Reza Mahroo EE7600: Machine Learning Project 4

In this project neural network is implemented for “load digits” and “load diabetics” data sets, and the effect of hyperparameters below on the performance of the model is studied.

1. Implement the neural network model for the digits dataset.
2. Plot the mis-classification rate and loss and compare in terms of
3. the hyper-parameters listed above.
4. Provide brief explanations for your results.

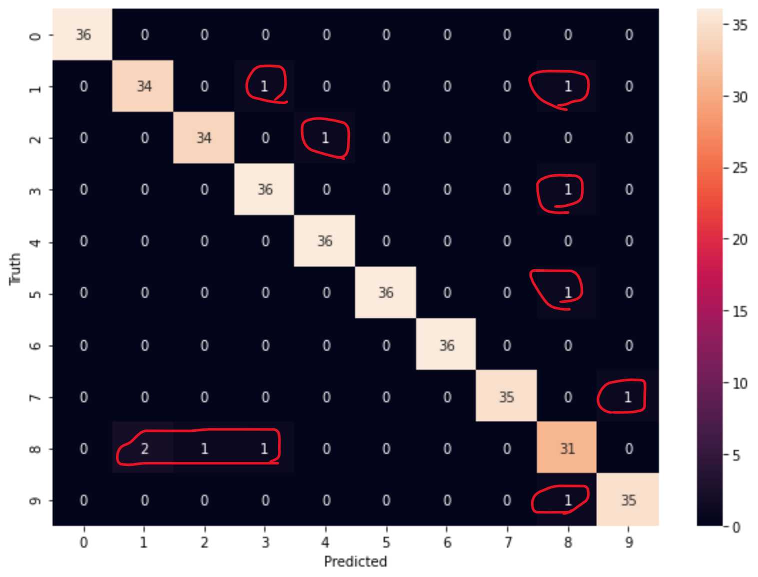
First, a very simple neural network with one dense hidden layer (32 neurons) and one output layer (10 neurons) is implemented. The hyper-parameters are set randomly so that the activation function in the hidden layer is set to “tanh,” and “softmax” for the output layer. “RMSprop” is selected as the optimizer and “sparse_categorical_crossentropy” for the loss function. Momentum and Regularization are not considered yet. Here are the results when the epoch is set to 50:

Elapsed time is 2.7 seconds.

Zero percent training error and 2.85 percent test error.

The value of the final loss is 0.008.

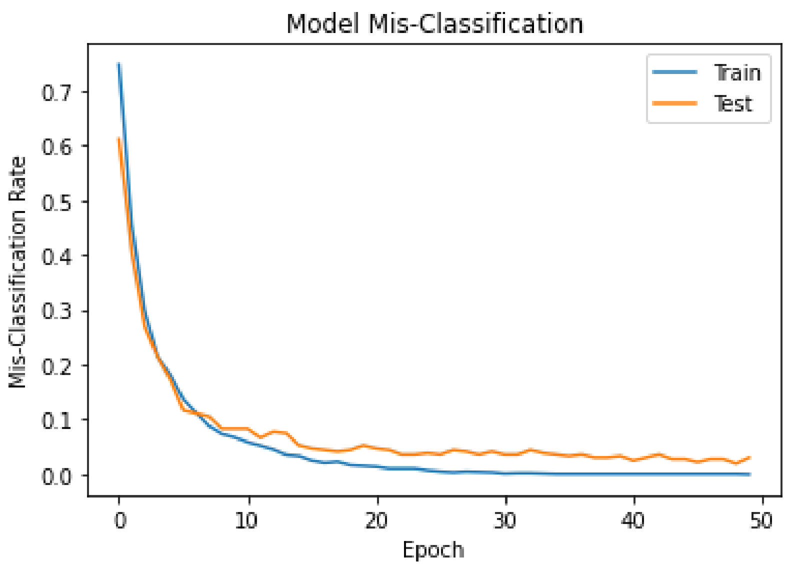
Confusion matrix for test set:



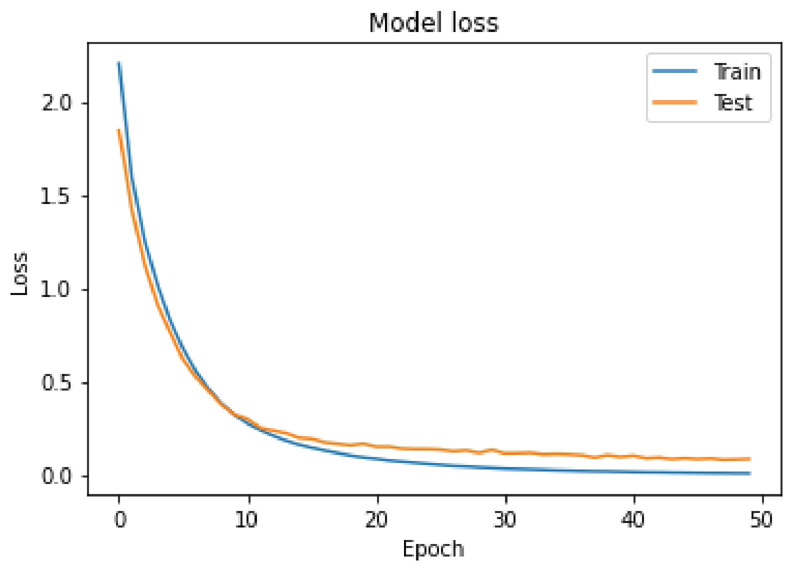
Truth \ Predicted	0	1	2	3	4	5	6	7	8	9
0	36	0	0	0	0	0	0	0	0	0
1	0	34	0	1	0	0	0	0	1	0
2	0	0	34	0	1	0	0	0	0	0
3	0	0	0	36	0	0	0	0	1	0
4	0	0	0	0	36	0	0	0	0	0
5	0	0	0	0	0	36	0	0	1	0
6	0	0	0	0	0	0	36	0	0	0
7	0	0	0	0	0	0	0	35	0	1
8	0	2	1	1	0	0	0	0	31	0
9	0	0	0	0	0	0	0	0	1	35

Truth
Predicted
35
30
25
20
15
10
5
0

Misclassification:



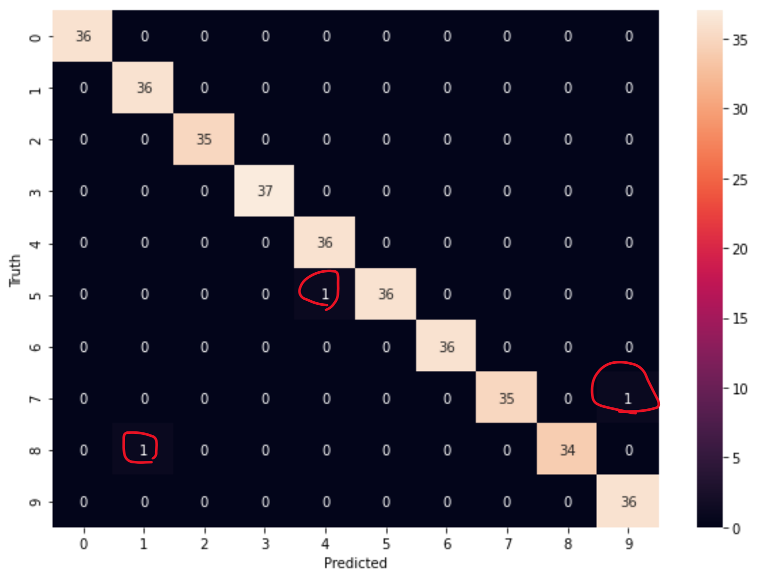
Loss:



