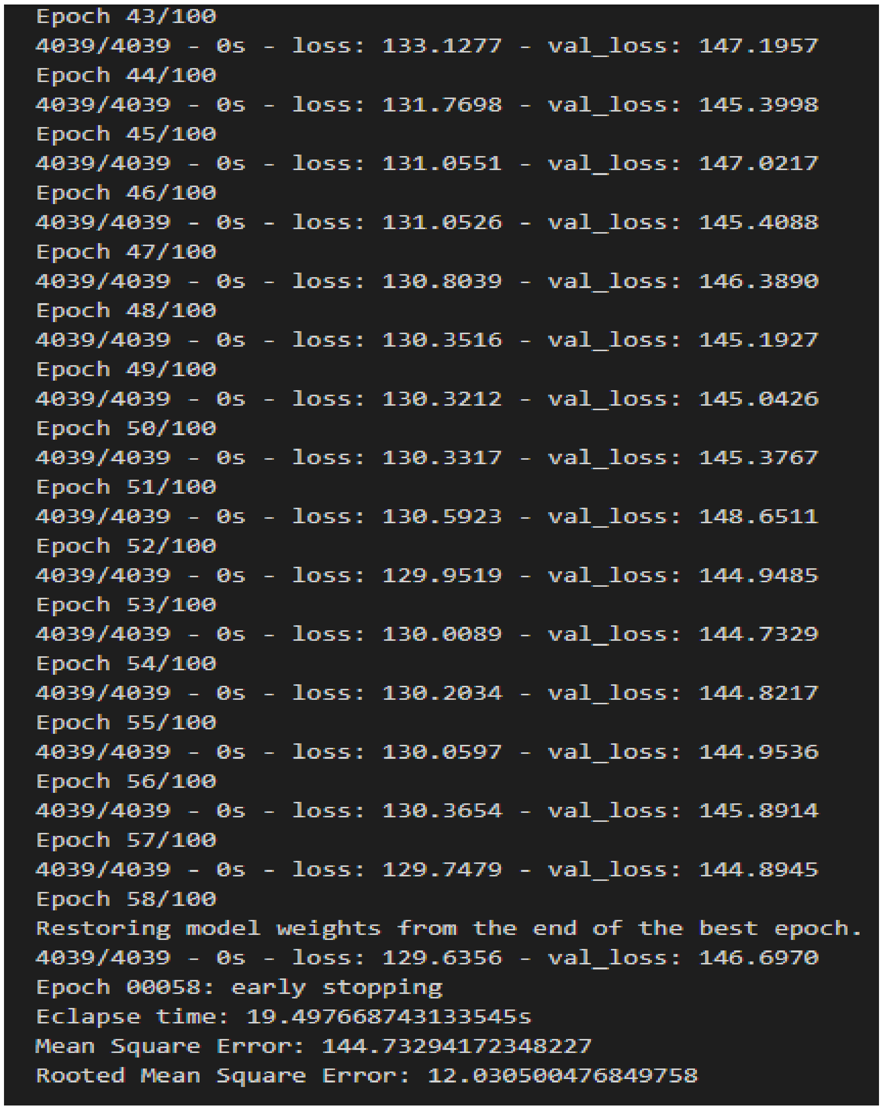
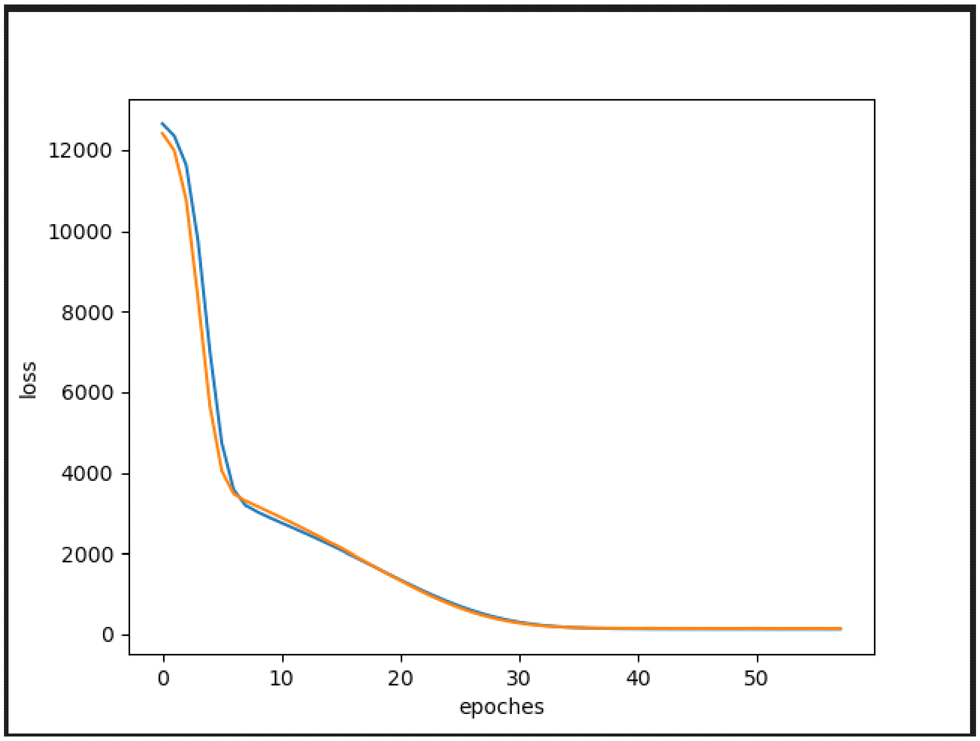
HW1 Regression and classification by MLP

