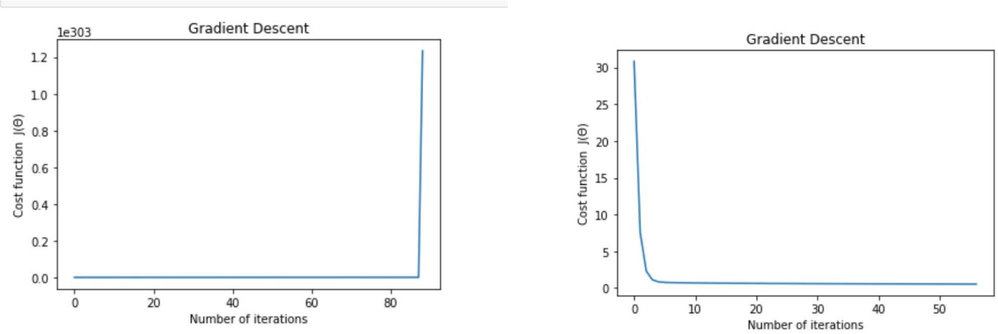
COL341 - Assignment 1

Valaya - 2019MT10731

February 12, 2023

1 Linear Regression

We construct our gradient descent algorithm such that for every iteration, it evaluates the relative decrease in cost function over validation data and breaks if this goes below a specified threshold. For learning rates 0.1 and 0.01, the MSE losses over training and validation data are diverging. The number of iterations are fixed to 200. A similar graph is obtained for $\alpha = 0.01$. For both these learning rates, MSE and MAE value for validation data blow up.



(a) MSE of training data, $\alpha = 0.1$ (b) MSE of training data, $\alpha = 0.001$

Figure 1: MSE vs iterations for different learning rates

For $\alpha = 0.001$, we obtain a converging graph for MSE. The table summarizes the MSE and MAE values for training and validation data.

$\alpha = 0.01$	Training	Validation
MSE	0.541	0.595
MAE	0.579	0.588

Figure-2 shows that around the value 0.003, the MSE over the validation data reaches a minimum value for a fixed number of iterations (200). Hence 0.029 is selected as our learning rate for the tuned linear regression model. The gradient descent algorithm returns if the relative decrease in the cost of validation data drops below a threshold at a given iteration which is determined to be 0.01. The estimated iteration came out to be 45. This means, the relative decrease in MSE on validation data after 45th iteration was very little. Hence, we plot a graph for i = 3 to 50 and observe MSE over training and validation data in section 3.6

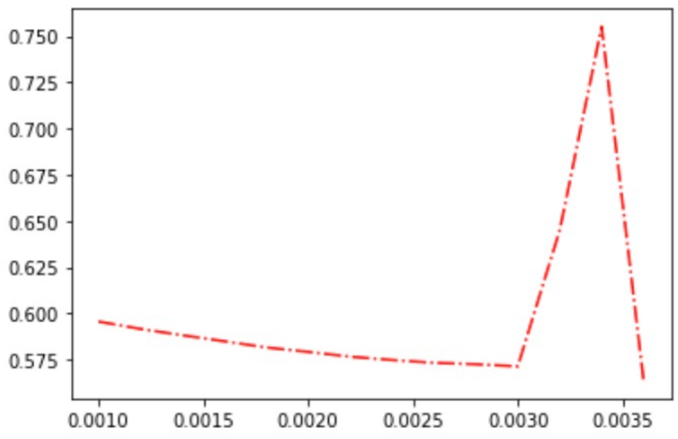


Figure 2: MSE of Validation data vs. Learning rate

2 Ridge Regression

The equation for the cost function $J(w)$for Ridge regression can be written as follows. The bias term w_0 is not regularised in this cost function.

$$(y - Xw)^T(y - Xw) + \lambda w^T w - \lambda {w_0}^2$$

The derivative of this cost function with respect to w comes out to be:

$$\nabla J(w) = 2X^T(Xw - y) + 2\lambda w - 2\lambda[w_0, 0...0]^T$$

Hence, the update function is given by -

$$w = w - \alpha \times \nabla J(w)$$

However, I've not used gradient descent for my ridge regression algorithm. Rather I've just set the gradient to 0, found the value of $w = (X^T X + \lambda I)^{-1} X^T y$ and have evaluated the corresponding loss. The following tables summarise my findings. For the parameter value equal to 25, the validation MSE/MAE loss is lesser than when the parameter value is

$\lambda = 5$	Training	Validation
MSE	0.0109	0.994
MAE	0.0788	0.801

$\lambda = 25$	Training	Validation
MSE	0.079	0.816
MAE	0.219	0.697

set to 5.

