COMP472 Project Report

Student: Daniel Codreanu

ID: 40245452

Section: NN

GitHub Repo: https://github.com/Kpytoo/COMP472_FinalProject_40245452.git

Shared with TAs

marzieh.adeli@gmail.com

shamita.datta119@gmail.com

rikinchauhan01@gmail.com

Model Architectures and Training

• Highlight the changes made for variants of each model, specifically noting how each deviate from the main model.

Gaussian Naive Bayes (2 models)

My personal model and the scikit learn model don't vary much. They are both basic Gaussian Naive Baye models.

Decision Tree (6 models)

My personal base model uses the Gini coefficient and a depth of 50. The first variant of my personal model also uses the Gini coefficient but a depth of 20 instead. The second variant of my personal model uses the Gini coefficient and a depth of 80.

The Scikit learn base model uses the Gini coefficient and a depth of 50. The first variant of the Scikit model also uses the Gini coefficient but a depth of 20 instead. The second variant of the Scikit model uses the Gini coefficient and a depth of 80.

Both my models and Scikit's models use the same parameters.

Multi-Layer Perceptron (3 models)

The MLP base model has 3 layers (as stated in the pdf project). It uses the cross-entropy loss and the SGD optimizer with momentum 9. The first variant of the MLP model has an added layer (a linear function). It also uses the cross-entropy loss and the SGD optimizer with momentum 9. The second variant of the MLP model has a smaller size within its hidden layer (from 512 to 256). It uses the cross-entropy loss and the SGD optimizer with momentum 9.

Convolutional Neural Network (3 models)

The CNN base model VGG11, has 11 layers (as stated in the pdf project), and all its parameters are set as specified in the pdf project. The first variant of the CNN has less convolutions layers (removed the 7th and 8th layers) resulting in 9 layers. It also utilizes the same parameters as specified in the pdf project. The second variant of the CNN had its kernel size adjusted from 3x3 to 2x2 in both convolution layers 3 and 5. Their respective padding parameters have also been adjusted from 1 to 0.

• Detail the training methodology: number of epochs, learning rate, loss function used, and other relevant training hyperparameters.

Gaussian Naive Bayes (2 models)

Both models use the fit function for training. No number of epochs, learning rate and loss function were required.

Decision Tree (6 models)

All models use the Gini coefficient for training. No number of epochs, learning rate and loss function were required.

Multi-Layer Perceptron (3 models)

All models use the cross-entropy loss functions, 5 epochs for training, learning rate of 0.01 for the SGD optimizer and a momentum of 9.

Convolutional Neural Network (3 models)

All models use the cross-entropy loss functions, 5 epochs for training, learning rate of 0.01 for the SGD optimizer and a momentum of 9.

• Mention any optimization algorithms or techniques used, like mini-batch gradient descent, Adam optimizer, etc.

No optimization algorithms or techniques used while making this project. The reason being is that I wanted to see the results of each model from a raw perspective, no influence from algorithms.

Evaluation

• Present the metrics (accuracy, precision, recall, and F1-measure) of the four ML models and their variants.

Gaussian Naive Bayes (2 models)

Model	Accuracy	Precision	Recall	F1-measure
My Naive Gaussian Bayes model	0.772	0.7759	0.772	0.77238
Scikit learn Naive Gausian Bayes model	0.772	0.7759	0.772	0.77238

Decision Tree (6 models)

Model	Accuracy	Precision	Recall	F1-measure
My Decision Tree model	0.578	0.578904	0.578	0.577820
My Decision Tree variant 1	0.578	0.578904	0.578	0.577820
My Decision Tree variant 2	0.578	0.578904	0.578	0.577820
Scikit Decision Tree model	0.575	0.578107	0.575	0.575866
Scikit Decision Tree variant 1	0.566	0.570999	0.566	0.568097
Scikit Decision Tree variant 2	0.568	0.573195	0.568	0.569868

Multi-Layer Perceptron (3 models)

Model	Accuracy	Precision	Recall	F1-measure
Perceptron Base model	0.762	0.787786	0.762	0.760251
Perceptron variant 1	0.749	0.792667	0.749	0.746552
Perceptron variant 2	0.770	0.787772	0.770	0.767831

Convolutional Neural Network (3 models)

Model	Accuracy	Precision	Recall	F1-measure
VGG11 Base model	0.207	0.181096	0.207	0.111002
VGG11 variant 1	0.100	0.010000	0.100	0.018182
VGG11 variant 2	0.236	0.182216	0.236	0.167278

• Offer insights into each model's performance relative to the others. For instance, if one model has a higher recall but lower precision, discuss its implications in the context of facial image analysis.

Gaussian Naive Bayes (2 models)

Both models have the same performance relative to each other, so in the context of facial image analysis, they will score the same results. Why would they have the same performance? I would guess that the way I created my model is the same way Scikit made theirs.

Decision Tree (6 models)

First off, if we look at my models and their variations, we can clearly see that something went wrong. Given they all have the same performance, no matter the depth of the tree, my model implementation is incorrect and thus falsifies my results. On the other hand, the comparison can be made with Scikit's models. The first model, with a depth of 50 has the highest accuracy compared to the other two variants, with depth 20 and 80 respectively. In a context of facial image analysis, this would mean that the tree with an in-between depth will correctly analyze the faces 57.5% of the time. My guess would be that a depth of 20 isn't enough to gather and classify information, and a depth of 80 produces overfitting which may throw our results out of the window. Looking at precision, recall and f1-measure, they follow suit to the base model with depth 50 being the best. In the context of facial image analysis, the base model will have less false positives and an overall better classification.

