COL341 : MACHINE LEARNING (Spring 2023)

Name: Stitiprajna Sahoo
Entry Number: 2020CS10394

Date: January 25 2023
Assignment: 1

Assignment 1: Linear Regression

Contents:

1 Basic Linear Regression

Implementaion of Gradient Descent algorithm on Linear Regression takes the cost function = MSE loss = $J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y'_i - \mathbf{w}^T\mathbf{x}_i)^2$, where y' is the actual output of the i^{th} sample, $\mathbf{w}$ is the weight vector and $\mathbf{x}_i$ is the features vector corresponding to the i^{th} sample of train dataset. Computing the gradient $\nabla_{\mathbf{w}}J(\mathbf{w})$ gives us the value of gradient with respect to each w_j as:

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{\partial(\frac{1}{N}\sum_{i=1}^{N}(y'_i - \mathbf{w}^T\mathbf{x}_i)^2)}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial(y'_i - \mathbf{w}^T\mathbf{x}_i)^2}{\partial w_j} = -\frac{1}{N}\sum_{i=1}^{N}2(y'_i - \mathbf{w}^T\mathbf{x}_i)\frac{\partial \mathbf{w}^T\mathbf{x}_i}{\partial w_j}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = -\frac{2}{N}\sum_{i=1}^{N}(y'_i - \mathbf{w}^T\mathbf{x}_i)x_i^{(j)} \tag{1}$$

From here, we get the update equation of each w_j ($w_j \leftarrow w_j - \alpha\frac{\partial J}{\partial w_j}$) as: (with α as the learning rate.)

$$w_j \leftarrow w_j + \alpha\frac{2}{N}\sum_{i=1}^{N}(y'_i - \mathbf{w}^T\mathbf{x}_i)x_i^{(j)} \tag{2}$$

I initialise the weights as $\frac{1}{\sqrt{N}}$, for $||\mathbf{w}||$ to be equal to 1. The stopping criteria for linear regression was chosen both ways: *maxit* (with a cap on maximum iterations) and *reltop* (a threshold on the step size of MSE) The validation dataset is represented by $\mathbf{X}^v = [\mathbf{x}_1^v, \mathbf{x}_2^v, ...\mathbf{x}_p^v]$ We also report the MAE loss on the train as well as validation datasets represented by $\frac{1}{N}\sum_{i=1}^{N}|y'_i - \mathbf{w}^T\mathbf{x}_i|$

1.1 Observations:

- Observed MSE and MAE on the *train data* with stopping criteria: **maxit**

Learning Rate(α)	Max Iterations	MSE	MAE
0.1	10	2.833528888966081e+38	1.657719121115341e+19
0.01	10	5.55946177383234e+19	7342830423.942273
0.001	4000	0.02761873684087757	0.12309727297768713

- Observed MSE and MAE on the *train data* with stopping criteria: **reltol**

Learning Rate(α)	Threshold	MSE	MAE
0.1	1000	inf	2.4488284409531112e+156
0.01	1000	inf	3.940901538239074e+154
0.001	0.00092	0.3797774344328462	0.49013941454898147

(Number of Iterations in $\alpha = 0.001$: 170, by setting the threshold according to validation dataset)

- Observed MSE and MAE on the *validation data* with stopping criteria: **maxit**

Learning Rate(α)	Max Iterations	MSE	MAE
0.1	10	3.0007712795308887e+38	1.7218195605061878e+19
0.01	10	5.887595952502615e+19	7626761911.579987
0.001	400	0.6093323148131842	0.5707821827041579

- Observed MSE and MAE on the *validation data* with stopping criteria: **reltol**

Learning Rate(α)	Threshold	MSE	MAE
0.1	10000	3.0007712795308887e+38	1.7218195605061878e+19
0.01	10000	5.887595952502615e+19	7626761911.579987
0.001	$3 \cdot 10^{-6}$	0.5663249967228129	0.5370967456801427

(Number of Iterations in $\alpha = 0.001$: 170)

1.2 Analysis:

Using learning rates α as 0.1 or 0.01, for both train and validation dataset makes the E_{in} and E_{valid} shoot out to ranges of millions or even more (E_{in} is diverging). So the correct learning rate out of the 3 options for which error converges is $\alpha = 0.001$
For $\alpha = 0.01$, the E_{in} dcreases with the number of iterations for train data, as maxit further increases, but MSE_{valid} on validation data first decreases, reaches a minima and then increases. This is in correspondence to the bias variance analysis graph, where E_{in} decreases with iterations, but E_{out} has minima, then increases.

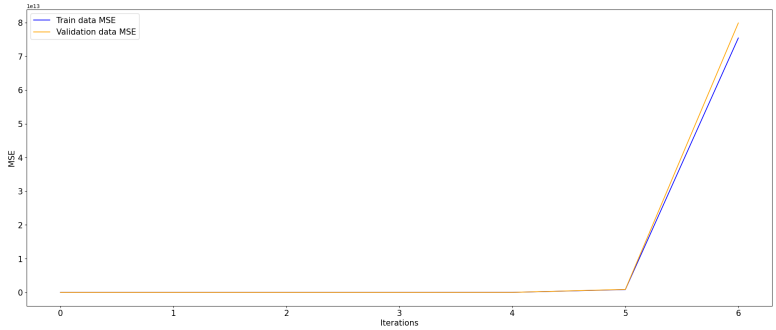

Figure 1: Linear Regression: MSE on train and validation data shooting out in case of $\alpha = 0.01$ and $\alpha = 0.1$

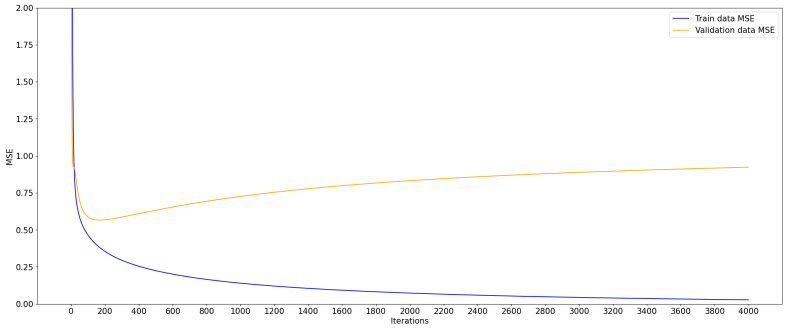

Figure 2: Linear Regression: MSE on train and validation data in $\alpha = 0.001$

For $\alpha = 0.001$: On experimental determination, the no. of iterations for which E_{valid} is minimum comes out to be 170. And at this point, the relative decrease in cost function for validation data is $3 \cdot 10^{-6}$, and at this point in the graph, the relative decrease in the cost function of E_{train} is 0.00092. *(There's no suitable threshold for $\alpha = 0.1/0.01$, as error always diverges and keeps on increasing)* So setting this value as the threshold gives final MSE and MAE as:

- For train data: MSE = 0.3797774344328462 MAE = 0.49013941454898147
- For validation data: MSE = 0.5663249967228129 MAE = 0.5370967456801427

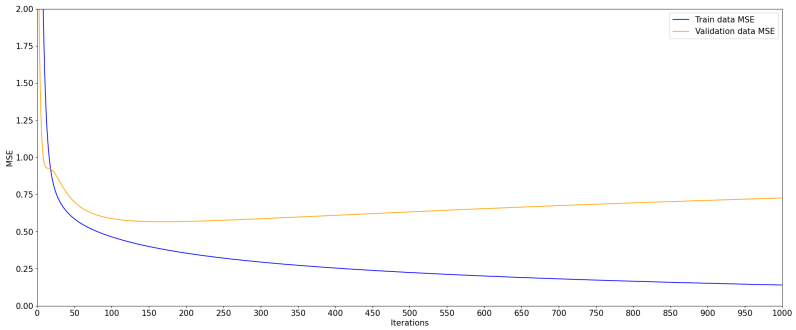

Figure 3: At 170 iterations, E_{valid} becomes the minimum, and then increases, so relative loss threshold $= 3 \cdot 10^{-6}$