3 Linear Regression using Scikit-learn

On using scikit-learn library's classifier on our training data, we get the following results for MSE and MAE over validation data:

$$MSE = 1.0246616206131949$$

$$MAE = 0.8320303303899533$$

3.1 Comparison with linear regression model

This model's MSE and MAE losses are higher than the linear regression model in section 3.1. A possible reason for this could be the higher value of the learning rate used by the sci-kit-learn classifier.

3.2 Comparison with ridge regression model

The MSE and MAE losses for this model are higher than the ridge regression model in section 3.2. This is consistent with theoretical findings of ridge regression being an improvement over linear regression as it adds a penalty term to the cost function and regularises.

4 Feature-Selection

4.1 Select K best

First, we use selectkbest method to extract 10 features. In order to reduce the MSE loss, we increase the number of iterations to 1000. Our findings are as follows - On comparing this with the model we constructed using all the features, we can

	Training	Validation
MSE	1.205	1.919
MAE	0.848	1.089

comment that even though selecting a subset of features certainly increases the computational speed, however in terms of accuracy, looking at the MSE/MAE losses, it can be said that the model using all the feautures worked better.

4.2 Select from model

For SelectFromModel method also, we increase the number of iterations and get the following results. This model shows worse results than the k-best model. Consequently, as compared to the linear regression model which used all features, it is giving more MSE/MAE loss.

	Training	Validation
MSE	1.128	2.869
MAE	0.840	1.123

5 Classification

We use maximum likelihood estimation by minimising the following cost and then use gradient descent to find its minima. We find the gradient descent with respect to each θ_r

$$C(\theta) = -\frac{1}{N} \sum \log\left(P(y_i \mid x_i, \theta)\right)$$

$$= -\frac{1}{N} \sum \log \frac{e^{\theta_{y_i}^T x_i}}{1 + \sum_{p=1}^{8} e^{\theta_p^T x_i}}$$

$$\nabla_{\theta_r} C(\theta) = \frac{1}{N} \sum \left(\frac{e^{\theta_r^T x_i}}{1 + \sum_{p=1}^{8} e^{\theta_p^T x_i}} \right) x_i - \frac{1}{N} \sum x_i$$

update eqn :→

$$\theta_r = \theta_r - \alpha \nabla_{\theta_r} C(\theta)$$

Figure 3: Cost function and update function

5.1 Linear Regression

For the model implemented in 3.1, we fix $\alpha = 0.0029$, and relative tolerate at 0.001. The maximum iterations that the algorithm goes through for the given set is around 45. We plot the MSE for training and validation data for this model and observe how it changes with changing number of iterations.

We can observe that the MSE for validation is more than that for the training data which makes sense given that our model is trained using the latter hence it is bound to give less error when evaluated on the same.

5.2 Select K-Best

The relative decrease in MSE value of validation data never drops below the specified threshold in our Kbest model. Therefore we plot the points for iteration = 3 to iteration = 1000. These values are greater than the MSE values for model 3.1.