Multi-Layer Perceptron (3 models)

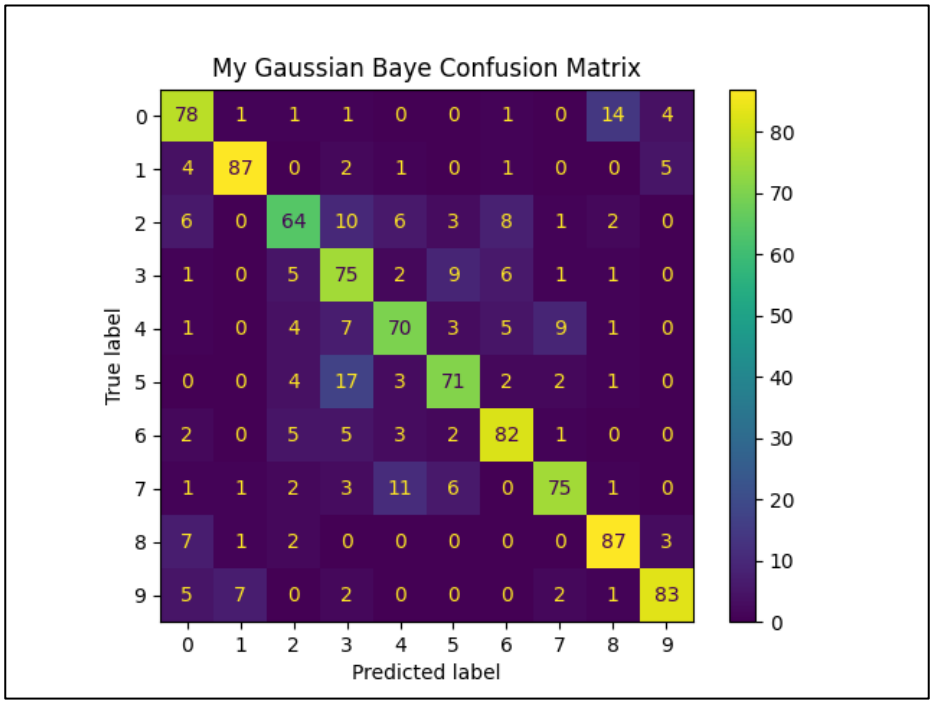
Looking at the 2nd variant of the model, the one that has a smaller size within it hidden layers, we notice that out of all the three models, it has the highest accuracy, but it does not have the highest precision, which goes to the 1st variant of the model, the one with more layers. Why could that be? Well, it would mean that having more layers enables more classification

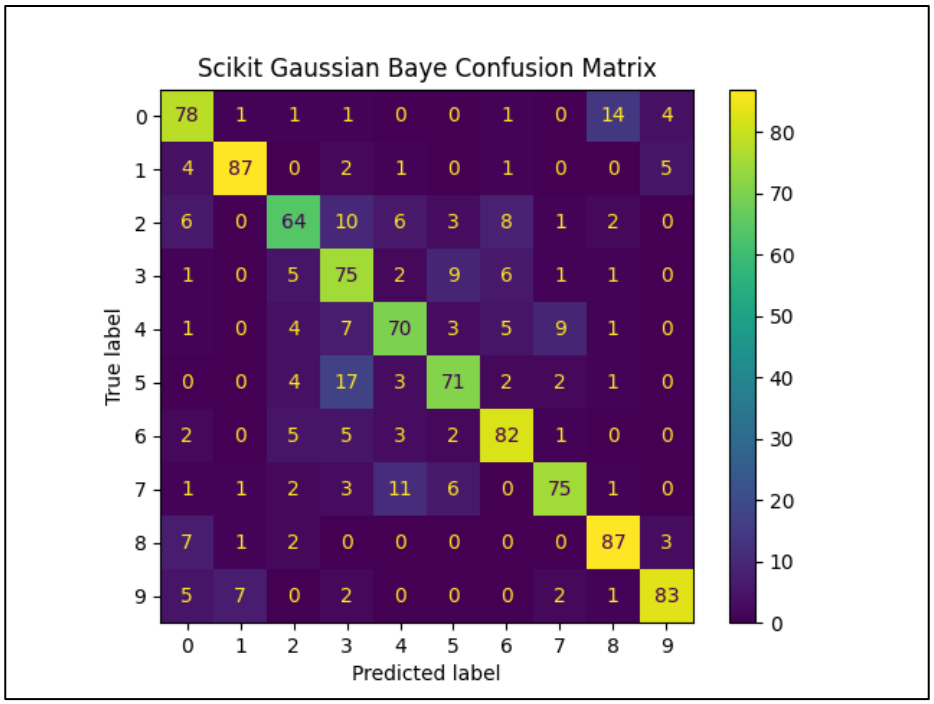
computations and results in a better prediction of true positives, although its accuracy is the lowest. Overall, in a facial image analysis system, the 2nd variant model will perform the best given its good accuracy, recall and f1-measure.

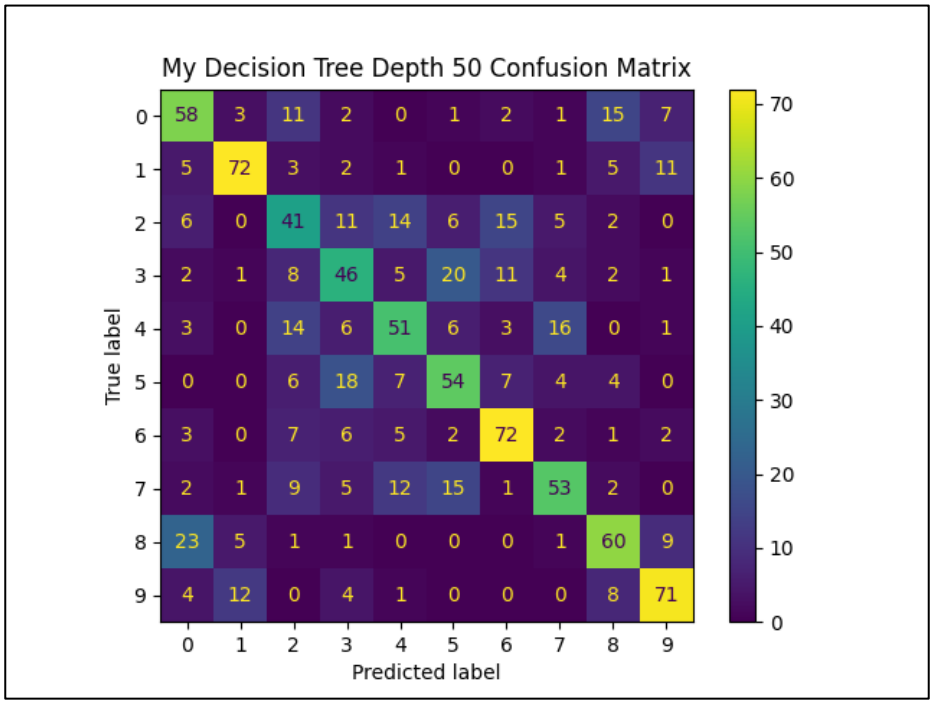
Convolutional Neural Network (3 models)

