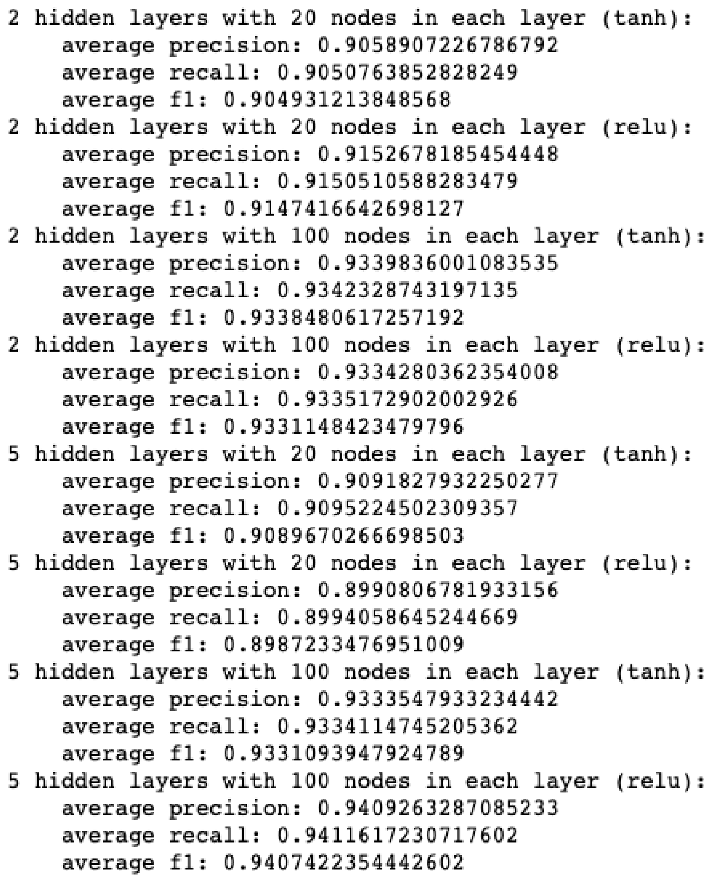
CS 5644: Assignment 4 report

1.1 Question 1 Multilayer Perceptron Classifier

MLP classifier is created with the argument max_iter = 1000 and solver = lbfgs. It uses MinMaxScaler for scaling and 5-fold for cross-validation. The following are the results of different MLP architectures in different scenarios.



```
2 hidden layers with 20 nodes in each layer (tanh):
    average precision: 0.9058907226786792
    average recall: 0.9050763852828249
    average f1: 0.904931213848568
2 hidden layers with 20 nodes in each layer (relu):
    average precision: 0.9152678185454448
    average recall: 0.9150510588283479
    average f1: 0.9147416642698127
2 hidden layers with 100 nodes in each layer (tanh):
    average precision: 0.9339836001083535
    average recall: 0.9342328743197135
    average f1: 0.9338480617257192
2 hidden layers with 100 nodes in each layer (relu):
    average precision: 0.9334280362354008
    average recall: 0.9335172902002926
    average f1: 0.9331148423479796
5 hidden layers with 20 nodes in each layer (tanh):
    average precision: 0.9091827932250277
    average recall: 0.9095224502309357
    average f1: 0.9089670266698503
5 hidden layers with 20 nodes in each layer (relu):
    average precision: 0.8990806781933156
    average recall: 0.8994058645244669
    average f1: 0.8987233476951009
5 hidden layers with 100 nodes in each layer (tanh):
    average precision: 0.9333547933234442
    average recall: 0.9334114745205362
    average f1: 0.9331093947924789
5 hidden layers with 100 nodes in each layer (relu):
    average precision: 0.9409263287085233
    average recall: 0.9411617230717602
    average f1: 0.9407422354442602
```