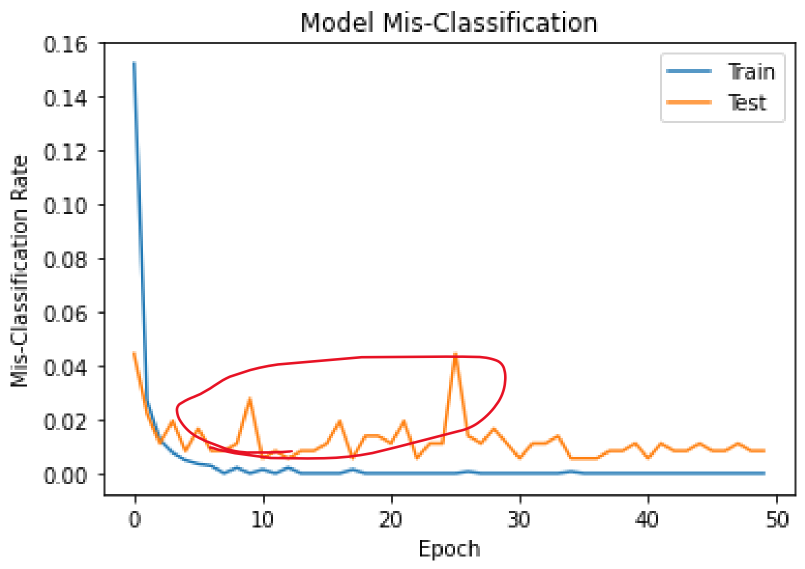
1. Number of the hidden unit

To see the effect of the number of neurons in the hidden layer, I increased the number of them to 500, and here are the results:
Elapsed time is 3.97 seconds.
Zero percent training error and 0.8 percent test error.
The value of the final loss is 0.041.

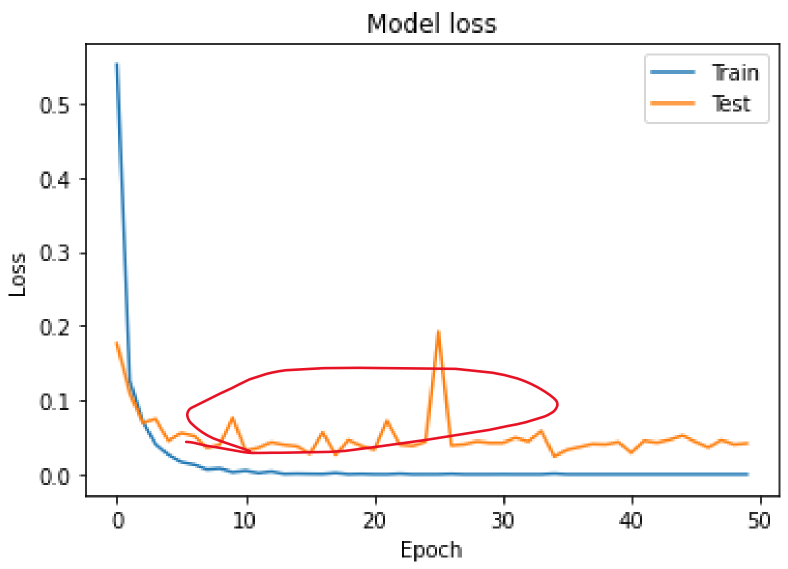
Confusion matrix for test set:



Misclassification:

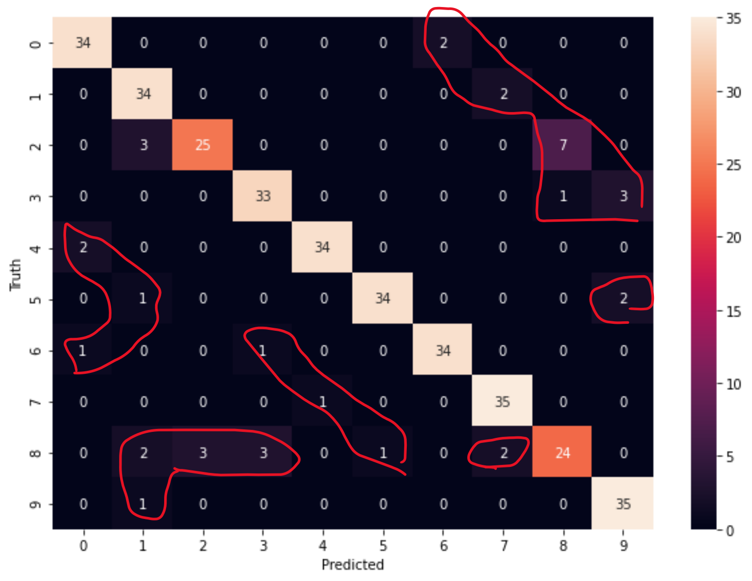


Loss:



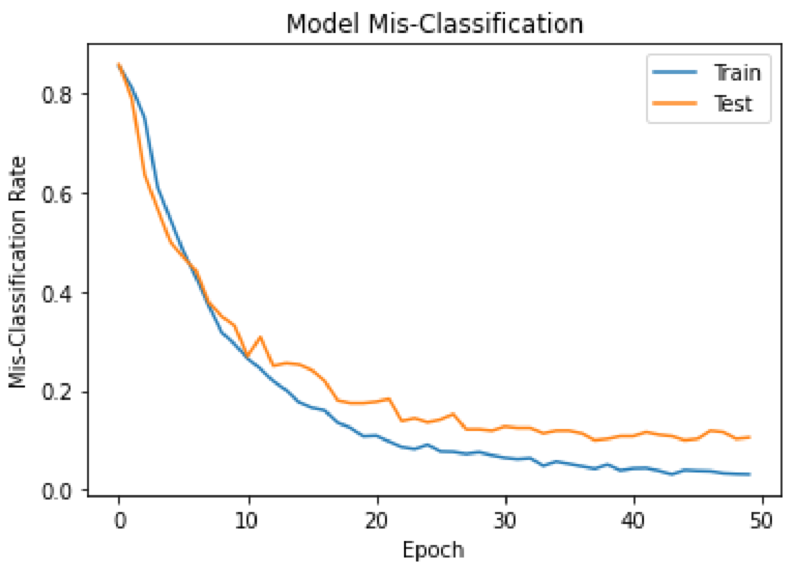
As the mis-classification and loss plots show, this case is prone to overfitting. However, it has not significantly happened in this case. Generally speaking, choosing too many neurons may lead to overfitting, high variance, and increases the time it takes to train the network. Whereas if fewer neurons are chosen, it will lead to underfitting and high statistical bias.
I also increased the number of hidden layers to 10 and received the following results:

Confusion matrix for test set:

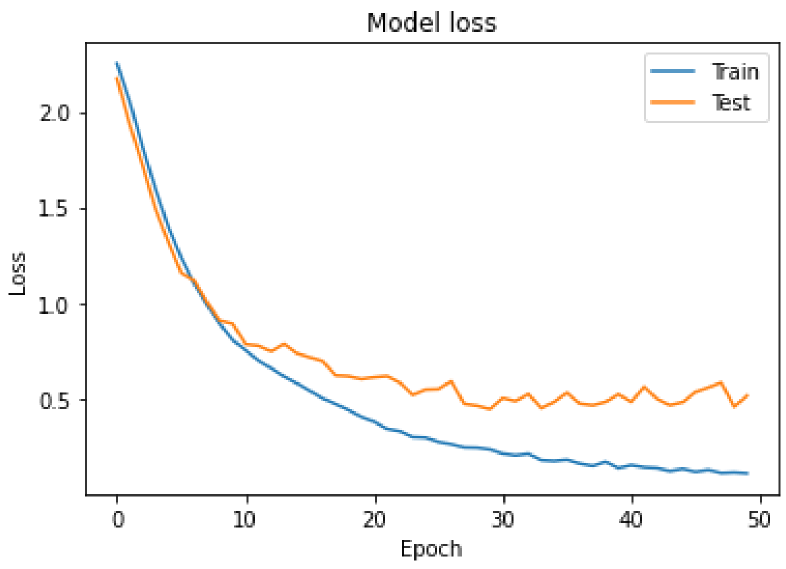


Truth \ Predicted	0	1	2	3	4	5	6	7	8	9
0	34	0	0	0	0	0	2	0	0	0
1	0	34	0	0	0	0	0	2	0	0
2	0	3	25	0	0	0	0	0	7	0
3	0	0	0	33	0	0	0	0	1	3
4	2	0	0	0	34	0	0	0	0	0
5	0	1	0	0	0	34	0	0	0	2
6	1	0	0	1	0	0	34	0	0	0
7	0	0	0	0	1	0	0	35	0	0
8	0	2	3	3	0	1	0	2	24	0
9	0	1	0	0	0	0	0	0	0	35

Misclassification:



Loss:



In this case, the problem needs more time to be solved in more epochs, and it is highly prone to overfitting when I increase the number of layers.

Increasing the number of hidden units and/or layers may lead to overfitting. It will make it easier for the neural network to memorize the training set, that is, to learn a function that perfectly separates the training set but that does not generalize to unseen data. On the other hand, the less number of neurons and/or layers might cause underfitting.

2. The activation function

It is a critical part of the design of a neural network. The choice of activation function in the hidden layer will control how well the model learns the training dataset. The choice of activation function in the output layer will define the type of predictions the model can make. The default activation function for hidden layers is the ReLU function in most commercial simulators. The activation function for output layers depends on the type of prediction problem. Since it is a multiclass classification, I check both “softmax” and “sigmoid” activation functions for the output layer. Also, “tanh” and “relu” are checked for the hidden layer.

	Hidden	Output	Hidden	Output	Hidden	Output	Hidden	Output
	Tanh	Softmax	Relu	Softmax	Tanh	Sigmoid	Relu	sigmoid
Train Err	0.003		0		0.0045		0	
Test Err	0.0.02		0.006		0.03		0.02	
Loss	0.0047		0.0021		0.023		0.002	
Time(s)	3.03		2.6		2.77		2.8	

The output activation function “softmax” provides a better probability for the truth label than the “sigmoid” activation function.

We prefer not to hire nonlinear activation functions (sigmoid,tanh) in the hidden layers. Although their use in the hidden layers allows neural networks to learn complex mapping functions, they effectively prevent the learning algorithm from working with deep (large layers) networks because they fail to receive effective gradient information (because of their saturation in large values). In fact, vanishing gradients make it difficult to know which direction the parameters should move to improve the cost function.

This multiclass data set’s most favorable activation functions are “relu” for hidden layers and “softmax” for the output layer.

3. Choice of optimization method

All other hyperparameters are set to the default above and the optimization method change. The results are reported in the following table:

	RMSprop	Adam	Adamax	Nadam	Ftrl	Sgd	Adadelta
Train Err	0.002	0	0.03	0.002	0.13	0.02	0.94
Test Err	0.033	0.033	0.05	0.02	0.12	0.04	0.94
Loss	0.0116	0.0237	0.1489	0.027	1.14	0.1167	2.85
Time (s)	3.04	2.7	2.5	2.9	2.5	2.79	2.5

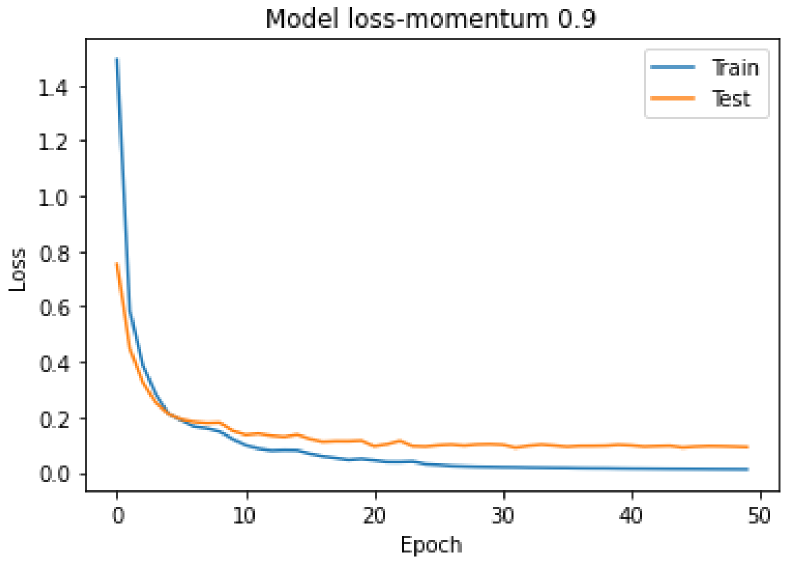
Optimization methods are responsible for reducing losses and providing the most accurate results in a reasonable time. In this case, since the scale of the dataset and neural network are small, the time does not significantly change. However, the accuracy of the training process might slightly change. Adam has provided the best results in terms of training accuracy. Adam optimizer works with momentums of first and second order. The idea behind Adam is that we don’t want to roll so fast just because we can jump over the minimum. We want to decrease the velocity a little bit for a careful search. In addition to storing an exponentially decaying average of past squared gradients like Adadelta, Adam also keeps an exponentially decaying average of past gradients. Nadam optimizer that has provided the best overall accuracy is Adam with Nesterov momentum. Theoretically, Nadam outperforms Adam, but sometimes RMSProp gives the best performance.

4. Momentum

The optimizer is set to be “sgd,” the parameters are set to default, and only the momentum changes.

momentum	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Train Err	0.009	0.02	0.02	0.006	0.009	0.006	0.003	0.0007	0.003	0.90
Test Err	0.04	0.05	0.04	0.03	0.05	0.03	0.03	0.03	0.04	0.903
Loss	0.115	0.093	0.082	0.078	0.07	0.05	0.04	0.022	0.01	11.05

Momentum is a technique that improves both training speed and accuracy. This term improves the speed of convergence by bringing some eigen components of the system closer to critical damping. Since this is a small case, the solution time does not significantly vary by changing the momentum (almost 2.5 seconds for all cases). From the table, we can see that increasing the momentum to 0.9 results in a decrease in loss. This trend would change from 0.9 to 1, it fails to converge to the optimal point, and the accuracy significantly decreases.