For test set: I have set the following parameters and stopping criteria for better prediction on test set:

1. Learning Rate: 0.001 (optimal for both train and validation)
2. Stopping criteria: Upper cap on maxit = 500, with another stop criteria which will break the loop if the threshold decrease value (3e-06 acc. to validation set)

2 Ridge-Regression

The new cost function to be minimised given as (where $\theta = [\theta_1, \theta_2, ...\theta_n]^T$)

$$J(\theta) = \frac{1}{m}(\sum_{i=1}^{m}(\theta_i^T - y_i)^2 + \lambda||\theta||^2) \tag{3}$$

which gives the loss function as $J(\theta) = \frac{1}{m}(\sum_{i=1}^{m}(\sum_{j=1}^{n}\theta_j x_{ij} - y_i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2)$
I have taken cost function as $\frac{1}{m}$ of the loss function to normalise the loss, i.e. take the **average of loss over all samples**. This is done to remove the effect of the number of samples in the training set on the gradient function, which affects the learning rate was well. More samples can diverge the loss function, or more λ can also diverge the loss, if we don't average the cost function (divide it by m)
Taking the gradient of cost function with respect to θ gives:

$$\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{m}(\sum_{i=1}^{m}(\sum_{j=1}^{n}\theta_j x_{ij} - y_i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2)$$

Taking partial derivates: for each θ_j, $\frac{\partial J(\theta)}{\partial \theta_j} = \frac{\partial(\frac{1}{m}(\sum_{i=1}^{m}(\sum_{j=1}^{n}\theta_j x_{ij} - y_i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2))}{\partial \theta_j}$. So we get:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}(2\sum_{i=1}^{m}(\sum_{j=1}^{n}\theta_j x_{ij} - y_i)x_{ij} + 2\lambda\theta_j) \tag{4}$$

Using the update equation as: $\theta_j \leftarrow \theta_j - \alpha\frac{\partial J(\theta)}{\partial \theta_j}$ The final update equation we get for ridge regression is:

$$\theta_j \leftarrow \theta_j - \frac{2\alpha}{m}(\sum_{i=1}^{m}(\sum_{j=1}^{n}\theta_j x_{ij} - y_i)x_{ij} + \lambda\theta_j) \tag{5}$$

I have initialised $\theta = [1, 0, 0, ..., 0, 0]^T$ and reported the results:

2.1 Observations:

α = Learning Rate, λ = Ridge Parameter, = Maximum Iterations
Observed MSE and MAE on train-data with stopping criteria = **maxit**:

α	λ	maxit	MSE	MAE
0.001	5	1000	0.14709422876494585	0.299351162142144
0.001	25	1000	0.16661858512628935	0.3198250695073164
0.001	25	4000	0.08843581079714866	0.23022400148766367
0.01	5	10	7.822890703983119e+18	2754336043.341179

Observed MSE and MAE on train-data with stopping criteria = **reltop**:

α	λ	threshold	MSE	MAE
0.001	5	0.000822261039661154	0.3742713162925253	0.48987865920751805
0.001	25	0.0007678289777322544	0.3719658573573478	0.48831598972441587

Observed MSE and MAE on validation-data with stopping criteria = **maxit**:

α	λ	maxit	MSE	MAE
0.001	25	300	0.5836869777479505	0.43224430937815667
0.001	25	1000	0.7068181866505379	0.6426290797802485
0.001	5	1000	0.7308610795736342	0.65662116488724
0.01	5	10	8.284874883643818e+18	2860940346.0901365

Observed MSE and MAE on validation-data with stopping criteria = **reltop**:

α	λ	threshold	MSE	MAE
0.001	25	0.000002	0.5854118795765293	0.5753134681590322
0.001	5	$7 \cdot 10^{-7}$	0.585406840206038	0.5752554046777596

Number of iterations $\alpha = 0.001$, $\lambda = 25$: 138
Number of iterations $\alpha = 0.001, \lambda = 5$: 132

2.2 Analysis:

Here too, the error E_{in} diverges for both values of $\alpha = 0.1/0.01$ for both λ values. The error converges for only rate = 0.001

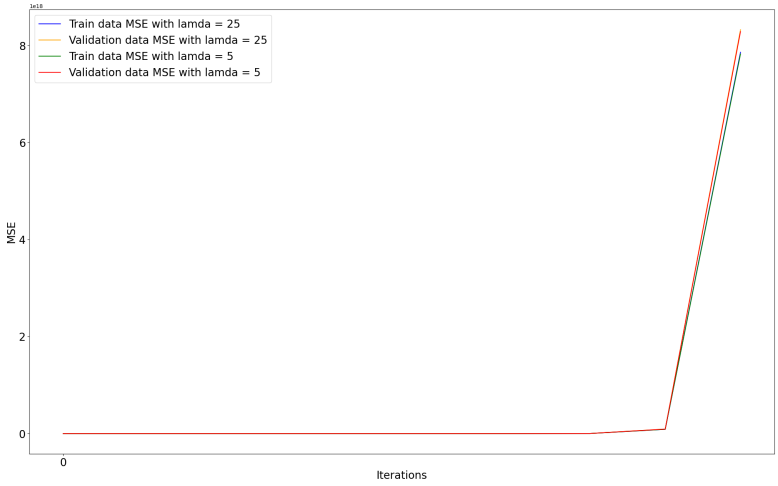

Figure 4: Ridge Regression: MSE on train and validation data at $\lambda = 5, \alpha = 0.1$

Here, λ parameter gives weight factor < 1 to the weights i.e. θ (as compared to Linear regression which gives 1). This makes the and more closer to the average θ:

- $variance()_{\lambda=25} < variance()_{\lambda=5}$
- $variance()_{\lambda=25} < variance()_{\lambda=5}$

But bias increases on increasing λ
For linear regression $\lambda = 0$ So, as we increase the λ parameter, the variance factor becomes less.
But in the case of bias factor, ≈ 0.027 in linear regression, but $\approx 0.14, (\lambda = 5), \approx 0.16(\lambda = 25)$ in Ridge Regression. Thus bias increases on increasing λ
Therefore, for bias-variance analysis, as we introduce higher λ parameter making the model more from Linear to ridge regression, variance decreases but bias increases.

