COL774 Assignment 3 Report

Name: *Anup Lal Nayak*
Entry Number: 2022CS51827

Part A: Decision Trees

(a) Decision Tree from Scratch

- Implemented a decision tree using mutual information as splitting criterion.
- Handled both **categorical** and **numerical** features.
- Used a recursive `build_tree()` method to construct the tree.
- Predictions made via depth-first traversal.
- Evaluated accuracy on train and test data for different max depths.

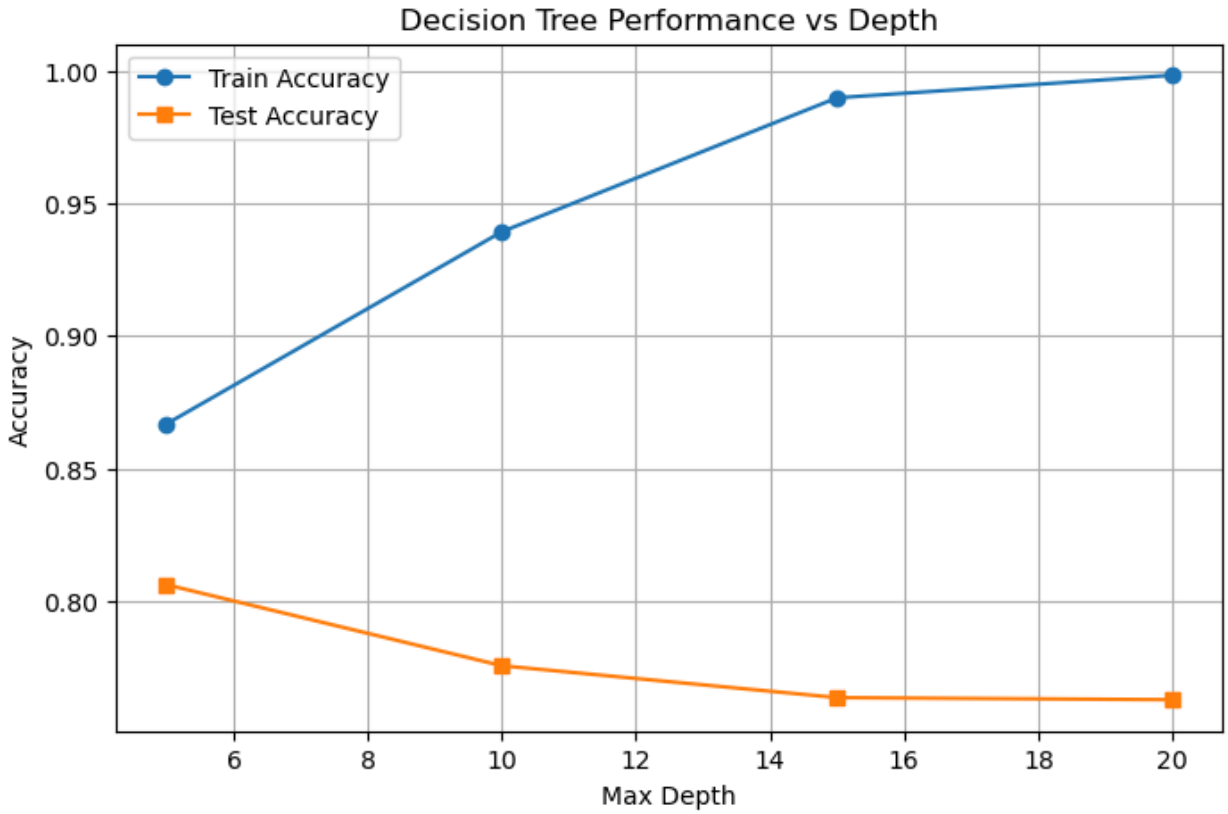
Depth vs Accuracy Table:

Max Depth	Train Accuracy	Test Accuracy
5	0.8669	0.8062
10	0.9393	0.7756
15	0.9898	0.7636

20	0.9983	0.7628

Test accuracy decreased as max depth increased.

Figure:



(b) One-Hot Encoded Decision Tree

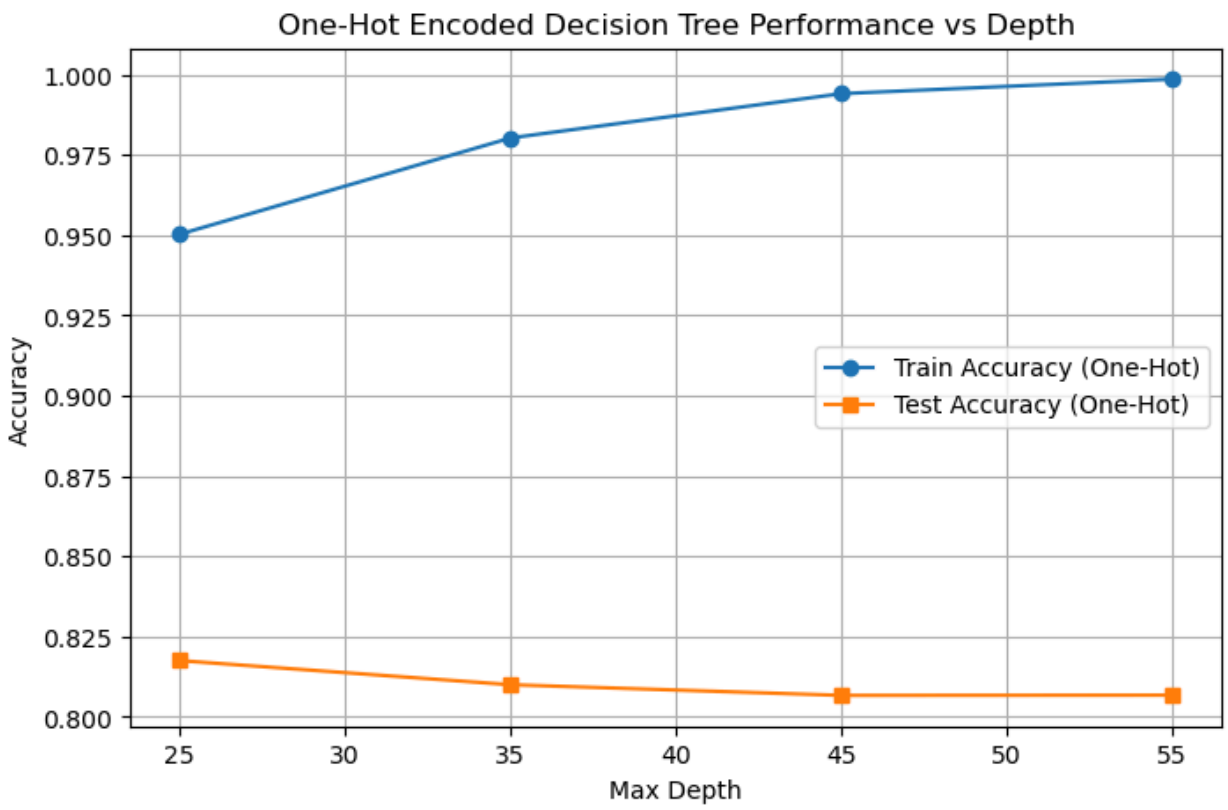
- Applied one-hot encoding to categorical attributes.
- Trained the same decision tree on encoded data.
- Used depths = {25, 35, 45, 55}.

Depth vs Accuracy Table (One-Hot Encoded):

Max Depth	Train Accuracy	Test Accuracy
25	0.9504	0.8175
35	0.9803	0.8100
45	0.9941	0.8067
55	0.9986	0.8068

Higher accuracy then without one hot encoding seen.

Figure:
One-Hot Tree Accuracy vs Depth



(c) Post-Pruning

- Used validation set to prune a fully grown tree.
- Iteratively replaced internal nodes with leaf predictions if validation accuracy improved.
- Compared tree size vs validation accuracy.

Figure:
Validation Accuracy vs Tree Size (for different starting depths)

Observation:

- Pruning reduced overfitting significantly.
- Validation accuracy peaked at smaller tree sizes than the original tree.

(d) Scikit-learn Decision Tree

- Used `DecisionTreeClassifier` with:
    - `criterion='entropy'`
    - Depths = {25, 35, 45, 55}
- Also tested post-pruning using `ccp_alpha` values.

Comparison:

Depth=25 → Train Acc: 0.9576, Test Acc: 0.8210, Val Acc: 0.8243

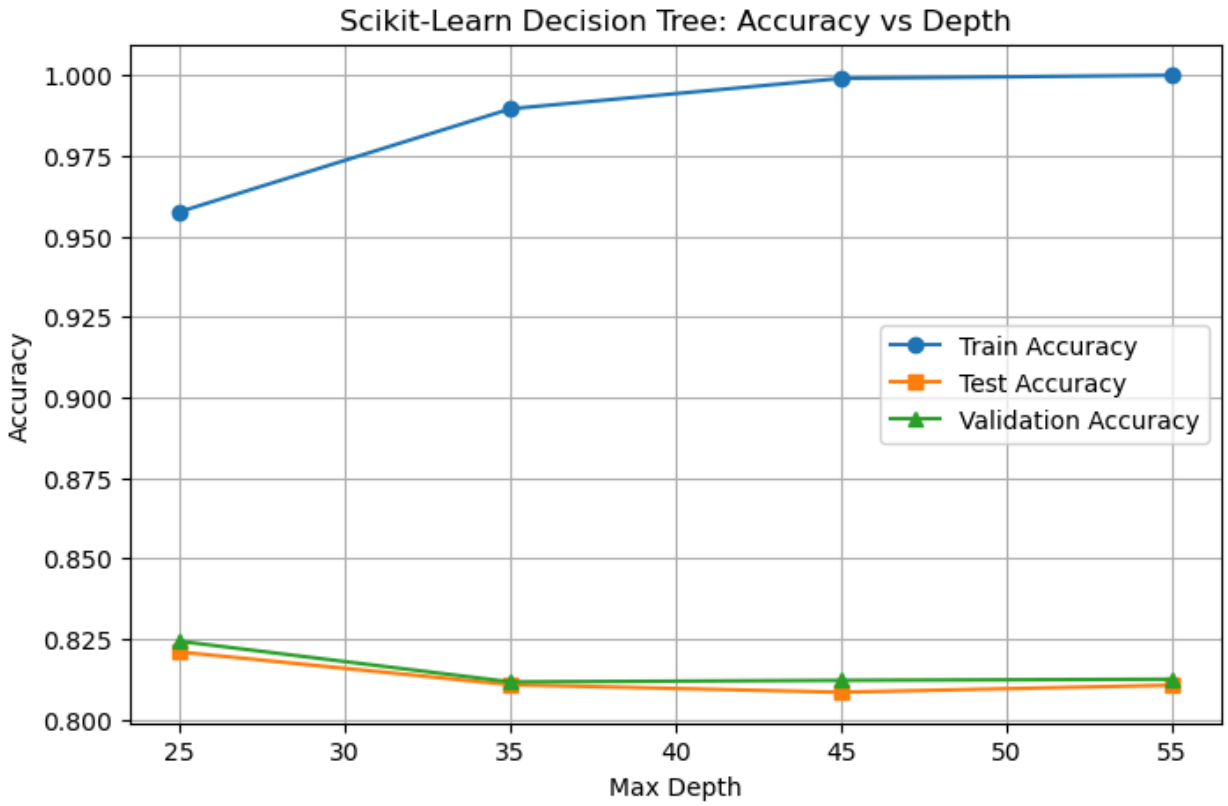
Depth=35 → Train Acc: 0.9896, Test Acc: 0.8108, Val Acc: 0.8117