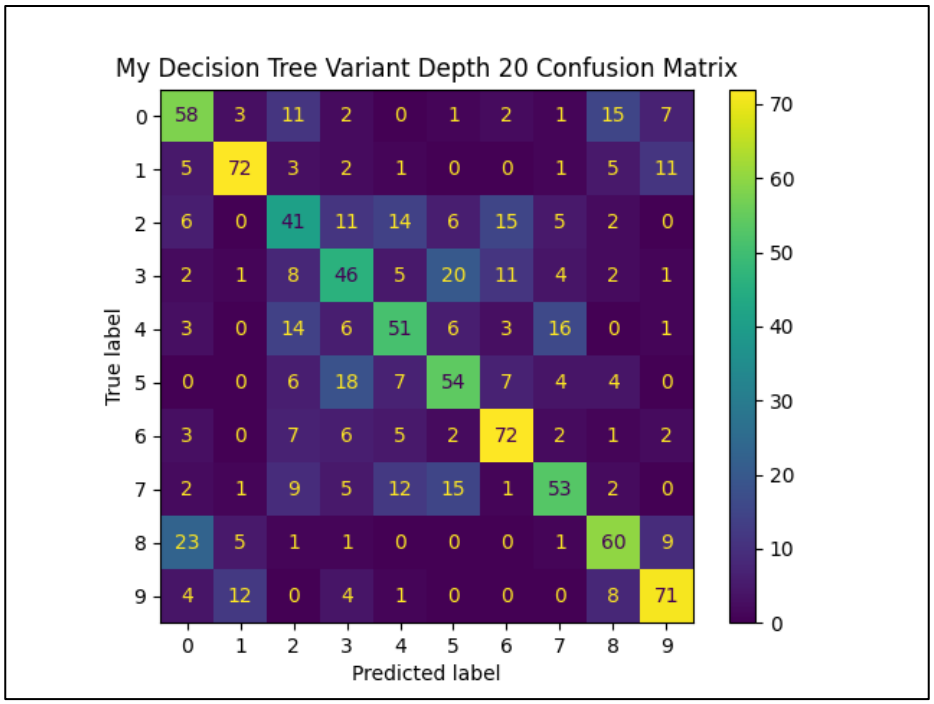
Taking a look at the 2nd variant of the model, the one having its kernel size reduced from 3x3 to 2x2 in the 3rd and 5th convolution layer, it has the highest value for all metrics considering all models. A smaller kernel size might possibly mean that it is able to process features more efficiently as the image gets smaller. We can also notice in the first variant of the model, the one that has less convolution layers, having the worst metrics out of all models. This implies that convolution layers are very necessary to capture features and details within an image.

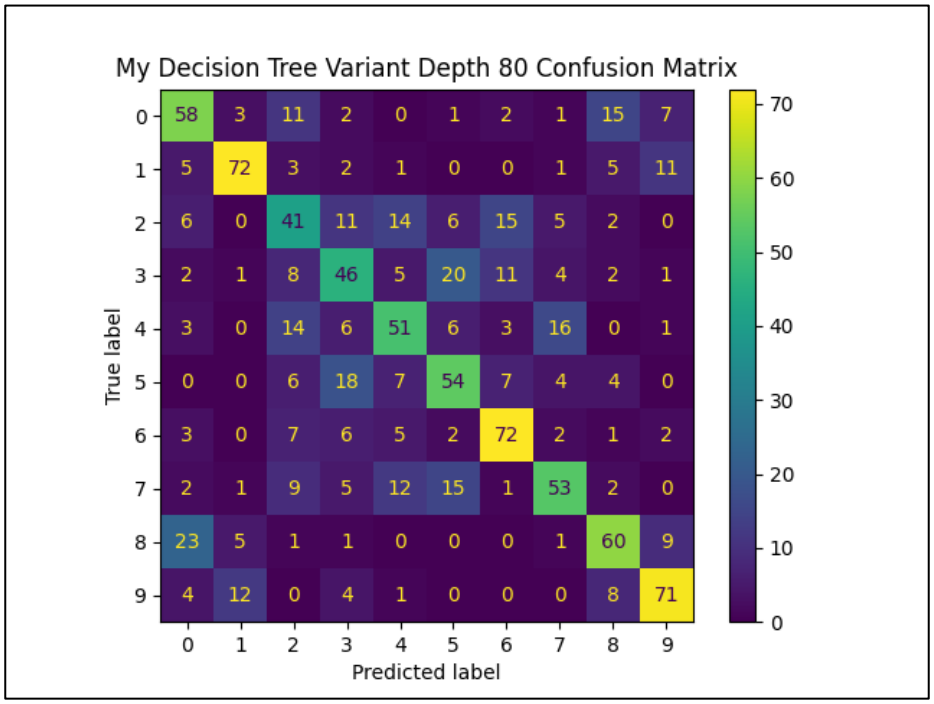
- Display the confusion matrices for each model.

