Experiment Report: Exploring MLP models' performances

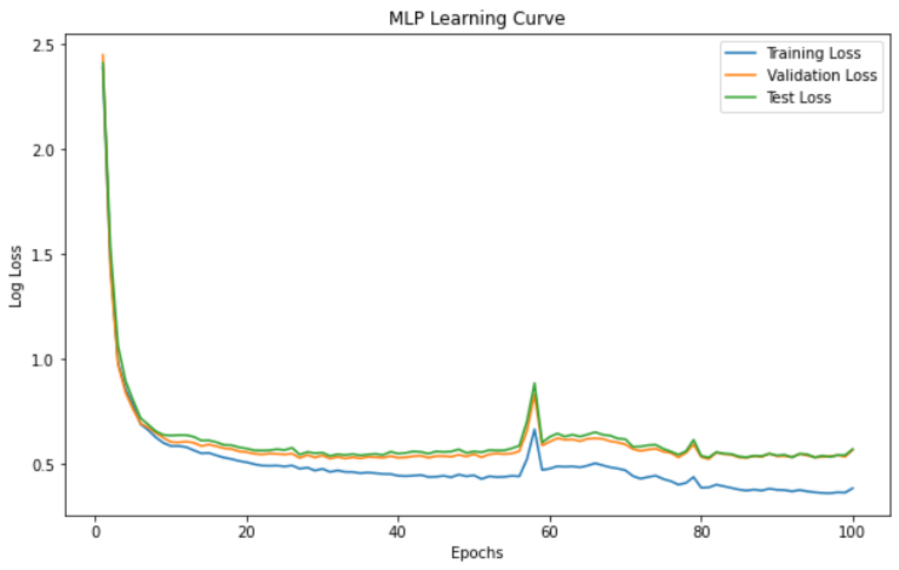
Introduction

In this experiment, the MLP classifier was trained using partial fit. Over 100 epochs, tracking training loss, validation loss, and test loss. Training error, cross-validation error, and test error served as the main evaluation measures. The objective was to find out how a Multi-Layer Perceptron (MLP) classifier would perform with various changes. To examine the influence of these hyperparameters on training convergence, generalization, and overall model performance, the MLP was trained on a healthcare dataset and assessed on both a validation set and a test set.

Observations and Results

MLP Model

Initially, I set up a MLP model with 100 neurons in the first layer and 50 in the second, the training result was shown below.

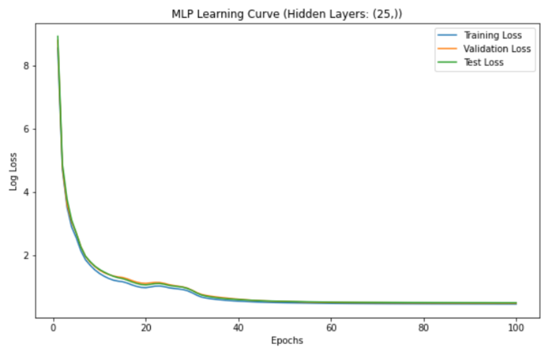


```
Training Error: 0.17
Cross-Validation Error: 0.25
Test Error: 0.25
```

Fig 1. MLP curve (100, 50,)

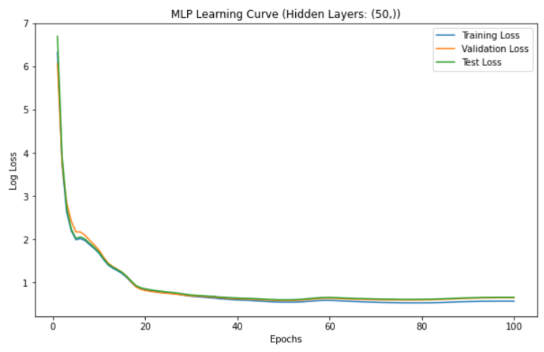
Then, I proceed to find out how the numbers of layers will affect our training model. This is achieved by setting up test by using only one layer with 25, 50, and 100 neurons as first. Followed by adding the second and third layer, each of the layer contains half of the number of neurons than the previous.

```
Hidden Layers: (25,)
Training Error: 0.21
Cross-Validation Error: 0.23
```



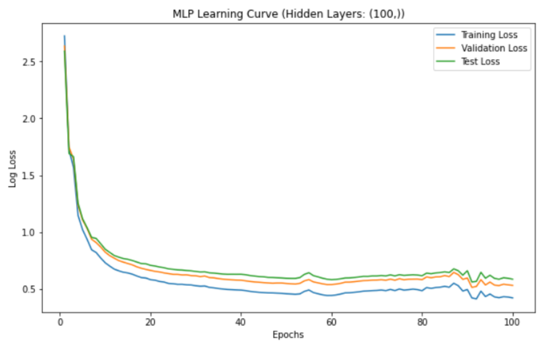
```
Test Error: 0.23
---------------------------------------------------------------------------
```

```
Hidden Layers: (50,)
Training Error: 0.24
Cross-Validation Error: 0.27
```