Depth=45 → Train Acc: 0.9990, Test Acc: 0.8085, Val Acc: 0.8122

Depth=55 → Train Acc: 1.0000, Test Acc: 0.8107, Val Acc: 0.8125

Similar test accuracy as compared to one hot encoding test accuracy seen.

Accuracy trend remains almost same with depth.



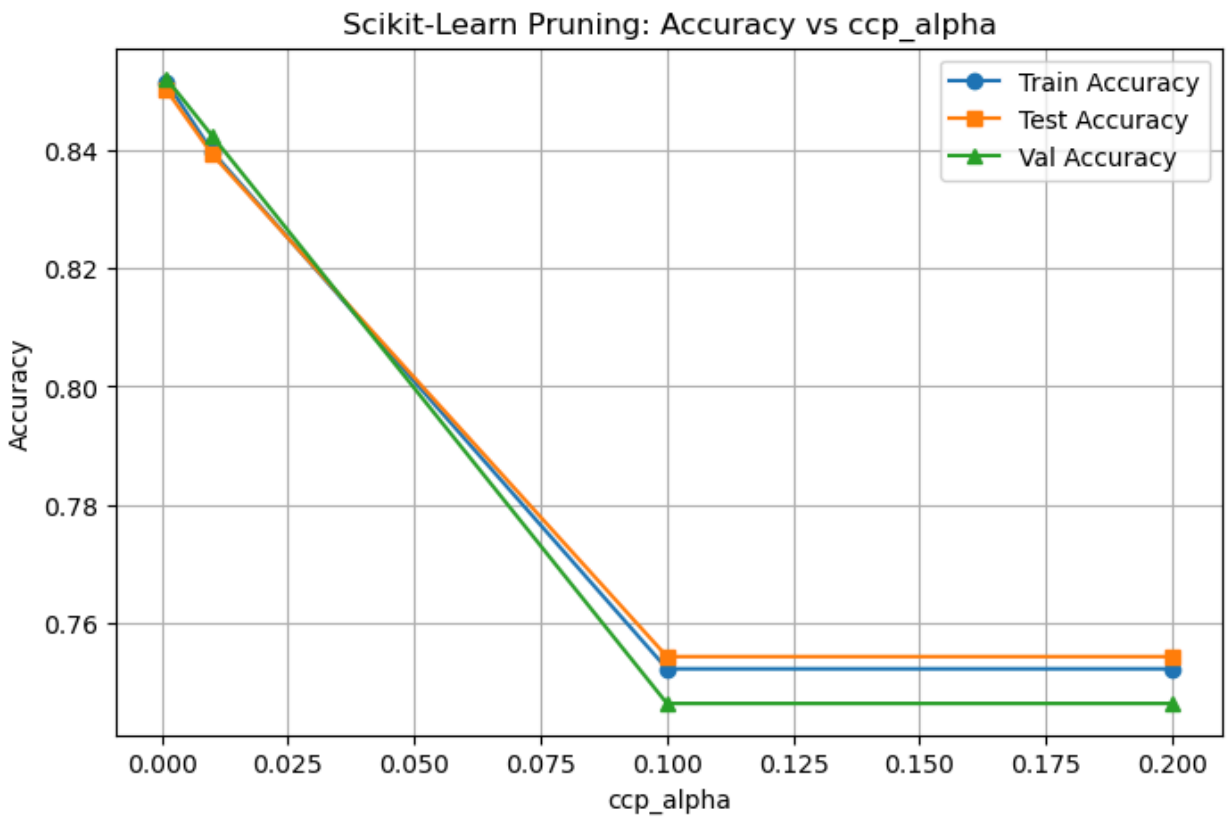
ccp_alpha=0.001 → Train Acc: 0.8514, Test Acc: 0.8501, Val Acc: 0.8520

ccp_alpha=0.010 → Train Acc: 0.8397, Test Acc: 0.8392, Val Acc: 0.8424

ccp_alpha=0.100 → Train Acc: 0.7522, Test Acc: 0.7543, Val Acc: 0.7464

ccp_alpha=0.200 → Train Acc: 0.7522, Test Acc: 0.7543, Val Acc: 0.7464

Highest accuracy observed with alpha = 0.001



(e) Random Forest

- Used `RandomForestClassifier` with OOB score for tuning:
  - `n_estimators`: [50, 150, 250, 350]
  - `max_features`: [0.1, 0.3, 0.5, 0.7, 1.0]
  - `min_samples_split`: [2, 4, 6, 8, 10]

Best Parameters:

- `Best Parameters (OBB score): {'max_features': 0.3, 'min_samples_split': 10, 'n_estimators': 150}`
- `According to F1-score: max_features: 0.7, min_samples_split: 10, n_estimators: 350`