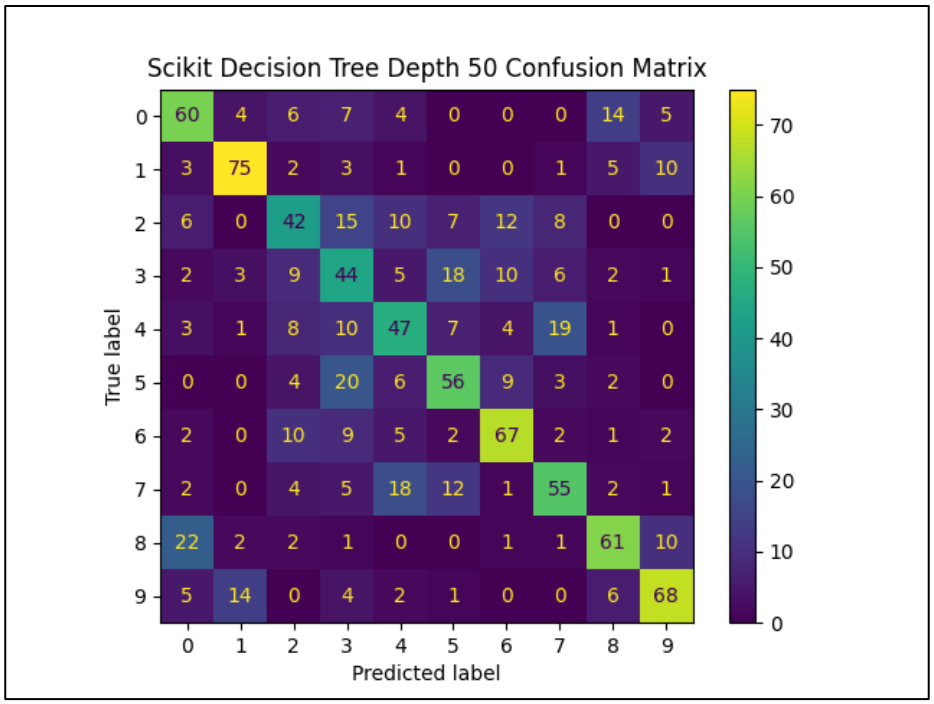


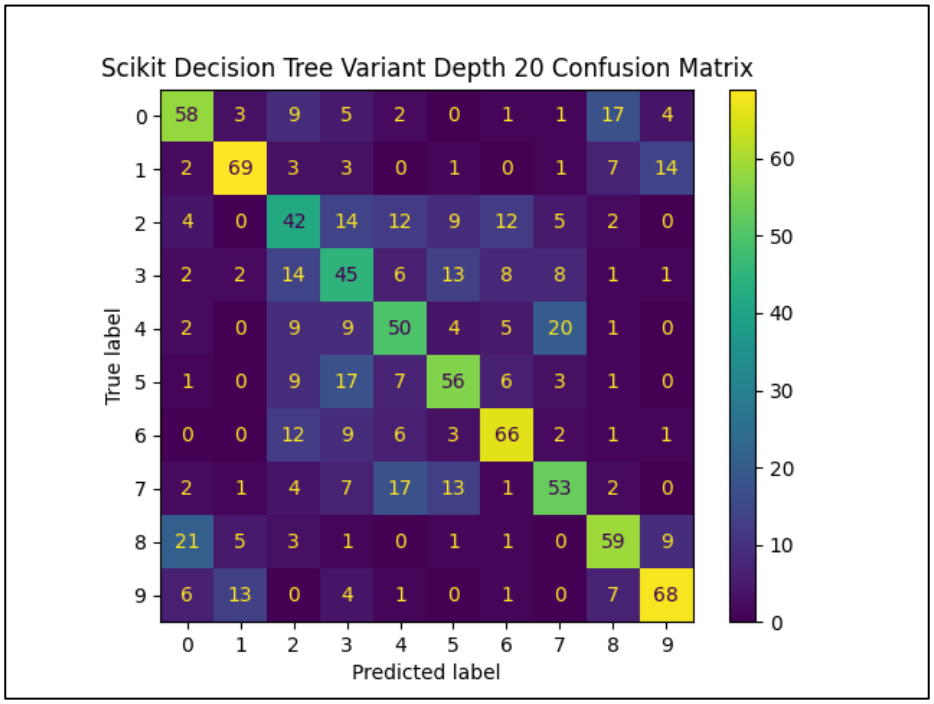

Scikit Gaussian Baye Confusion Matrix
True label
Predicted label
0 78 1 1 1 0 0 1 0 14 4
1 4 87 0 2 1 0 1 0 0 5
2 6 0 64 10 6 3 8 1 2 0
3 1 0 5 75 2 9 6 1 1 0
4 1 0 4 7 70 3 5 9 1 0
5 0 0 4 17 3 71 2 2 1 0
6 2 0 5 5 3 2 82 1 0 0
7 1 1 2 3 11 6 0 75 1 0
8 7 1 2 0 0 0 0 0 87 3
9 5 7 0 2 0 0 0 2 1 83
0 1 2 3 4 5 6 7 8 9
80 70 60 50 40 30 20 10 0

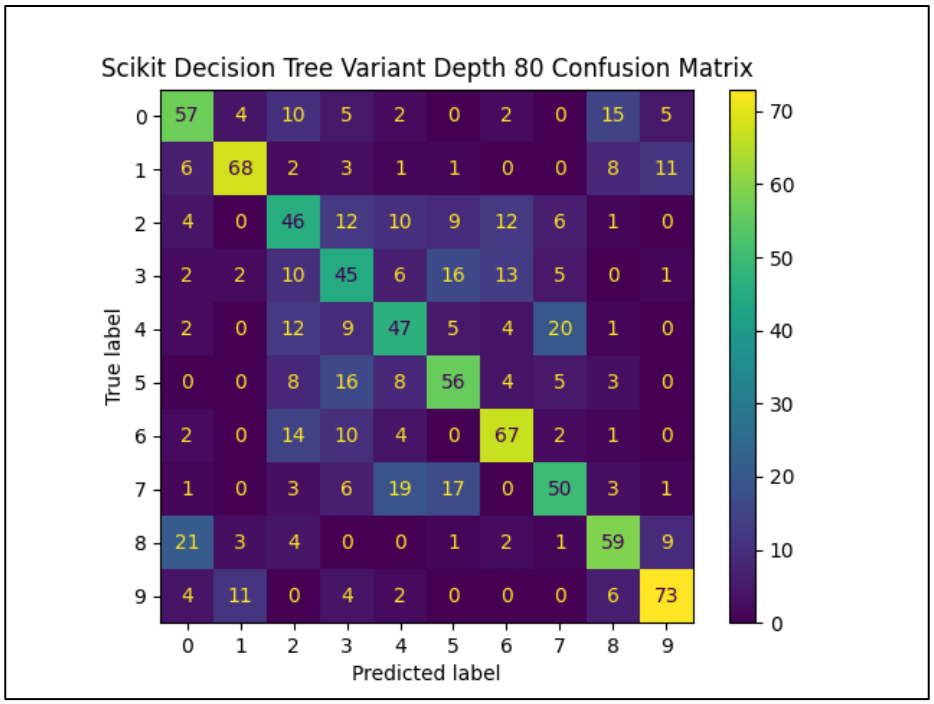

My Decision Tree Depth 50 Confusion Matrix
True label
Predicted label
0 58 3 11 2 0 1 2 1 15 7
1 5 72 3 2 1 0 0 1 5 11
2 6 0 41 11 14 6 15 5 2 0
3 2 1 8 46 5 20 11 4 2 1
4 3 0 14 6 51 6 3 16 0 1
5 0 0 6 18 7 54 7 4 4 0
6 3 0 7 6 5 2 72 2 1 2
7 2 1 9 5 12 15 1 53 2 0
8 23 5 1 1 0 0 0 1 60 9
9 4 12 0 4 1 0 0 0 8 71
0 1 2 3 4 5 6 7 8 9
70 60 50 40 30 20 10 0

