project 4

Dipesh Poudel

1/4/2022

# Repeated K-Fold Cross Validation

## Reading the File

library(haven)  
bank\_loan\_df <- read\_sav("P4\_bankloan\_5000\_clients.sav")

## Changing the data type of variables

bank\_loan\_df$defaulted\_loan<-as.factor(bank\_loan\_df$defaulted\_loan)  
bank\_loan\_df$education\_level<-as.factor(bank\_loan\_df$education\_level)

## Splitting the data into train and test set

set.seed(1234)  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

ind<-sample(2,nrow(bank\_loan\_df),replace=T,prob = c(0.7,0.3))  
train\_data<-bank\_loan\_df[ind==1,]  
test\_data<-bank\_loan\_df[ind==2,]

## Setting Up the Train Control

rep\_cv\_train\_control<-trainControl(method = "repeatedcv",number = 10,repeats = 3)

## Logistic Regression With Repeated Cross Validation

### Training Logistic Regression Model

logistic\_clf1<-train(defaulted\_loan~.,  
 data=train\_data,  
 method="glm",  
 family="binomial",  
 trControl=rep\_cv\_train\_control  
)  
summary(logistic\_clf1)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.6490 -0.6635 -0.3442 0.1409 3.2833   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.235986 0.272446 -4.537 5.72e-06 \*\*\*  
## age 0.006492 0.008297 0.782 0.4339   
## education\_level2 0.227329 0.110244 2.062 0.0392 \*   
## education\_level3 0.260781 0.156468 1.667 0.0956 .   
## education\_level4 0.285038 0.186776 1.526 0.1270   
## education\_level5 0.020994 0.447370 0.047 0.9626   
## current\_employ\_year -0.182777 0.012678 -14.416 < 2e-16 \*\*\*  
## current\_address\_year -0.094317 0.010300 -9.157 < 2e-16 \*\*\*  
## income\_household -0.002470 0.003879 -0.637 0.5244   
## debt\_income\_ratio 0.099652 0.012885 7.734 1.04e-14 \*\*\*  
## credit\_card\_debt 0.425066 0.044558 9.540 < 2e-16 \*\*\*  
## other\_debts 0.006704 0.030495 0.220 0.8260   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 3994.4 on 3524 degrees of freedom  
## Residual deviance: 2850.2 on 3513 degrees of freedom  
## AIC: 2874.2  
##   
## Number of Fisher Scoring iterations: 6

### Making the Prediction

predicted\_val\_log1<-predict(logistic\_clf1,newdata = test\_data)

### Confusion Matrix for Evaluation

confusionMatrix(predicted\_val\_log1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1038 191  
## 1 76 170  
##   
## Accuracy : 0.819   
## 95% CI : (0.7984, 0.8383)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 2.487e-09   
##   
## Kappa : 0.4513   
##   
## Mcnemar's Test P-Value : 3.022e-12   
##   
## Sensitivity : 0.9318   
## Specificity : 0.4709   
## Pos Pred Value : 0.8446   
## Neg Pred Value : 0.6911   
## Prevalence : 0.7553   
## Detection Rate : 0.7037   
## Detection Prevalence : 0.8332   
## Balanced Accuracy : 0.7013   
##   
## 'Positive' Class : 0   
##

## KNN Model with Repeated Cross validation

### Training KNN Model

knn\_clf1<-train(defaulted\_loan~.,data = train\_data,  
 method="knn",  
 trControl=rep\_cv\_train\_control  
 )

### Getting the Result of the Model

knn\_clf1$result

## k Accuracy Kappa AccuracySD KappaSD  
## 1 5 0.7651051 0.3108629 0.01793387 0.04889046  
## 2 7 0.7734288 0.3193413 0.01600585 0.04842754  
## 3 9 0.7749418 0.3174315 0.01861342 0.05753687