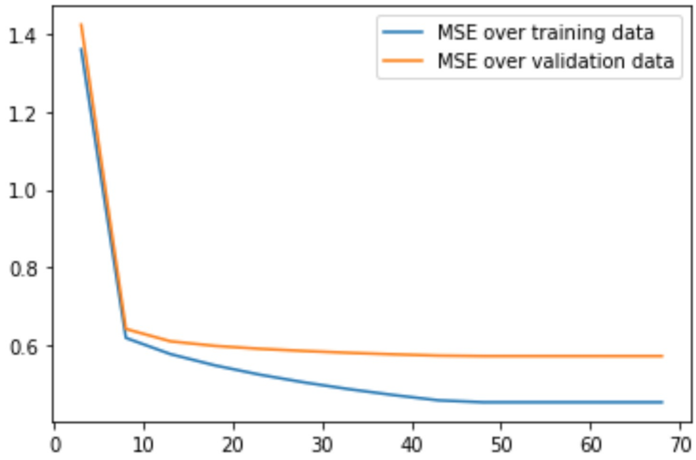


Figure 4: MSE of Validation and training data for linear regression

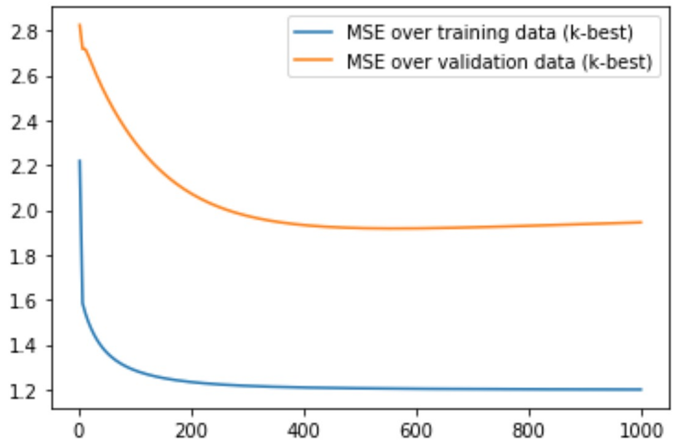


Figure 5: MSE of Validation and training data using K-best Model

5.3 Select From Model

The relative decrease in MSE value of validation data never drops below the specified threshold in our model. Therefore we plot the points for iteration = 3 to iteration = 2000. These values are greater than the MSE values for model 3.1.

5.4 Data Normalisation

Performing section 3.1 on normalised data gives similar results as before. We get diverging MSE/MAE losses for learning rate equal to 0.1 and 0.01. For learning rate = 0.001, we get converging MSE/MAE values. The following table tabulates the results for the same. In order to get the optimum value of the learning rate, we plot a graph of MSE on validation data

$\alpha = 0.001$	Training	Validation
MSE	1.426	2.029
MAE	1.192	1.301

Figure 6: MSE of Validation and training data using Select-from-Model

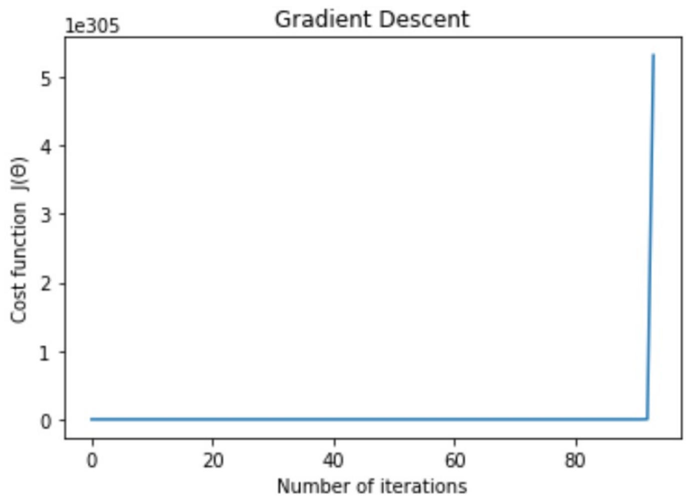


Figure 7: MSE of training data, $\alpha = 0.1$

vs different learning rates. We observe that for alpha = 0.0044, we are getting a minimal MSE. We find that by setting the learning rate to this value, our algorithm jumps out for the 1378th iteration since the relative MSE decrease starts going below the threshold value at that iteration. The following table summarises the results for this optimal learning rate.

$\alpha = 0.0044$	Training	Validation
MSE	0.00386	0.534
MAE	0.06193	0.555

Graph is plotted for approx 1400 points as follows -

5.4.1 Sampling of normalised data for 1/4

Even though the MSE/MAE losses for training are very small, they are significantly high over the validation data. The following table summarises our results for the most optimum value of learning rate (0.002).

