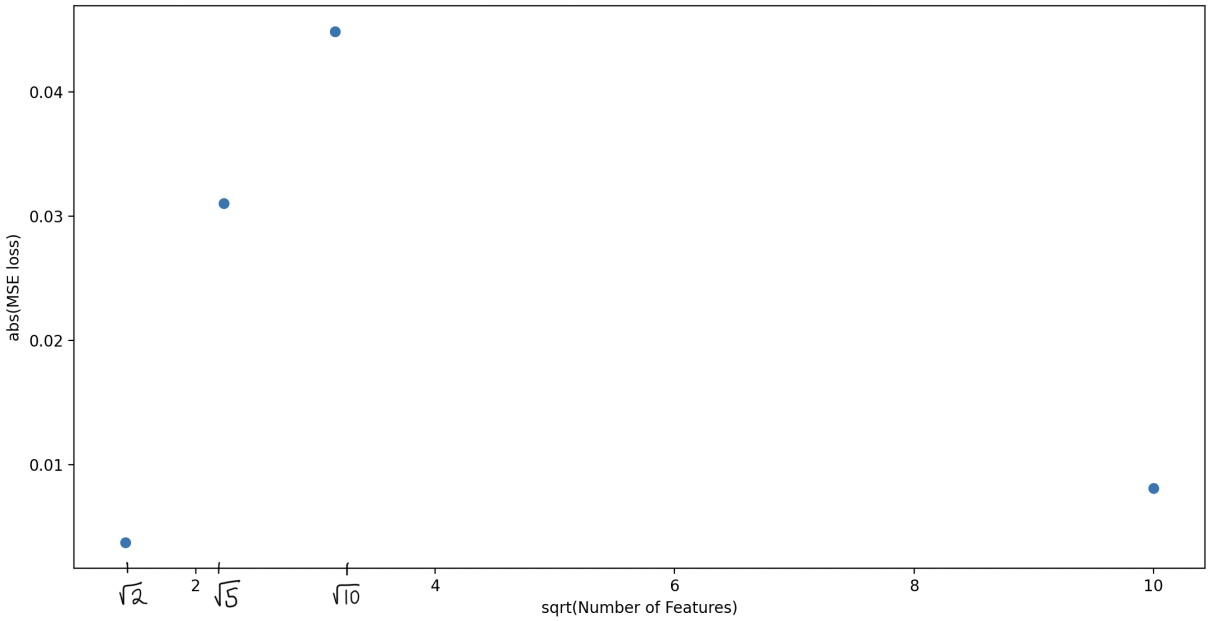
Choosing best 100 and 1000 features

- It is not possible to train the data with learning rates 0.1 and 0.01 for more features so we will be studying only learning rate = 0.001.
- `SelectFromModel` **produces better result and hence is used in this analysis** (We have used the reltol stopping using the threshold obtained from 3.1)



We can observe that as the number of features increase the MSE loss is decreasing but the difference between the losses for 1000 best features and all the features is very small. This shows that more features lead to better results however a lot of features are less important and do not play any significant role in improving the model.

3.7 Generalization Analysis



Observations

- We observe a linear relation between the square root of the number of features and absolute value of (E_in - E_out) as expected, however the value for 100 features does not follow this trend and is unexpectedly low.
- Since the generalization is only an upper bound, it is not surprising to see a dataset producing such results.