Scenario \ Activation funs	tanh	ReLU

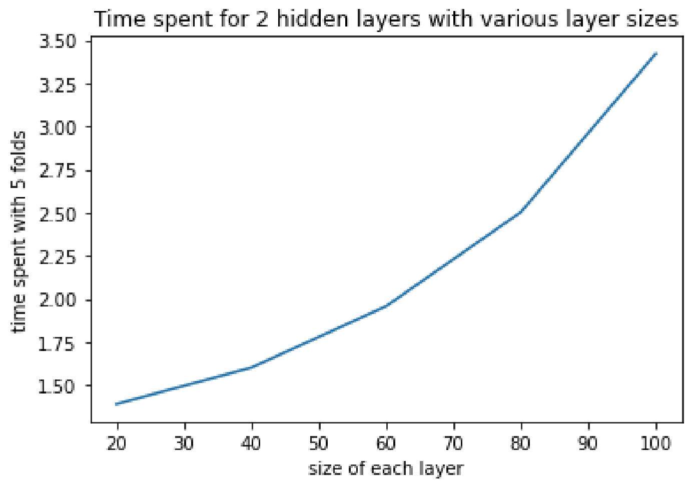
2 hidden layers with 20 nodes in each layer	precosion = 0.906
	recall =0.905
	f1 = 0.905
		precosion = 0.915
		recall = 0.915
		f1 = 0.91

2 hidden layers with 100 nodes in each layer	precosion = 0.934
	recall = 0.934
	f1 = 0.934
		precosion = 0.933
		recall = 0.934
		f1 = 0.933

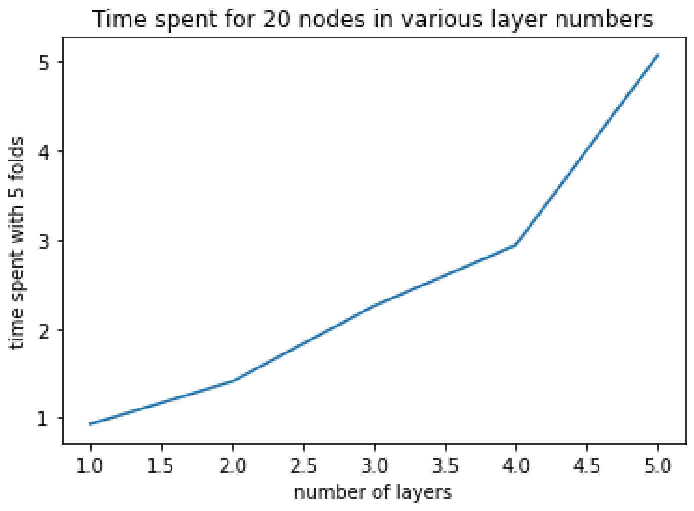
5 hidden layers with 20 nodes in each layer	precosion = 0.91
	recall = 0.91
	f1 = 0.91
		precosion = 0.899
		recall = 0.899
		f1 = 0.899

5 hidden layers with 100 nodes in each layer	precosion = 0.933
	recall = 0.933
	f1 = 0.933
		precosion = 0.941
		recall = 0.941
		f1 = 0.941

Observe the time spent to fit each model as the size of layers increases for various layer sizes. 5-fold cross-validation and ReLU activation function are applied. The number of layers is set to 2. The graph of the result is presented as follows.



Observe the time spent to fit each model as the number of layers increases for various numbers of layers. 5-fold cross-validation and ReLU activation function are applied. The size of layers is set to 20. See the graph of the result below.



From the results, we see that either we increase the size of each layer or increase the number of layers, the time it takes to fit the model mounting. The reason is that the more nodes we add to the model, the more calculations are needed for training.

If we stop using MinMaxScaler for simple/smallish networks, the input variables may have different scales. In that case, large input values can result in a model that learns large weight values. Consequently, the model may suffer from poor performance during learning and sensitivity to input values resulting in higher generalization errors.

1.2 Question 2

When the learning rate is set to 0.5, the performance will oscillate more and converge more quickly over training epochs than when the learning rate is set to 0.001. This is because the learning rate determines how much an updating step influences the current value of the weights, that is, a larger learning rate causes a bigger divergence of weights.