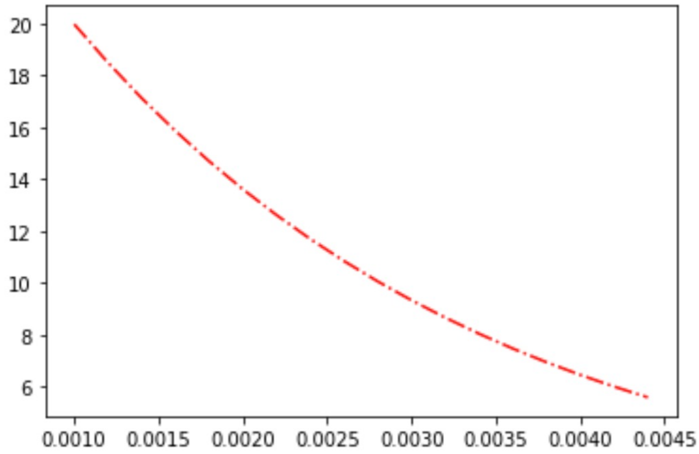


Figure 8: MSE of validation data vs learning rates

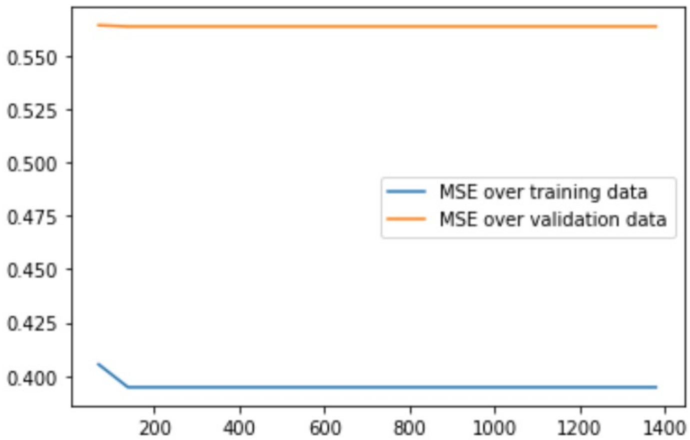


Figure 9: MSE of validation data and training data

$\alpha = 0.002$	Training	Validation
MSE	0.005	42.334
MAE	0.0569	5.277

5.4.2 Sampling of normalised data for 1/2

We have similar findings for 1/2 when using a learning rate - (0.002).

$\alpha = 0.002$	Training	Validation
MSE	0.08	48.02 MAE
0.23	4.831	

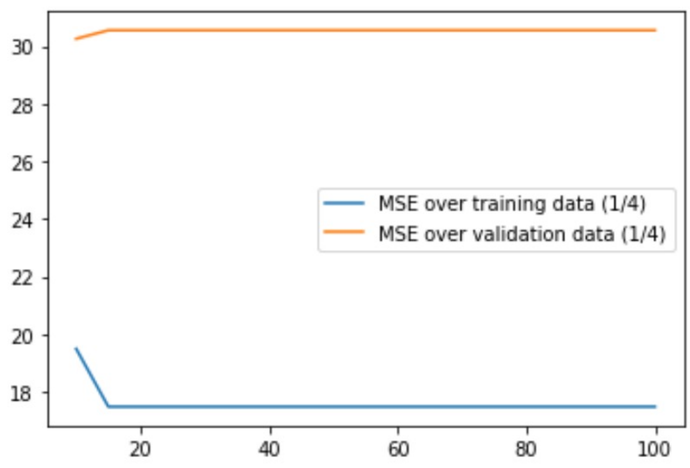


Figure 10: MSE of validation data and training data, using 1/4th of training data

