Indian Institute of Technology Delhi

COL 774 - Machine Learning

Assignment 3

Decision Trees and Neural Network Architecture

Garv Nagori
2021CS10549

Contents

§1 Part 1. Decision Trees

§1.1 a) Simple Decision Trees

In this part, I implemented a Decision Tree Model from scratch in Python. There are 10 features some of which are categorical while some are continuous-valued.
For categorical features, each node is split into k children where k is the possible number of values of the feature being split on.
For continuous-valued features, each node is split based on the median of all the possible values that data can have. Therefore, there are only 2 children present here.
The decision of the attribute to be split is based on the maximum loss in entropy achieved if we split using that feature. This also allows continuous-valued variables to be split again since the median always changes.
We can also control the maximum depth of the decision tree. I trained the model for depths of values $5, 10, 15, 20$ and 25. The following are my observations:

Depth: 5

Training Set	Testing Set
Correct = 6924	Correct = 554
Incorrect = 903	Incorrect = 413
Accuracy = 88.46%	Accuracy = 57.29%

Depth: 10

Training Set	Testing Set
Correct = 7807	Correct = 580
Incorrect = 20	Incorrect = 387
Accuracy = 99.74%	Accuracy = 59.98%

Depth: 15

Training Set	Testing Set
Correct = 7813	Correct = 580
Incorrect = 14	Incorrect = 387
Accuracy = 99.82%	Accuracy = 59.98%

Depth: 20

Training Set	Testing Set
Correct = 7813	Correct = 580
Incorrect = 14	Incorrect = 387
Accuracy = 99.82%	Accuracy = 59.98%

Depth: 25

Training Set	Testing Set
Correct = 7813	Correct = 580
Incorrect = 14	Incorrect = 387
Accuracy = 99.82%	Accuracy = 59.98%

Always 0 Prediction

Training Set	Testing Set
Correct = 3887	Correct = 487
Incorrect = 3940	Incorrect = 480
Accuracy = 49.66%	Accuracy = 50.36%

Always 1 Prediction

Training Set	Testing Set
Correct = 3940	Correct = 480
Incorrect = 3887	Incorrect = 487
Accuracy = 50.34%	Accuracy = 49.64%

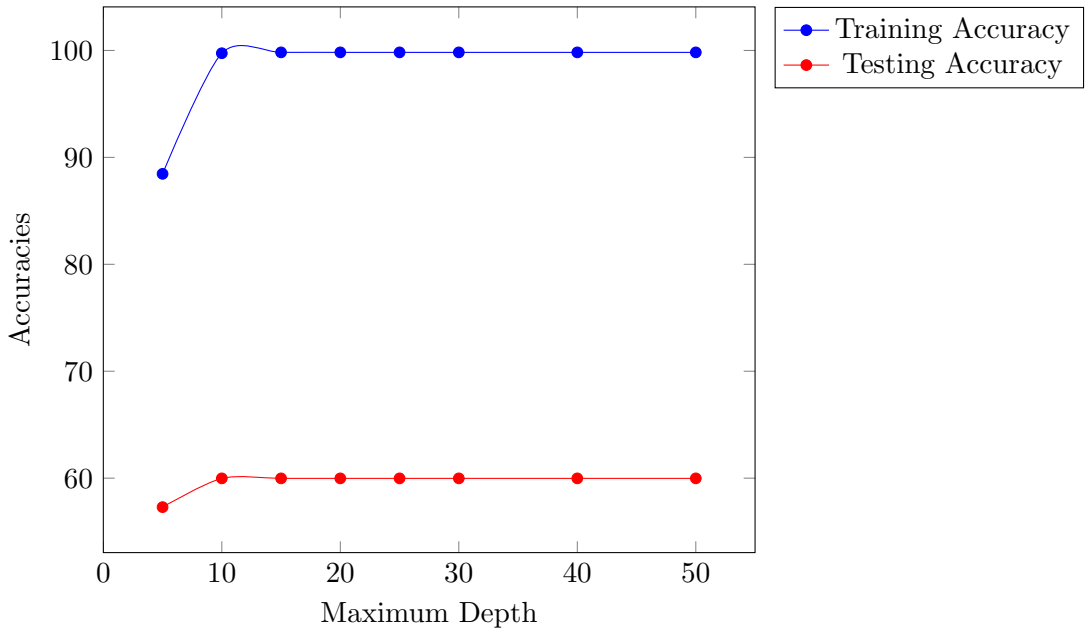


Figure 1: **Simple Decision Trees**

The testing accuracy is around 60% which is better than guessing randomly but still not enough to justify this model is great.
We can see that on increasing depth of the decision tree doesn't affect testing accuracy much but the training accuracy reaches almost 100%. This shows that our model is overfitting on the training data. We try to solve these issues with our model.

§1.2 b) One-Hot Encoding

In this part, we create k variables if there are k possible values of a particular categorical feature. All variables are assigned 0 or 1 based on if the variable is equal to a particular value or not.
I then ran the same Decision Tree Model as before but with these new types of features. Since there are extra features and a node can only split based on two, the maximum depth required will be greater but the time required to train will more or less be the same.
I trained the model for depths of values $15, 25, 35$ and 45. The following are my observations:

Depth: 15

Training Set		Testing Set	
Correct	= 5571	Correct	= 553
Incorrect	= 2256	Incorrect	= 414
Accuracy	= 71.18%	Accuracy	= 57.19%

Depth: 25

Training Set		Testing Set	
Correct	= 6615	Correct	= 608
Incorrect	= 1212	Incorrect	= 359
Accuracy	= 84.52%	Accuracy	= 62.87%

Depth: 35

Training Set		Testing Set	
Correct	= 7272	Correct	= 595
Incorrect	= 555	Incorrect	= 370
Accuracy	= 92.91%	Accuracy	= 61.74%

Depth: 45

Training Set		Testing Set	
Correct	= 7793	Correct	= 597
Incorrect	= 34	Incorrect	= 370
Accuracy	= 99.56%	Accuracy	= 61.74%

