MLP for Pima Indians Dataset with 10-fold cross validation

from keras.models import Sequential  
from keras.layers import LeakyReLU  
from keras.layers import Dense  
from sklearn.model\_selection import StratifiedKFold  
import numpy  
import pandas as pd  
import keras  
from keras.layers import Dense, Dropout, Flatten, Reshape, GlobalAveragePooling1D  
from keras.layers import Conv2D, MaxPooling2D, Conv1D, MaxPooling1D  
from keras.utils import np\_utils  
  
# fix random seed for reproducibility  
seed = 7  
numpy.random.seed(seed)  
  
# load pima indians dataset  
dataset = pd.read\_csv('data.csv', delimiter=',')  
# split into input (X) and output (Y) variables  
  
X = dataset.iloc[0:5001,0:180].values  
Y = dataset.iloc[5001:11500,0:180].values  
# define 10-fold cross validation test harness  
kfold = StratifiedKFold(n\_splits=10, shuffle=True, random\_state=seed)  
cvscores = []  
#X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=0)  
# 1D CNN neural network  
model\_m = Sequential()  
model\_m.add(Conv1D(4, 6, strides=1,input\_shape=(4097,4)))  
model\_m.add(LeakyReLU(alpha=0.1))  
model\_m.add(MaxPooling1D(4, strides=2))  
model\_m.add(Conv1D(4, 5, strides=1))  
model\_m.add(LeakyReLU(alpha=0.1))  
model\_m.add(MaxPooling1D(4, strides=2))  
model\_m.add(Conv1D(10, 4,strides=1))  
model\_m.add(LeakyReLU(alpha=0.1))  
model\_m.add(MaxPooling1D(10, strides=2))  
model\_m.add(Conv1D(10, 4, strides=1))  
model\_m.add(LeakyReLU(alpha=0.1))  
model\_m.add(MaxPooling1D(10, strides=2))  
model\_m.add(Conv1D(15, 4, strides=1))  
model\_m.add(LeakyReLU(alpha=0.1))  
model\_m.add(MaxPooling1D(15, strides=2))  
model\_m.add(Dense (50))  
model\_m.add(Dense (20))  
model\_m.add(Dense (3,activation='softmax'))  
print(model\_m.summary())  
"""  
# Compile model  
model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])  
# Fit the model  
model.fit(X[train], Y[train], epochs=150, batch\_size=3, verbose=0)  
# evaluate the model  
scores = model.evaluate(X[test], Y[test], verbose=0)  
  
print("%s: %.2f%%" % (model.metrics\_names[1], scores[1]\*100))  
cvscores.append(scores[1] \* 100)  
print("%.2f%% (+/- %.2f%%)" % (numpy.mean(cvscores), numpy.std(cvscores)))  
  
print("\n--- Fit the model ---\n")  
"""  
  
  
# The EarlyStopping callback monitors training accuracy:  
# if it fails to improve for two consecutive epochs,  
# training stops early  
"""  
callbacks\_list = [  
  keras.callbacks.ModelCheckpoint(  
      filepath='best\_model.{epoch:02d}-{val\_loss:.2f}.h5',  
      monitor='val\_loss', save\_best\_only=True),  
  keras.callbacks.EarlyStopping(monitor='acc', patience=1)  
]  
  
model\_m.compile(loss='categorical\_crossentropy',  
              optimizer='adam', metrics=['accuracy'])  
  
# Hyper-parameters  
BATCH\_SIZE = 3  
EPOCHS = 150  
  
# Enable validation to use ModelCheckpoint and EarlyStopping callbacks.  
history = model\_m.fit(X,  
                    Y,  
                    batch\_size=BATCH\_SIZE,  
                    epochs=EPOCHS,  
                    callbacks=callbacks\_list,  
                    validation\_split=0.2,  
                    verbose=1)  
"""