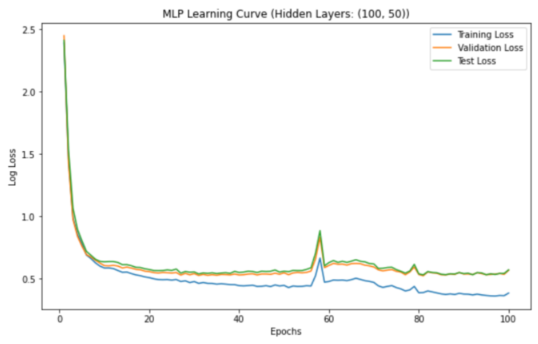
```
Test Error: 0.28
---------------------------------------------------------------------------
```

```
Hidden Layers: (100,)
Training Error: 0.19
Cross-Validation Error: 0.24
```



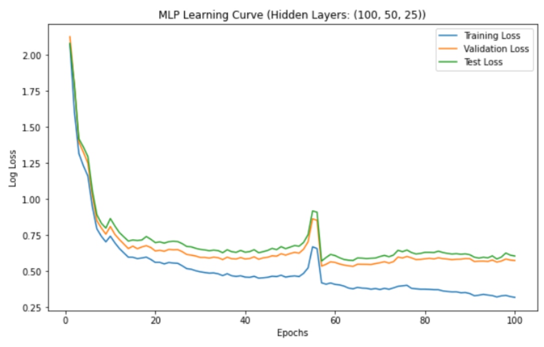
```
Test Error: 0.26
---------------------------------------------------------------------------
```

```
Hidden Layers: (100, 50)
Training Error: 0.17
Cross-Validation Error: 0.25
```



```
Test Error: 0.25
---------------------------------------------------------------------------
```

```
Hidden Layers: (100, 50, 25)
Training Error: 0.15
Cross-Validation Error: 0.23
```



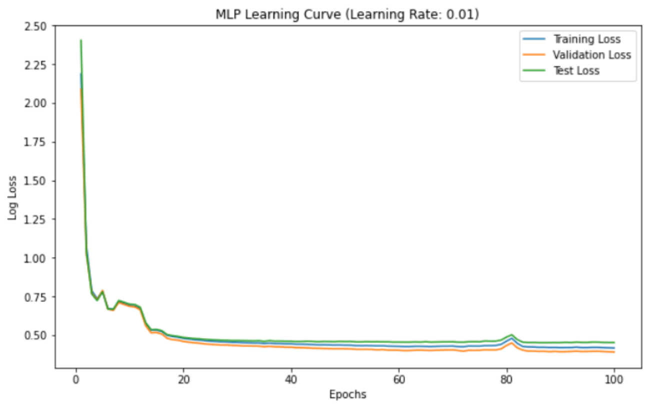
```
Test Error: 0.25
---------------------------------------------------------------------------
```

Fig 2. MLP with different number of neurons

Result show that the model fits well with lower numbers of neurons in a single layer and are showing signs of overfitting when number of neurons and layers increase. At this point, I decide to standardise the rest of the experiment by using a single layered MLP with 50 neurons.

At this point, I decided to add a hyperparameter “learning rate” to see its effect on the model. I selected four different learning rate, 0.001, 0.01, 0.1 and 0.5 to put into test. The outcome will return the best fitted learning rate model judging from the best validation error.

```
Best Learning Rate: 0.01
Training Error: 0.48
Best Validation Error: 0.38
```



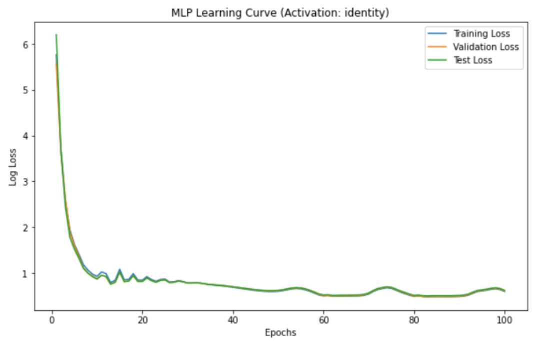
```
Test Error: 0.21
```

Fig 3. MLP (Learning rate = 0.01)

Result shows that the best model is the one with 0.01 learning rate. The model fitted well but the training and validation error was high. Surprisingly, test error went down.