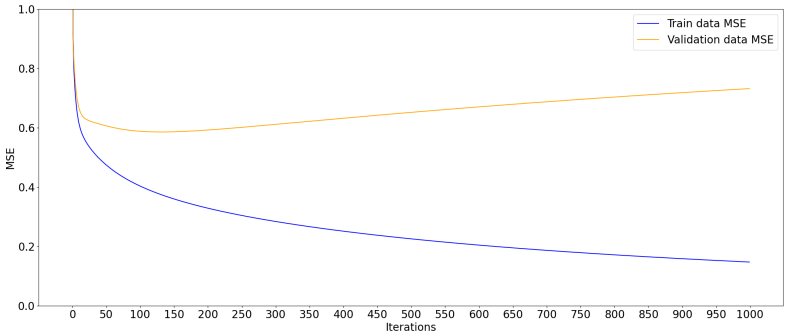

Figure 5: Ridge Regression: MSE on train and validation data at $\lambda = 5, \alpha = 0.001$

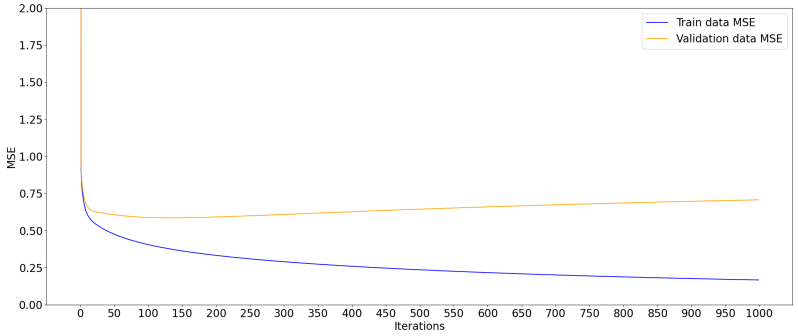

Figure 6: Ridge Regression: MSE on train and validation data for Ridge regression at $\lambda = 25, \alpha = 0.001$

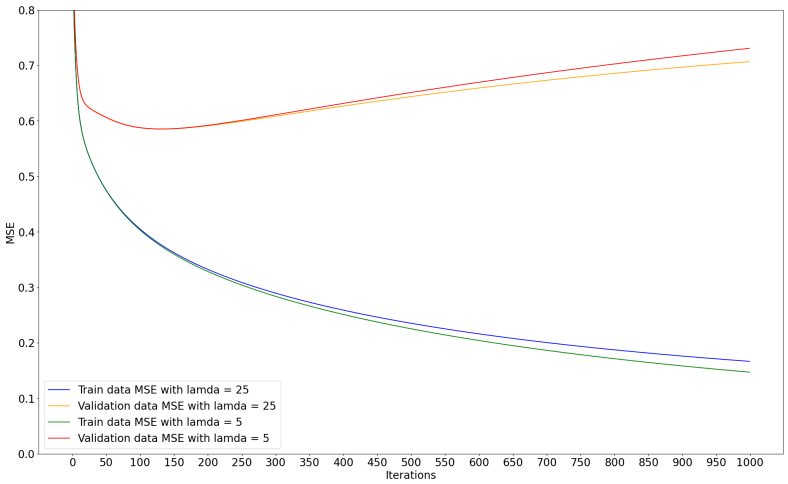

Figure 7: Ridge Regression: MSE analysis for $\alpha = 0.001$

For Test set: I have set the following parameters and stopping criteria for better prediction on test set:

1. Learning Rate: 0.001 (optimal for both train and validation)
2. Lamda (λ): 25 (better performance on optimally large λ)
3. Stopping criteria: Upper cap on maxit = 400, with another stop criteria which will break the loop if the threshold decrease value (7e-07 acc. to validation set)

3 Using Scikit-Learn Library

The MSE and MAE obtained on training the Scikit-Linear Regression Classifier using train data and testing on validation data:

Dataset	MSE	MAE
Validation data	0.9990127265458332	0.8122884749995689

Comparing the performance of 3 models on the validation set: Linear Regression, Ridge Regression and Scikit Classifier, with their respective minimum loss on validation sets (containing approx values):

Model	MSE	MAE
Linear Regression	0.5663	0.5371
Ridge Regression	0.5854	0.5753
Scikit Classifier	0.9990	0.8123

From the above table, it can be concluded that:

- The MSE (as well as MAE) on Linear Regression and Ridge regression are less that that on Scikit-classifier, on the validation data.

- This difference can be explained by the very small error on train data $\approx 10^{-20}$ in the scikit classifier.
- The Scikit classifer fits the data very well into the train dataset, but fails to learn the data better. That is the reason it performs poorer in the validation dataset.
- Whereas the best validation errors of my own implementations of Linear Regression as well as Ridge regression have been adjusted according to the graph, taking the stopping criteria as the point where is the lowest. The scikit classifiers ignores this, and outputs the minimum , at which point the was actually increasing. (acc. to Fig. 2)

4 Feature Selection

MSE and MAE for validation dataset on SeleckBest10 features on Linear Regression

Method	α	threshold	MSE	MAE
SelectBestK(K=10)	0.1	3e-06	0.9682461090218369	0.8101992392977756

MSE and MAE for validation dataset on Select from Model (with Ridge Estimator) features on Linear Regression

FeatureThreshold	α	threshold(reltop)	MSE	MAE
0.09	0.1	3e-06	inf	inf
0.1	0.1	3e-06	1.2695803409194946	0.7298967556962442
0.12	0.1	3e-06	1.2328736027239962	0.9180869114212079

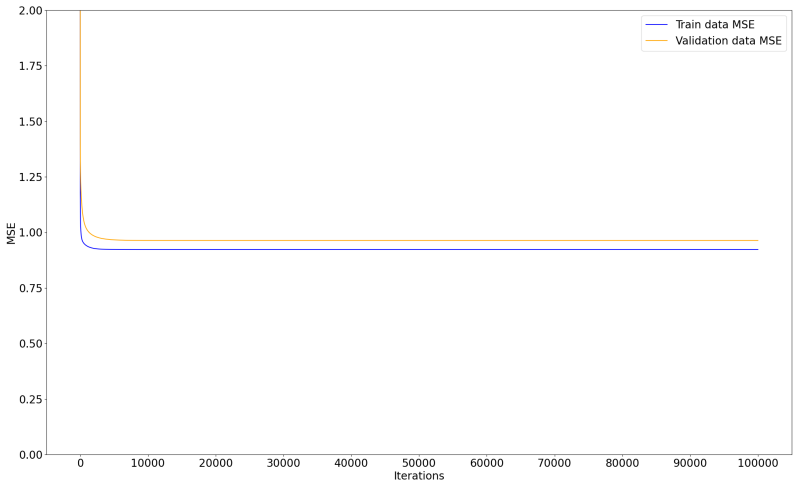

Figure 8: Linear Regression with Best10 Features: with $\alpha = 0.1$, E_{train} and E_{valid} converges to their minimum values at approx 40000 iterations with reltop threshold = 3e-06

Here using Select best k, the variance is quite low, which means − is very low value.

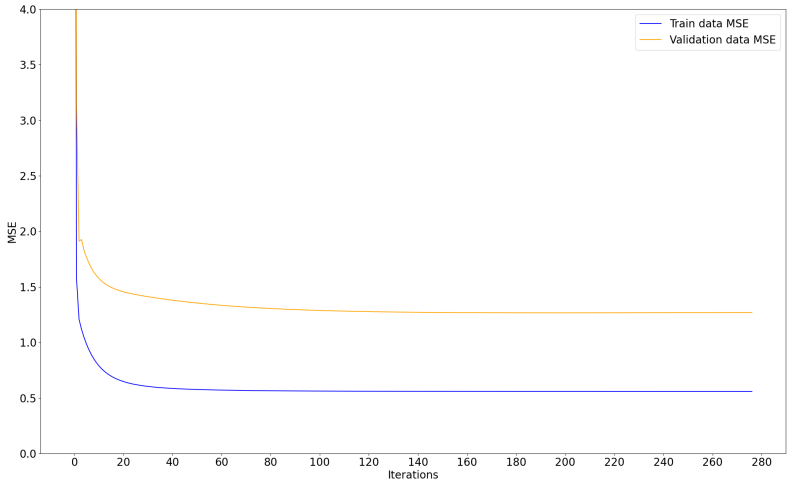

Figure 9: Linear Regression with Select from Model Features: with $\alpha = 0.1$, feature threshold = 0.1