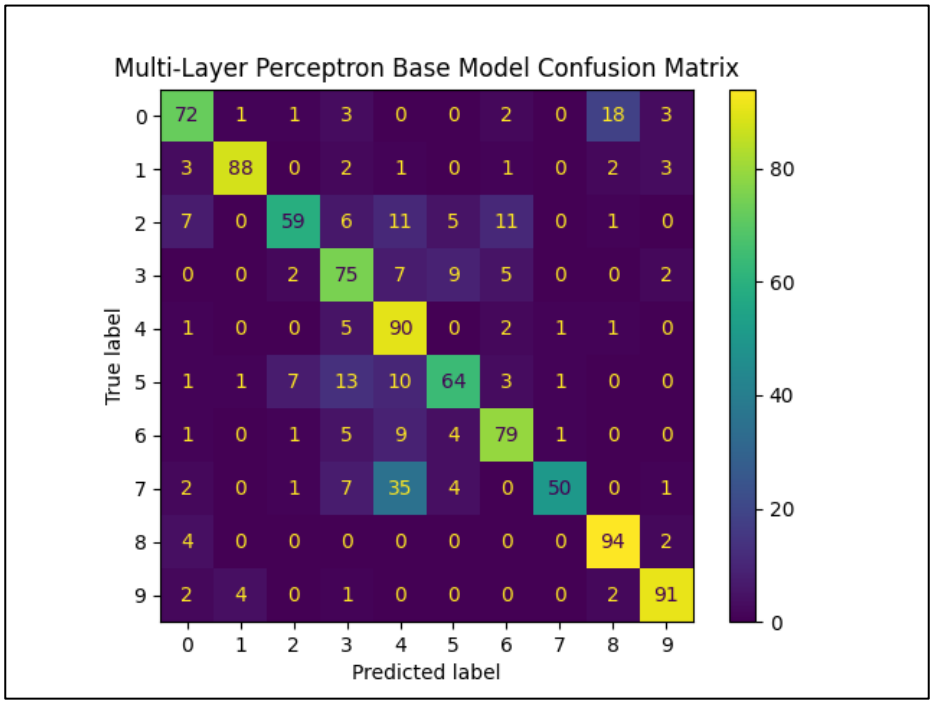

My Decision Tree Variant Depth 20 Confusion Matrix
True label
Predicted label
0 58 3 11 2 0 1 2 1 15 7
1 5 72 3 2 1 0 0 1 5 11
2 6 0 41 11 14 6 15 5 2 0
3 2 1 8 46 5 20 11 4 2 1
4 3 0 14 6 51 6 3 16 0 1
5 0 0 6 18 7 54 7 4 4 0
6 3 0 7 6 5 2 72 2 1 2
7 2 1 9 5 12 15 1 53 2 0
8 23 5 1 1 0 0 0 1 60 9
9 4 12 0 4 1 0 0 0 8 71
0 1 2 3 4 5 6 7 8 9
70 60 50 40 30 20 10 0

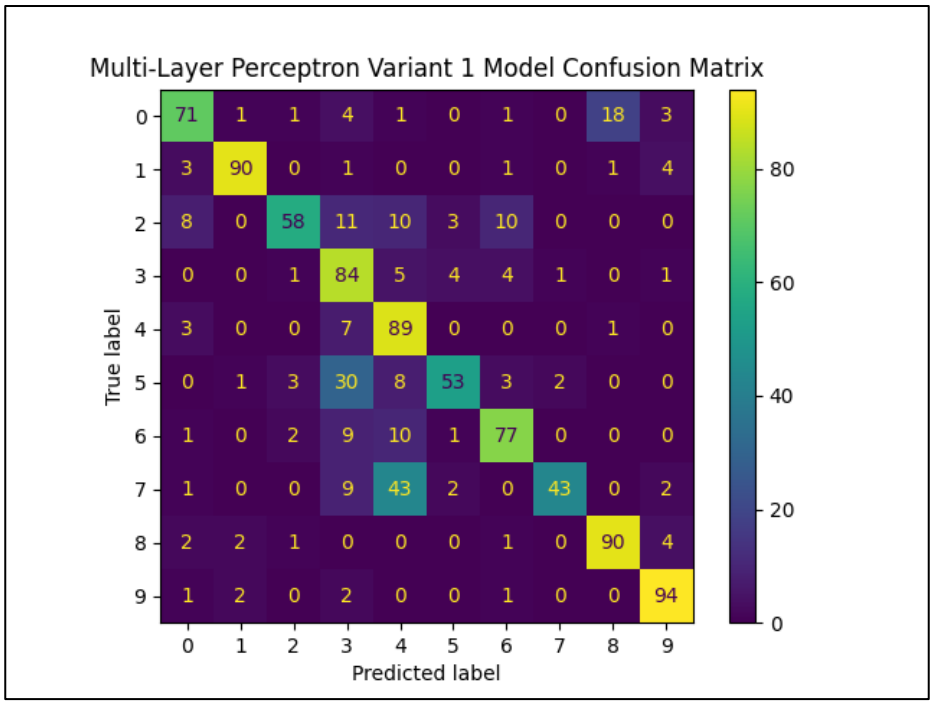

My Decision Tree Variant Depth 80 Confusion Matrix
True label
Predicted label
0 58 3 11 2 0 1 2 1 15 7
1 5 72 3 2 1 0 0 1 5 11
2 6 0 41 11 14 6 15 5 2 0
3 2 1 8 46 5 20 11 4 2 1
4 3 0 14 6 51 6 3 16 0 1
5 0 0 6 18 7 54 7 4 4 0
6 3 0 7 6 5 2 72 2 1 2
7 2 1 9 5 12 15 1 53 2 0
8 23 5 1 1 0 0 0 1 60 9
9 4 12 0 4 1 0 0 0 8 71
0 1 2 3 4 5 6 7 8 9
70 60 50 40 30 20 10 0