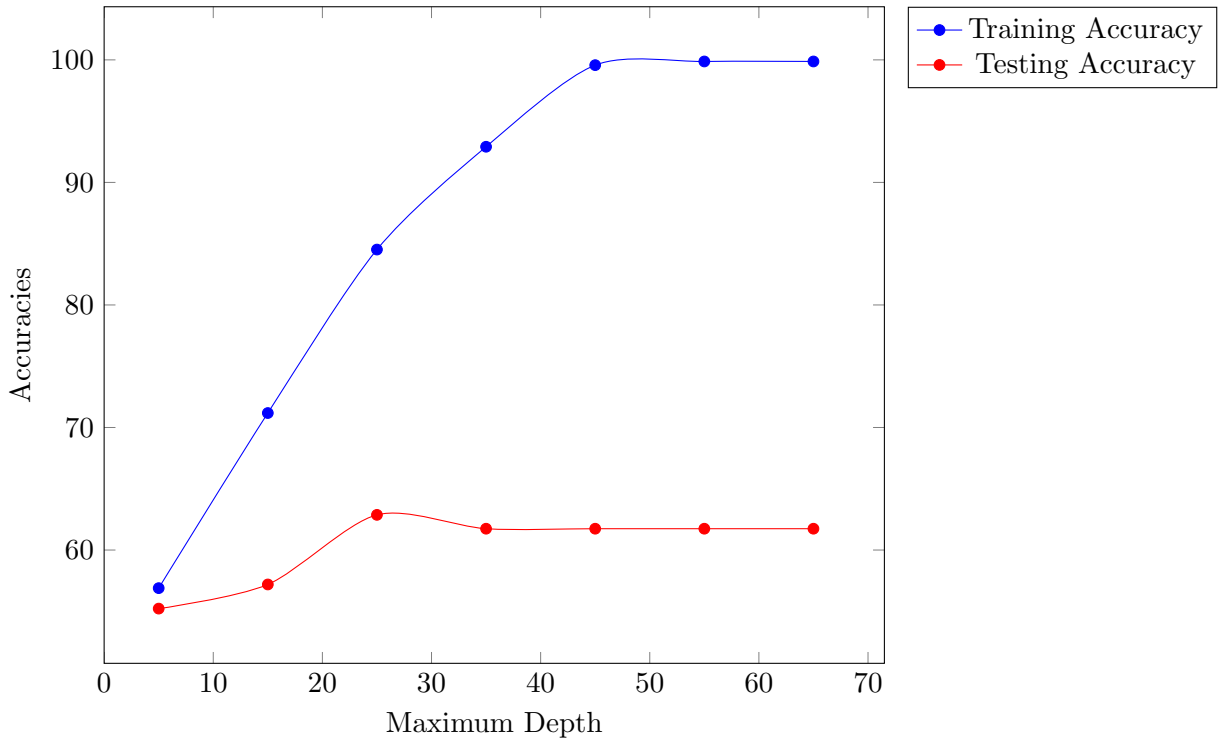


Figure 2: **Decision Trees with One Hot Encoding**

The accuracies for these increase over the Simple Decision Trees but it is also prone to overfitting at higher depths. However, this still shows progress in the right direction since the accuracies have increased by about 2%.

§1.3 c) Post Pruning using Validation Set

In this part, I now make use of the Validation Set to perform Post-Pruning on our tree.
Post Pruning is done as follows:

- Pick a node which is not a leaf and make it into a leaf node.
- Find validation accuracies over all such nodes.
- Make the node who increases the validation accuracy the most into a leaf node
- Repeat the process until every node pruning decreases the accuracy.

This step is done to ensure no overfitting to the training data.
I trained the model for depths of values $15, 25, 35$ and 45. The following are my observations:

Depth: 15

Training Set	Testing Set	Validation Set
Correct $= 5056$	Correct $= 545$	Correct $= 550$
Incorrect $= 2771$	Incorrect $= 422$	Incorrect $= 320$
Accuracy $= 64.60\%$	Accuracy $= 56.36\%$	Accuracy $= 63.22\%$

Depth: 25

Training Set	Testing Set	Validation Set
Correct = 5767	Correct = 591	Correct = 592
Incorrect = 2060	Incorrect = 376	Incorrect = 278
Accuracy = 73.68%	Accuracy = 61.12%	Accuracy = 68.05%

Depth: 35

Training Set	Testing Set	Validation Set
Correct = 6171	Correct = 585	Correct = 620
Incorrect = 1656	Incorrect = 382	Incorrect = 250
Accuracy = 78.84%	Accuracy = 60.50%	Accuracy = 71.26%

Depth: 45

Training Set	Testing Set	Validation Set
Correct = 6501	Correct = 588	Correct = 653
Incorrect = 1326	Incorrect = 379	Incorrect = 217
Accuracy = 83.06%	Accuracy = 60.81%	Accuracy = 75.06%

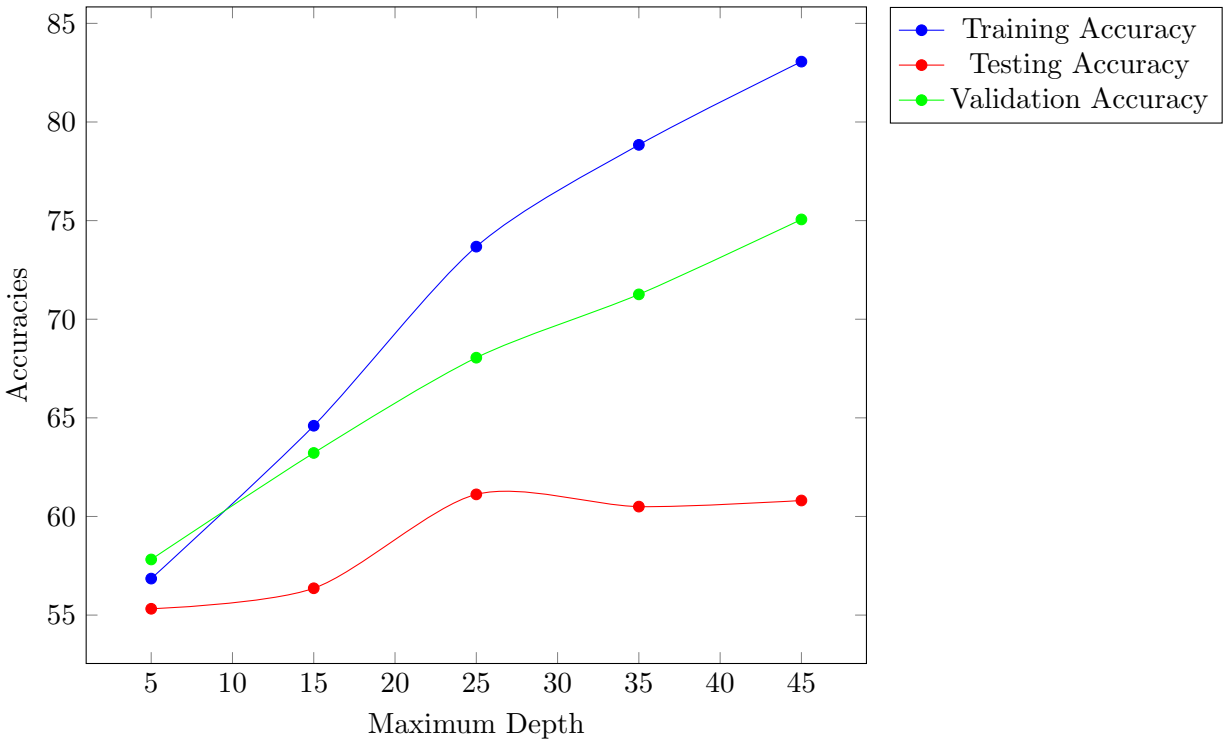
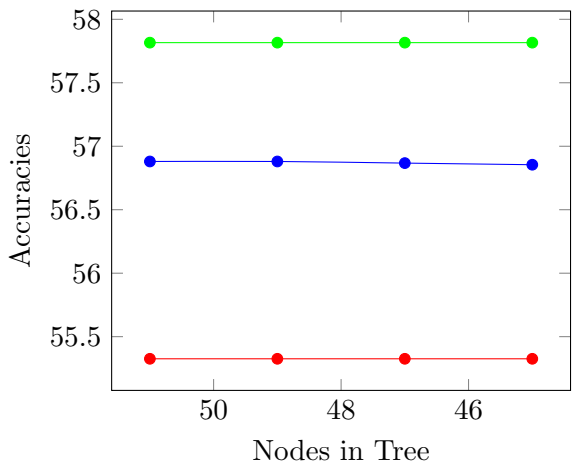
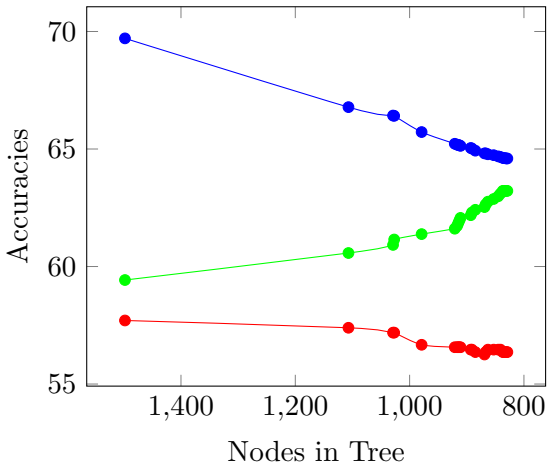