### Confusion Matrix for Model Evaluation

predicted\_val\_knn1<-predict(knn\_clf1,newdata = test\_data)

confusionMatrix(predicted\_val\_knn1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1019 226  
## 1 95 135  
##   
## Accuracy : 0.7824   
## 95% CI : (0.7604, 0.8032)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.007801   
##   
## Kappa : 0.329   
##   
## Mcnemar's Test P-Value : 3.99e-13   
##   
## Sensitivity : 0.9147   
## Specificity : 0.3740   
## Pos Pred Value : 0.8185   
## Neg Pred Value : 0.5870   
## Prevalence : 0.7553   
## Detection Rate : 0.6908   
## Detection Prevalence : 0.8441   
## Balanced Accuracy : 0.6443   
##   
## 'Positive' Class : 0   
##

## Naïve Bayes classifier

### Training the Model

library(naivebayes)

## naivebayes 0.9.7 loaded

nb\_clf1<-train(defaulted\_loan~.,  
 data=train\_data,  
 method="naive\_bayes",  
 usepoisson = TRUE,  
 trControl=rep\_cv\_train\_control  
 )

summary(nb\_clf1)

##   
## ================================== Naive Bayes ==================================   
##   
## - Call: naive\_bayes.default(x = x, y = y, laplace = param$laplace, usekernel = TRUE, usepoisson = TRUE, adjust = param$adjust)   
## - Laplace: 0   
## - Classes: 2   
## - Samples: 3525   
## - Features: 11   
## - Conditional distributions:   
## - KDE: 11  
## - Prior probabilities:   
## - 0: 0.7461  
## - 1: 0.2539  
##   
## ---------------------------------------------------------------------------------

### Making Prediction on Test Data

predicted\_val\_nb1<-predict(nb\_clf1,newdata = test\_data)

### Confusion Matrix for Model Evaluation

confusionMatrix(predicted\_val\_nb1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1094 308  
## 1 20 53  
##   
## Accuracy : 0.7776   
## 95% CI : (0.7555, 0.7986)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.02363   
##   
## Kappa : 0.1764   
##   
## Mcnemar's Test P-Value : < 2e-16   
##   
## Sensitivity : 0.9820   
## Specificity : 0.1468   
## Pos Pred Value : 0.7803   
## Neg Pred Value : 0.7260   
## Prevalence : 0.7553   
## Detection Rate : 0.7417   
## Detection Prevalence : 0.9505   
## Balanced Accuracy : 0.5644   
##   
## 'Positive' Class : 0   
##

## Support Vector Machine (SVM) Model

### Training the Model

svm\_clf1<-train(defaulted\_loan~.,  
 data=train\_data,  
 method="svmLinear",  
 trControl=rep\_cv\_train\_control,  
 )

svm\_clf1

## Support Vector Machines with Linear Kernel   
##   
## 3525 samples  
## 8 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3172, 3172, 3173, 3172, 3172, 3173, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.803591 0.3896318  
##   
## Tuning parameter 'C' was held constant at a value of 1

### Making the Prediction for test data

predicted\_val\_svm1<-predict(svm\_clf1,newdata = test\_data)

### Confusion Matrix for Model Evaluation

confusionMatrix(predicted\_val\_svm1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1055 218  
## 1 59 143  
##   
## Accuracy : 0.8122   
## 95% CI : (0.7913, 0.8318)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 9.898e-08   
##   
## Kappa : 0.4032   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9470   
## Specificity : 0.3961   
## Pos Pred Value : 0.8288   
## Neg Pred Value : 0.7079   
## Prevalence : 0.7553   
## Detection Rate : 0.7153   
## Detection Prevalence : 0.8631   
## Balanced Accuracy : 0.6716   
##   
## 'Positive' Class : 0   
##

## Decision Tree Model

dtree\_clf1<-train(defaulted\_loan~.,  
 data = train\_data,  
 method="rpart",  
 parms = list(split = "information"),  
 tuneLength=10,  
 trControl=rep\_cv\_train\_control  
 )

dtree\_clf1

## CART   
##   
## 3525 samples  
## 8 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3172, 3173, 3173, 3172, 3172, 3173, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.002793296 0.7823196 0.3521905  
## 0.002979516 0.7835485 0.3489152  
## 0.003072626 0.7840206 0.3495299  
## 0.003351955 0.7835466 0.3399202  
## 0.004469274 0.7850591 0.3386658  
## 0.005586592 0.7848705 0.3342402  
## 0.006703911 0.7819413 0.3186385  
## 0.024581006 0.7692707 0.2886086  
## 0.027374302 0.7677572 0.2879412  
## 0.060335196 0.7572640 0.2116461  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.004469274.