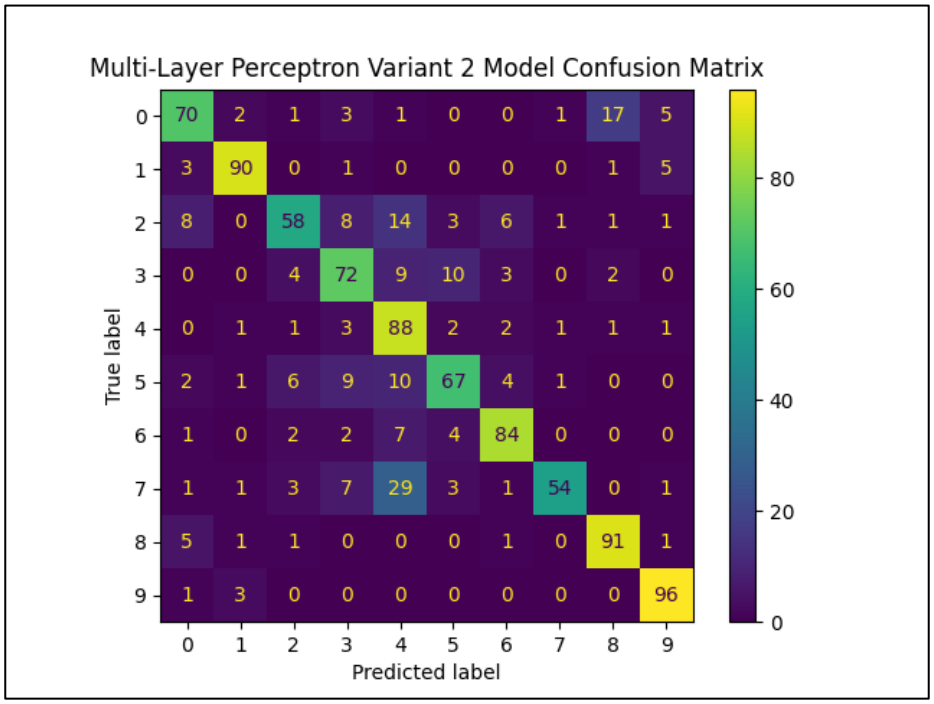

Scikit Decision Tree Depth 50 Confusion Matrix
True label
Predicted label
0 60 4 6 7 4 0 0 0 14 5
1 3 75 2 3 1 0 0 1 5 10
2 6 0 42 15 10 7 12 8 0 0
3 2 3 9 44 5 18 10 6 2 1
4 3 1 8 10 47 7 4 19 1 0
5 0 0 4 20 6 56 9 3 2 0
6 2 0 10 9 5 2 67 2 1 2
7 2 0 4 5 18 12 1 55 2 1
8 22 2 2 1 0 0 1 1 61 10
9 5 14 0 4 2 1 0 0 6 68
0 1 2 3 4 5 6 7 8 9
70 60 50 40 30 20 10 0

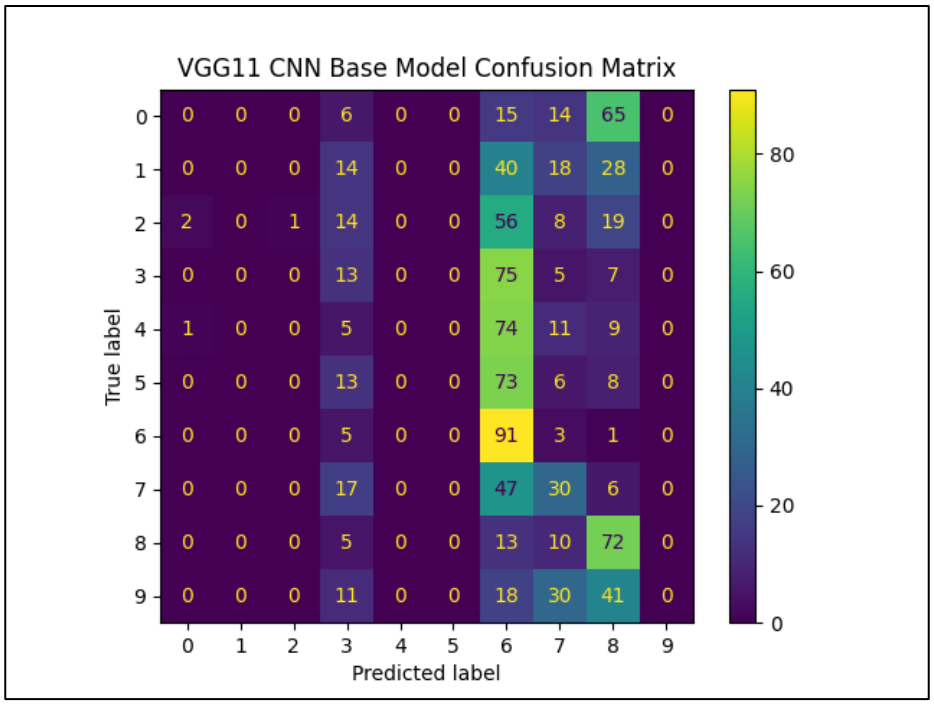

Scikit Decision Tree Variant Depth 20 Confusion Matrix
True label
Predicted label
0 58 3 9 5 2 0 1 1 17 4
1 2 69 3 3 0 1 0 1 7 14
2 4 0 42 14 12 9 12 5 2 0
3 2 2 14 45 6 13 8 8 1 1
4 2 0 9 9 50 4 5 20 1 0
5 1 0 9 17 7 56 6 3 1 0
6 0 0 12 9 6 3 66 2 1 1
7 2 1 4 7 17 13 1 53 2 0
8 21 5 3 1 0 1 1 0 59 9
9 6 13 0 4 1 0 1 0 7 68
0 1 2 3 4 5 6 7 8 9
60 50 40 30 20 10 0