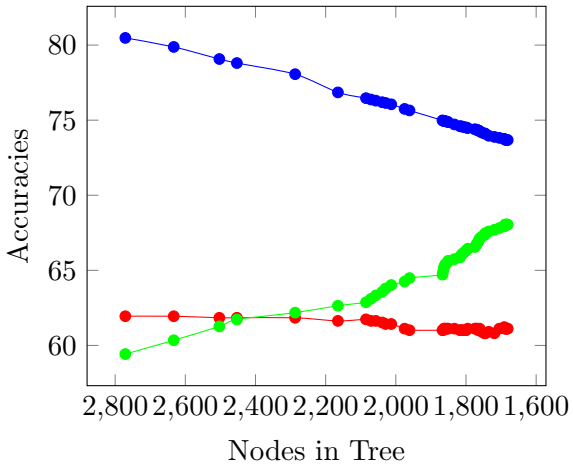
Figure 3: **Decision Trees after Post Pruning**



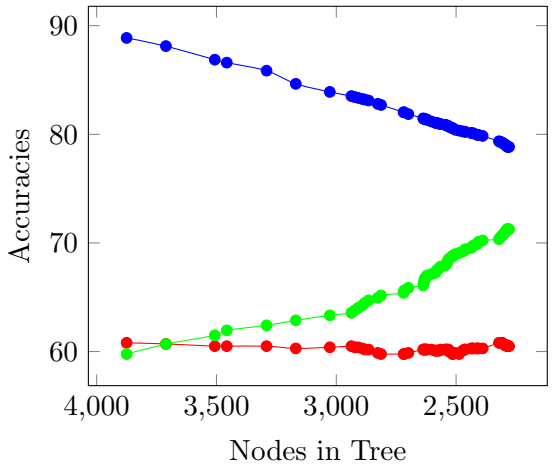
(a) Depth = 5



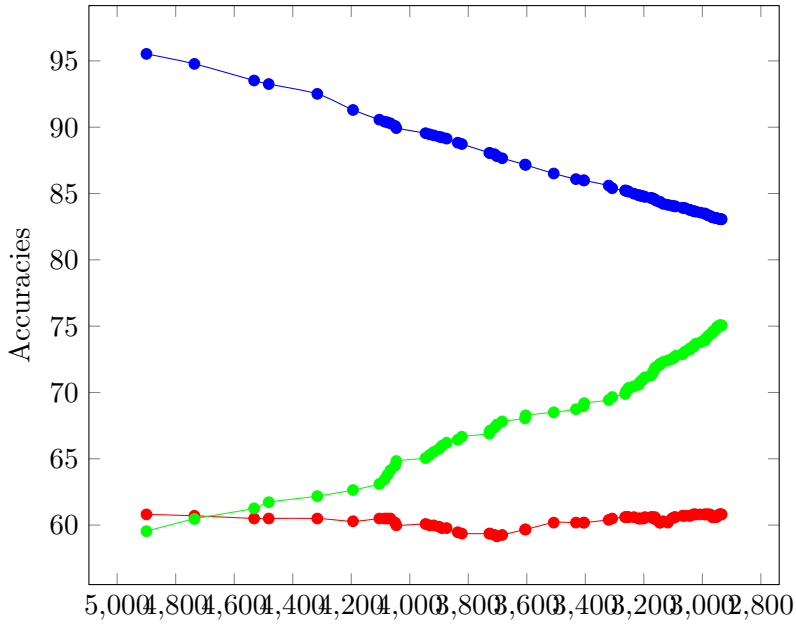
(b) Depth = 15



(c) Depth = 25



(d) Depth = 35



(e) Depth = 45

Figure 4: Accuracies vs Number of Nodes in Decision Tree

We can see that Training Accuracy decreases as we prune more and more nodes but we increase the Validation and Testing Accuracies.
As we increase the size of pruning, the Validation Accuracy dramatically increases and is around 75% for depth = 45.
Thus, this method is a good way of not overfitting to the training dataset and use pruning to increase accuracy on data never seen before.

§1.4 d) SciKit Learn's Decision Trees

in this part, I use SciKit Learn to train Decision Trees on the same training data set.
First, I vary the maximum depth as $15, 25, 35$ and 45 and train. These are the accuracies I found.

Depth: 15

Training Set	Testing Set	Validation Set
Correct = 6461	Correct = 595	Correct = 567
Incorrect = 1366	Incorrect = 372	Incorrect = 303
Accuracy = 82.55%	Accuracy = 61.53%	Accuracy = 65.17%

Depth: 25

Training Set	Testing Set	Validation Set
Correct = 7733	Correct = 602	Correct = 545
Incorrect = 94	Incorrect = 365	Incorrect = 325
Accuracy = 98.80%	Accuracy = 62.25%	Accuracy = 62.64%

Depth: 35

Training Set	Testing Set	Validation Set
Correct = 7824	Correct = 606	Correct = 542
Incorrect = 3	Incorrect = 361	Incorrect = 328
Accuracy = 99.96%	Accuracy = 62.67%	Accuracy = 62.30%

Depth: 45

Training Set	Testing Set	Validation Set
Correct = 7827	Correct = 611	Correct = 549
Incorrect = 0	Incorrect = 356	Incorrect = 321
Accuracy = 100%	Accuracy = 63.18%	Accuracy = 63.10%

