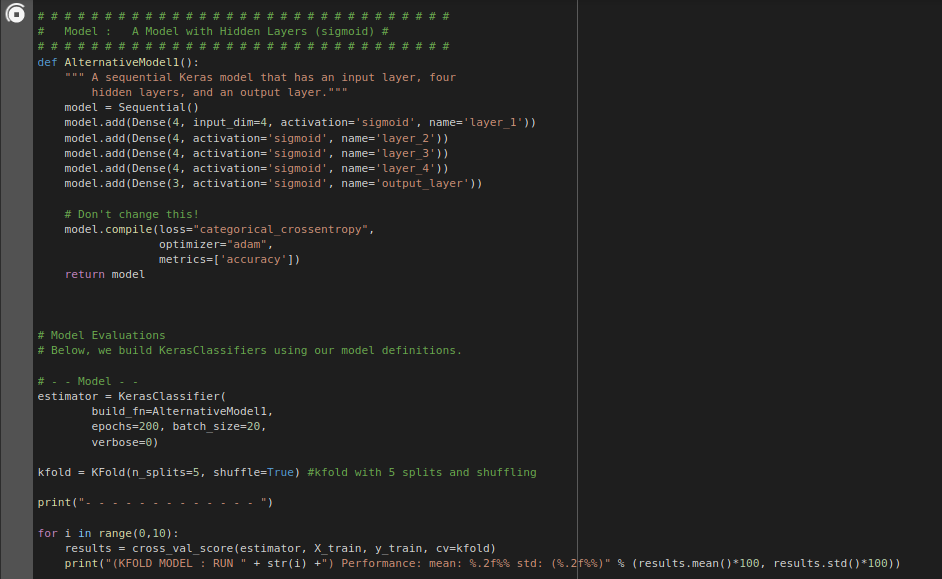
**1. Using the Iris dataset, implement a Keras classifier that uses a neural network model with one input layer, four hidden layers, and an output layer. The input layer has four neurons, each hidden layer has four neurons, and the output layer has three neurons. Each layer should use the sigmoid activation function. Evaluate the classifier with KFold validation across five splits with shuffling. Run the evaluation 10 times, and report the performance mean and standard deviation for each run. Alongside the performance metrics, take a screenshot of your model’s implementation in your source code. [15pts]**



(KFOLD MODEL : RUN 0) Performance: mean: 22.86% std: (3.56%)

(KFOLD MODEL : RUN 1) Performance: mean: 27.62% std: (5.55%)

(KFOLD MODEL : RUN 2) Performance: mean: 27.62% std: (9.71%)

(KFOLD MODEL : RUN 3) Performance: mean: 26.67% std: (16.11%)

(KFOLD MODEL : RUN 4) Performance: mean: 34.29% std: (9.23%)

(KFOLD MODEL : RUN 5) Performance: mean: 33.33% std: (6.02%)

(KFOLD MODEL : RUN 6) Performance: mean: 29.52% std: (5.55%)

(KFOLD MODEL : RUN 7) Performance: mean: 23.81% std: (5.22%)

(KFOLD MODEL : RUN 8) Performance: mean: 32.38% std: (8.19%)

(KFOLD MODEL : RUN 9) Performance: mean: 28.57% std: (4.26%)

**2. Following your implementation in the prior question, you decide that it makes more**

**sense to simplify the model with two layers. However, you still need to identify the appropriate set of hyperparameters for your neural network. Implement a neural network model that allows you to configure the number of neurons and the activation functions in the network’s hidden layers. Use Scikit-Learn’s GridSearchCV function to identify the optimal hyperparameters. [15pts]**