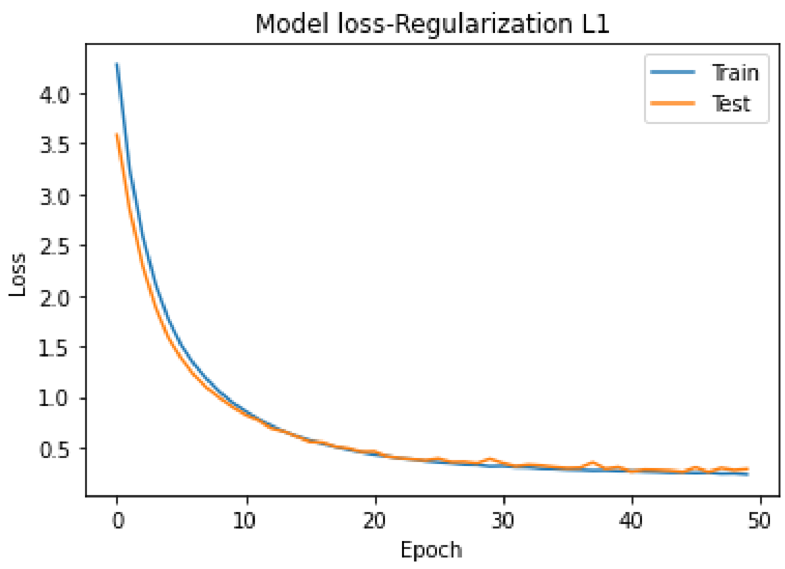


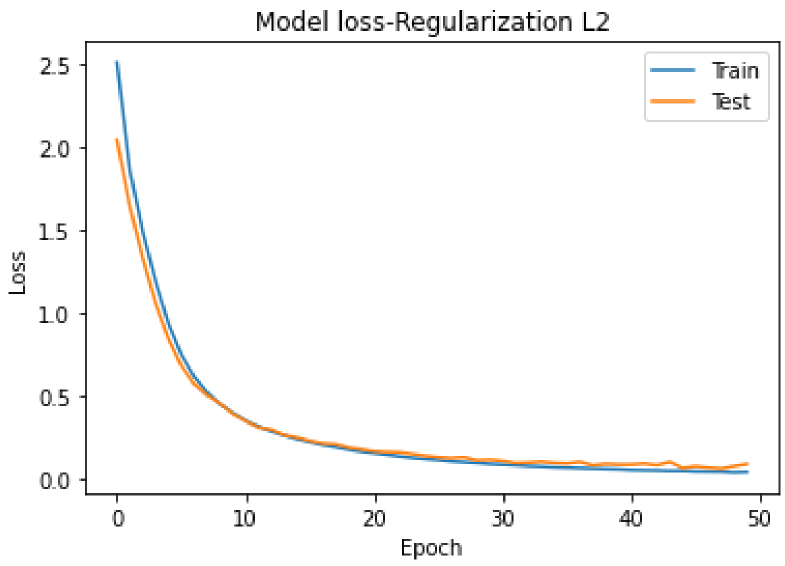
5. Regularization

Among all the regularization methods, we only add the regularization terms l_1 and l_2And evaluate their effect.

	l_1	l_2
Train Err	0.031	0.003
Test Err	0.041	0.03
Loss	0.24	0.043

Regularization is a set of strategies used to reduce the generalization error. From a practical standpoint, l_1 tends to shrink coefficients to zero whereas l_2 tends to shrink coefficients evenly. l_1 is useful for feature selection so that we can drop any variables associated with coefficients zero. On the other hand l_2 is useful when you have collinear/codependent features. In this case l_2 outperforms l_1 as it provides better errors and value for the loss.

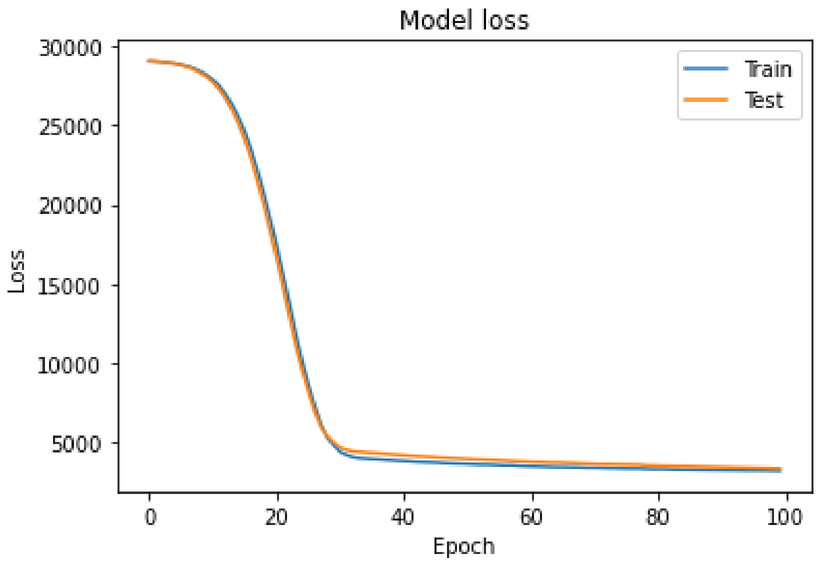

Model loss-Regularization L1
Train
Test
Loss
Epoch
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
10
20
30
40
50


Model loss-Regularization L2
Train
Test
Loss
Epoch
2.5
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50

1. Implement the neural network model for the diabetics dataset.
2. Plot the loss and compare in terms of
3. the hyper-parameters listed above.
4. Study the subjects of partial dependence (PD) and Individual Conditional Expectation (ICE) from sklearn and/or the references listed there. Plot the PD and ICE graphs versus all single features in the dataset. The link is provided below.
5. Provide brief explanations for your result.

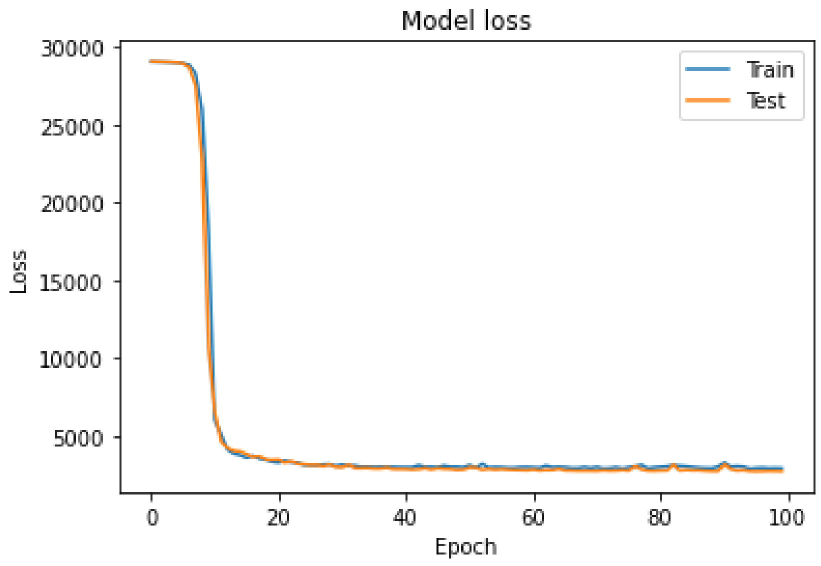
First, a very simple neural network with three dense hidden layers (32, 10, 10 neurons) and one output layer (1 neuron) is implemented. The hyper-parameters are set to default so that the activation function in the hidden layers is set to “relu,” and “linear” is set for the output layer. “adam” is selected as the optimizer, and “mean_squared_error” for the loss function. Momentum and Regularization are not considered yet. Here are the results when the epoch is set to 100:

Loss:



1. Number of the hidden unit

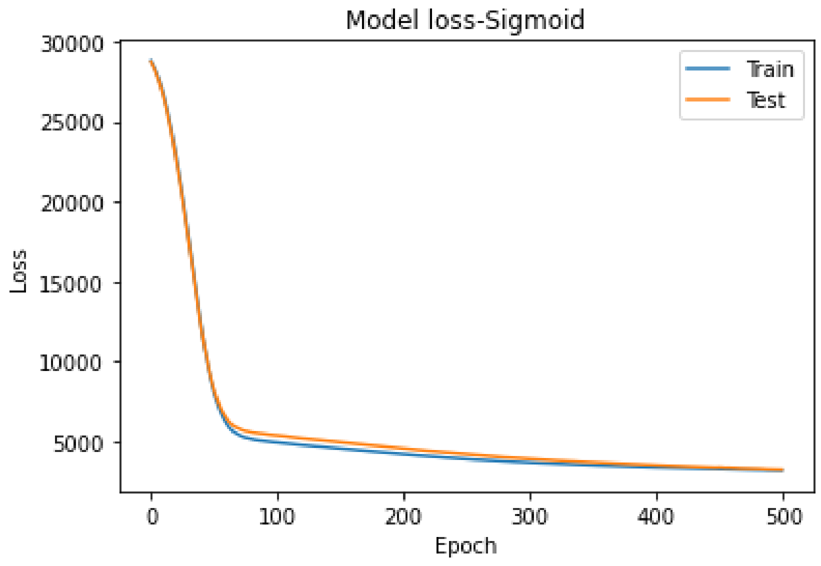
The number of hidden layers is increased to 10, and the number of neurons is (128, 64, 32, 10, 10, 10, 10, 10, 10, 10). The loss function is plotted below:

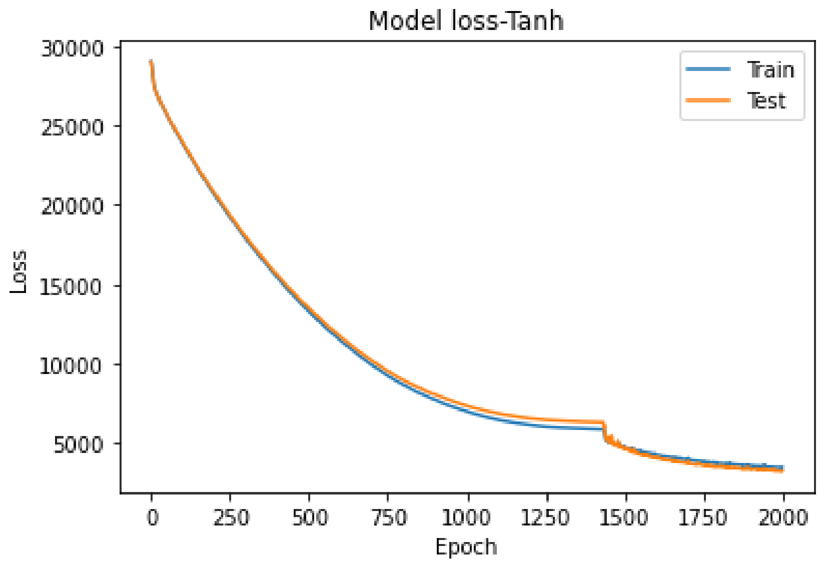


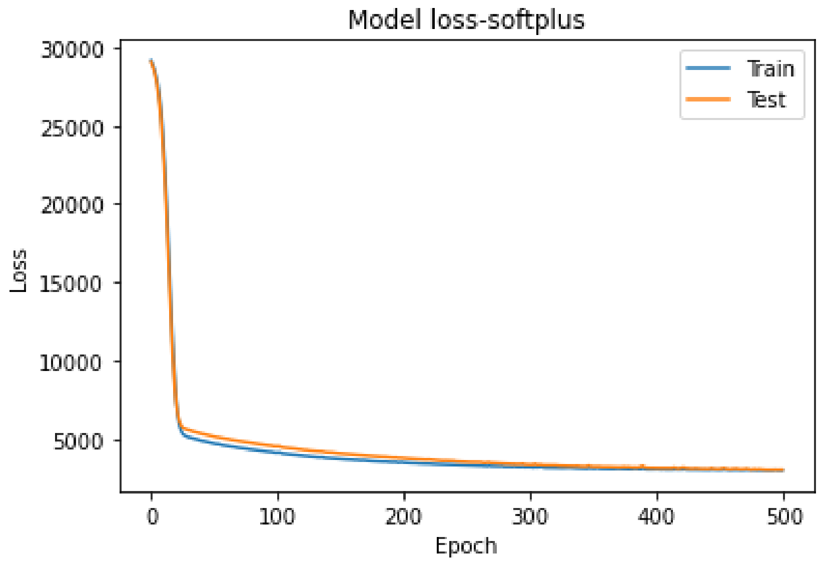
As can be seen from the picture, the loss value has settled down faster than before in less than 20 epochs. Moreover, the MSE has slightly decreased.

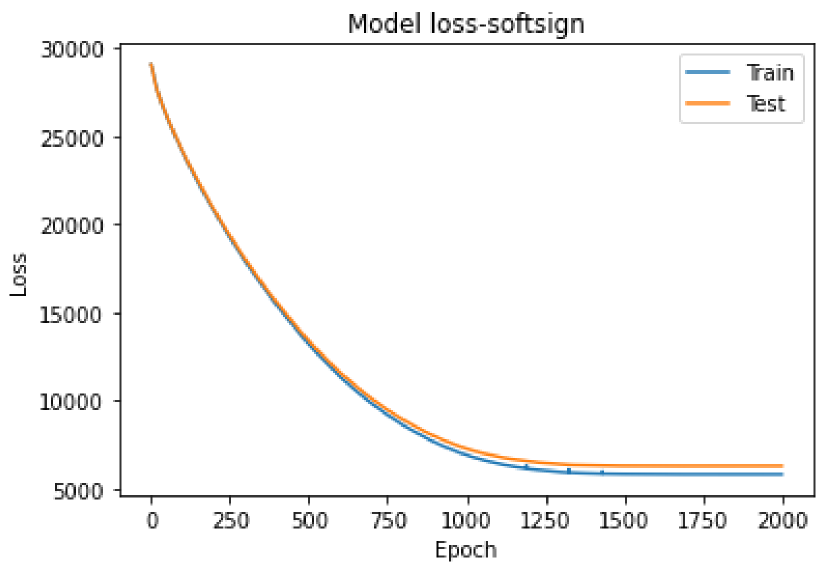
2. The activation function

Only the hidden layers activation function has changed for this study, and loss is plotted.