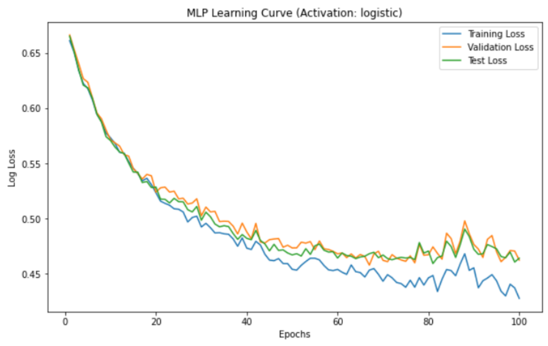
I also tried to change the activation functions to Identity Activation, Logistic Activation, Tanh Activation and ReLU Activation. Different activation functions do influence the model's performance. Identity and ReLU activation showed the best training performance and good generalizations. Logistic and tanh activations have risk of overfitting.

```
Activation Function: identity
Training Error: 0.26
Cross-Validation Error: 0.26
```



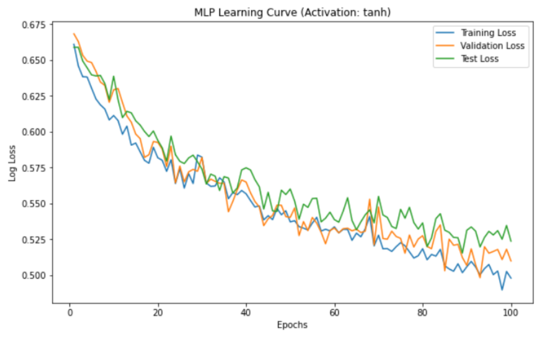
```
Test Error: 0.26
```

```
Activation Function: logistic
Training Error: 0.20
Cross-Validation Error: 0.22
```



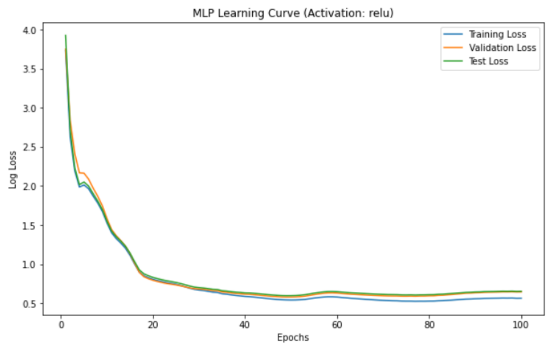
```
Test Error: 0.23
```

```
Activation Function: tanh
Training Error: 0.25
Cross-Validation Error: 0.26
```



```
Test Error: 0.27
```

```
Activation Function: relu
Training Error: 0.24
Cross-Validation Error: 0.27
```

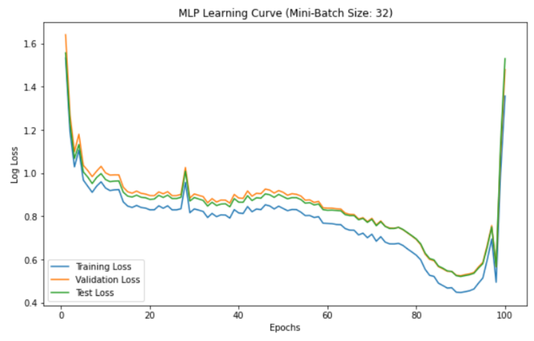


```
Test Error: 0.28
```

Fig 4. MLP with different activation functions

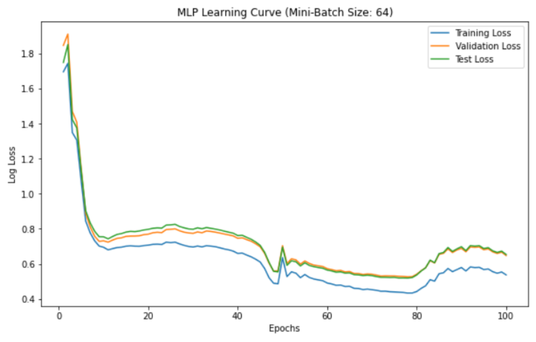
Lastly, by changing the mini-batch size to 32, 64 and 128. We can see the improvement of the fitness of the model with increasing number of batch sizes. Batch size 128 gives the best generalisation performance with lowest risk of overfitting.

```
Mini-Batch Size: 32
Training Error: 0.36
Cross-Validation Error: 0.38
```



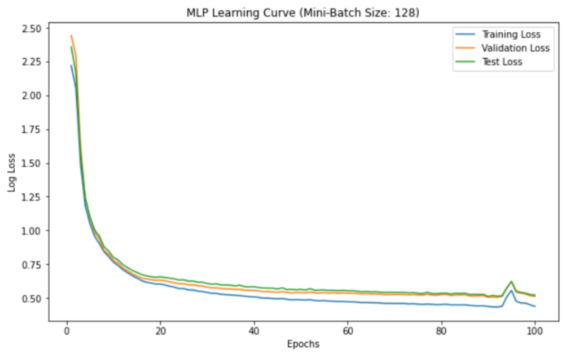
```
Test Error: 0.38
--------------------------------------------------------------------------------
```

```
Mini-Batch Size: 64
Training Error: 0.23
Cross-Validation Error: 0.26
```



```
Test Error: 0.28
--------------------------------------------------------------------------------
```

```
Mini-Batch Size: 128
Training Error: 0.20
Cross-Validation Error: 0.22
```



```
Test Error: 0.23
--------------------------------------------------------------------------------
```

Fig 5. MLP with different Mini-batch sizes