Performance:

Params={'max_features': 0.3, 'min_samples_split': 10, 'n_estimators': 150} → OOB Acc=0.8599, Train Acc=0.9532, Test Acc=0.8556, Val Acc=0.8589

Params={'max_features': 0.7, 'min_samples_split': 10, 'n_estimators': 350} → OOB Acc=0.8583, Train Acc=0.9655, Test Acc=0.8543, Val Acc=0.8603

Part B: Neural Networks

(a) Neural Network from Scratch

- Implemented fully-connected feedforward NN with:
    - Sigmoid in hidden layers
    - Softmax at output
    - Cross-entropy loss
    - Mini-batch SGD

(b) Single Hidden Layer Experiments

- Hidden layer sizes: `[1], [5], [10], [50], [100]`

Results:
All results are for 200 epochs.
Training samples: 26640
Features: 2352
Classes: 43

Training with 1 hidden units

Training time: 91.27 seconds

Train Metrics:

Accuracy : 0.1212

Macro Precision: 0.0130

Macro Recall : 0.0516

Macro F1 : 0.0188

Weighted F1 : 0.0434

Test Metrics:

Accuracy : 0.1305

Macro Precision: 0.0141

Macro Recall : 0.0530

Macro F1 : 0.0204

Weighted F1 : 0.0492

Training with 5 hidden units

Training time: 135.61 seconds

Train Metrics:

Accuracy : 0.6831

Macro Precision: 0.4769

Macro Recall : 0.4614

Macro F1 : 0.4296

Weighted F1 : 0.6326

Test Metrics:

Accuracy : 0.6277

Macro Precision: 0.3872

Macro Recall : 0.3979

Macro F1 : 0.3767

Weighted F1 : 0.5841

Training with 10 hidden units

Training time: 128.03 seconds

Train Metrics:

Accuracy : 0.8750

Macro Precision: 0.8422

Macro Recall : 0.7725

Macro F1 : 0.7709

Weighted F1 : 0.8629

Test Metrics:

Accuracy : 0.7888

Macro Precision: 0.6570

Macro Recall : 0.6474

Macro F1 : 0.6364

Weighted F1 : 0.7755

Training with 50 hidden units

Training time: 349.70 seconds

Train Metrics:

Accuracy : 0.9646

Macro Precision: 0.9698

Macro Recall : 0.9543

Macro F1 : 0.9615

Weighted F1 : 0.9645

Test Metrics:

Accuracy : 0.8792

Macro Precision: 0.8741

Macro Recall : 0.8225

Macro F1 : 0.8370

Weighted F1 : 0.8773

Training with 100 hidden units

Training time: 479.62 seconds

Train Metrics:

Accuracy : 0.9702

Macro Precision: 0.9747

Macro Recall : 0.9612

Macro F1 : 0.9673

Weighted F1 : 0.9701

Test Metrics:

Accuracy : 0.8797

Macro Precision: 0.8800

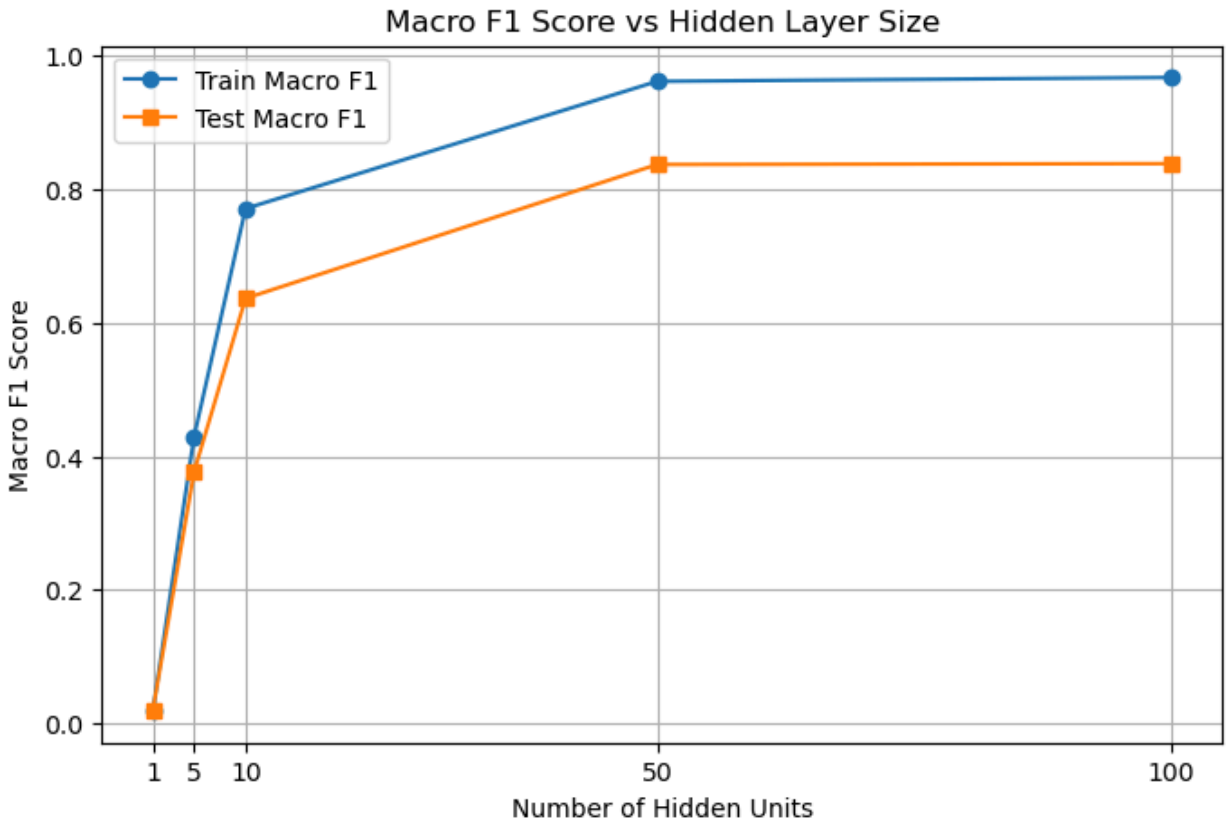
Macro Recall : 0.8212

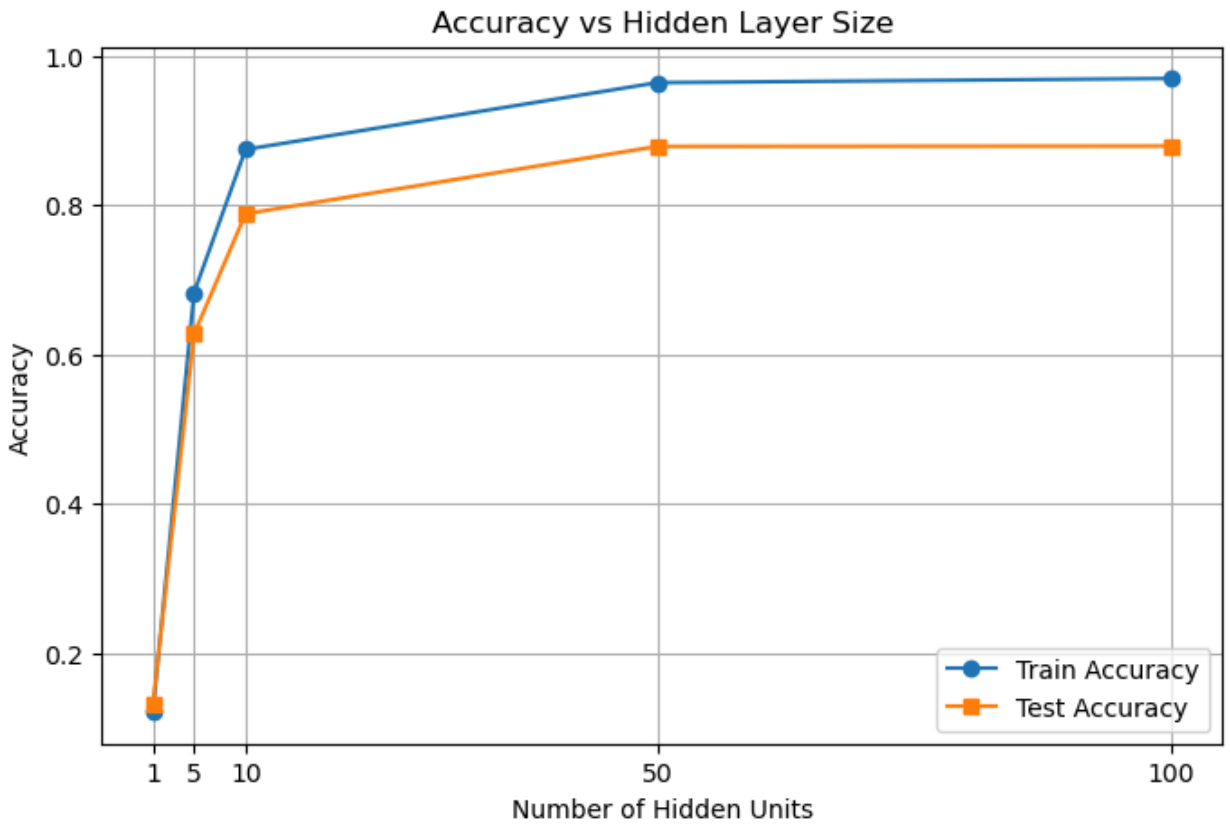
Macro F1 : 0.8382

Weighted F1 : 0.8777

Accuracy increased as hidden units increased.

Figure:
Macro F1 vs Hidden Units





(c) Depth Experiment (Sigmoid)

- Architectures tried:
    - `[512]`, `[512, 256]`, `[512, 256, 128]`, `[512, 256, 128, 64]`

Results:

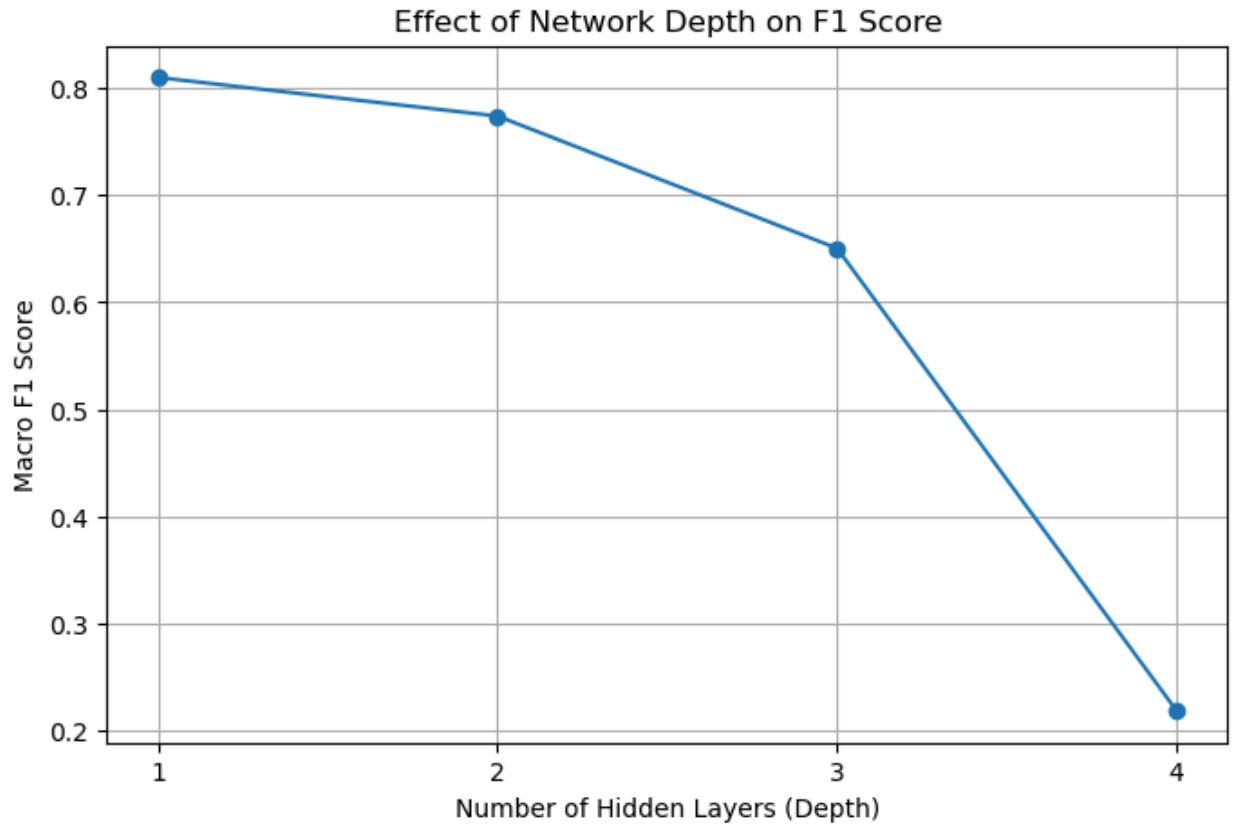
Depth	Hidden Layers	Accuracy	Macro Precision	Macro Recall	Macro F1	Weighted F1
1	[512]	0.8580	0.8602	0.7875	0.8102	0.8560
2	[512, 256]	0.8443	0.8045	0.7655	0.7741	0.8429

3	**[512, 256, 128]**	**0.7713**	**0.6876**	**0.6421**	**0.6504**	**0.7649**
4	**[512, 256, 128, 64]**	**0.4408**	**0.2146**	**0.2505**	**0.2180**	**0.3784**

Accuracy decreased with more layers since epochs are same, and higher depth layers need more epochs to converge.

Figure:

Macro F1 vs Depth (Sigmoid, fixed LR)