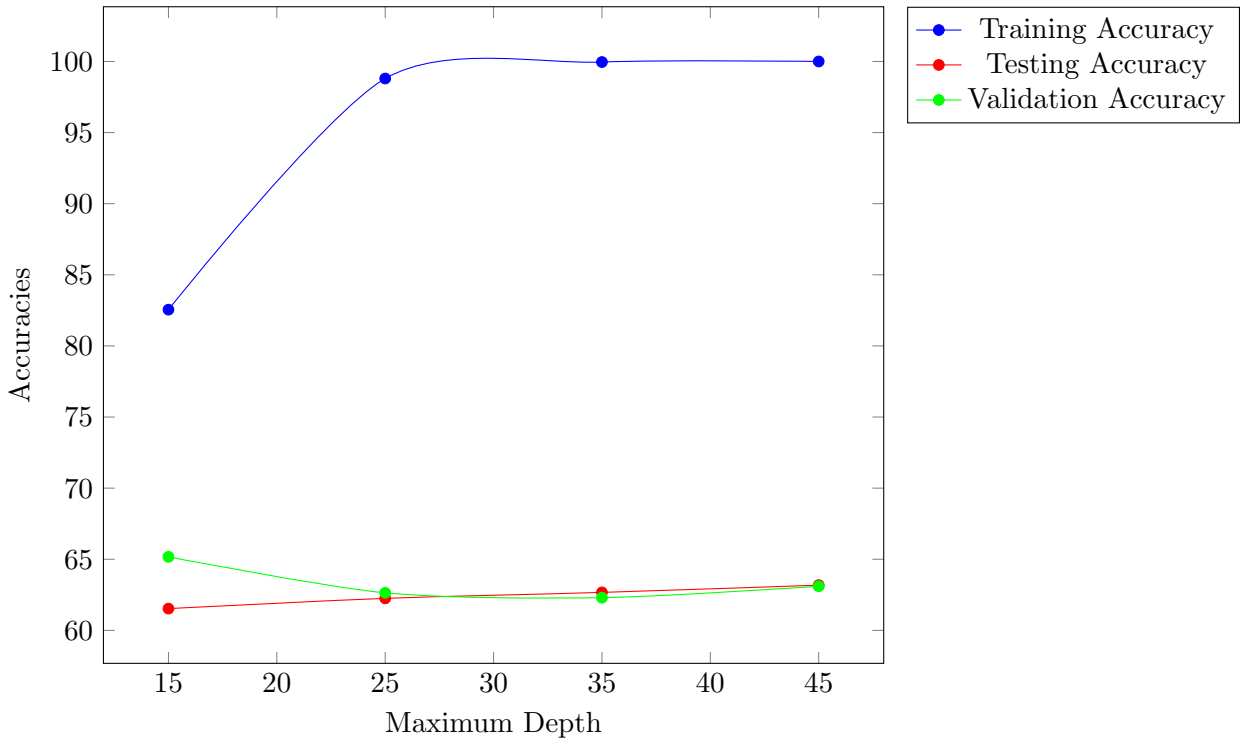


Figure 5: **SciKit Learn's Decision Trees for Varying Maximum Depths**

The validation accuracy is best for depth 15.
Then, I varied the value of *ccp_alpha* parameters as $0.001, 0.01, 0.1$ and 0.2. These are my observations.

CCP Alpha: 0.001

Training Set	Testing Set	Validation Set
Correct = 5425	Correct = 611	Correct = 557
Incorrect = 2402	Incorrect = 356	Incorrect = 313
Accuracy = 69.31%	Accuracy = 63.18%	Accuracy = 64.02%

CCP Alpha: 0.01

Training Set	Testing Set	Validation Set
Correct = 4183	Correct = 501	Correct = 435
Incorrect = 3644	Incorrect = 466	Incorrect = 435
Accuracy = 53.44%	Accuracy = 51.81%	Accuracy = 50.00%

CCP Alpha: 0.1

Training Set	Testing Set	Validation Set
Correct = 3940	Correct = 480	Correct = 412
Incorrect = 3887	Incorrect = 487	Incorrect = 458
Accuracy = 50.34%	Accuracy = 49.64%	Accuracy = 47.36%

CCP Alpha: 0.2

Training Set	Testing Set	Validation Set
Correct = 3940	Correct = 480	Correct = 412
Incorrect = 3887	Incorrect = 487	Incorrect = 458
Accuracy = 50.34%	Accuracy = 49.64%	Accuracy = 47.36%

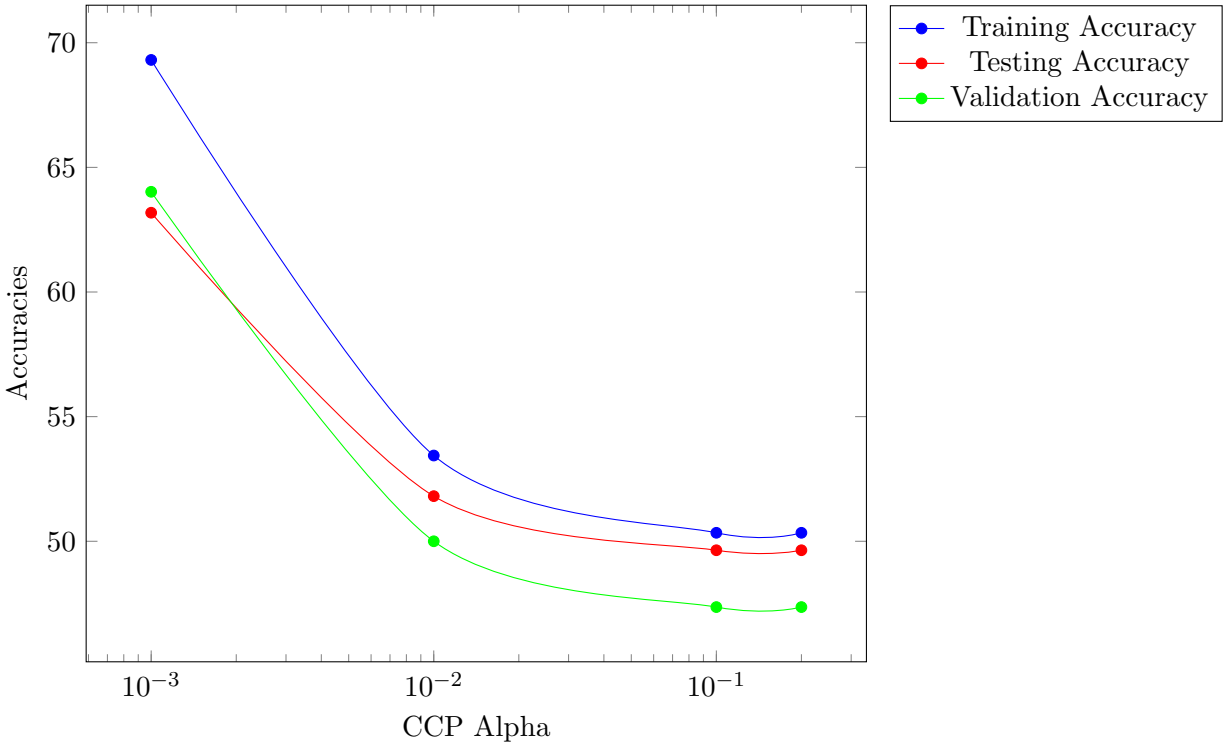


Figure 6: **SciKit Learn's Decision Trees with varying CCP Alpha**

The best value of CCP Alpha is 0.001 as evidenced by the validation set accuracy.
Here also we can see that SciKit Learn's implementation gives only slightly better accuracy than my own implementation although training it is much faster.
Also, we can see that Post Pruning is very effective since accuracy is much greater than even SciKit Learn's implementations.

§1.5 e) Random Forests

In this part, I trained Random Forests, which are Decision Trees constructed by bootstrapping the training data.
Each of the following hyperparameter configuration was used by performing a Grid Search to train the forest and then the Out-of-Bag Accuracy was used to select the best fit among them. The Training and Validation Accuracy are much better than previous parts.