Name: Xiaotong Sun, XSEDE name: xs018

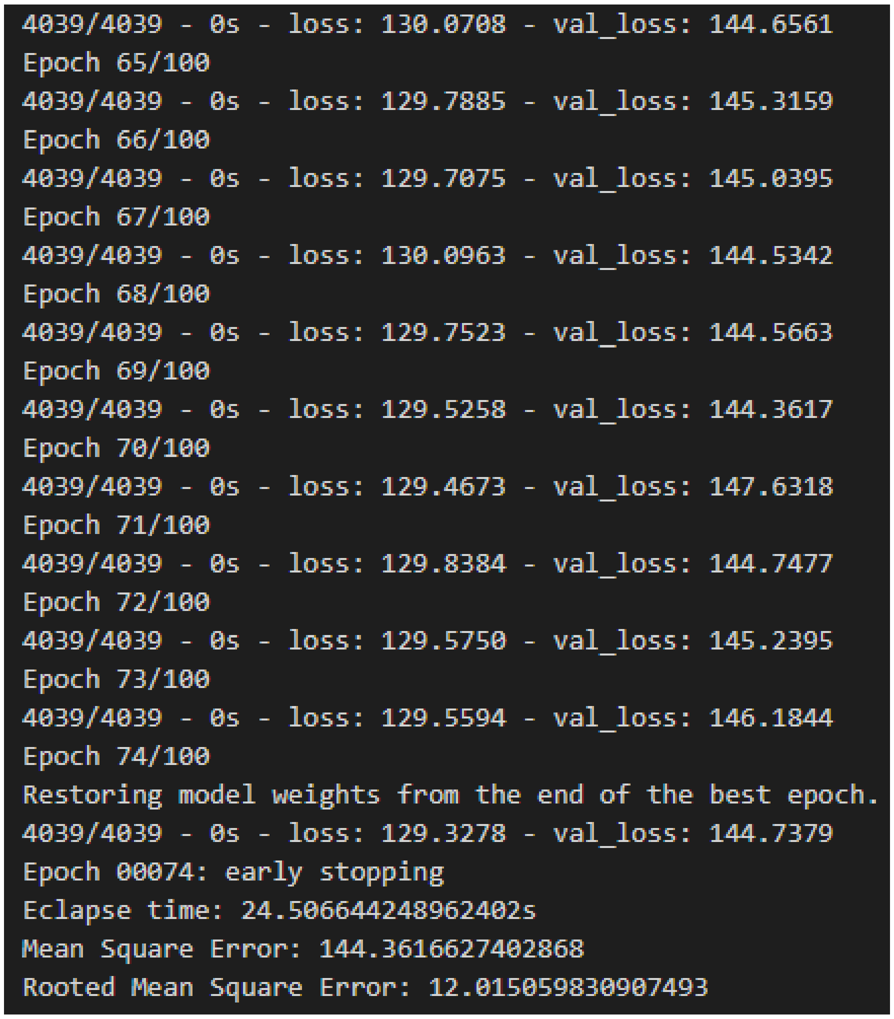
Q 1. a) In my first try, I built a MLP with 3 layers with dense 8, 4, and 1, the active function is "relu". I set the epochs as 100 times and I used early stopping. I trained and tested the model for two times. The first one it stopped at epochs 58, which the eclapse time is 19.5s, MSE is 144.73. The loss for the training is 129.64, and the validation loss is 146.7. The screenshots of result and graph is blow: blue line is train loss, orange line is validation loss.



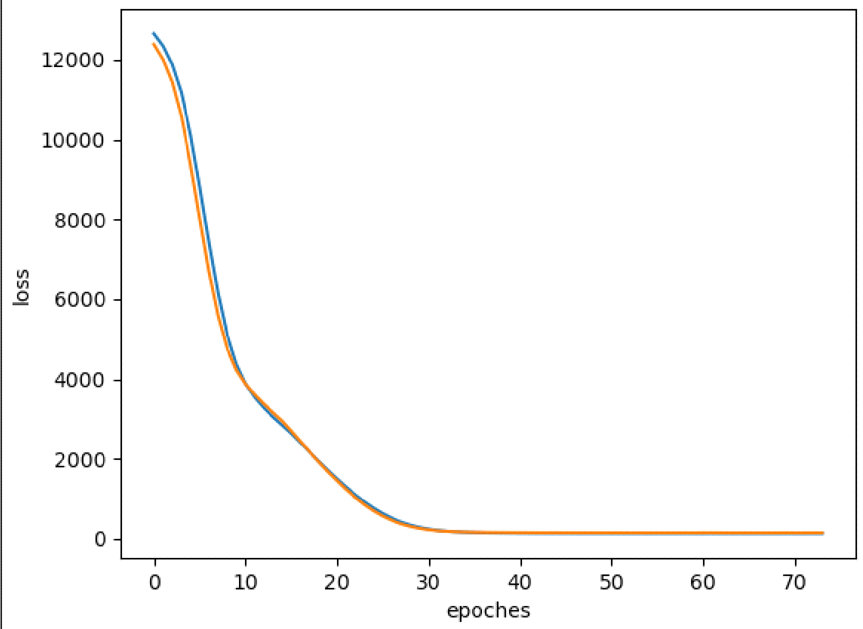
```
Epoch 43/100
4039/4039 - 0s - loss: 133.1277 - val_loss: 147.1957
Epoch 44/100
4039/4039 - 0s - loss: 131.7698 - val_loss: 145.3998
Epoch 45/100
4039/4039 - 0s - loss: 131.0551 - val_loss: 147.0217
Epoch 46/100
4039/4039 - 0s - loss: 131.0526 - val_loss: 145.4088
Epoch 47/100
4039/4039 - 0s - loss: 130.8039 - val_loss: 146.3890
Epoch 48/100
4039/4039 - 0s - loss: 130.3516 - val_loss: 145.1927
Epoch 49/100
4039/4039 - 0s - loss: 130.3212 - val_loss: 145.0426
Epoch 50/100
4039/4039 - 0s - loss: 130.3317 - val_loss: 145.3767
Epoch 51/100
4039/4039 - 0s - loss: 130.5923 - val_loss: 148.6511
Epoch 52/100
4039/4039 - 0s - loss: 129.9519 - val_loss: 144.9485
Epoch 53/100
4039/4039 - 0s - loss: 130.0089 - val_loss: 144.7329
Epoch 54/100
4039/4039 - 0s - loss: 130.2034 - val_loss: 144.8217
Epoch 55/100
4039/4039 - 0s - loss: 130.0597 - val_loss: 144.9536
Epoch 56/100
4039/4039 - 0s - loss: 130.3654 - val_loss: 145.8914
Epoch 57/100
4039/4039 - 0s - loss: 129.7479 - val_loss: 144.8945
Epoch 58/100
Restoring model weights from the end of the best epoch.
4039/4039 - 0s - loss: 129.6356 - val_loss: 146.6970
Epoch 00058: early stopping
Eclapse time: 19.497668743133545s
Mean Square Error: 144.73294172348227
Rooted Mean Square Error: 12.030500476849758
```