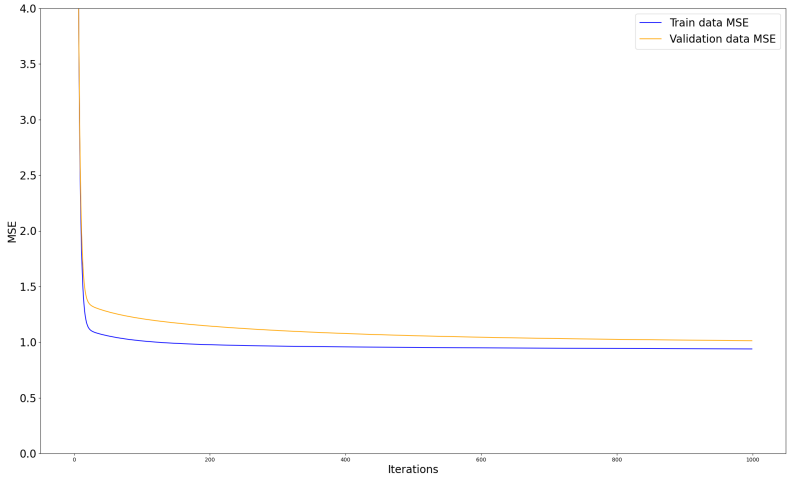

Figure 10: Linear Regression with Select from Model Features: with $\alpha = 0.1$, feature threshold = 0.12

By using Select from model selection function to select function gives us higher variance $E_{in} - E_{valid}$ than select best k.

Stopping criteria = reltop with reltop threshold as 3e-06 (according to observations in validation data) Comparing the performance of Linear regression with best implementations of scikit's SelectfromModel, SelectBestK and all features(implementaion of 3.1), via approx Validation MSE and Train data MSE:

Selected Features	Valid-MSE	Train-MSE
All Features	0.5663	0.3797
Select Best 10	0.9682	0.9235
Select from Ridge Estimator	1.2328	0.5879

Selecting few features out of the 2048 dimensional feature reports more MSE and MAE than the whole feature set. Less dimensions of feature $\implies$ small hypothesis () set $\implies$ decrease in model complexity.
Decrease in model complexity means higher bias (from bias variance analysis), i.e. higher , which results in Underfitting. Out of select best k and selectfromModel, selectfromModel has a higher variance. According to variance:

$$selectbestK < allfeatures < selectfromModel$$

So, selecting a few features out of the dataset makes convergence easier, but bias value (average error) increases.

5 Classification

For this method of multiclass regression, we make a matrix θ, where it contains 8 vectors θ_k, containing the feature vector that calculates the probability of a sample belonging to respective label k.

$P(y-i=k|_i) = e^{(\theta_k)_i^T}/1 + \sum_{r=1}^{p} e^{(\theta_p)_i^T}$...(5.1)

So we need argmax$(P(y_i = k|_i))$, which means argmin$(-\log(P(y^i = k|^i)))$

We need to minimize $-\log(P(y^i = k|^i))$

So we take cost function as (n = number of samples)

$$J(\theta) = -\sum_{i=1}^{n}(\sum_{k=1}^{9} \mathbb{I}(y^i = k)\log(P(y_i = k|_i))) \tag{6}$$

We need to minimise this cost function so we take the gradient $\nabla_(\theta)$

For each θ_{kj}, which is the j^{th} feature of label k vector, We have 2 cases:

1. If $y_i = k$: then $\mathbb{I}(y^i = k) = 1$ for k=y_i and for else it is 0, which means $\frac{\partial J(\theta)}{\theta_{kj}} = -\sum_{i=1}^{n} \frac{1}{P(y_i=k|_i)} \frac{\partial P(y_i=k|_i)}{\partial \theta_{kj}}$

   - Suppose $\sum_{r=1}^{p} e^{(\theta_p)_i^T} = s$, and $e^{(\theta_k)_i^T} = t$, from 5.1, we have $P(y_i = k|_i) = \frac{t}{s}$ we will have $-\frac{\partial J(\theta)}{\theta_{kj}} = \frac{s}{t}\frac{s\partial t/\partial\theta_{kj} - t\partial s/\partial\theta_{kj}}{s^2}$ over all i's
   - $\partial t/\partial\theta_{kj} = e^{(\theta_k)_i^T} \cdot x_i^j = tx_i^j$
   - $\partial s/\partial\theta_{kj} = e^{(\theta_k)_i^T} \cdot x_i^j = tx_i^j$
   - $-\frac{\partial J(\theta)}{\theta_{kj}} = \frac{s}{t}\frac{s\partial t/\partial\theta_{kj} - t\partial s/\partial\theta_{kj}}{s^2} = stx_i^j - t^2x_i^j/s^2 = x_i^j\frac{s}{t}(\frac{st-t^2}{s^2}) = x_i^j(1-\frac{t}{s})$ over all i's

- $-\frac{\partial J(\theta)}{\theta_{kj}} = \sum_{i=1}^{n} x_i^j(1 - \frac{t}{s})$

For the second case, $y_i \neq k$ so we have $\partial t/\partial\theta_{kj} = 0$,

- So the derivative becomes $-\frac{\partial J(\theta)}{\theta_{kj}} = \frac{s}{t}\frac{s\partial t/\partial\theta_{kj} - t\partial s/\partial\theta_{kj}}{s^2} = \frac{s}{t}\frac{0 - t\partial s/\partial\theta_{kj}}{s^2} = x_i^j(0 - \frac{t}{s})$ over all i's
- $-\frac{\partial J(\theta)}{\theta_{kj}} = \sum_{i=1}^{n} x_i^j(0 - \frac{t}{s})$

Taking the 2 cases together, via an identity function we can make $\frac{\partial J(\theta)}{\theta_{kj}} = -\sum_{i=1}^{n}(\mathbb{I}(y^i = k) - \frac{t}{s})x_i^j$

$$\implies \frac{\partial J(\theta)}{\theta_{kj}} = -\sum_{i=1}^{n}(\mathbb{I}(y^i = k) - \frac{e^{(\theta_k)_i^T}}{1 + \sum_{r=1}^{p} e^{(\theta_p)_i^T}})x_i^j \tag{7}$$

Therfore, the update equation becomes $\theta_{kj} \leftarrow \theta_{kj} - \alpha\frac{\partial J(\theta)}{\theta_{kj}}$ (where α is learning rate)

$$\theta_{kj} \leftarrow \theta_{kj} + \alpha\sum_{i=1}^{n}[\mathbb{I}(y^i = k) - \frac{e^{(\theta_k)_i^T}}{1 + \sum_{r=1}^{p} e^{(\theta_p)_i^T}}]x_i^j \tag{8}$$

5.1 Observation:

I have assigned the θ parameters initially randomly using python's numpy.random.rand function. The following table shows the outputs MSE on train and validation data

α	maxit	Train MSE	Valid MSE
0.0001	100	0.2937062937062937	0.95
0.0001	20	0.7132867132867133	0.75
0.001	100	0.46853146853146854	0.8
0.01	40	0.8041958041958042	0.9

5.2 Analysis:

- With increasing iterations, MSE on train data goes on decreasing, but MSE on validation fluctuates a lot. With small learning rate such as $\alpha = 0.0001$ as in figure 11, the training error finally converges at around 100 iterations but validation error starts increasing during those iterations. Optimal fitting of train data lies at around 40 iterations. But the curve is not very smooth as compared to Linear or Ridge regression, as there are multiple labelled hypothesis, so the function may not be *differentiable* at some points
- But with higher learning rates such as $\alpha = 0.01$, there are lot of jumps in the plot of MSE loss for both training as well validation dataset. As the graph is *not differentiable*, (for both train and validation), such learning rates do not work.

