CONCLUSION

Key Findings:

- The multi-class classification model implemented using a neural network performed reasonably well in classifying the synthetic dataset with five distinct classes.

- The model achieved satisfactory accuracy, precision, recall, and F1-score metrics on the test set, indicating its effectiveness in distinguishing between different classes.

Challenges :

- One challenge was adapting the neural network architecture and loss function to a support multi-class classification. This required modifying the output layer to have five neurons and using the CrossEntropyLoss function for calculating the loss.

- Handling the data preprocessing, such as one-hot encoding the class labels, was essential for ensuring compatibility with the neural network model.

- Debugging and troubleshooting errors, such as tensor shape mismatches and attribute errors, required careful attention to detail and understanding of PyTorch functionality.

Potential Improvements and Further Experiments:

- Experimenting with different network architectures, including varying the number of hidden layers and neurons, could help optimize the model's performance further.

- Fine-tuning hyperparameters such as learning rate, batch size, and number of epochs may lead to better convergence and improved accuracy.

- Data augmentation techniques could be explored to increase the diversity of the training dataset, potentially enhancing the model's ability to generalize to unseen data.

- Investigating more advanced neural network architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), could be beneficial for capturing complex patterns in the data, especially for tasks with high-dimensional inputs or sequential data.