Exercise

K-Fold Manual Validation Using Scikit Learn StratifiedKFold Class & Grid Search

# MLP for Pima Indians Dataset with 10-fold cross validation from keras.models import Sequential

from keras.layers import Dense

from sklearn.model\_selection import StratifiedKFold import numpy

# fix random seed for reproducibility seed = 7

numpy.random.seed(seed)

# load pima indians dataset

dataset = numpy.loadtxt("/gdrive/My Drive/DEEPLEARNINGLAB/diabetes.csv", delimiter=",")

# split into input (X) and output (Y) variables X = dataset[:,0:8]

Y = dataset[:,8]

# define 10-fold cross validation test harness

kfold = StratifiedKFold(n\_splits=10, shuffle=True, random\_state=seed) cvscores = []

for train, test in kfold.split(X, Y): # create model

model = Sequential()

model.add(Dense(12, input\_dim=8, kernel\_initializer='uniform' , activation= 'relu' ))

model.add(Dense(8, kernel\_initializer='uniform' , activation= "relu" )) model.add(Dense(1, kernel\_initializer= 'uniform' , activation= "sigmoid"

))

# Compile model

model.compile(loss= "binary\_crossentropy" , optimizer= "adam" , metrics=[ "accuracy" ])

# Fit the model

model.fit(X[train], Y[train], epochs=150, batch\_size=10, 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)))

from keras.models import Sequential from keras.layers import Dense

from keras.wrappers.scikit\_learn import KerasClassifier from sklearn.model\_selection import GridSearchCV import numpy

# Function to create model, required for KerasClassifier

def create\_model(optimizer= "rmsprop" , init= "glorot\_uniform" ): # create model

model = Sequential()

model.add(Dense(12, input\_dim=8, kernel\_initializer=init, activation= "relu" ))

model.add(Dense(8,kernel\_initializer=init, activation= "relu" )) model.add(Dense(1, kernel\_initializer=init, activation= "sigmoid" )) # Compile model

model.compile(loss= "binary\_crossentropy" , optimizer=optimizer, metrics=[ "accuracy" ])

return model

# fix random seed for reproducibility seed = 7

numpy.random.seed(seed)

# load pima indians dataset

dataset = numpy.loadtxt("/gdrive/My Drive/DEEPLEARNINGLAB/diabetes.csv", delimiter=",")

# split into input (X) and output (Y) variables X = dataset[:,0:8]

Y = dataset[:,8] # create model

model = KerasClassifier(build\_fn=create\_model, verbose=0) # grid search epochs, batch size and optimizer

optimizers = [ "rmsprop" , "adam" ]

init = [ "glorot\_uniform" , "normal" , "uniform" ] epochs = [50, 100, 150]

batches = [5, 10, 20]

param\_grid = dict(optimizer=optimizers, nb\_epoch=epochs, batch\_size=batches, init=init)

grid = GridSearchCV(estimator=model, param\_grid=param\_grid) grid\_result = grid.fit(X, Y)

# summarize results

print("Best: %f using %s" % (grid\_result.best\_score\_, grid\_result.best\_params\_))

means = grid\_result.cv\_results\_[ "mean\_test\_score" ] stds = grid\_result.cv\_results\_[ "std\_test\_score" ] params = grid\_result.cv\_results\_[ "params" ]

for mean, stdev, param in zip(means, stds, params): print("%f (%f) with: %r" % (mean, stdev, param))