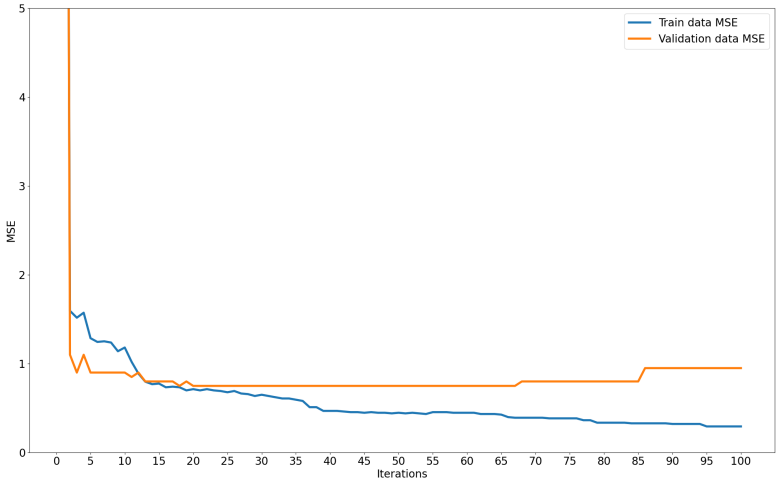

Figure 11: Multiclassifier regression: Train and Validation MSE on running the classifier for 100 iterations with $\alpha = 0.0001$

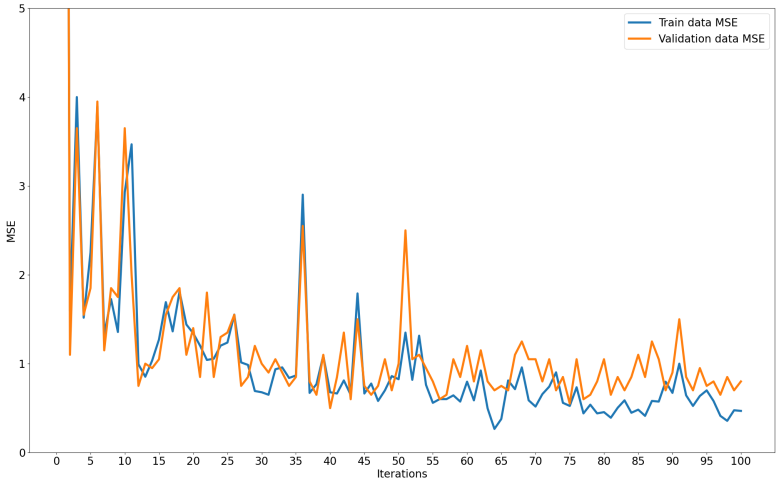

Figure 12: Multiclassifier regression: Train and Validation MSE on running the classifier for 100 iterations with $\alpha = 0.001$

- On comparing with different models: this model shows an error of 0.75 on validation set, as compared to 0.56 in linear model and 0.96 on scikit feature selection. so it performs better than scikit feature selection, but poorer than linear regression. It takes very less iteration for the MSE on validation set to converge with a proper learning rate (here 0.0001)

For test set: I have set the following parameters and stopping criteria for better prediction on test set:

1. Learning Rate: 0.0001 (optimal for both train and validation)

2. Stopping criteria: Upper cap on maxit = 50, here we can not apply the reltop criteria as the function is not differentiable at some points with minute fluctuations too.

6 Visualisation

6.1 Plots for MSE loss:

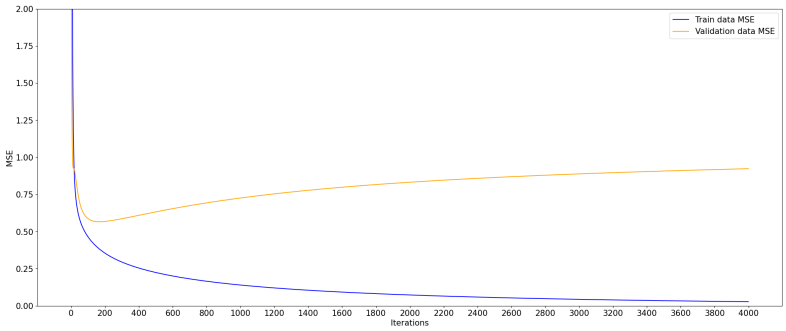

Figure 13: Linear Regression: MSE on train and validation data in $\alpha = 0.001$

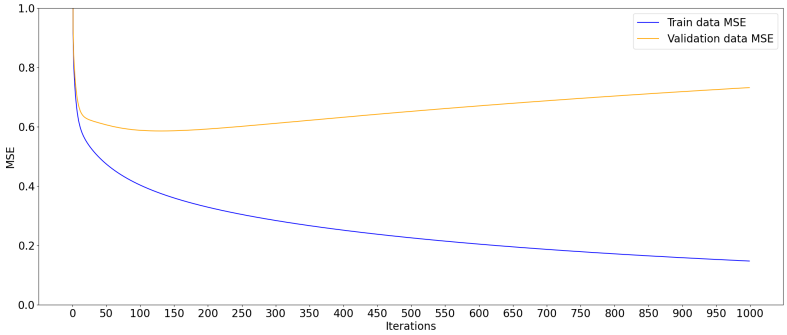

Figure 14: Ridge Regression: MSE on train and validation data at $\lambda = 5, \alpha = 0.001$

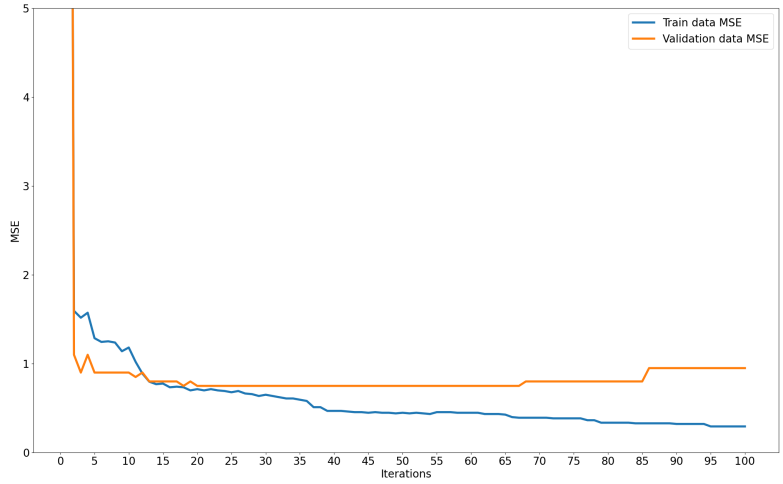

Figure 15: Multiclassifier regression: Train and Validation MSE on running the classifier for 100 iterations with $\alpha = 0.0001$

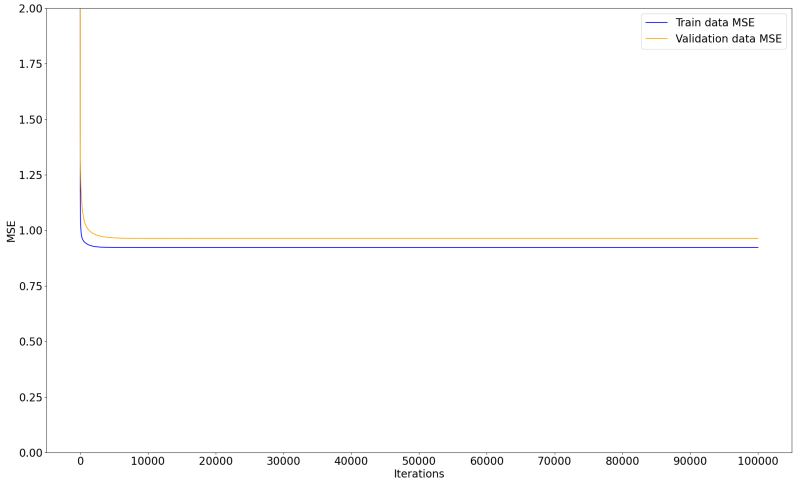

Figure 16: Linear Regression with Best10 Features: with $\alpha = 0.1$, E_{train} and E_{valid} converges