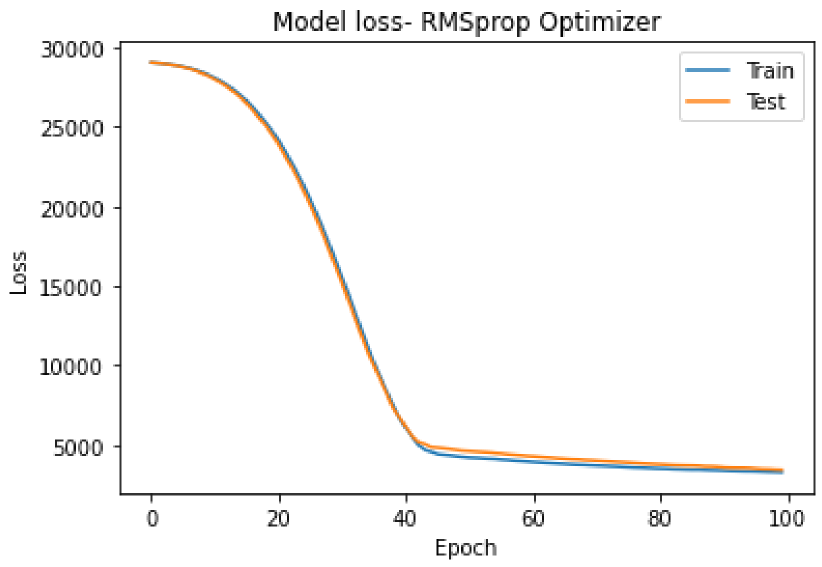



Model loss-Tanh
Train
Test
Loss
Epoch
30000
25000
20000
15000
10000
5000
0
250
500
750
1000
1250
1500
1750
2000

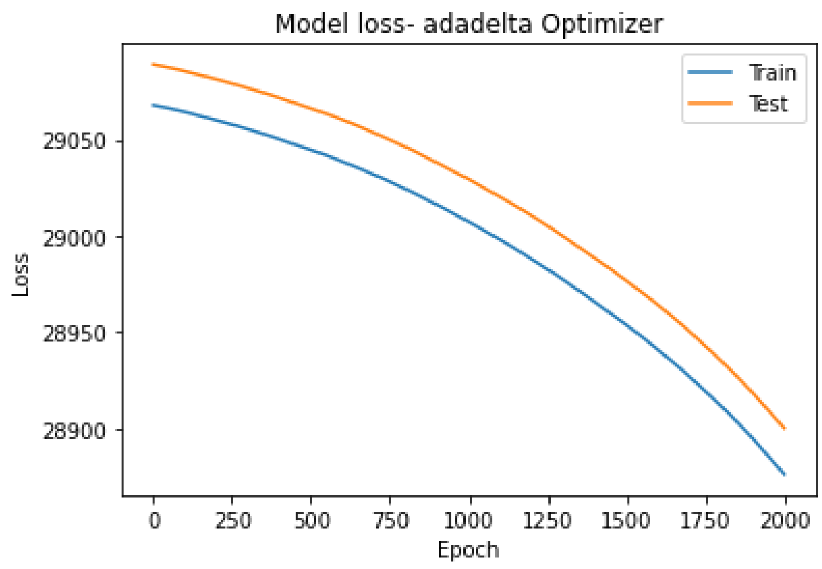

Model loss-softplus
Train
Test
Loss
Epoch
30000
25000
20000
15000
10000
5000
0
100
200
300
400
500

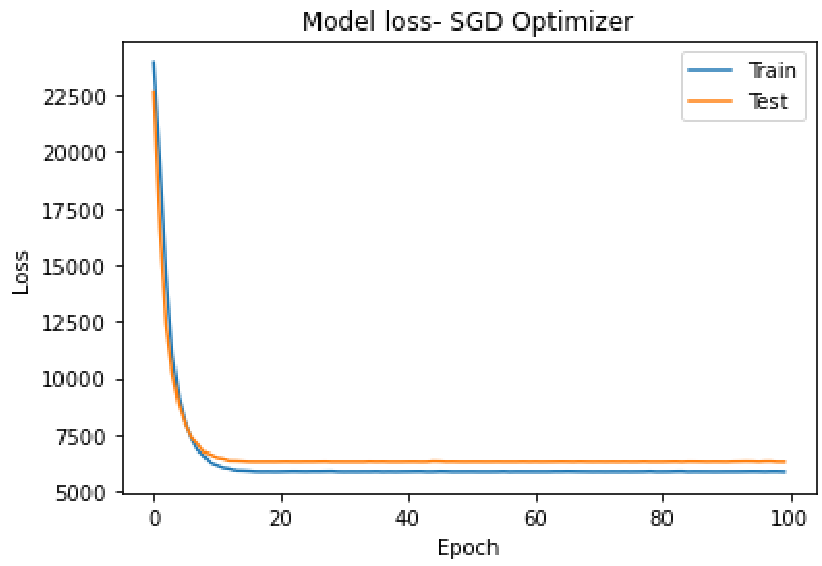


3. Choice of optimization method



Still, adam outperforms RMSprop in terms of speed and also errors. However, RMSprop shows that it is a good choice for this dataset.



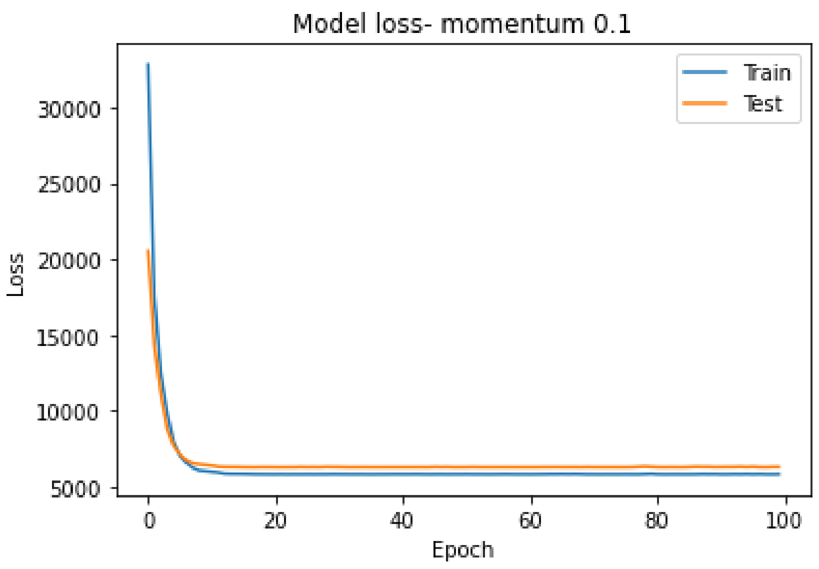


SDG converges fast, but not good results for loss and errors.