Best: 0.809524 using {'activation\_func': 'relu', 'neurons': 30}

0.561905 (0.203818) with: {'activation\_func': 'linear', 'neurons': 1}

0.657143 (0.163299) with: {'activation\_func': 'linear', 'neurons': 2}

0.780952 (0.117417) with: {'activation\_func': 'linear', 'neurons': 5}

0.504762 (0.211677) with: {'activation\_func': 'linear', 'neurons': 10}

0.476190 (0.110246) with: {'activation\_func': 'linear', 'neurons': 15}

0.685714 (0.209956) with: {'activation\_func': 'linear', 'neurons': 20}

0.628571 (0.209956) with: {'activation\_func': 'linear', 'neurons': 25}

0.438095 (0.158794) with: {'activation\_func': 'linear', 'neurons': 30}

0.390476 (0.107750) with: {'activation\_func': 'sigmoid', 'neurons': 1}

0.409524 (0.155329) with: {'activation\_func': 'sigmoid', 'neurons': 2}

0.552381 (0.210388) with: {'activation\_func': 'sigmoid', 'neurons': 5}

0.419048 (0.168762) with: {'activation\_func': 'sigmoid', 'neurons': 10}

0.457143 (0.141902) with: {'activation\_func': 'sigmoid', 'neurons': 15}

0.304762 (0.013469) with: {'activation\_func': 'sigmoid', 'neurons': 20}

0.314286 (0.000000) with: {'activation\_func': 'sigmoid', 'neurons': 25}

0.628571 (0.245781) with: {'activation\_func': 'sigmoid', 'neurons': 30}

0.561905 (0.176640) with: {'activation\_func': 'tanh', 'neurons': 1}

0.523810 (0.203818) with: {'activation\_func': 'tanh', 'neurons': 2}

0.800000 (0.101686) with: {'activation\_func': 'tanh', 'neurons': 5}

0.571429 (0.163299) with: {'activation\_func': 'tanh', 'neurons': 10}

0.514286 (0.207348) with: {'activation\_func': 'tanh', 'neurons': 15}

0.609524 (0.097124) with: {'activation\_func': 'tanh', 'neurons': 20}

0.457143 (0.123443) with: {'activation\_func': 'tanh', 'neurons': 25}

0.628571 (0.207348) with: {'activation\_func': 'tanh', 'neurons': 30}

0.457143 (0.163299) with: {'activation\_func': 'relu', 'neurons': 1}

0.571429 (0.176126) with: {'activation\_func': 'relu', 'neurons': 2}

0.514286 (0.145686) with: {'activation\_func': 'relu', 'neurons': 5}

0.533333 (0.175093) with: {'activation\_func': 'relu', 'neurons': 10}

0.628571 (0.222539) with: {'activation\_func': 'relu', 'neurons': 15}

0.580952 (0.088320) with: {'activation\_func': 'relu', 'neurons': 20}

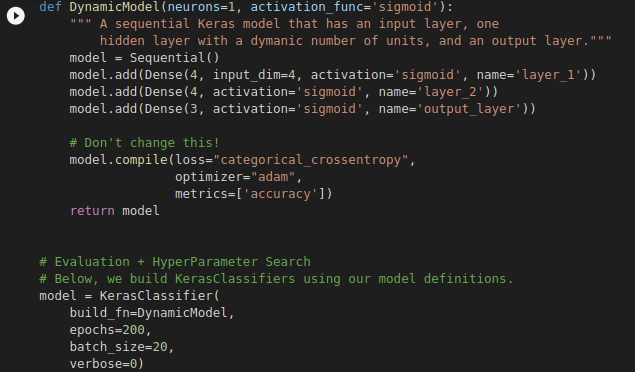
0.304762 (0.035635) with: {'activation\_func': 'relu', 'neurons': 25}

0.809524 (0.117417) with: {'activation\_func': 'relu', 'neurons': 30}

**(a) Below, include a screenshot of your model’s implementation that clearly shows your**

**Sequential() Keras model and its ability to configure the number of neurons and the**

**activation functions in both hidden layers. [5pts]**

****

**(b) State the optimal number of neurons and choice of activation function for both layers as observed via GridSearchCV. [5pts]**

*The optimal number of 'neurons': 30 , with the 'activation\_func': 'relu' .*

**(c) Re-run the GridSearchCV technique on your neural network again, and you will find**

**that your output may suggest a different set of hyperparameters perform best. With this**

**variability in mind, what steps could you take with GridSearchCV to know that you’ve truly reached the optimal set of hyperparameters? [5pts]**

*We have re-ran the GridSearchCV technique and found out that it somehow suggests different hyperparameters for our model and having that in mind and to tackle this thing, we suggest that we should increase the number of epochs for the search to find the optimal hyperparameters for us and don’t settle or get stuck at local minimum situation.*