Model	MSE train	MSE valid
Linear regression	0.0276	0.5663
Ridge Regression	0.088	0.5854
Scikit Feature Selection	0.5879	0.9682
Multiclass Classifier	0.293	0.75

Comparision:

- Best fitting of the train Data is possible in Linear (and Ridge) regression. So the bias is least in case of linear regression. Order of E_{train} in the models:

- Best performance on validation data is shows by linear regression
- Minimum iterations to achieve convergence is there in Multiclass classifier, but each iteration takes longer time, as in each iteration, we need $k_{labels} \times n_{features}$ to get updated.
- The variance is least by using less features in Scikit feature selector, as the model is a simple one (hypothesis set is smaller) which results in higher bias (higher E_{in}) but less variance ($E_{in} - E_{valid}$)

6.2 Normalisation of Features:

Normalised x_i' is calculated using the equation $x_i' = \frac{x_i - \mu_i}{\sigma_i}$, where μ_i and σ_i are the mean and standard deviations of $x_i s$ of all samples.
Performing Linear Regression on normalised dataset yields the following results, only learning rate = 0.001 results in convergence of errors, other α make the error overshoot/-diverge. Hence only $\alpha = 0.001$ results are noted:

α	maxit	Train MSE	Validation MSE
0.001	1000	0.5164402477232332	2.6419769946047524
0.001	2000	0.009427190471665159	1.8439750804098054

The plots of MSE loss for train and validation data over iterations:

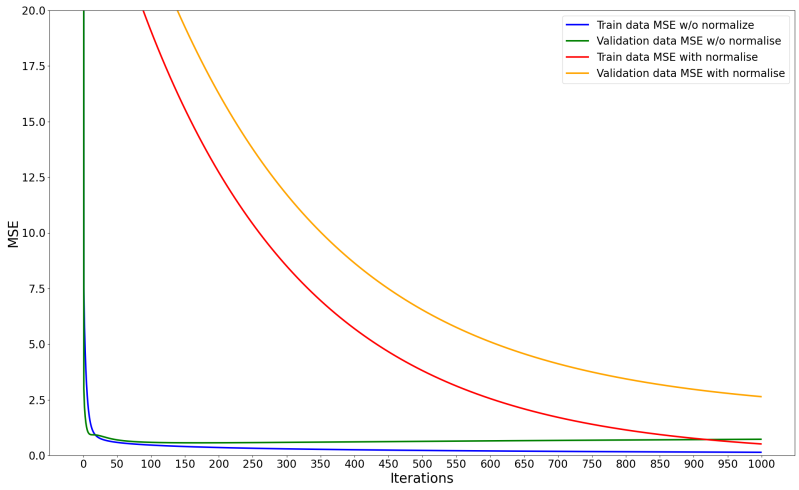

Figure 17: MSE of both train and validation converges faster without normalising

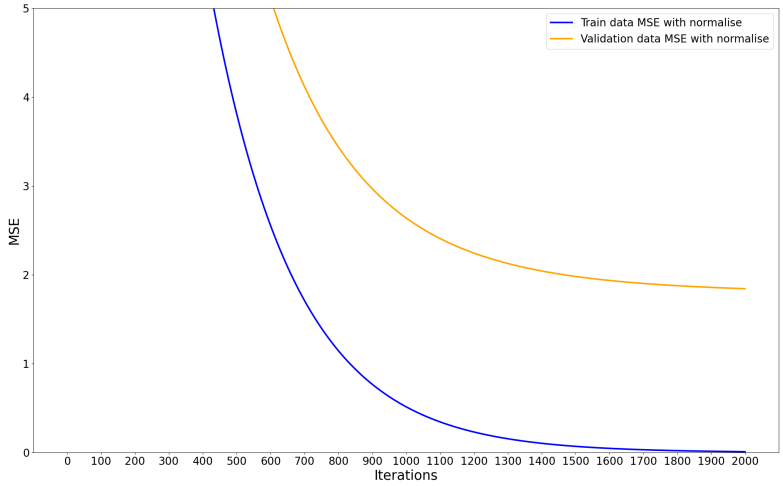

Figure 18: MSE of both train and validation data with $\alpha = 0.001$, still higher after 2000 iterations

So, this shows that on normalising the features, the error converges very slowly. It takes much more iterations for the train error as well as validation error to converge. Without normalising, linear regression converged validation MSE within 400 iterations with $\alpha = 0.001$ to ≈ 0.6, but with normalising the MSE of validation set is still 1.84 after 2000 iterations

6.3 Varying Train data size:

The following MSE were observed on converging the errors for different train data sizes, with maximum iterations run as 500:

Train size	Train MSE	Validation MSE
1	0.22471193651058025	0.6324096569178523
3/4	0.16701530849650476	0.5475047596416959
1/2	0.10389710691017466	0.5240187301842355
1/4	0.010878084745051596	1.0564612681658536

Observations and Analysis:

- On decreasing the size of train data, with the same number of iterations MSE on train data decreases (as there are less data points now, so it is easier to fit into the data better into the weights).
- Validation MSE for data size = 1/4 is much more than the other train data sizes. This is because much less data points make − bound larger (this bound $E_{in} - E_{out}$ depends on $\sqrt{1/N}$ acc. to Hoeffding's Bound where N is the size of train dataset, so less N makes bound larger). Even if is less it makes more, learning doesnt happens well. This makes the variance more even if E_{in} decreases/approaches more to 0

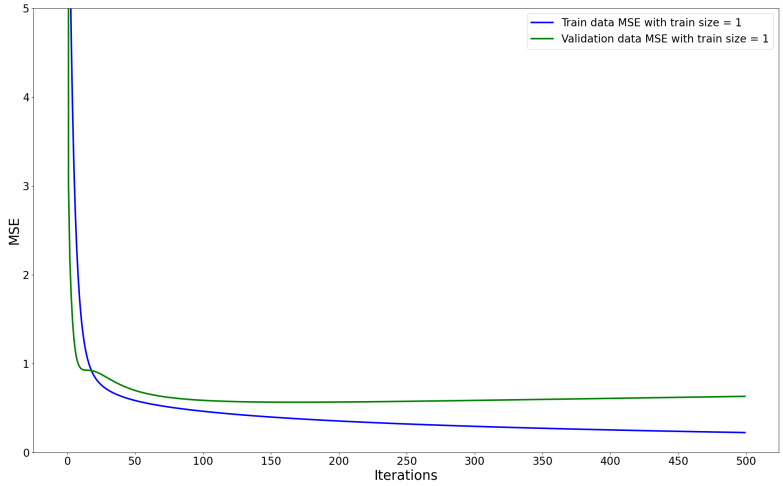

Figure 19: Train and Validation MSE over 500 iterations on train data size = 1 with $\alpha = 0.001$