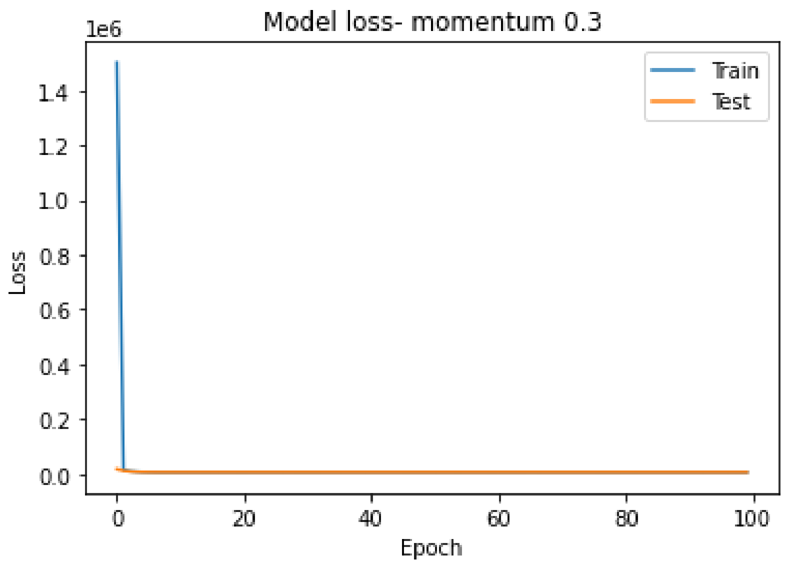
4. Momentum

The optimizer is set to be “sgd,” the parameters are set to default, and only the momentum changes.

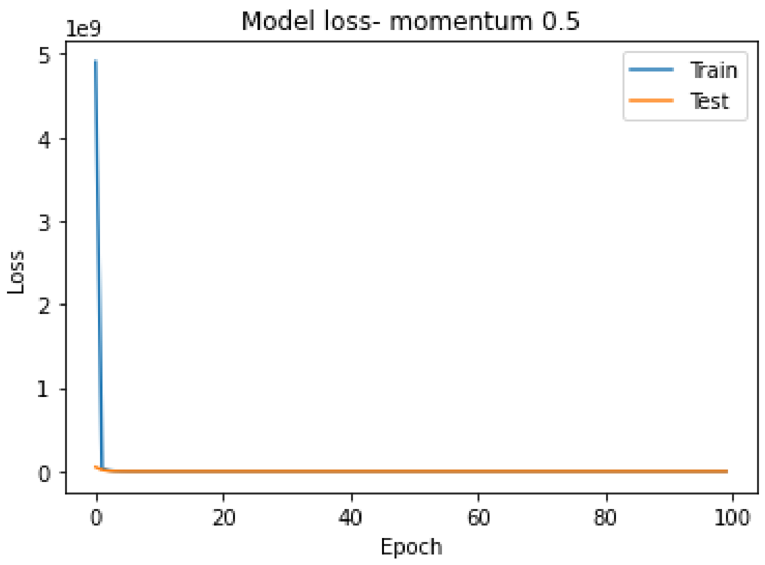
momentum	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Loss	5840	5840	5843	5843	5861	5877	5841	5907	5987	Nan

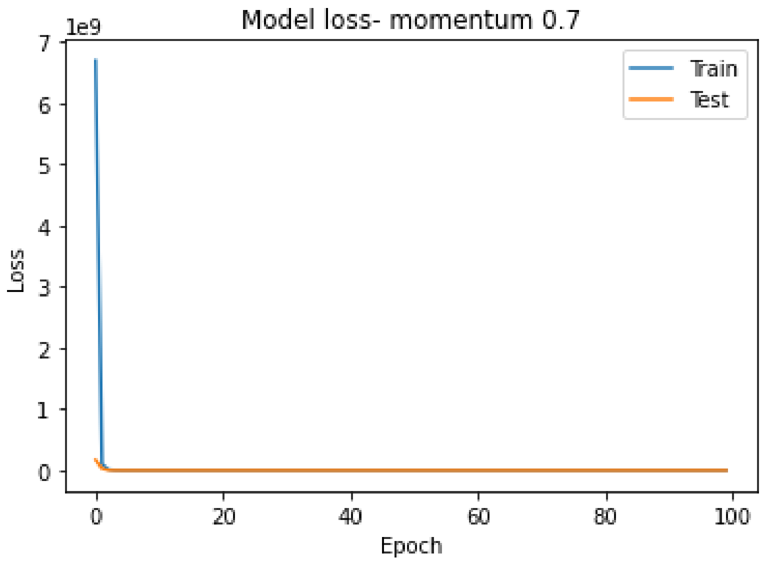


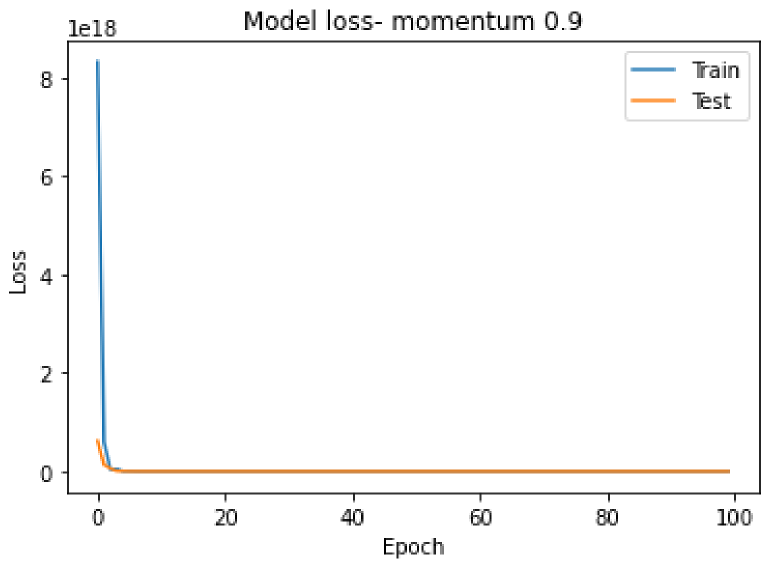
Model loss- momentum 0.1
Train
Test
Loss
30000
25000
20000
15000
10000
5000
0
20
40
60
80
100
Epoch



Model loss- momentum 0.3
1e6
Train
Test
Loss
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
Epoch