For the second one stopped at end of the epoch at 74, which the eclapse time is 24.51s, MSE is 144.36. The loss for the training is 129.33, and the validation loss is 144.74. The screenshots of result and graph is blow: blue line is train loss, orange line is validation loss.

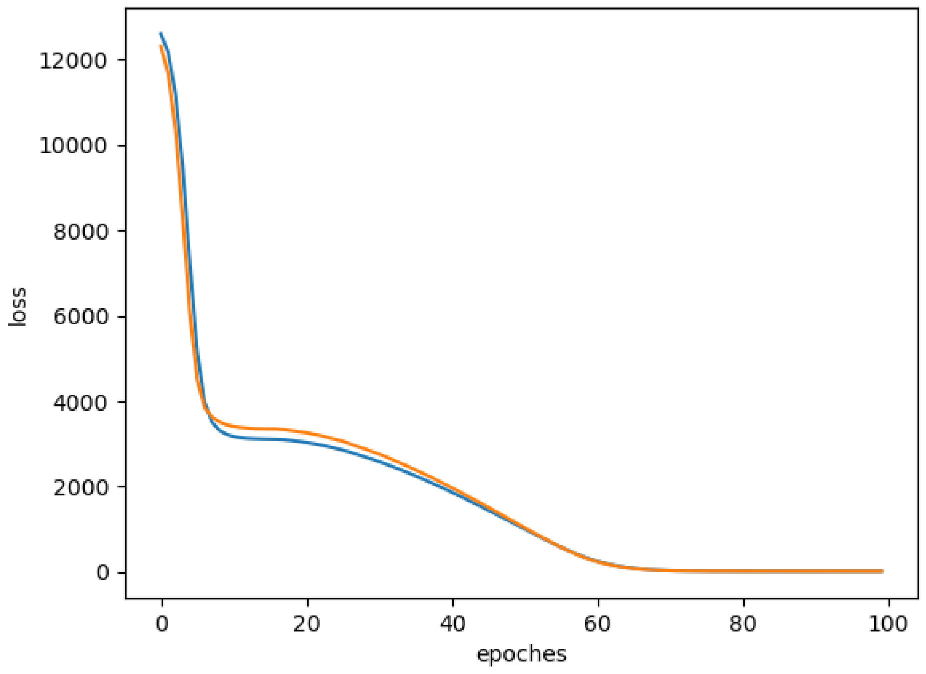


```
4039/4039 - 0s - loss: 130.0708 - val_loss: 144.6561
Epoch 65/100
4039/4039 - 0s - loss: 129.7885 - val_loss: 145.3159
Epoch 66/100
4039/4039 - 0s - loss: 129.7075 - val_loss: 145.0395
Epoch 67/100
4039/4039 - 0s - loss: 130.0963 - val_loss: 144.5342
Epoch 68/100
4039/4039 - 0s - loss: 129.7523 - val_loss: 144.5663
Epoch 69/100
4039/4039 - 0s - loss: 129.5258 - val_loss: 144.3617
Epoch 70/100
4039/4039 - 0s - loss: 129.4673 - val_loss: 147.6318
Epoch 71/100
4039/4039 - 0s - loss: 129.8384 - val_loss: 144.7477
Epoch 72/100
4039/4039 - 0s - loss: 129.5750 - val_loss: 145.2395
Epoch 73/100
4039/4039 - 0s - loss: 129.5594 - val_loss: 146.1844
Epoch 74/100
Restoring model weights from the end of the best epoch.
4039/4039 - 0s - loss: 129.3278 - val_loss: 144.7379
Epoch 00074: early stopping
Eclapse time: 24.506644248962402s
Mean Square Error: 144.3616627402868
Rooted Mean Square Error: 12.015059830907493
```



In order to get the same result, add ‘set seeds’ into the code. The ‘random_sate’ fixed the dividing of train and validation dataset.