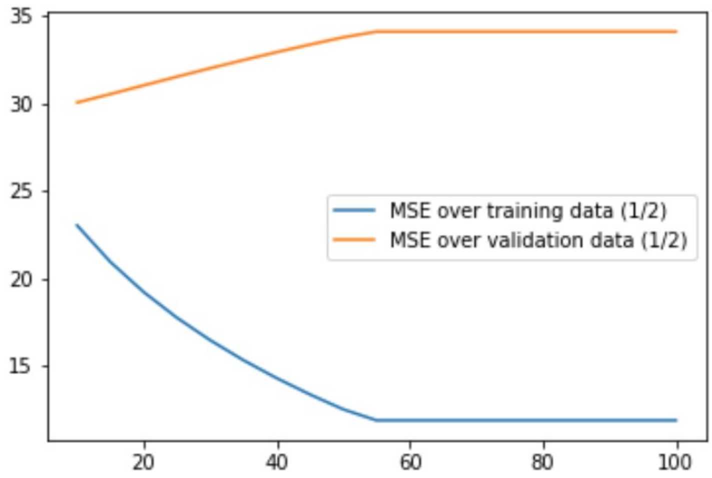


Figure 11: MSE of validation data and training data, using 1/2th of training data

5.4.3 Sampling of normalised data for 3/4

Sampling 75 per cent of the training data gives better results on MSE validation loss than the previous two, as summarised in the table below. Even though the losses for the training data are slightly higher, it can be explained by the fact that due to the small training set in the previous two cases, due to overfitting, those losses were coming out to be small.

$\alpha = 0.004$	Training	Validation
MSE	0.933	22.5
MAE	0.857	3.73

5.5 Dividing data

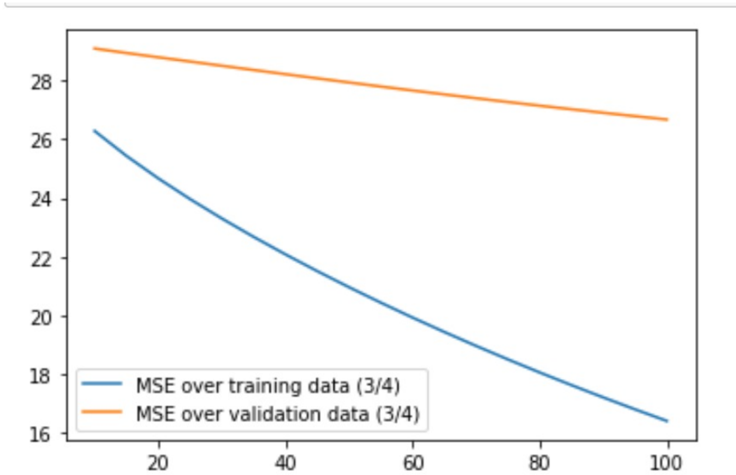


Figure 12: MSE of validation data and training data, using 3/4th of training data