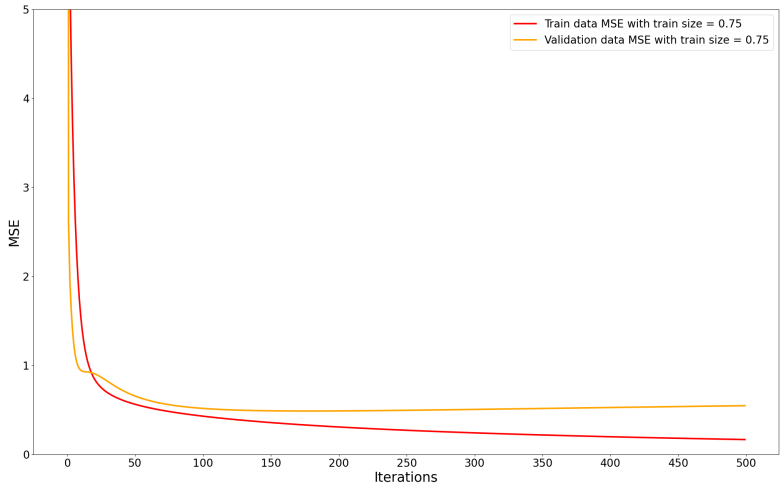

Figure 20: Train and Validation MSE over 500 iterations on train data size = 0.75 with $\alpha = 0.001$

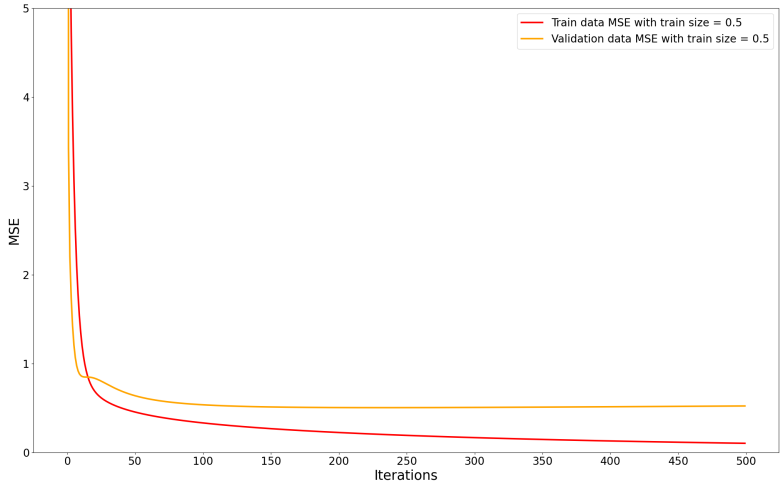

Figure 21: Train and Validation MSE over 500 iterations on train data size = 0.5 with $\alpha = 0.001$

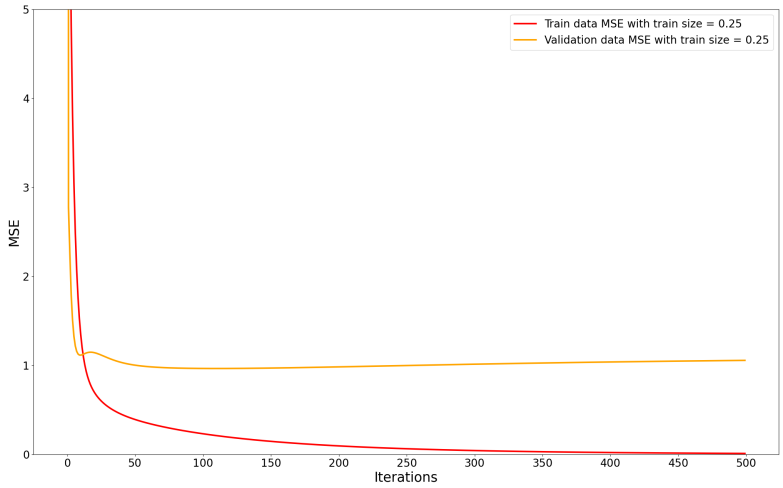

Figure 22: Train and Validation MSE over 500 iterations on train data size = 0.25 with $\alpha = 0.001$

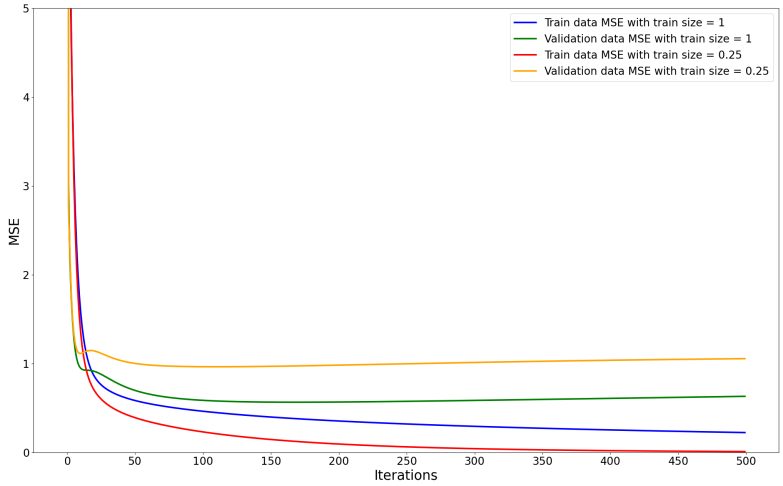

Figure 23: Comparative graph on Train and Validation MSE over 500 iterations on train data size = 1 and 0.25 with $\alpha = 0.001$. Shows the increase in validation error (also variance) due to less train size (N) and larger bound on $E_{in} - E_{valid}$

6.4 Dividing Train data into 2 halves:

The following table shows the mean absolute difference between Model 1 and Model 2 on the **predictions on set1 and set2 of train data respectively**, when Model1 is trained on one half of the dataset and Model 2 is trained on other half. (Both using Linear regression and Ridge regression)

Model	MAE between both halves
Linear Regression (3.1)	0.23399111593874644
Ridge Regression (3.2)	0.503221019908088

The following table shows the mean absolute difference between Model 1 and Model 2 on the **predictions on validation set**, when Model1 is trained on one half of the dataset and Model 2 is trained on other half. (Both using Linear regression and Ridge regression)

Model	MAE between both halves
Linear Regression (3.1)	1.4792945824195727
Ridge Regression (3.2)	1.2633092086827136

Analysis:

- MAE is a measure of variance of the model. So higher MAE can depict that the variance of the model is higher (*here, Ridge has less variance on validation set, which means the predicted hypothesis would be closer to each other*)

6.5 MSE loss v/s Score values:

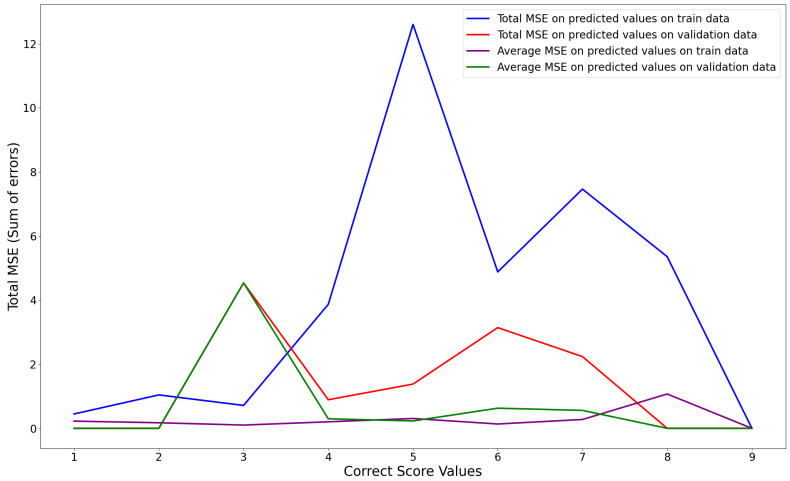

Figure 24: Plot showing Total MSE and Average MSE vs correct score values

Analysis:

- The total MSE plot shows peaks for some score values (both train and validation predictions) *For example for score values = 5, 7, total MSE is very high which means more wrong predictions for those scores, which may account to more correct scores corresponding to these labels*
- The average MSE plot shows less peak and is more uniform for train data (*almost 0 average MSE for nearly all scores*)

