from sklearn.model\_selection import StratifiedKFold

# Define 10-fold cross validation

cv= StratifiedKFold(n\_splits=10, random\_state=None, shuffle=True)

# Loop over cross validation folds

acc = []

for train\_index, test\_index in cv.split(X, Y):

x\_train\_fold, x\_test\_fold = X[train\_index], X[test\_index]

y\_train\_fold, y\_test\_fold = Y[train\_index], Y[test\_index]

# Calculate class weights for current split

class\_weights = class\_weight.compute\_class\_weight(class\_weight = "balanced", classes = np.unique(y\_train\_fold), y = y\_train\_fold )

class\_weights = {0: class\_weights[0],1: class\_weights[1]}

# Rebuild the DNN model, to not continue training on the previously trained model

model12 = build\_DNN(input\_shape,2,20,learning\_rate = 0.1, optimizer = "adam")

# Fit the model with training set and class weights for this fold

history12 = model12.fit(x\_train\_fold,y\_train\_fold,batch\_size=batch\_size,epochs=epochs, class\_weight = class\_weights)

# Evaluate the model using the test set for this fold

score = model12.evaluate(Xtest,Ytest,batch\_size = batch\_size)

# Save the test accuracy in an array

acc.append(score[1])

# Calculate and print mean and std of accuracies

print("mean: {}.".format(np.mean(acc)))

print("std: {}.".format(np.std(acc)))

For regression, we would need to change the activation function in the finallayer, the metric used and the loss function that we use.

Question 20: Dropout provides a better way to intergate uncertanity into our model without having to increase the computation needs to do so. CV would require more computation to have this uncertainity into the model