(d) Adaptive Learning Rate

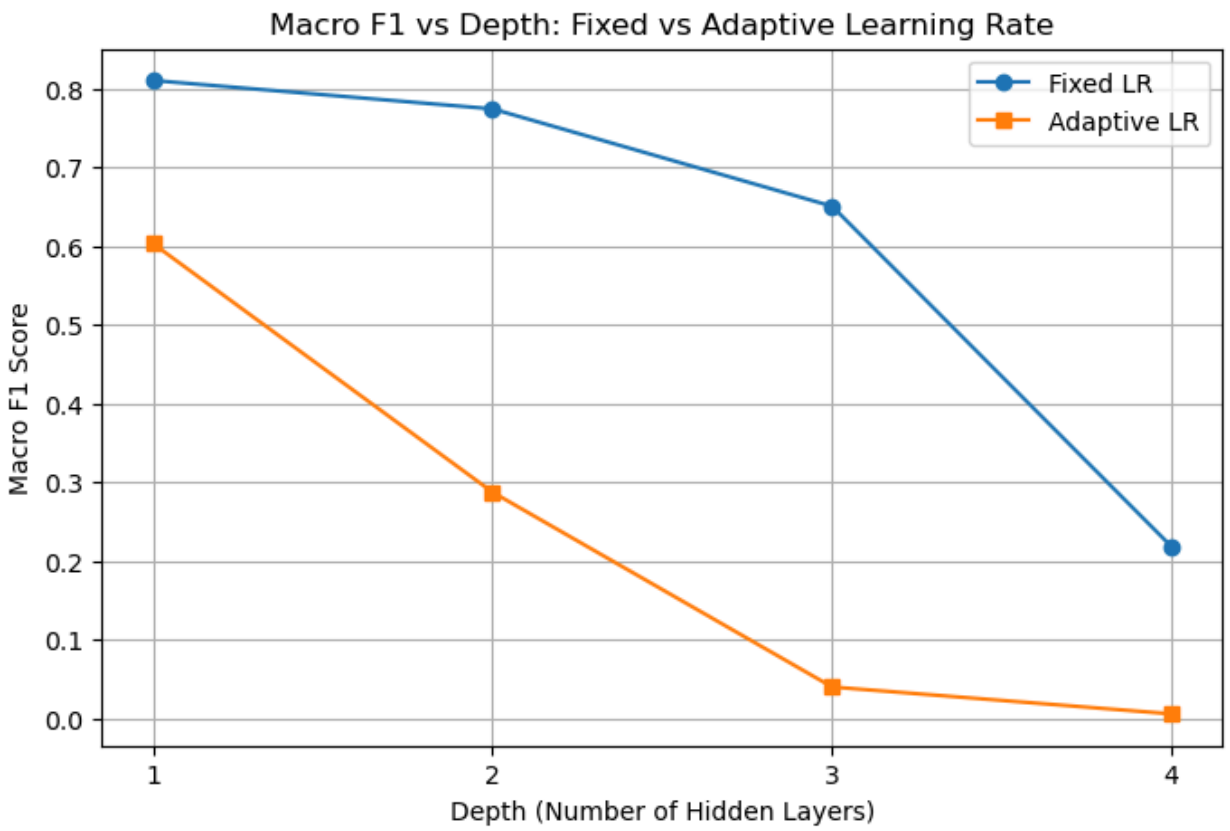
- Used rule:
  `η_e = η_0 / sqrt(e)`

Model Performance Summary

Hidden Layer Configuration	Accuracy	Macro Precision	Macro Recall	Macro F1 Score
[512]	0.7214	0.6860	0.5831	0.6031
[512, 256]	0.5006	0.3557	0.3032	0.2873
[512, 256, 128]	0.1532	0.0530	0.0671	0.0400
[512, 256, 128, 64]	0.0707	0.0034	0.0286	0.0058

Figure:
Macro F1 vs Depth (Adaptive LR vs Fixed LR)

F1 score decreased as depth increased for the same reason as stated above.



(e) ReLU Activation

- Replaced sigmoid with ReLU in hidden layers
- Observed faster convergence and better test accuracy in most cases

Model Performance Summary

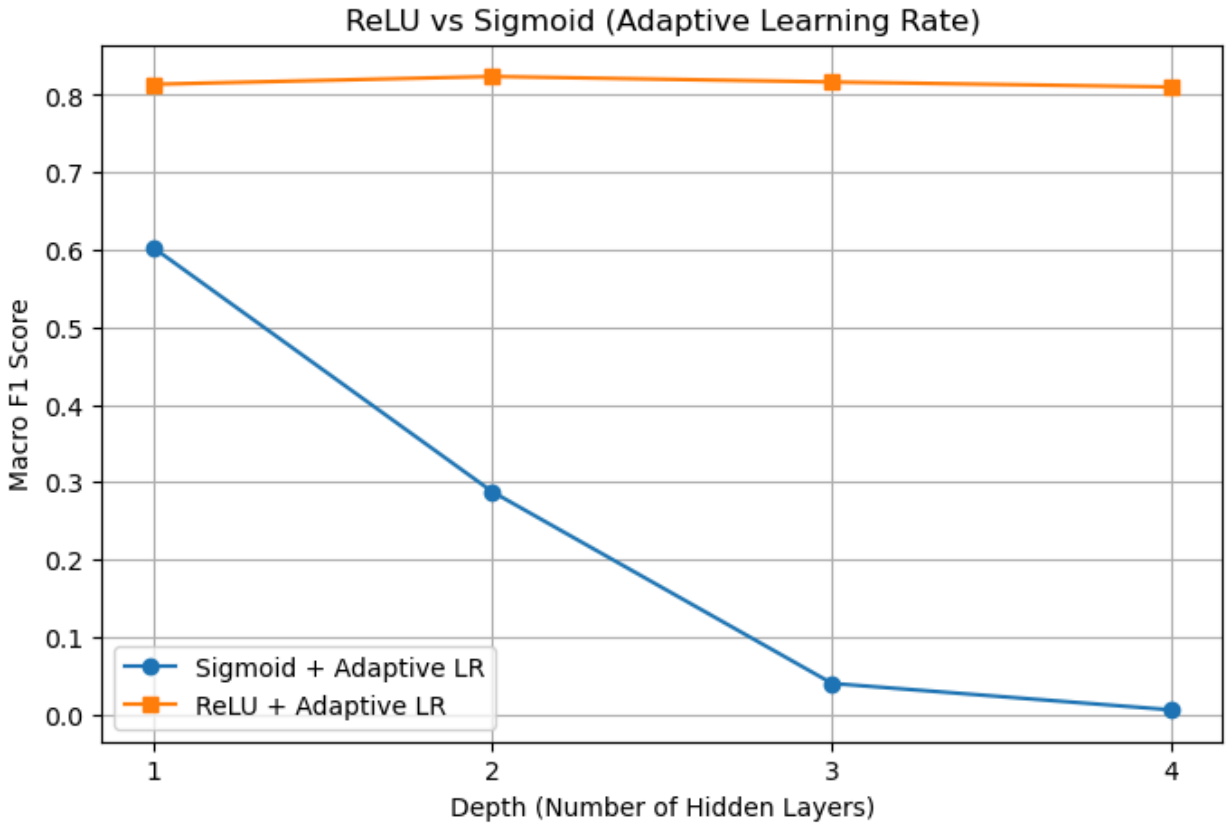
Hidden Layer Configuration	Accuracy	Macro Precision	Macro Recall	Macro F1 Score
[512]	0.8572	0.8594	0.7907	0.8137
[512, 256]	0.8736	0.8371	0.8128	0.8167

[512, 256, 128]	0.8735	0.8370	0.8128	0.8167
[512, 256, 128, 64]	0.8690	0.8220	0.8141	0.8103

Figure:

ReLU vs Sigmoid Performance (Macro F1 vs Depth)

Relu showed better results as compared to sigmoid, and f1 score remained same as depth increased.