6.6 MSE loss vs Number of Features

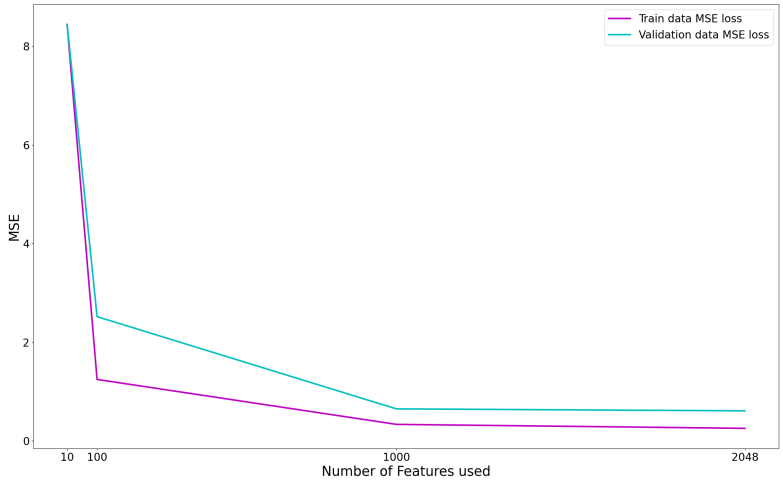

Figure 25: MSE loss over different number of features by using SelectBestK library of scikit, number of iterations = 300 and $\alpha = 0.001$

Analysis:
(On running same number of iterations with same learning rate)

- With increase in number of features taken for training: **Train MSE decreases**, **Validation MSE also decreases**
- The difference between Train MSE and Validation MSE ($E_{in} - E_{valid}$) also decreases due to increase in N (as per Hoeffding's bound)

7 Generalisation Analysis

Here, as we do not have validation set with us, provided only train set, we have to use **maxit** stopping criteria and train the data with converging

- For dimension = 2: $\alpha = 0.1$, number of iterations for which converges = 1524
- For dimension = 5: $\alpha = 0.1$, number of iterations for which converges = 1661
- For dimension = 10: $\alpha = 0.1$, number of iterations for which converges = 2640
- For dimension = 100: $\alpha = 0.01$, number of iterations for which converges = 100000 (approx) (As error diverges for higher learning rates)

Dimension	E_{out}(test)	E_{in}(train)	$E_{out} - E_{in}$
2	1.3915431258094955	0.7546412112262914	0.6369019145832041
5	1.0141969480101949	0.666023451458872	0.3481734965513229
10	1.133140548865089	0.8307547324943876	0.3023858163707014
100	5.747276900988118	0.16133199201514786	5.58594490897297

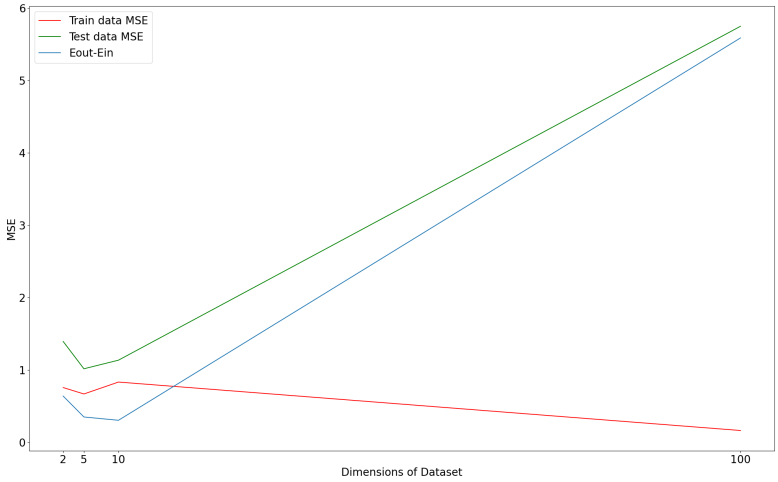

Figure 26: Plot of − for all 4 dimensions of datasets

Analysis:

- On increase in the dimension d of the data:
  - decreases, as we have a complex model with a large hypothesis set, we can fit the data better, which in facts takes more iterations to converge and results in overfitting the data
  - increases, as we have overfitted the data by converging , but as the bound is $O(\sqrt{\frac{d}{N}})$, becomes distant from .
  - − increases, as the bound is $O(\sqrt{\frac{d}{N}})$, so − proportional to $O(\sqrt{d})$.

8 One vs Rest Method

(Bonus Part) In One vs Rest method, we have to draw 9 boundaries between the datasets, each boundary representing the hypothesis that separates one label from all other labels. These 9 boundaries would be trained by treating y values as 1 if it corresponds to index, else 0.
The logistic regression hypothesis function for prediction is

$$h_i(\mathbf{x}_i) = \frac{1}{1 + e^{w_k^T \mathbf{x}_i}}$$

So we make a 2d matrix = $[w_{kj}]_{9\times m}$ where kth vector w_k represents hypothesis that separates label k from others. To make this, we have the update equation as:

$$w_{kj} \leftarrow w_{kj} + \frac{\alpha}{N}(l_i - h_i(\mathbf{x}_i))x_{ij} \tag{9}$$

We have true label l_i is 1 if $y_i == k$ and 0 if not.

So l_i can be replaced by identity function:

$$w_{kj} \leftarrow w_{kj} + \frac{\alpha}{N}(\mathbb{I}(y_i = k) - h_i(\mathbf{x}_i))x_{ij} \tag{10}$$

8.1 Observations and Analysis:

The following accuracy and MSE was observed on running *maxit* iterations for each boundary, with the given α, by training with train set and testing on validation dataset.

α	maxit	MSE	Accuracy(in %)
0.01	100	0.7	60
0.01	300	0.75	55

On training the dataset for 300 iterations each, testing on validation set gives results:

- True values: [3, 6, 4, 4, 4, 7, 7, 5, 5, 5, 5, 7, 6, 6, 5, 5, 6, 6, 7, 8]
- Predicted val: [5, 6, 5, 4, 5, 7, 6, 6, 5, 5, 5, 5, 6, 6, 5, 5, 6, 7, 6, 7]

As we can see from the table, MSE increases as well as accuracy decreases on increasing number of iterations to 300 $\implies$ overfitting of train data. So it doesnt perform well on validation set.

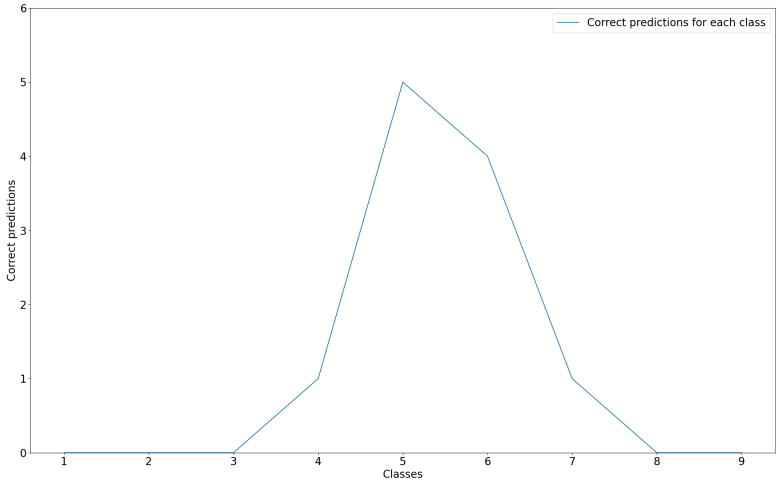

Figure 27: Plot showing correct predictions on validation set by One vs Rest method on 300 iterations