

# Create a scorer assigning more cost to false positives

def my\_scorer(y\_test, y\_est, cost\_fp=10.0, cost\_fn=1.0):

tn, fp, fn, tp = confusion\_matrix(y\_test, y\_est).ravel()

return cost\_fp\*fp + cost\_fn\*fn

# Fit a DecisionTreeClassifier to the data and compute the loss

clf = DecisionTreeClassifier(random\_state=2).fit(X\_train, y\_train)

print(my\_scorer(y\_test, clf.predict(X\_test)))

# Refit with same seed, downweighting subjects weighing > 80

weights = [0.5 if w > 80 else 1.0 for w in X\_train.weight]

clf\_weighted = DecisionTreeClassifier(random\_state=2).fit(

X\_train,y\_train,sample\_weight=weights)

print(my\_scorer(y\_test, clf\_weighted.predict(X\_test)))