(f) Scikit-learn MLPClassifier

- Used:
    - `activation='relu'`
    - `solver='sgd'`
    - `learning_rate='invscaling'`
    - `alpha=0`, `batch_size=32`

Comparison (Best Scores):

[{'hidden_layers': [512],
  'accuracy': 0.5937450514647664,
  'macro avg': {'precision': 0.5334312659038734,
   'recall': 0.3975096937010486,
   'f1-score': 0.3997423389102254,
   'support': 12630.0},
  'weighted avg': {'precision': 0.5705842700692266,
   'recall': 0.5937450514647664,
   'f1-score': 0.5428703776416801,
   'support': 12630.0}},
 'train_time': 1160.8289635181427},
{'hidden_layers': [512, 256],
  'accuracy': 0.5883610451306414,
  'macro avg': {'precision': 0.49995749991931704,
   'recall': 0.4041608764508157,
   'f1-score': 0.4083657816997896,
   'support': 12630.0},

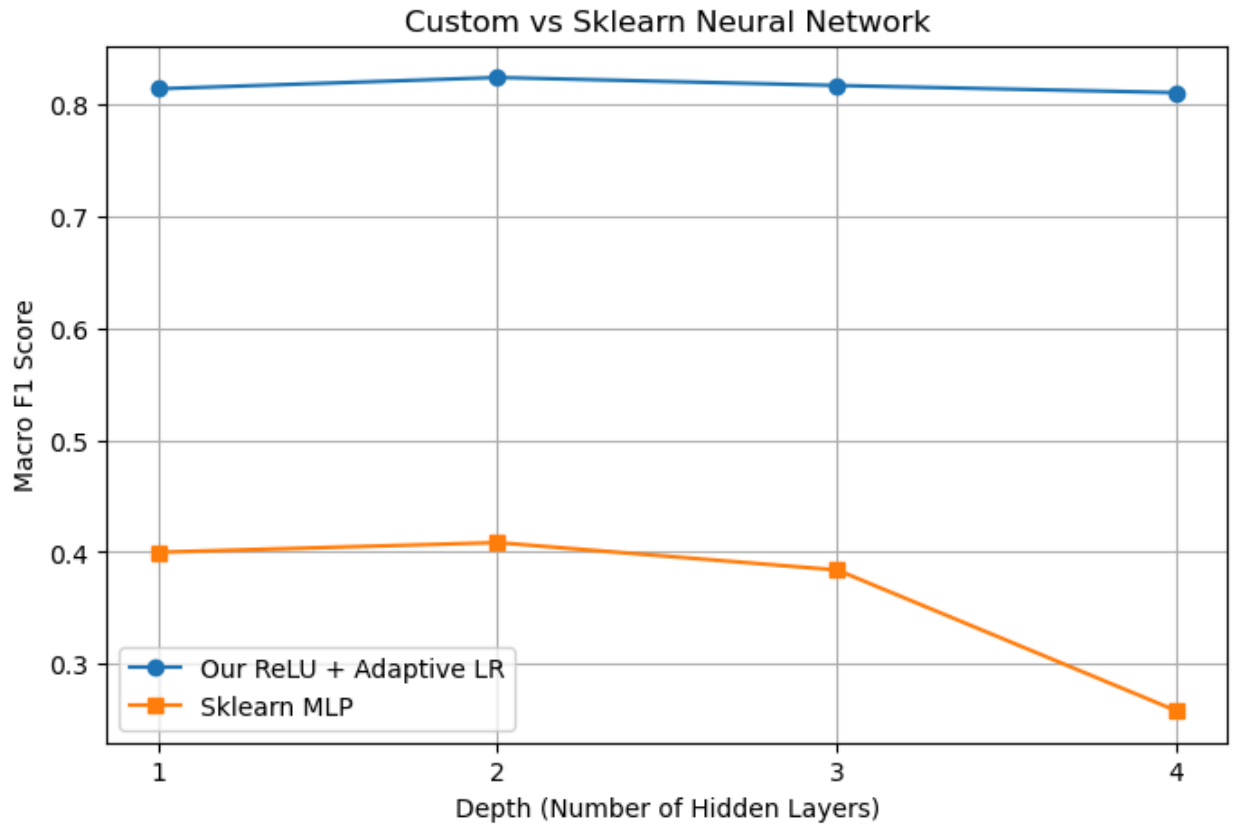
'weighted avg': {'precision': 0.5624392148685554,
'recall': 0.5883610451306414,
'f1-score': 0.5471809403158763,
'support': 12630.0}},
'train_time': 1514.7257151603699},
{'hidden_layers': [512, 256, 128],
'accuracy': 0.5702296120348377,
'macro avg': {'precision': 0.4456749673618586,
'recall': 0.3860030002203916,
'f1-score': 0.3839189334156959,
'support': 12630.0},
'weighted avg': {'precision': 0.5228400481932333,
'recall': 0.5702296120348377,
'f1-score': 0.5246420631580202,
'support': 12630.0}},
'train_time': 1672.269194841385},
{'hidden_layers': [512, 256, 128, 64],
'accuracy': 0.4874901029295329,
'macro avg': {'precision': 0.2902545837633678,
'recall': 0.28533683776808144,
'f1-score': 0.2580082038236769,
'support': 12630.0},
'weighted avg': {'precision': 0.41623331463098057,
'recall': 0.4874901029295329,

'f1-score': 0.42254223909009503,

'support': 12630.0}},

'train_time': 1762.469577550888}]

Figure:
Sklearn vs Custom NN Performance



Similar trend shown as implemented neural network.