Parameters	Performance
$n\_estimators = 350$	Out-of-Bag (OOB) Accuracy = 73.20%
$max\_features = 10$	Validation Accuracy = 71.84%
$min\_samples_split = 0.5$	Training Accuracy = 95.92%

§2 Part 2. Neural Networks

§2.1 a) Neural Network from First Principles

In this part, I wrote a class for a Neural Network from scratch. The inputs are the input and output neurons, the hidden layer sizes and their corresponding activation fucntions.
The neural network is trained using Backpropogation and we can even specify the batch size to use for Gradient Descent. The default value of Batch size is 32 for Mini Batch Graident Descent and the default value of learning rate is 0.01.

§2.2 b) Single Hidden Layer

In this part, I trained this neural network with a single hidden layer consisting of varying number of neurons. The following are my observations:

Number of Neurons: 1

Training Set

Classes	Precision	Recall	F1 Score
1	0.93	0.83	0.90
2	0.65	0.69	0.67
3	0.54	0.65	0.56
4	0.46	0.52	0.50
5	0.89	0.66	0.74

Accuracy: 68.21%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.94	0.85	0.92
2	0.62	0.67	0.64
3	0.57	0.64	0.54
4	0.48	0.58	0.51
5	0.83	0.67	0.72

Accuracy: 67.40%

Number of Neurons: 5

Training Set

Classes	Precision	Recall	F1 Score
1	0.95	0.93	0.93
2	0.74	0.79	0.76
3	0.62	0.65	0.63
4	0.47	0.59	0.53
5	0.88	0.69	0.75

Accuracy: 70.21%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.93	0.92	0.93
2	0.75	0.68	0.71
3	0.57	0.62	0.57
4	0.47	0.56	0.52
5	0.82	0.67	0.70

Accuracy: 70.10%

Number of Neurons: 10

Training Set

Classes	Precision	Recall	F1 Score
1	0.95	0.93	0.94
2	0.79	0.81	0.80
3	0.63	0.66	0.65
4	0.49	0.58	0.54
5	0.85	0.70	0.75

Accuracy: 74.56%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.93	0.92	0.93
2	0.75	0.68	0.71
3	0.57	0.62	0.57
4	0.47	0.56	0.52
5	0.82	0.67	0.70

Accuracy: 71.70%

Number of Neurons: 50

Training Set

Classes	Precision	Recall	F1 Score
1	0.96	0.95	0.96
2	0.85	0.83	0.84
3	0.66	0.73	0.69
4	0.60	0.64	0.62
5	0.85	0.76	0.80

Accuracy: 77.17%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.97	0.96	0.96
2	0.83	0.80	0.81
3	0.61	0.72	0.66
4	0.62	0.61	0.61
5	0.81	0.75	0.78

Accuracy: 75.60%

Number of Neurons: 100

Training Set

Classes	Precision	Recall	F1 Score
1	0.97	0.96	0.96
2	0.86	0.83	0.84
3	0.67	0.74	0.70
4	0.61	0.65	0.63
5	0.85	0.77	0.81

Accuracy: 79.59%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.97	0.96	0.96
2	0.83	0.80	0.81
3	0.61	0.72	0.66
4	0.62	0.61	0.61
5	0.81	0.75	0.78

Accuracy: 78.10%

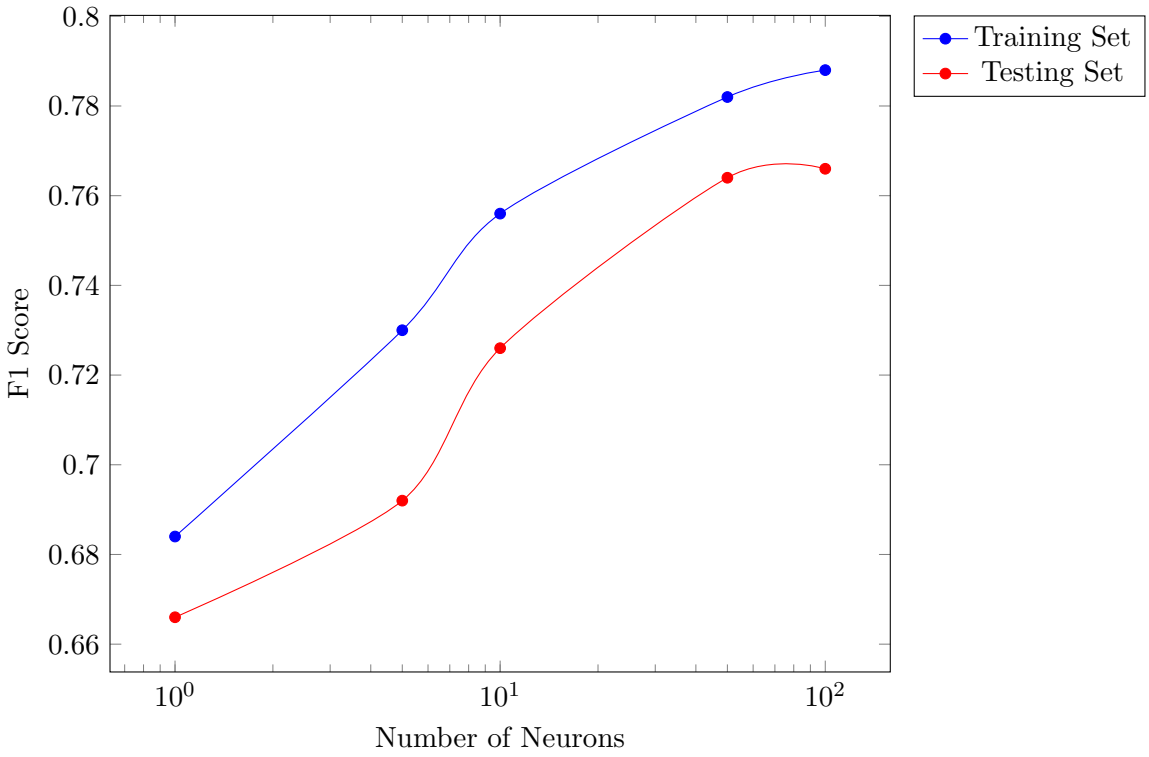


Figure 7: **F1 Score vs Number of Neurons**

The stopping criteria I chose was based on two factors. If the number of epochs became greater than some maximum threshold (1000 here), the neural network training was stopped. Or if the difference is loss became less than a certain threshold difference for fixed (10) consecutive iterations.
The F1 Score and the Accuracy increased as the number of neurons in the hidden layer increased. The increase is higher when the number of neurons change in the beginning but as we approach even higher values the difference is less striking. Going from 50 to 100 neurons only changed the values ever so slightly.

§2.3 c) Depth of Hidden Layers

In this part, I experimented with the depth of the neural network. I increased the depth from 1 to 4 and with each increase I decreased the number of neurons in the next layer to half of its previous layer. The architectures I tested with was:

- 1 Hidden Layer with 512 nodes.
- 2 Hidden Layers with 512 and 256 nodes.
- 3 Hidden Layers with 512, 256 and 128 nodes.
- 4 Hidden Layers with 512, 256, 128 and 64 nodes.