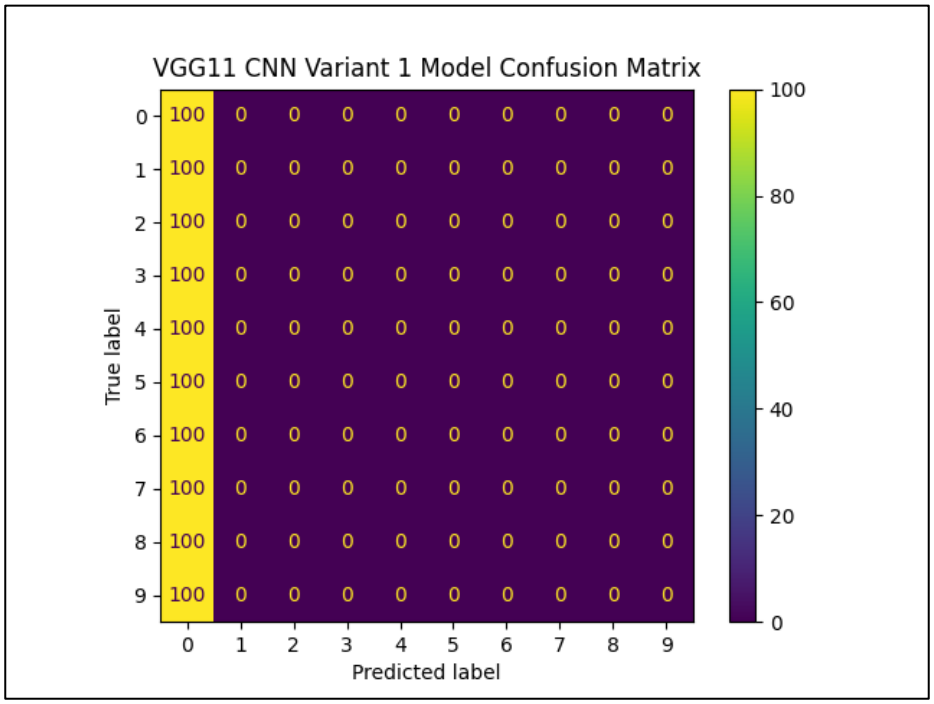

Scikit Decision Tree Variant Depth 80 Confusion Matrix
True label
Predicted label
0 57 4 10 5 2 0 2 0 15 5
1 6 68 2 3 1 1 0 0 8 11
2 4 0 46 12 10 9 12 6 1 0
3 2 2 10 45 6 16 13 5 0 1
4 2 0 12 9 47 5 4 20 1 0
5 0 0 8 16 8 56 4 5 3 0
6 2 0 14 10 4 0 67 2 1 0
7 1 0 3 6 19 17 0 50 3 1
8 21 3 4 0 0 1 2 1 59 9
9 4 11 0 4 2 0 0 0 6 73
0 1 2 3 4 5 6 7 8 9
70 60 50 40 30 20 10 0

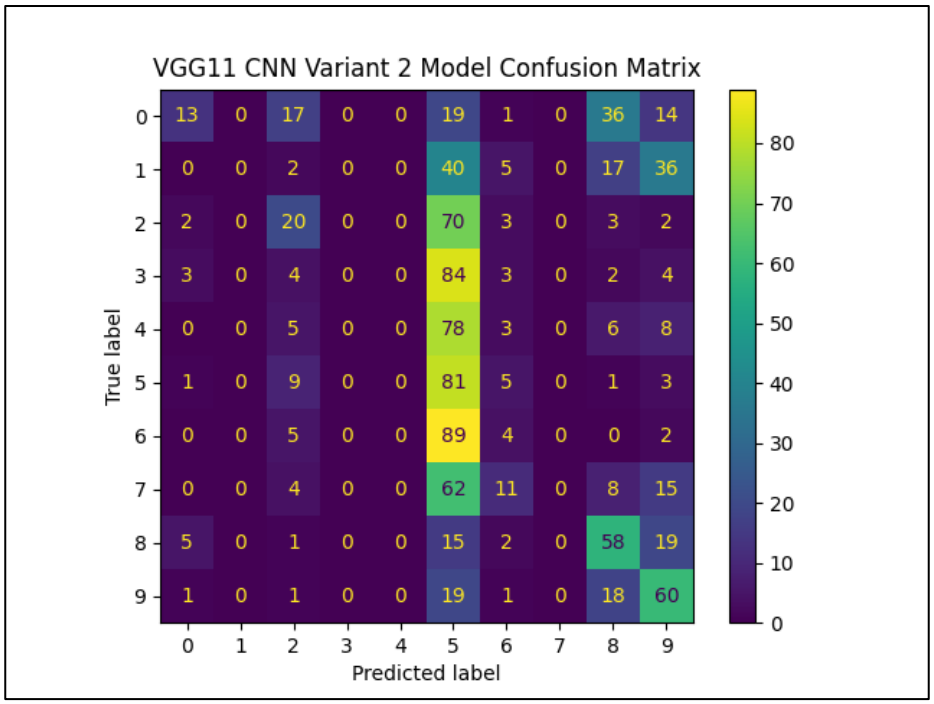

Multi-Layer Perceptron Base Model Confusion Matrix
True label
Predicted label
0 72 1 1 3 0 0 2 0 18 3
1 3 88 0 2 1 0 1 0 2 3
2 7 0 59 6 11 5 11 0 1 0
3 0 0 2 75 7 9 5 0 0 2
4 1 0 0 5 90 0 2 1 1 0
5 1 1 7 13 10 64 3 1 0 0
6 1 0 1 5 9 4 79 1 0 0
7 2 0 1 7 35 4 0 50 0 1
8 4 0 0 0 0 0 0 0 94 2
9 2 4 0 1 0 0 0 0 2 91
0 1 2 3 4 5 6 7 8 9
80 60 40 20 0


Multi-Layer Perceptron Variant 1 Model Confusion Matrix
True label
Predicted label
0 71 1 1 4 1 0 1 0 18 3
1 3 90 0 1 0 0 1 0 1 4
2 8 0 58 11 10 3 10 0 0 0
3 0 0 1 84 5 4 4 1 0 1
4 3 0 0 7 89 0 0 0 1 0
5 0 1 3 30 8 53 3 2 0 0
6 1 0 2 9 10 1 77 0 0 0
7 1 0 0 9 43 2 0 43 0 2
8 2 2 1 0 0 0 1 0 90 4
9 1 2 0 2 0 0 1 0 0 94
0 1 2 3 4 5 6 7 8 9
80
60
40
20
0


Multi-Layer Perceptron Variant 2 Model Confusion Matrix
True label
Predicted label
0 70 2 1 3 1 0 0 1 17 5
1 3 90 0 1 0 0 0 0 1 5
2 8 0 58 8 14 3 6 1 1 1
3 0 0 4 72 9 10 3 0 2 0
4 0 1 1 3 88 2 2 1 1 1
5 2 1 6 9 10 67 4 1 0 0
6 1 0 2 2 7 4 84 0 0 0
7 1 1 3 7 29 3 1 54 0 1
8 5 1 1 0 0 0 1 0 91 1
9 1 3 0 0 0 0 0 0 0 96
0 1 2 3 4 5 6 7 8 9
80
60
40
20
0