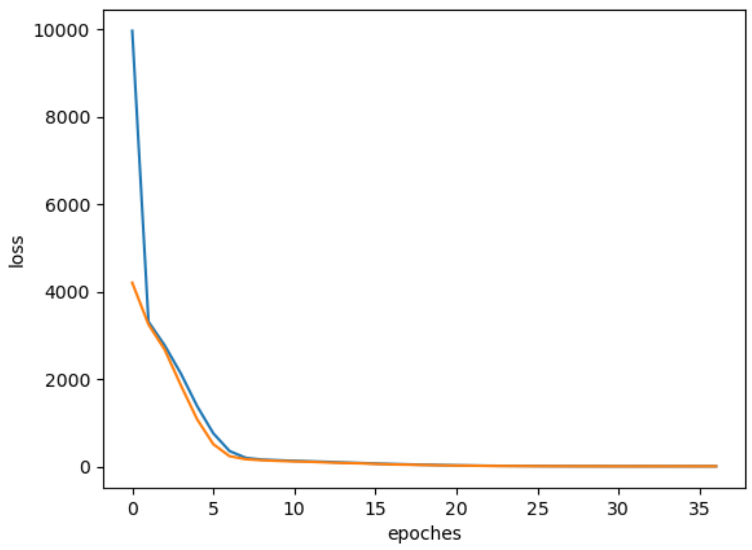
After that, I got the model stopped at end of the epoch at 100, which the eclapse time is 31.89s, MSE is 16.9852, RME is 4.12. The loss for the training is 17.45, and the validation loss is 16.98. The graph is blow: blue line is train loss, orange line is validation loss.



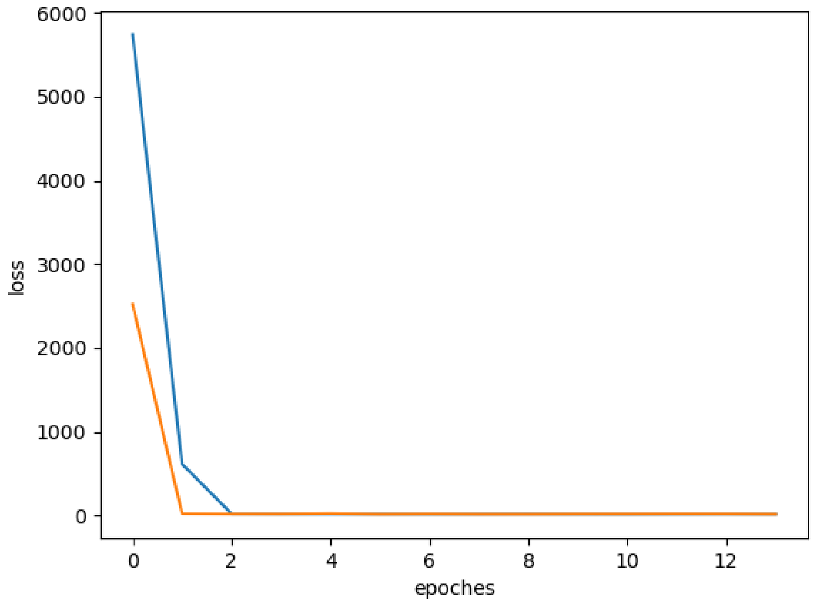
After I try to train and test the model for another two times, I get the same loss for training and validation with the upper one, but the training time is a little difference which is 30.30s, 30.64s.

The next tries all used fixed seeds.

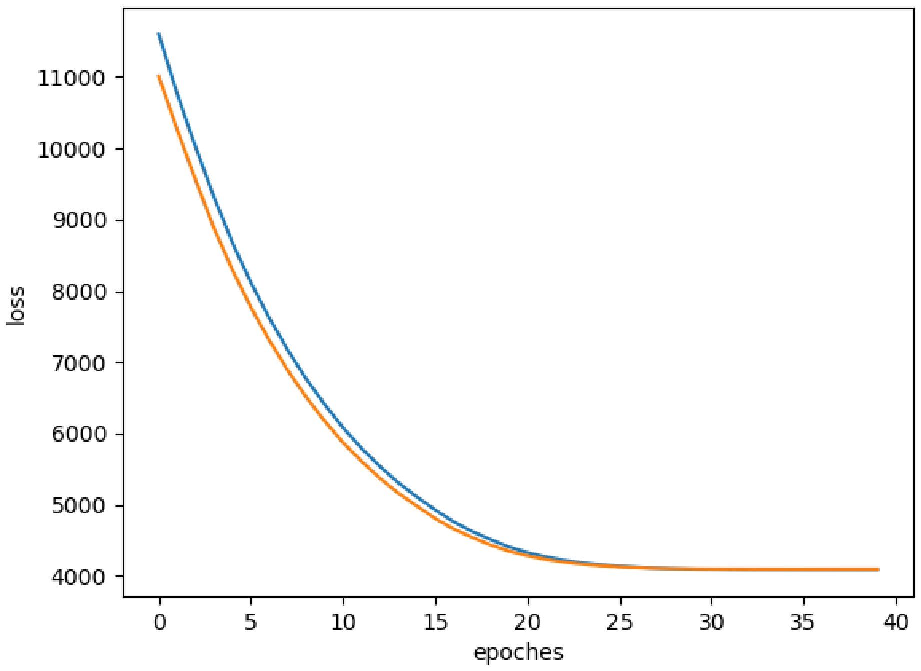
After I changed the number of dense layers with dense 256, 64, 1, the model stopped at end of the epoch at 37, which the eclapse time is 15.58s, MSE is 13.68, RMSE is 3.7. The loss for the training is 14.27, and the validation loss is 15. The graph of training and validation loss is below: blue line is train loss, orange line is validation loss.



The I added dense layers as 256, 512, 128, 64, 1, the model stopped at end of the epoch at 14, which the eclapse time is 10.7s, MSE is 13.48, RMSE is 3.67. The loss for the training is 15.46, and the validation loss is 14.44. The graph of training and validation loss is below: blue line is train loss, orange line is validation loss.