The following are my observations:

Depth: 1

Training Set

Classes	Precision	Recall	F1 Score
1	0.97	0.94	0.95
2	0.84	0.82	0.83
3	0.63	0.72	0.67
4	0.59	0.62	0.60
5	0.84	0.76	0.80

Accuracy: 77.34%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.96	0.96	0.96
2	0.80	0.78	0.89
3	0.59	0.68	0.63
4	0.63	0.58	0.60
5	0.78	0.74	0.76

Accuracy: 76.20%

Depth: 2

Training Set

Classes	Precision	Recall	F1 Score
1	0.96	0.99	0.97
2	0.87	0.90	0.89
3	0.73	0.78	0.76
4	0.63	0.66	0.64
5	0.88	0.76	0.81

Accuracy: 81.87%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.99	0.99	0.98
2	0.83	0.86	0.85
3	0.68	0.74	0.71
4	0.47	0.56	0.52
5	0.82	0.67	0.70

Accuracy: 79.90%

Depth: 3

Training Set

Classes	Precision	Recall	F1 Score
1	0.97	0.99	0.98
2	0.90	0.94	0.92
3	0.80	0.81	0.80
4	0.62	0.69	0.65
5	0.89	0.76	0.82

Accuracy: 84.21%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.93	0.92	0.93
2	0.75	0.68	0.71
3	0.57	0.62	0.57
4	0.47	0.56	0.52
5	0.82	0.67	0.70

Accuracy: 81.70%

Depth: 4

Training Set

Classes	Precision	Recall	F1 Score
1	0.96	0.98	0.97
2	0.89	0.93	0.91
3	0.80	0.79	0.80
4	0.60	0.69	0.65
5	0.89	0.76	0.82

Accuracy: 83.82%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.97	0.96	0.96
2	0.83	0.80	0.81
3	0.61	0.72	0.66
4	0.62	0.61	0.61
5	0.81	0.75	0.78

Accuracy: 82.30%

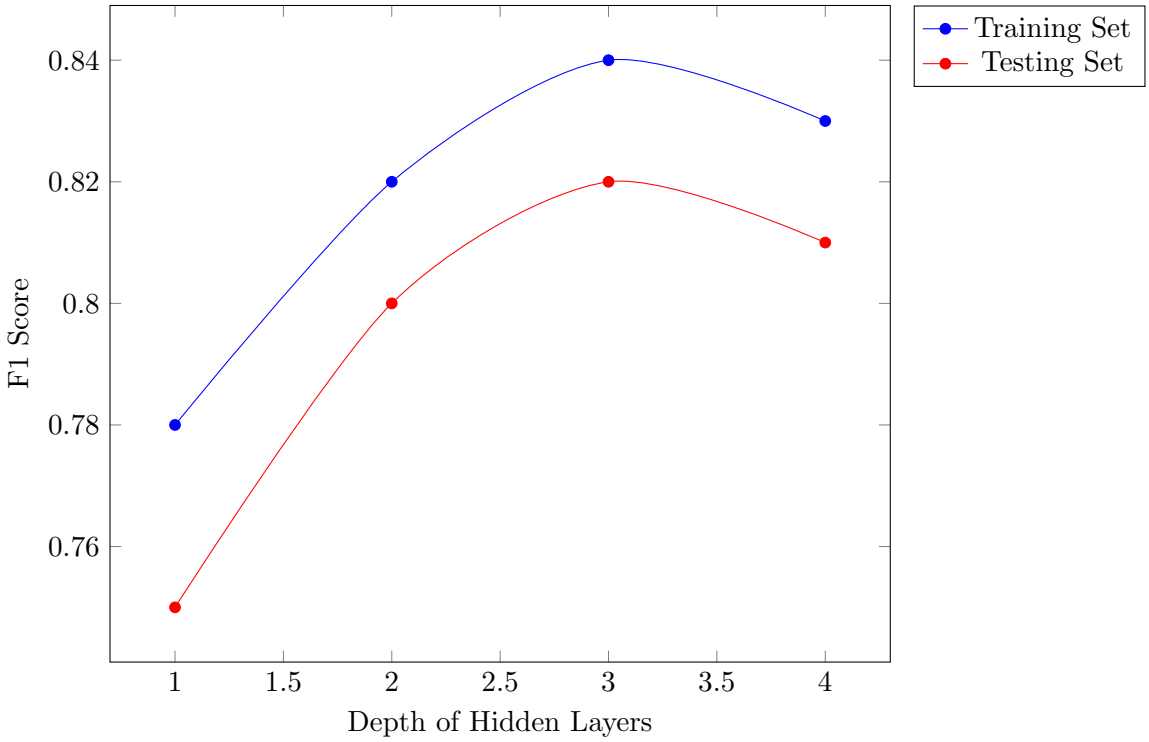


Figure 8: **F1 Score vs Depth of Hidden Layers**

I used the same stopping criteria as before, maximum epochs = 100 and consecutive 10 with loss in error equal to 0.001.
The accuracy and performance of the model increase with an increase in depth generally. However, the time to train and converge also increases dramatically with each increase in depth.
For depth = 4, it did not converge even in 1000 epochs and hence, the accuracy did not increase that much but was in fact lesser than with depth = 3 as well.

§2.4 d) Adaptive Learning Rate

In this part, I experimented with an adaptive learning rate which slowly decreases with the number of epochs. The idea is that as the network reaches the minimum, we decrease the learning rate so that it doesn't overshoot the minimum and start diverging again.
I particulary experimented with the following learning rate where $\eta_0 = 0.01$ and e is the current epoch

number.

$$\eta_e = \frac{\eta_0}{\sqrt{e}}$$

The following are my observations:

Depth: 1

Training Set

Classes	Precision	Recall	F1 Score
1	0.89	0.92	0.90
2	0.68	0.70	0.69
3	0.53	0.51	0.52
4	0.51	0.46	0.48
5	0.68	0.72	0.70

Accuracy: 66.43%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.90	0.90	0.90
2	0.66	0.64	0.65
3	0.49	0.48	0.48
4	0.42	0.43	0.42
5	0.58	0.60	0.59

Accuracy: 62.20%

Depth: 2

Training Set

Classes	Precision	Recall	F1 Score
1	0.90	0.94	0.92
2	0.73	0.74	0.73
3	0.57	0.56	0.56
4	0.53	0.48	0.50
5	0.70	0.74	0.72

Accuracy: 69.17%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.91	0.91	0.91
2	0.69	0.68	0.68
3	0.52	0.48	0.50
4	0.46	0.46	0.46
5	0.63	0.66	0.64

Accuracy: 65.10%

Depth: 3

Training Set

Classes	Precision	Recall	F1 Score
1	0.90	0.94	0.92
2	0.75	0.74	0.74
3	0.60	0.57	0.58
4	0.53	0.48	0.50
5	0.69	0.76	0.73

Accuracy: 70.09%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.91	0.93	0.92
2	0.70	0.67	0.69
3	0.53	0.49	0.51
4	0.44	0.48	0.46
5	0.66	0.67	0.69

Accuracy: 65.80%

Depth: 4

Training Set

Classes	Precision	Recall	F1 Score
1	0.92	0.96	0.94
2	0.78	0.78	0.78
3	0.63	0.60	0.61
4	0.57	0.52	0.54
5	0.73	0.80	0.76

Accuracy: 73.10%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.94	0.94	0.94
2	0.73	0.73	0.73
3	0.57	0.54	0.56
4	0.46	0.45	0.46
5	0.64	0.70	0.67