VGG11 CNN Base Model Confusion Matrix
True label
Predicted label
0 0 0 6 0 0 15 14 65 0
0 0 0 14 0 0 40 18 28 0
2 0 1 14 0 0 56 8 19 0
0 0 0 13 0 0 75 5 7 0
1 0 0 5 0 0 74 11 9 0
0 0 0 13 0 0 73 6 8 0
0 0 0 5 0 0 91 3 1 0
0 0 0 17 0 0 47 30 6 0
0 0 0 5 0 0 13 10 72 0
0 0 0 11 0 0 18 30 41 0
0 1 2 3 4 5 6 7 8 9
0 20 40 60 80


VGG11 CNN Variant 1 Model Confusion Matrix
True label
Predicted label
100 0 0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8 9
0 20 40 60 80 100


VGG11 CNN Variant 2 Model Confusion Matrix
True label
Predicted label
0 13 0 17 0 0 19 1 0 36 14
1 0 0 2 0 0 40 5 0 17 36
2 2 0 20 0 0 70 3 0 3 2
3 3 0 4 0 0 84 3 0 2 4
4 0 0 5 0 0 78 3 0 6 8
5 1 0 9 0 0 81 5 0 1 3
6 0 0 5 0 0 89 4 0 0 2
7 0 0 4 0 0 62 11 0 8 15
8 5 0 1 0 0 15 2 0 58 19
9 1 0 1 0 0 19 1 0 18 60
0 1 2 3 4 5 6 7 8 9
80 70 60 50 40 30 20 10 0

• Identify which classes were most frequently confused and discuss any model-specific reasons that might be behind these misclassifications.

The Naive Bayes models, the Decision Trees models and the MLP models all somewhat decently cover and predict all classes. The outlier comes mainly from the CNN models. We can see that the base VGG11 model predicted mostly class 6 and a little bit of class 8. The first variant was the most disastrous one where it predicted all images to be of class 0. And the second variant predicted mostly class 5 and a little bit of class 8 and 9. The reason behind these misclassifications could be because of the low number of epochs given for training. Since these models had little training, their classification was very elementary and unsuccessful. Especially the first variant which has less convolution layers than the base model, predicted all values to be of class 0.

• Highlight well-recognized classes and speculate on the reasons behind their success.

The most well-recognized classes were class 1, 6, 8, 9. My guess would be that they have specific color patterns and specific features that are easily classified within any model. They could also be easily differentiated from other classes which makes them stand out whenever training a model.

• Reflect on how depth (for the decision tree, MLP, and CNN) influenced performance. Did it seem to make the model capture more detailed features or overfit?

Decision Tree (6 models)

Looking at Scikit Learn's models since my personal models failed, we can see that a small depth (depth 20) had the lowest metrics out of all three models, which means that it did not capture enough details from the features to correctly classify the images. On the other hand, a large depth (depth 80) also did not have the highest metrics, although higher than the smallest depth model. This might mean that we reached overfitting where the model doesn't appropriately classify the test data. Finally, we can see that a depth in the middle (depth 50) had the best

metrics where it did not overfit too much the data, but it manages to capture important details from the features.

Multi-Layer Perceptron (3 models)

Looking at variant 1 of the model, which has one more layer than the base, we notice that all its metrics are lower, except the precision, which is higher than the base model. Why could that be? My guess would be that out of all the true predicted values, it predicted the most actual true values, but in general, its true predicted values weren't as many as the base model. This would lead to a high precision but an actual low accuracy score. In this case, we could say that it overfit the test data.

Convolutional Neural Network (3 models)

In the case of CNNs, the first variant of the model has 2 less convolution layers than the base, and the results are drastic. The precision is very low for the first variant, meaning that it did not capture any important detail from the features. That is also why its accuracy is very low. Compared to the base model, where it did somewhat capture some detail, but given the low epoch number of 5 for training, it is expected.

• Discuss how layer size (for MLP) and kernel size (for CNN) variations affected the model's recognition abilities.

Multi-Layer Perceptron (3 models)

Looking at the second variant of the MLP, its hidden layers are of smaller sizes than the base MLP model, and it has higher metrics overall. A smaller size could reduce the risk of overfitting, which is what may have happened in our case. The MLP variant with smaller layer size has better metrics than the base model, whereas the base model may have overfit the data.

Convolutional Neural Network (3 models)

Looking at the second variant of the CNN, its kernel size is smaller in the convolution layer 3 and 5, compared to the base model, resulting in higher metrics overall. This could be the case since

a smaller kernel may process smaller images more efficiently down the line of feature processing. We can see this in the precision and recall metric of the variant. It was able to capture more important features than the base model.

• Summarize the primary findings: which model performed best and why?

In conclusion, the best model that fit our test data was the Gaussian Naive Bayes model. It had the highest accuracy, precision, recall and f1-measure out of all the models in this project. It could be in the fact that its fit method, where calculating the prior probability of each class and then determining the posterior probability to classify the images, accordingly, is simple and does not overfit the data. The second closest model was the second variant of the MLP. I believe that if it were given more epochs for training then it would succeed the Gaussian Naive Bayes model. We have also found that removing convolution layers from the CNN model results in disastrous classification results, as seen in the variant 1 of the CNN model.

Sources

[1] PyTorch. https://pytorch.org/ (accessed November 20, 2024)

[2] Torchvision. https://pytorch.org/vision/stable/index.html (accessed November 20, 2024)

[3] Pandas. https://pandas.pydata.org/docs/index.html (accessed November 21, 2024)

[4] NumPy. https://numpy.org/ (accessed November 21, 2024)

[5] Scikit-Learn. https://scikit-learn.org/stable/ (accessed November 19, 2024)

[6] A. Krizhevsky. The CIFAR-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html (accessed November 18, 2024)

[7] Pickle. https://docs.python.org/3/library/pickle.html (accessed November 21, 2024)

[8] J. Hunter, D. Dale, E. Firing, M. Droettboom. Matplotlib. https://matplotlib.org/stable/ (accessed November 20, 2024)

[9] Resnet18. https://pytorch.org/vision/main/models/generated/torchvision.models.resnet18.html (accessed November 19, 2024)

[10] Enozeren, Building a Decision Tree From Scratch with Python. https://medium.com/@enozeren/building-a-decision-tree-from-scratch-324b9a5ed836 (accessed November 19, 2024)

[11] Training with PyTorch. https://pytorch.org/tutorials/beginner/introyt/trainingyt.html (accessed November 20, 2024)

[12] S. R. Rath. Implementing VGG11 from Scratch using PyTorch. https://debuggercafe.com/implementing-vgg11-from-scratch-using-pytorch/ (accessed November 20, 2024)