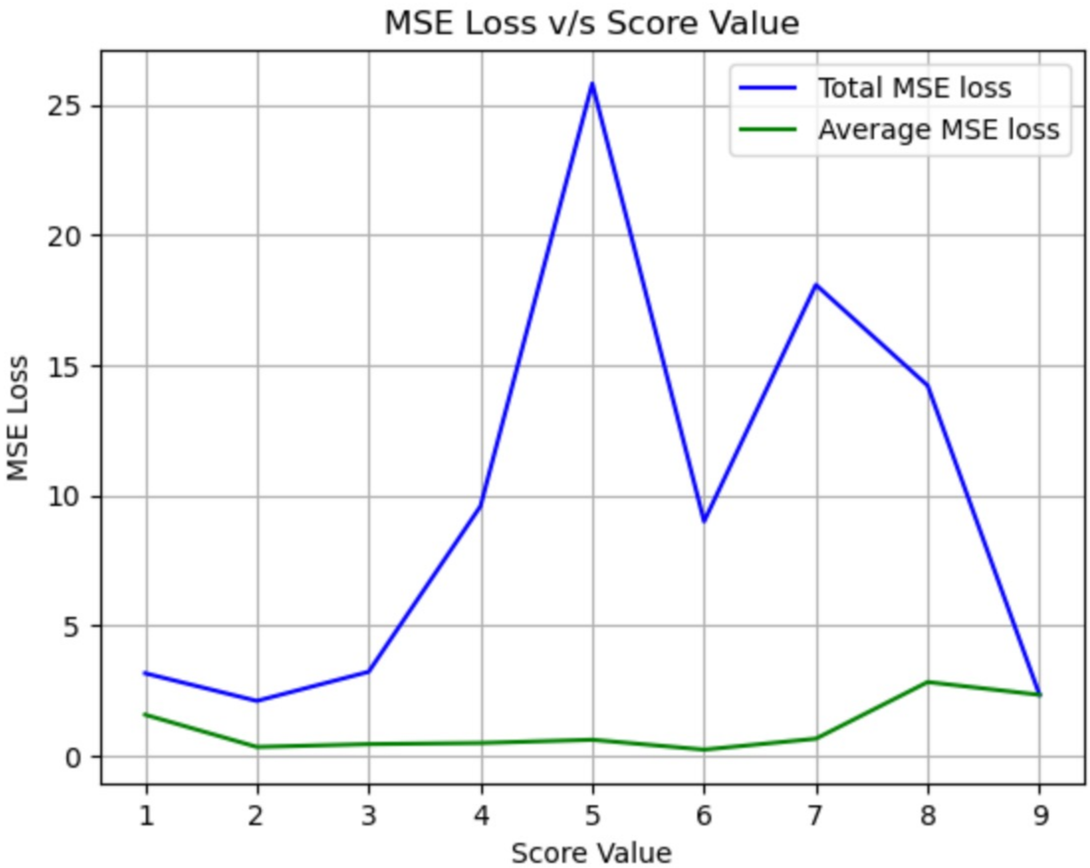


Figure 13: MSE loss vs Score values

```
y_b1_est

0     2.927815
1     4.739399
2     5.337741
3     4.389242
4     4.365176
5     4.313331
6     7.301439
7     6.217863
8     5.392599
9     4.548955
10    4.780665
11    4.971322
12    6.315437
13    5.058005
14    6.565401
15    4.317198
16    5.531044
17    5.259034
18    6.219442
19    5.706438
20    7.743937
dtype: float64
```

```
y_b2_est

0     2.114852
1     5.266587
2     5.625679
3     4.600322
4     4.039794
5     4.527410
6     6.711016
7     5.948948
8     5.374596
9     4.964772
10    5.225866
11    5.131813
12    6.828501
13    5.199316
14    6.410344
15    4.337493
16    6.074321
17    4.797188
18    6.480031
19    5.187194
20    7.463245
dtype: float64
```

```
dff

0     0.812963
1     0.527188
2     0.287938
3     0.211081
4     0.325382
5     0.214080
6     0.590422
7     0.268915
8     0.018003
9     0.415818
10    0.445201
11    0.160491
12    0.513064
13    0.141311
14    0.155057
15    0.020295
16    0.543278
17    0.461846
18    0.260589
19    0.519245
20    0.280691
dtype: float64
```

Figure 14: predictions using training set 1 and training set 2 and their difference (linear regression)

	y_b1_est_ridge	y_b2_est_ridge	dff_ridge
0	1.586032	2.086383	0.500351
1	4.554787	5.985410	1.430624
2	5.391144	5.566006	0.174862
3	4.654285	5.443859	0.789574
4	4.285786	4.462602	0.176816
5	4.386402	5.013574	0.627172
6	7.527569	6.851723	0.675845
7	6.031235	5.878055	0.153180
8	4.969919	4.455827	0.514093
9	4.576925	4.894759	0.317834
10	4.552285	5.469600	0.917315
11	5.456708	4.857469	0.599239
12	6.063716	6.834795	0.771079
13	4.292570	4.852205	0.559635
14	7.030020	5.952962	1.077059
15	4.342704	4.272604	0.070099
16	4.752258	6.115655	1.363397
17	5.956825	4.972994	0.983831
18	6.311183	6.123791	0.187393
19	5.552896	5.896403	0.343507
20	8.243487	8.269698	0.026211
dtype:	float64	float64	float64

Figure 15: predictions using training set 1 and training set 2 and their difference (ridge regression)