### Making the Prediction for test data

predicted\_val\_dtree1<-predict(dtree\_clf1,newdata = test\_data)

### Confusion Matrix for Model Evaluation

confusionMatrix(predicted\_val\_dtree1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1022 209  
## 1 92 152  
##   
## Accuracy : 0.7959   
## 95% CI : (0.7744, 0.8162)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.0001207   
##   
## Kappa : 0.3801   
##   
## Mcnemar's Test P-Value : 2.292e-11   
##   
## Sensitivity : 0.9174   
## Specificity : 0.4211   
## Pos Pred Value : 0.8302   
## Neg Pred Value : 0.6230   
## Prevalence : 0.7553   
## Detection Rate : 0.6929   
## Detection Prevalence : 0.8346   
## Balanced Accuracy : 0.6692   
##   
## 'Positive' Class : 0   
##

## Artifical Neural Network (ANN) Model

### Training the Model

ann\_clf1 <- train(defaulted\_loan ~ ., data = train\_data,   
 method = "nnet",  
 preProcess = c("center","scale"),   
 maxit = 250, # Maximum number of iterations  
 tuneGrid = data.frame(size = 1, decay = 0),  
 # tuneGrid = data.frame(size = 0, decay = 0),skip=TRUE, # Technically, this is log-reg  
 metric = "Accuracy",  
 trControl=rep\_cv\_train\_control)