Model loss- momentum 0.5
1e9
Loss
Epoch
Train
Test

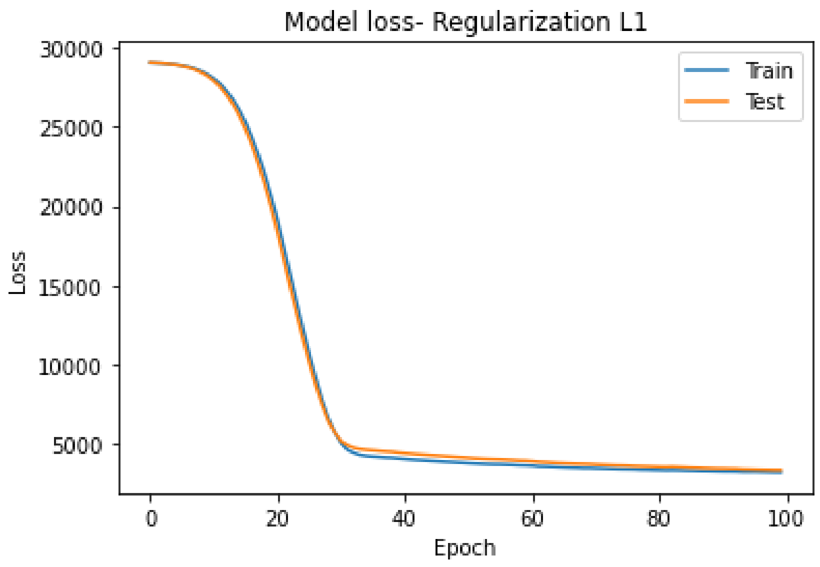

Model loss- momentum 0.7
1e9
Loss
Epoch
Train
Test

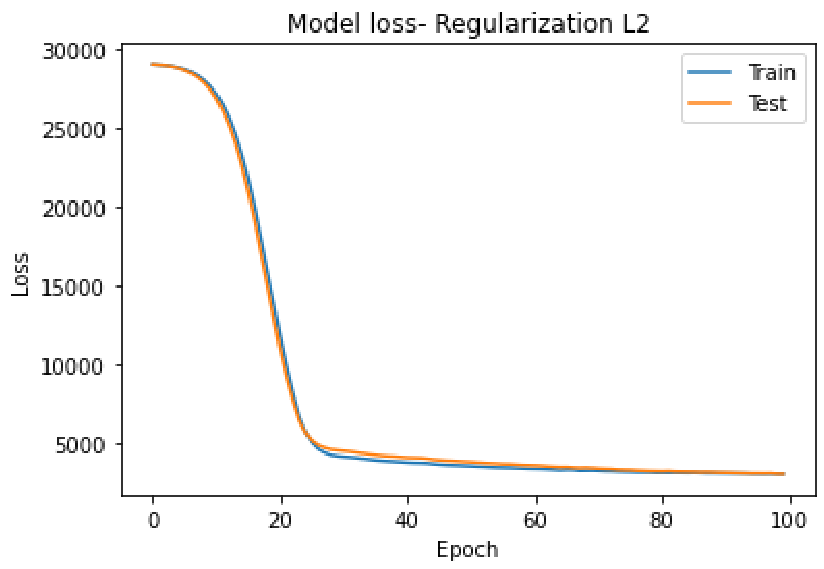


As we increase the momentum's value, the loss function also increases. However, the problem converges faster.

5. Regularization

Among all the regularization methods, we only add the regularization terms l_1 and l_2And evaluate their effect.

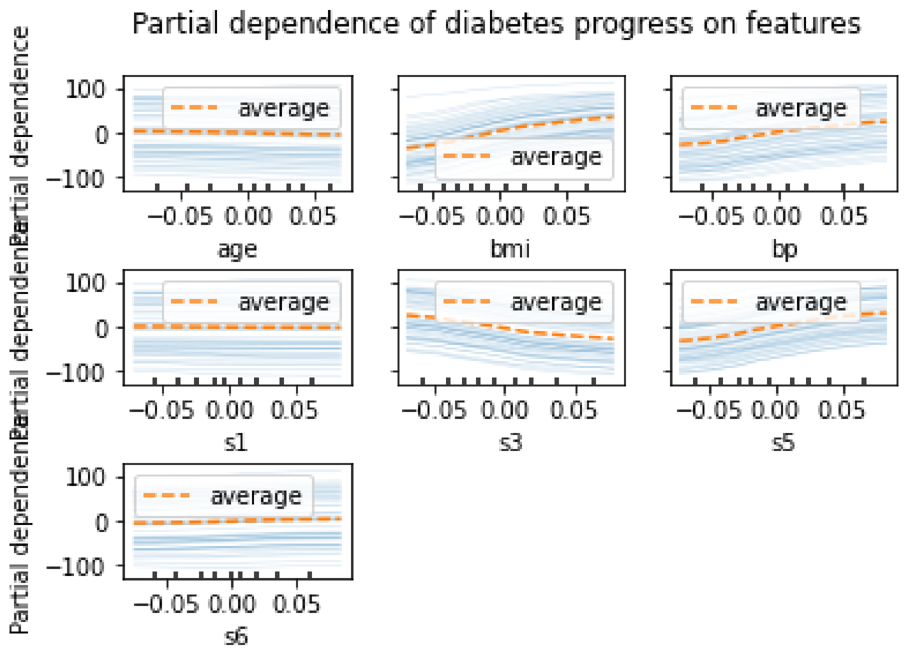




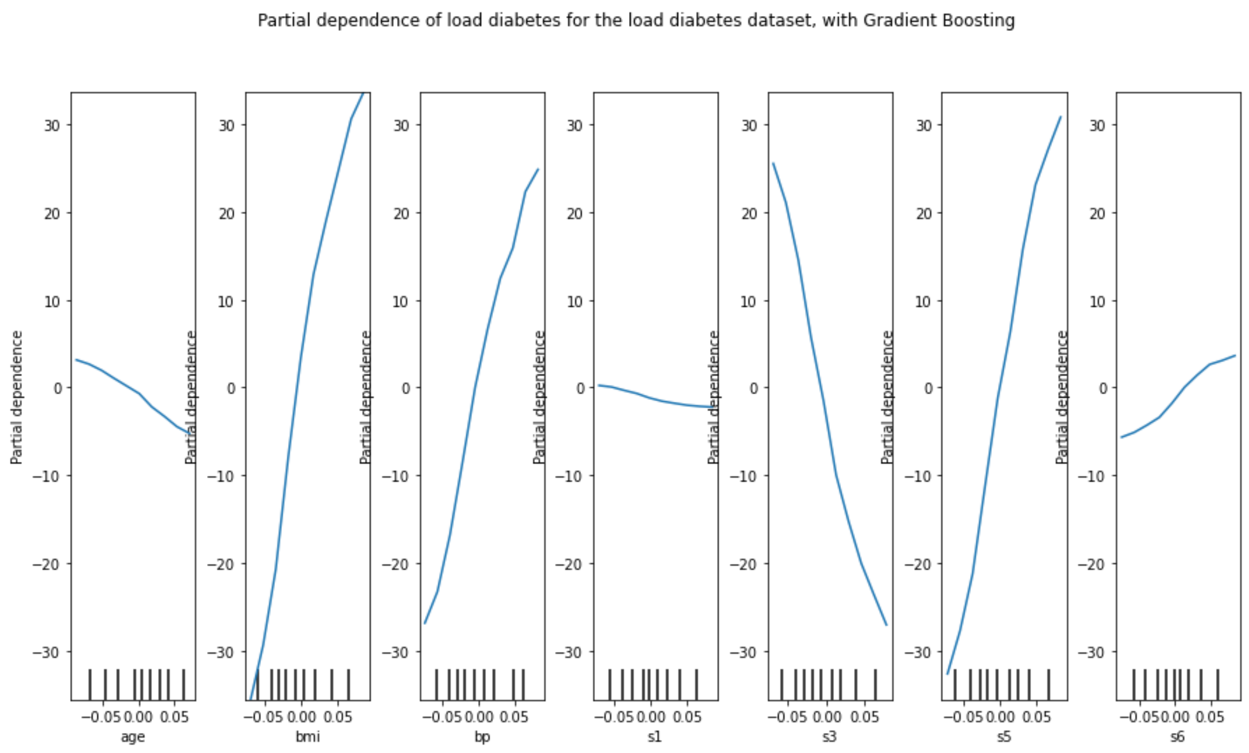
l_2 converges faster and with a better value for the loss.

Partial dependence (PD) and Individual Conditional Expectation (ICE):

Individual Conditional Expectation (ICE) plots display one line per instance that shows how the instance's prediction changes when a feature changes. Partial dependence (PD) plots show the dependence between the target function and a set of features of interest.



2D Interaction:



The above plots show how is the relationship between the features and the target value. For instance, features “age” and “s1” slightly negatively correlate with the target. Also, “s6” has a slightly positive correlation. “bmi,” “bp,” and “s5” all have positive correlation, but “s3” is negatively correlated to the target.