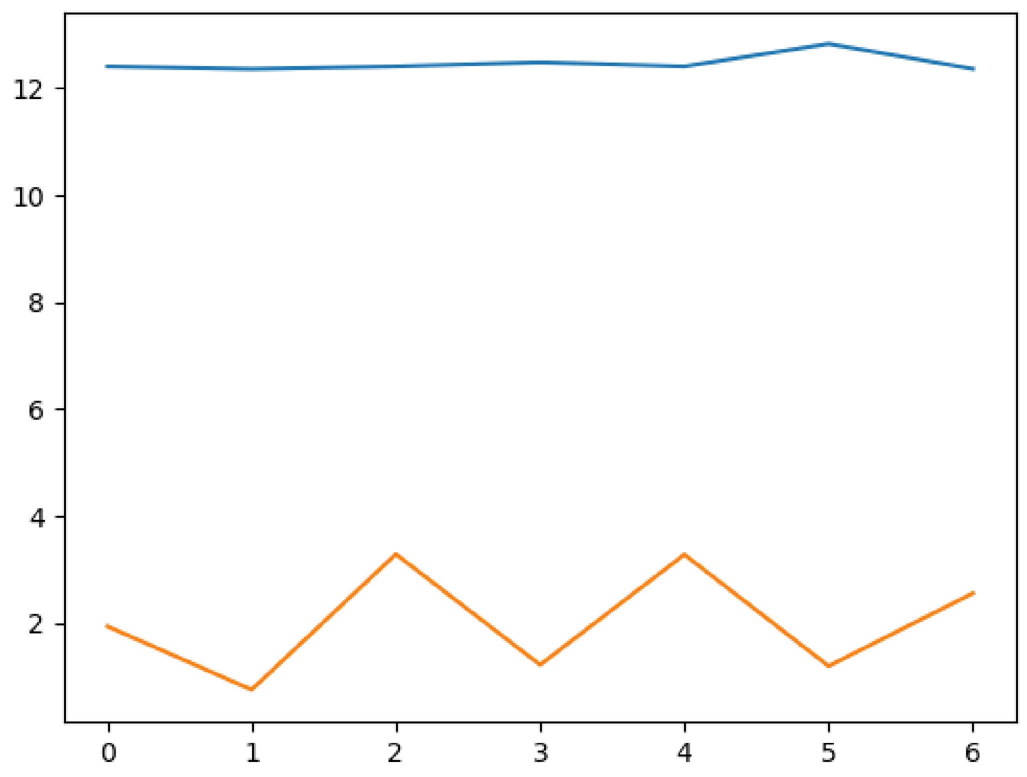
Then I changed the active function “relu” to “sigmoid”, the model stopped at end of the epoch at 40, which the eclapse time is 17.2s, MSE is 4099.78, RMSE is 3.67. The loss for the training is 4089.39, and the validation loss is 4100. The graph of training and validation loss is below: blue line is train loss, orange line is validation loss.



The reason why the “sigmoid” is not good as “relu”, I guess it might because “sigmoid” is from 0 to 1, which may lead to the vanish of gradient.

Q1. b)

From part a, we got some good hyper parameters with dense layers as 256, 512, 128, 64, 1, and active function is “relu”. Then I used those hyper parameters to do the 100-fold cross validation. After 100-fold cross validation, the loss of the best model is output. In order to get the best model, I compared the validation loss for each fold and get the minimize one. Dr. Li said from his courses in BiliBili: the average loss of the folds always be used as a loss output for the model. I didn’t use the average because find the best model can be a step for choosing the hyper parameters. When we save the best model from 100-fold validation, we can use the hyper parameters in the best model to train the model with full dataset (We split the full dataset as training, validation, and test) so that we can get a model has better performance. I get the eclapse time is 334.36s, MSE is 8.21, RMSE is 2.87. The loss for the training is 12.41, and the validation loss is 8.86. The graph is attached below: blue line is train loss, orange line is validation loss.



I used early stopping in the 100-fold cross validation. When the validation loss is too small, it will stop and take that validation loss as that fold.

Q2. Readme: When train the model, commend the part of test. When test the model, commend the part of train

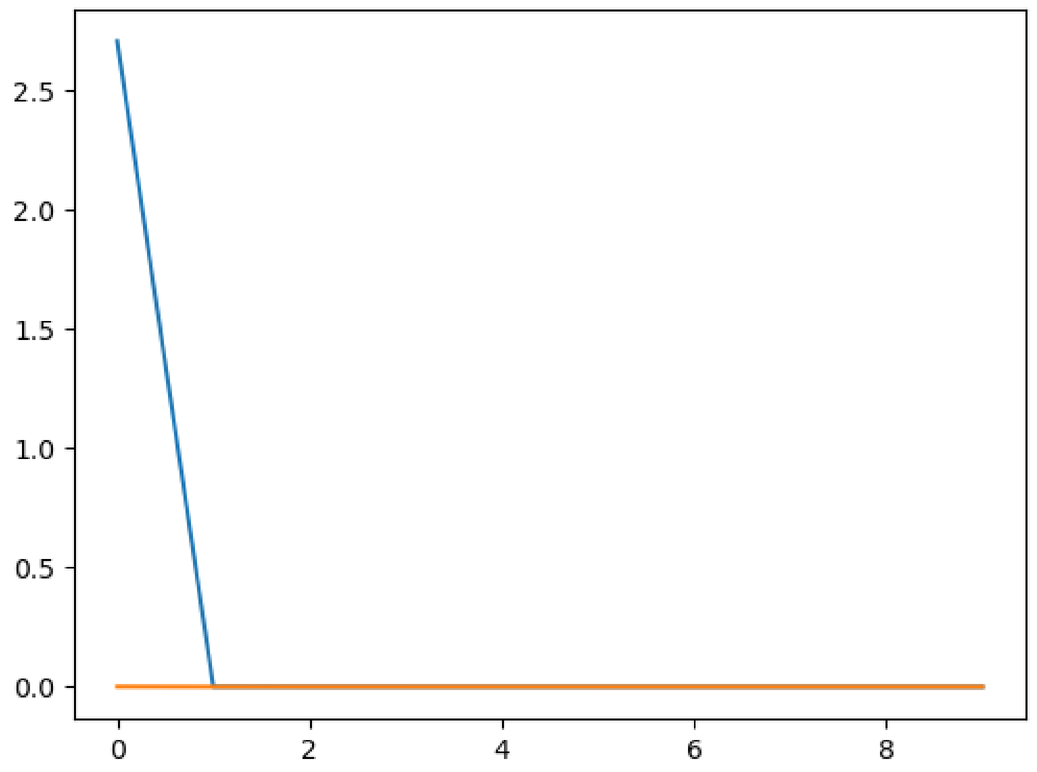
a), b), and c)