## # weights: 14  
## initial value 2023.149797   
## iter 10 value 1396.519266  
## iter 20 value 1339.721019  
## iter 30 value 1291.143105  
## iter 40 value 1285.932746  
## iter 50 value 1279.022143  
## iter 60 value 1277.575386  
## iter 70 value 1277.538891  
## iter 80 value 1277.170176  
## final value 1277.080129   
## converged  
## # weights: 14  
## initial value 2075.998442   
## iter 10 value 1685.968290  
## iter 20 value 1429.788753  
## iter 30 value 1398.698201  
## iter 40 value 1380.325772  
## iter 50 value 1343.667446  
## iter 60 value 1300.588363  
## iter 70 value 1291.661419  
## iter 80 value 1288.774192  
## iter 90 value 1286.795333  
## iter 100 value 1286.645154  
## iter 110 value 1286.569504  
## iter 120 value 1286.445126  
## final value 1286.438147   
## converged  
## # weights: 14  
## initial value 2307.401589   
## iter 10 value 1382.621705  
## iter 20 value 1318.848750  
## iter 30 value 1299.579451  
## iter 40 value 1298.166463  
## iter 50 value 1296.179017  
## iter 60 value 1295.850513  
## iter 70 value 1295.847257  
## iter 80 value 1295.813297  
## iter 90 value 1295.804795  
## final value 1295.804482   
## converged  
## # weights: 14  
## initial value 2634.070303   
## iter 10 value 1524.268698  
## iter 20 value 1395.671928  
## iter 30 value 1368.490884  
## iter 40 value 1337.965137  
## iter 50 value 1309.141591  
## iter 60 value 1299.060672  
## iter 70 value 1298.713286  
## iter 80 value 1297.609515  
## iter 90 value 1297.332195  
## iter 100 value 1297.330163  
## iter 110 value 1297.195503  
## iter 120 value 1297.125295  
## final value 1297.125172   
## converged  
## # weights: 14  
## initial value 2590.855955   
## iter 10 value 1553.270296  
## iter 20 value 1390.064714  
## iter 30 value 1353.838503  
## iter 40 value 1326.129503  
## iter 50 value 1289.182883  
## iter 60 value 1276.407980  
## iter 70 value 1276.009298  
## iter 80 value 1273.850230  
## iter 90 value 1273.707804  
## iter 100 value 1273.704175  
## final value 1273.703721   
## converged  
## # weights: 14  
## initial value 1930.479881   
## iter 10 value 1384.499921  
## iter 20 value 1314.575992  
## iter 30 value 1282.100710  
## iter 40 value 1278.443980  
## iter 50 value 1272.017716  
## iter 60 value 1270.889612  
## iter 70 value 1270.876800  
## iter 80 value 1270.832039  
## final value 1270.831379   
## converged  
## # weights: 14  
## initial value 2663.967914   
## iter 10 value 1396.718254  
## iter 20 value 1318.279690  
## iter 30 value 1274.306674  
## iter 40 value 1271.672161  
## iter 50 value 1266.122410  
## iter 60 value 1265.551462  
## iter 70 value 1265.538659  
## iter 80 value 1265.536468  
## final value 1265.536125   
## converged  
## # weights: 14  
## initial value 2772.603285   
## iter 10 value 1552.501121  
## iter 20 value 1340.778431  
## iter 30 value 1296.805987  
## iter 40 value 1292.018901  
## iter 50 value 1286.088233  
## iter 60 value 1285.214303  
## iter 70 value 1285.208337  
## iter 80 value 1285.064494  
## final value 1285.046914   
## converged  
## # weights: 14  
## initial value 1852.636110   
## iter 10 value 1425.261378  
## iter 20 value 1375.504339  
## iter 30 value 1314.692033  
## iter 40 value 1303.238874  
## iter 50 value 1292.916713  
## iter 60 value 1290.452098  
## iter 70 value 1290.397707  
## iter 80 value 1289.954672  
## iter 90 value 1289.875769  
## final value 1289.874678   
## converged  
## # weights: 14  
## initial value 2820.190738   
## iter 10 value 1554.587786  
## iter 20 value 1413.742630  
## iter 30 value 1347.174623  
## iter 40 value 1325.338714  
## iter 50 value 1291.297234  
## iter 60 value 1280.095921  
## iter 70 value 1279.817675  
## iter 80 value 1278.358529  
## iter 90 value 1278.047963  
## iter 100 value 1278.035790  
## iter 110 value 1277.951916  
## iter 120 value 1277.918397  
## final value 1277.917660   
## converged  
## # weights: 14  
## initial value 2227.721605   
## iter 10 value 1440.128135  
## iter 20 value 1368.233842  
## iter 30 value 1315.984060  
## iter 40 value 1296.332188  
## iter 50 value 1283.441965  
## iter 60 value 1281.728902  
## iter 70 value 1281.619841  
## iter 80 value 1281.170901  
## iter 90 value 1281.134853  
## iter 90 