Accuracy: 68.10%

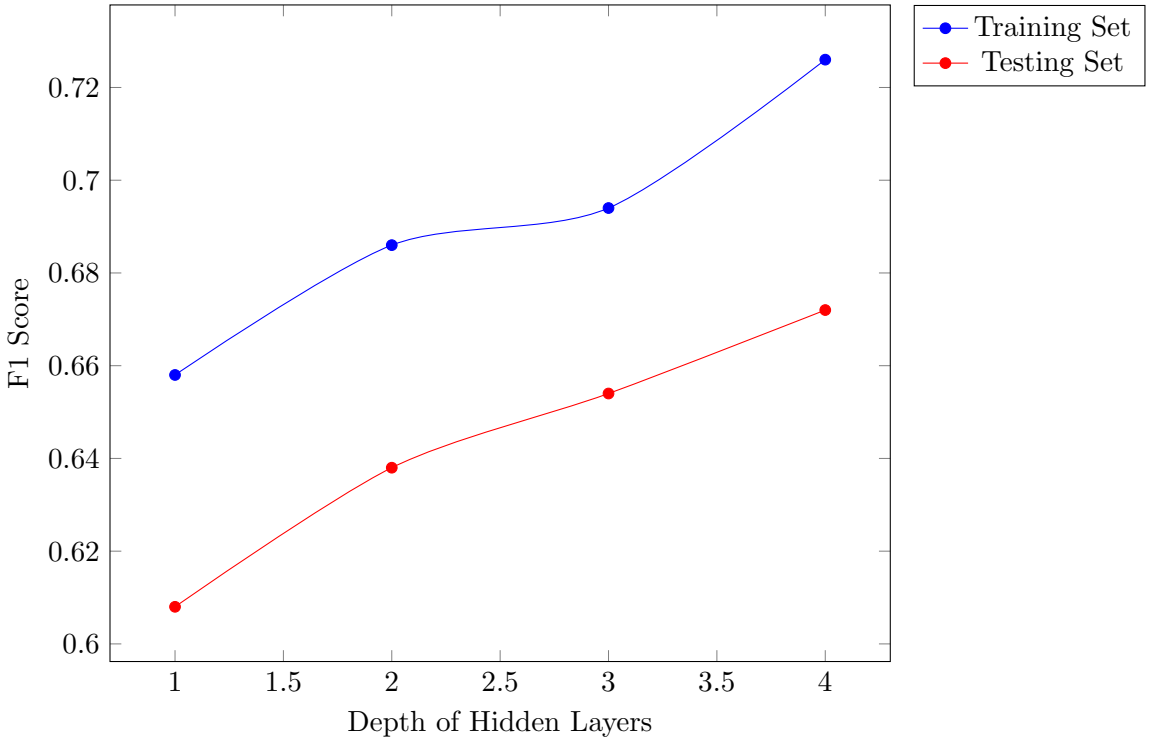


Figure 9: **F1 Score vs Depth of Hidden Layers with Adaptive Learning Rate**

We can see that the accuracies and f1 score have decreased by adding an adaptive learning rate. This is because now the learning rate becomes very small as the number of epochs becomes larger and larger. Since the gradient of a sigmoid function is already small, it is insufficient to drive it closer and closer towards the minimum.
This suggests that an adaptive learning rate shouldn't be used with sigmoid whose derivative becomes very small far apart from zero.

§2.5 e) ReLU Activation Function

To solve the problem arising in the above section of gradient becoming very less, I tried another activation function commonly used.
I used the Rectified Linear Unit or 'ReLU' Activation function instead of sigmoid. It is given by.

$$ReLU(x) = max(0, x)$$

This function is very close to a linear function yet induces just the right amount of non-linearity which makes this an ideal function. Although this function isn't differentiable at $x = 0$, we use the concept of subgradients which essentially allows us to pick any value between 0 and 1.
This function's gradient is 1 if x is positive and 0 if it is negative. This allows us to use the adaptive learning rate effectively since the gradient doesn't determine how far the input is from 0.
I trained my neural network again for the same set of architectures. The following are my observations:

Depth: 1

Training Set

Classes	Precision	Recall	F1 Score
1	0.91	0.89	0.90
2	0.73	0.72	0.73
3	0.62	0.61	0.61
4	0.52	0.59	0.55
5	0.80	0.74	0.77

Accuracy: 71.89%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.92	0.91	0.91
2	0.69	0.70	0.70
3	0.56	0.61	0.58
4	0.56	0.52	0.54
5	0.71	0.70	0.71

Accuracy: 70.20%

Depth: 2

Training Set

Classes	Precision	Recall	F1 Score
1	0.98	0.97	0.98
2	0.87	0.89	0.88
3	0.76	0.76	0.76
4	0.63	0.67	0.65
5	0.84	0.79	0.81

Accuracy: 81.67%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.99	0.98	0.98
2	0.84	0.84	0.84
3	0.69	0.73	0.71
4	0.65	0.63	0.64
5	0.79	0.79	0.79

Accuracy: 78.90%

Depth: 3

Training Set

Classes	Precision	Recall	F1 Score
1	0.99	0.99	0.99
2	0.93	0.97	0.95
3	0.86	0.89	0.87
4	0.76	0.80	0.78
5	0.94	0.83	0.88

Accuracy: 88.79%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.98	0.99	0.99
2	0.88	0.90	0.89
3	0.71	0.83	0.77
4	0.75	0.68	0.72
5	0.87	0.81	0.84

Accuracy: 82.40%

Depth: 4

Training Set

Classes	Precision	Recall	F1 Score
1	0.99	1.00	0.99
2	0.96	0.98	0.97
3	0.92	0.95	0.93
4	0.88	0.90	0.89
5	0.91	0.91	0.94

Accuracy: 93.23%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.99	0.99	0.99
2	0.89	0.93	0.91
3	0.73	0.84	0.78
4	0.76	0.66	0.71
5	0.84	0.81	0.83