Split the dataset as train, validation, and test is the most difficult step for me. Firstly, I defined a get_label function. In that function, the file_path is convert to a list of path components. Then the one hot encode is used to dummy the pre or post image. Then a decode_img function is defined to read the a image as 3D unit8 tensor, which has the dimension of channels. Then resize the image into a canonical input size(e.g., 240x240) and divide the intensity by 255. After that, a function called process_path is defined to convert the raw data as string, get_label and decode_img is called in the function. The function configure_for_performance is defined to input the image by batch. And the data is shuffled by buffer size 1000 which will make the dataset more randomly. Then the function create_dataset is defined, which includes the data path, and batch size.

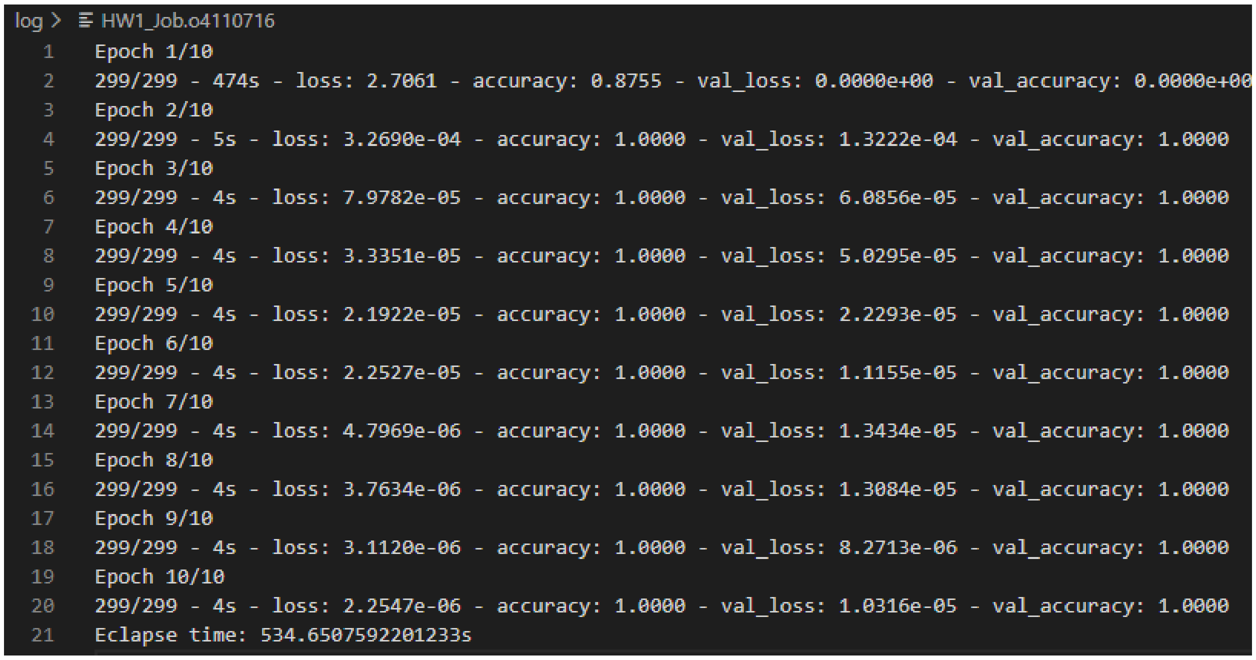
Then built the train function with parameters model, train data, validation data and checkpoint which is used to save the weights after training.

The test function is built to do the predication. The bigger probability of the predication is taken as the output.

In the main(), the batch_size is 64, and training epoch is 10. The model test used the results in checkpoint which is the weight from training. The graph of validation and train loss is attached below: blue line is train loss, orange line is validation loss.

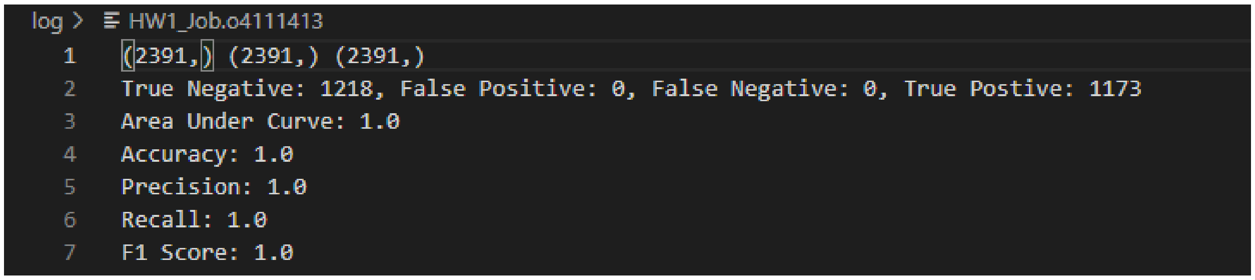


The validation accuracy is 1, the eclapse time is 534.65s. The details for the training time of each epoch is shown in the screenshot below:



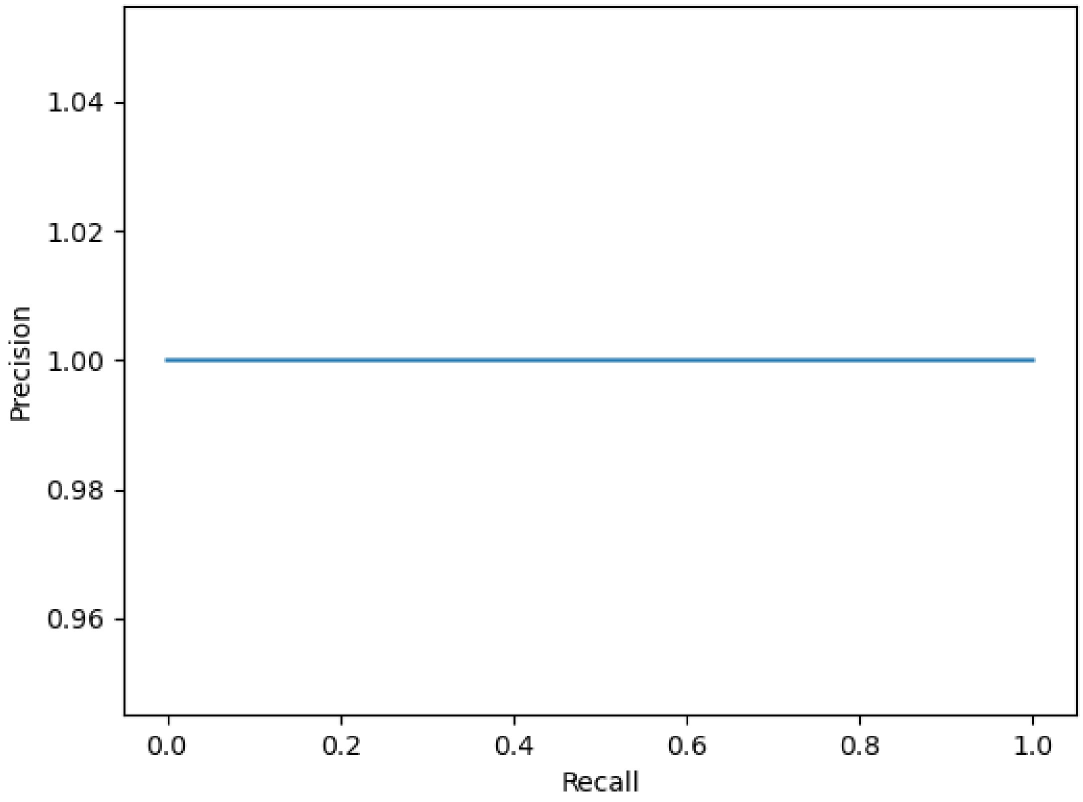
```
log > HW1_Job.o4110716
Epoch 1/10
299/299 - 474s - loss: 2.7061 - accuracy: 0.8755 - val_loss: 0.0000e+00 - val_accuracy: 0.0000e+00
Epoch 2/10
299/299 - 5s - loss: 3.2690e-04 - accuracy: 1.0000 - val_loss: 1.3222e-04 - val_accuracy: 1.0000
Epoch 3/10
299/299 - 4s - loss: 7.9782e-05 - accuracy: 1.0000 - val_loss: 6.0856e-05 - val_accuracy: 1.0000
Epoch 4/10
299/299 - 4s - loss: 3.3351e-05 - accuracy: 1.0000 - val_loss: 5.0295e-05 - val_accuracy: 1.0000
Epoch 5/10
299/299 - 4s - loss: 2.1922e-05 - accuracy: 1.0000 - val_loss: 2.2293e-05 - val_accuracy: 1.0000
Epoch 6/10
299/299 - 4s - loss: 2.2527e-05 - accuracy: 1.0000 - val_loss: 1.1155e-05 - val_accuracy: 1.0000
Epoch 7/10
299/299 - 4s - loss: 4.7969e-06 - accuracy: 1.0000 - val_loss: 1.3434e-05 - val_accuracy: 1.0000
Epoch 8/10
299/299 - 4s - loss: 3.7634e-06 - accuracy: 1.0000 - val_loss: 1.3084e-05 - val_accuracy: 1.0000
Epoch 9/10
299/299 - 4s - loss: 3.1120e-06 - accuracy: 1.0000 - val_loss: 8.2713e-06 - val_accuracy: 1.0000
Epoch 10/10
299/299 - 4s - loss: 2.2547e-06 - accuracy: 1.0000 - val_loss: 1.0316e-05 - val_accuracy: 1.0000
Eclapse time: 534.6507592201233s
```