value 1281.134850  
## iter 90 value 1281.134850  
## final value 1281.134850   
## converged  
## # weights: 14  
## initial value 2272.078969   
## iter 10 value 1700.735294  
## iter 20 value 1506.491264  
## iter 30 value 1440.349154  
## iter 40 value 1368.736493  
## iter 50 value 1315.981319  
## iter 60 value 1295.178432  
## iter 70 value 1291.358723  
## iter 80 value 1287.693202  
## iter 90 value 1287.275176  
## iter 100 value 1287.158231  
## iter 110 value 1287.025819  
## final value 1287.019179   
## converged  
## # weights: 14  
## initial value 2367.202574   
## iter 10 value 1464.520042  
## iter 20 value 1360.767832  
## iter 30 value 1321.368147  
## iter 40 value 1303.034486  
## iter 50 value 1286.147505  
## iter 60 value 1282.397332  
## iter 70 value 1282.242085  
## iter 80 value 1281.108492  
## iter 90 value 1280.745727  
## iter 100 value 1280.744676  
## iter 110 value 1280.583758  
## iter 120 value 1280.500901  
## final value 1280.500690   
## converged  
## # weights: 14  
## initial value 1984.595780   
## iter 10 value 1418.682761  
## iter 20 value 1310.193381  
## iter 30 value 1285.665829  
## iter 40 value 1284.465386  
## iter 50 value 1281.369087  
## iter 60 value 1280.924733  
## iter 70 value 1280.920216  
## iter 80 value 1280.861217  
## iter 90 value 1280.849474  
## iter 90 value 1280.849470  
## iter 90 value 1280.849469  
## final value 1280.849469   
## converged  
## # weights: 14  
## initial value 2686.297777   
## iter 10 value 1610.978443  
## iter 20 value 1421.641614  
## iter 30 value 1382.876147  
## iter 40 value 1381.338959  
## final value 1381.138847   
## converged  
## # weights: 14  
## initial value 1996.628654   
## iter 10 value 1315.945479  
## iter 20 value 1289.143529  
## iter 30 value 1280.492832  
## iter 40 value 1279.259374  
## iter 50 value 1278.298362  
## iter 60 value 1277.817955  
## iter 70 value 1277.797976  
## iter 80 value 1277.737987  
## iter 90 value 1277.657522  
## iter 100 value 1277.653559  
## iter 100 value 1277.653557  
## iter 100 value 1277.653555  
## final value 1277.653555   
## converged  
## # weights: 14  
## initial value 2075.083323   
## iter 10 value 1326.043553  
## iter 20 value 1291.478662  
## iter 30 value 1281.371795  
## iter 40 value 1280.493315  
## iter 50 value 1279.955925  
## iter 60 value 1279.797880  
## final value 1279.795883   
## converged  
## # weights: 14  
## initial value 1959.331375   
## iter 10 value 1394.272522  
## iter 20 value 1356.149145  
## iter 30 value 1304.047308  
## iter 40 value 1295.323818  
## iter 50 value 1285.086235  
## iter 60 value 1283.594176  
## iter 70 value 1283.585464  
## iter 80 value 1282.998802  
## iter 90 value 1282.824186  
## iter 100 value 1282.822444  
## iter 100 value 1282.822439  
## final value 1282.822354   
## converged  
## # weights: 14  
## initial value 2417.789458   
## iter 10 value 1449.091651  
## iter 20 value 1373.854077  
## iter 30 value 1339.962879  
## iter 40 value 1310.741226  
## iter 50 value 1294.400913  
## iter 60 value 1286.055212  
## iter 70 value 1285.907371  
## iter 80 value 1285.149081  
## iter 90 value 1285.043787  
## final value 1285.043702   
## converged  
## # weights: 14  
## initial value 2888.453317   
## iter 10 value 1464.256748  
## iter 20 value 1344.105614  
## iter 30 value 1294.298879  
## iter 40 value 1288.773657  
## iter 50 value 1282.217670  
## iter 60 value 1281.377534  
## iter 70 value 1281.367936  
## iter 80 value 1281.266537  
## final value 1281.251455   
## converged  
## # weights: 14  
## initial value 1887.372005   
## iter 10 value 1400.878062  
## iter 20 value 1365.599202  
## iter 30 value 1315.964013  
## iter 40 value 1301.620993  
## iter 50 value 1289.354081  
## iter 60 value 1286.223788  
## iter 70 value 1286.100957  
## iter 80 value 1285.361669  
## iter 90 value 1285.189718  
## final value 1285.188574   
## converged  
## # weights: 14  
## initial value 2008.575403   
## iter 10 value 1571.079561  
## iter 20 value 1461.535559  
## iter 30 value 1378.150901  
## iter 40 value 1346.377795  
## iter 50 value 1303.164239  
## iter 60 value 1292.224151  
## iter 70 value 1290.441503  
## iter 80 value 1286.948739  
## iter 90 value 1286.365614  
## iter 