Accuracy: 84.20%

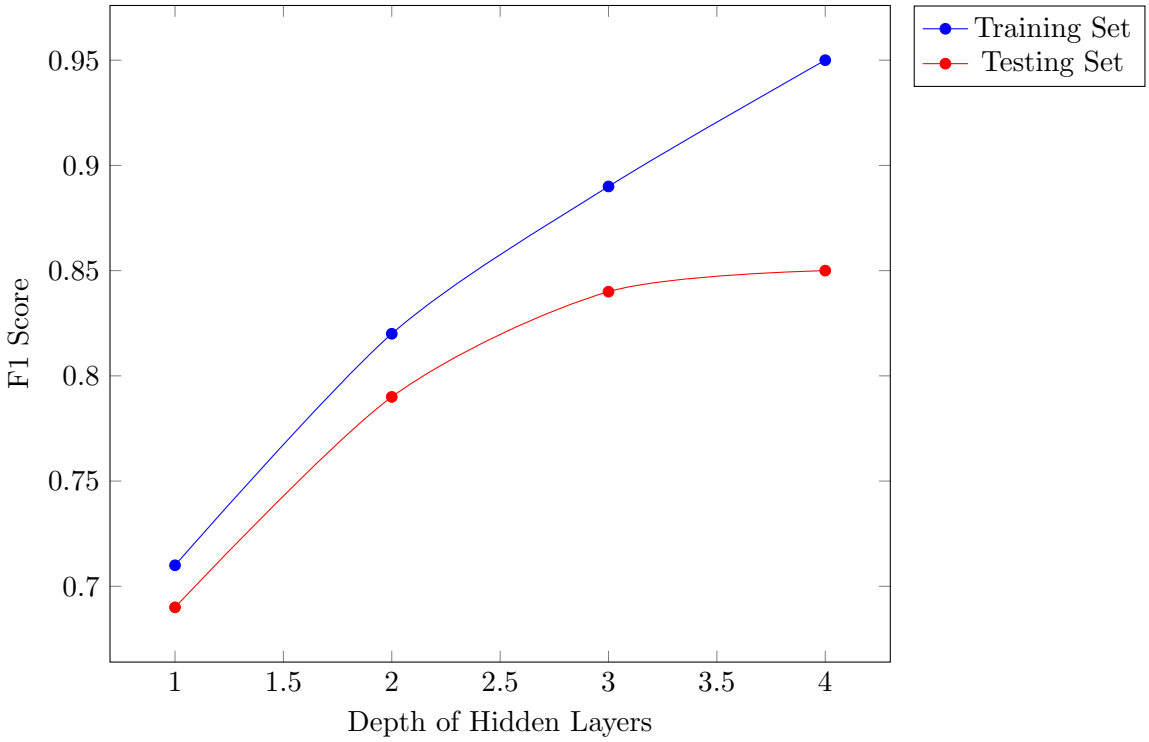


Figure 10: **F1 Score vs Depth of Hidden Layers with ReLU Activation Function**

We can see that ReLU performs much better than Sigmoid with Adaptive Learning rate. This is because It has a much larger gradient and so moves very quickly towards the minimum initially and the learning rate slows it down as it reaches the minimum point. It achieves convergence within 1000 epochs.

§2.6 f) SciKit Learn's MLP CLassifier

In this part, I use SciKit Learn's MLP Classifier for Neural Network. I trained 4 models corresponding to the different types of architectures of hidden layers used above.

The following are my observations:

Depth: 1

Training Set

Classes	Precision	Recall	F1 Score
1	0.86	0.89	0.88
2	0.68	0.68	0.68
3	0.54	0.53	0.54
4	0.50	0.45	0.47
5	0.70	0.74	0.72

Accuracy: 65.02%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.90	0.90	0.90
2	0.65	0.64	0.65
3	0.54	0.51	0.52
4	0.49	0.51	0.50
5	0.69	0.70	0.69

Accuracy: 64.60%

Depth: 2

Training Set

Classes	Precision	Recall	F1 Score
1	0.87	0.90	0.88
2	0.69	0.70	0.69
3	0.56	0.54	0.55
4	0.50	0.47	0.49
5	0.71	0.73	0.72

Accuracy: 66.39%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.90	0.91	0.90
2	0.69	0.67	0.68
3	0.58	0.53	0.55
4	0.48	0.52	0.50
5	0.68	0.68	0.68

Accuracy: 66.30%

Depth: 3

Training Set

Classes	Precision	Recall	F1 Score
1	0.87	0.90	0.89
2	0.70	0.68	0.69
3	0.56	0.55	0.55
4	0.50	0.47	0.48
5	0.70	0.75	0.72

Accuracy: 67.16%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.90	0.90	0.90
2	0.67	0.67	0.67
3	0.58	0.52	0.55
4	0.48	0.53	0.51
5	0.68	0.68	0.68

Accuracy: 66.90%

Depth: 4

Training Set

Classes	Precision	Recall	F1 Score
1	0.88	0.89	0.88
2	0.69	0.70	0.69
3	0.56	0.54	0.55
4	0.50	0.46	0.48
5	0.70	0.74	0.72

Accuracy: 67.84%

Testing Set

Classes	Precision	Recall	F1 Score
1	0.90	0.91	0.91
2	0.67	0.66	0.67
3	0.56	0.51	0.53
4	0.48	0.53	0.51
5	0.68	0.69	0.69

Accuracy: 67.40%

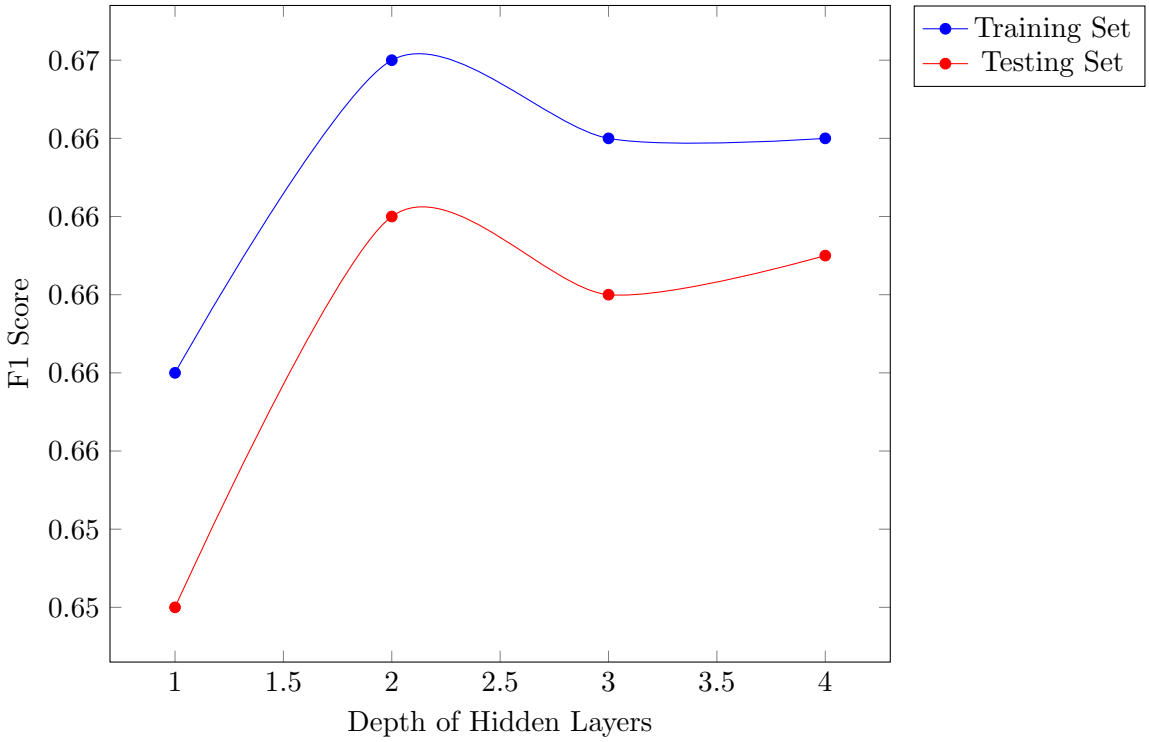


Figure 11: **F1 Score vs Depth of Hidden Layers for SciKit Learn's MLP Classifier**

Performance increases with depth generally, however the relative increase is very small. Compared to our own implementation the accuracies are much lower. This is because the default value of maximum epochs is 200 which is not enough to converged especially at higher depths, so the performance is not that impressive.

§3 Acknowledgements

I have used the style file from here[1] to produce this document.

[1]https://github.com/vEnhance/dotfiles/blob/main/texmf/tex/latex/evan/evan.sty