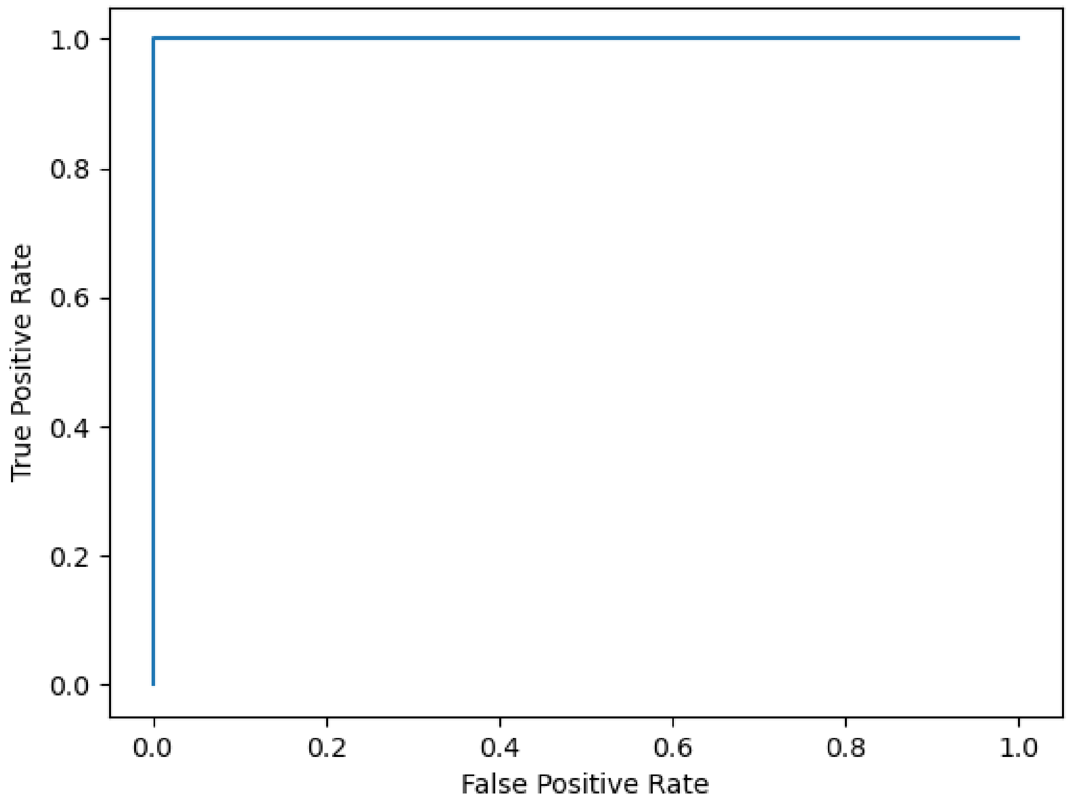
Finally, testing the MLP model by the reserved test data, the accuracy I got is 1.0, precision is 1.0, F1 score is 1.0, AUC is 1.0, recall is 1.0, confusion matrix is [1218,0;0,1173]. The screenshot of result as below:

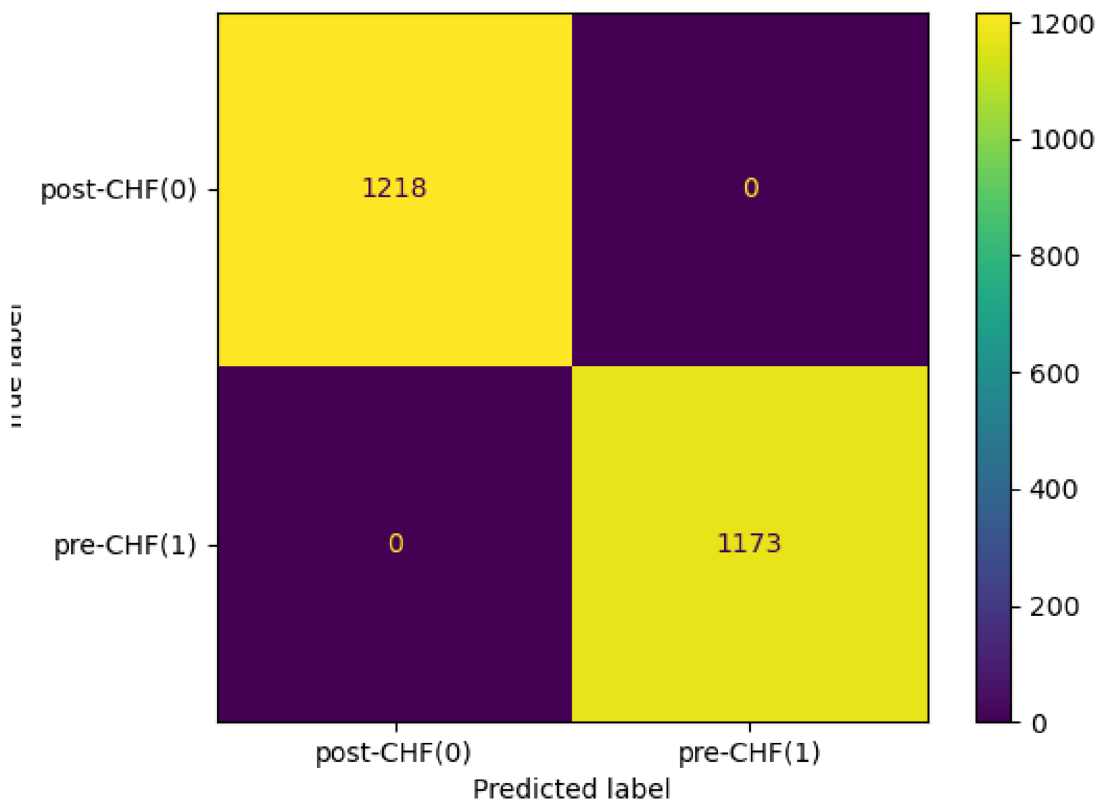


```
log > HW1_Job.o4111413
(2391,) (2391,) (2391,)
True Negative: 1218, False Positive: 0, False Negative: 0, True Postive: 1173
Area Under Curve: 1.0
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

The graph of confusion predication, ROC, and confusion matrix are below:


1.04
1.02
1.00
0.98
0.96
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall


True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


post-CHF(0)
pre-CHF(1)
True label
1218
0
0
1173
post-CHF(0)
pre-CHF(1)
Predicted label
1200
1000
800
600
400
200
0