100 value 1286.361314  
## iter 110 value 1286.259684  
## iter 120 value 1286.230082  
## final value 1286.229702   
## converged  
## # weights: 14  
## initial value 2288.209682   
## iter 10 value 1394.544393  
## iter 20 value 1321.918177  
## iter 30 value 1290.546847  
## iter 40 value 1282.351934  
## iter 50 value 1275.497362  
## iter 60 value 1274.862112  
## iter 70 value 1274.830259  
## iter 80 value 1274.691475  
## iter 90 value 1274.670924  
## iter 90 value 1274.670920  
## iter 90 value 1274.670909  
## final value 1274.670909   
## converged  
## # weights: 14  
## initial value 2139.539512   
## iter 10 value 1456.601762  
## iter 20 value 1348.540521  
## iter 30 value 1299.888023  
## iter 40 value 1289.722011  
## iter 50 value 1275.268458  
## iter 60 value 1273.118391  
## iter 70 value 1272.974080  
## iter 80 value 1272.390348  
## iter 90 value 1272.084470  
## iter 100 value 1272.077913  
## final value 1272.076800   
## converged  
## # weights: 14  
## initial value 2546.649428   
## iter 10 value 1472.101624  
## iter 20 value 1368.314323  
## iter 30 value 1332.561487  
## iter 40 value 1308.375382  
## iter 50 value 1294.272762  
## iter 60 value 1291.694111  
## iter 70 value 1291.632784  
## iter 80 value 1291.206270  
## iter 90 value 1291.136016  
## final value 1291.135930   
## converged  
## # weights: 14  
## initial value 2335.878721   
## iter 10 value 1671.231451  
## iter 20 value 1525.411096  
## iter 30 value 1394.550867  
## iter 40 value 1362.396427  
## iter 50 value 1328.724481  
## iter 60 value 1325.663487  
## iter 70 value 1325.381421  
## final value 1325.359257   
## converged  
## # weights: 14  
## initial value 2310.099754   
## iter 10 value 1626.583836  
## iter 20 value 1429.109093  
## iter 30 value 1393.021653  
## iter 40 value 1378.556393  
## iter 50 value 1377.644767  
## final value 1377.576515   
## converged  
## # weights: 14  
## initial value 2118.655109   
## iter 10 value 1494.719324  
## iter 20 value 1368.621992  
## iter 30 value 1326.569503  
## iter 40 value 1320.412929  
## iter 50 value 1293.806933  
## iter 60 value 1288.219990  
## iter 70 value 1286.310937  
## iter 80 value 1286.258114  
## iter 90 value 1286.079770  
## iter 100 value 1286.030067  
## final value 1286.028515   
## converged  
## # weights: 14  
## initial value 2085.508909   
## iter 10 value 1367.230484  
## iter 20 value 1331.111227  
## iter 30 value 1292.119793  
## iter 40 value 1288.248447  
## iter 50 value 1285.913616  
## iter 60 value 1285.069499  
## iter 70 value 1285.036987  
## final value 1285.036583   
## converged  
## # weights: 14  
## initial value 2001.443726   
## iter 10 value 1618.960466  
## iter 20 value 1420.722848  
## iter 30 value 1377.521088  
## iter 40 value 1317.413609  
## iter 50 value 1292.071170  
## iter 60 value 1278.838320  
## iter 70 value 1274.764541  
## iter 80 value 1274.458125  
## iter 90 value 1274.111448  
## iter 100 value 1273.897486  
## iter 110 value 1273.895405  
## iter 120 value 1273.892889  
## iter 130 value 1273.867364  
## iter 140 value 1273.851089  
## iter 140 value 1273.851085  
## final value 1273.850949   
## converged  
## # weights: 14  
## initial value 3049.618042   
## iter 10 value 1657.349057  
## iter 20 value 1511.673346  
## iter 30 value 1468.305091  
## iter 40 value 1461.231457  
## iter 50 value 1460.123297  
## iter 50 value 1460.123290  
## iter 50 value 1460.123290  
## final value 1460.123290   
## converged

### Making the Predictions for Test data

predicted\_val\_ann1<-predict(ann\_clf1,newdata = test\_data)

### Confusion Matrix for the Model Evaluation

confusionMatrix(predicted\_val\_ann1,test\_data$defaulted\_loan)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1015 178  
## 1 99 183  
##   
## Accuracy : 0.8122   
## 95% CI : (0.7913, 0.8318)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 9.898e-08   
##   
## Kappa : 0.4514   
##   
## Mcnemar's Test P-Value : 2.778e-06   
##   
## Sensitivity : 0.9111   
## Specificity : 0.5069   
## Pos Pred Value : 0.8508   
## Neg Pred Value : 0.6489   
## Prevalence : 0.7553   
## Detection Rate : 0.6881   
## Detection Prevalence : 0.8088   
## Balanced Accuracy : 0.7090   
##   
## 'Positive' Class : 0   
##