project 4

Dipesh Poudel

1/4/2022

# 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

cv\_train\_control<-trainControl(method = "cv",number = 10)

## Logistic Regression With Cross Validation

### Training Logistic Regression Model

logistic\_clf1<-train(defaulted\_loan~.,  
 data=train\_data,  
 method="glm",  
 family="binomial",  
 trControl=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 Cross validation

### Training KNN Model

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

### Getting the Result of the Model

knn\_clf1$result

## k Accuracy Kappa AccuracySD KappaSD  
## 1 5 0.7668056 0.3138335 0.01709039 0.03827953  
## 2 7 0.7727611 0.3210782 0.01581367 0.03762291  
## 3 9 0.7744568 0.3184971 0.01934934 0.05507607

### 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=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=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)   
## Summary of sample sizes: 3173, 3172, 3173, 3173, 3173, 3172, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8033994 0.3865814  
##   
## 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=cv\_train\_control  
 )  
dtree\_clf1

## CART   
##   
## 3525 samples  
## 8 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 3172, 3172, 3173, 3173, 3173, 3172, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.002793296 0.7790191 0.3351323  
## 0.002979516 0.7798698 0.3382538  
## 0.003072626 0.7798698 0.3382538  
## 0.003351955 0.7838374 0.3440170  
## 0.004469274 0.7832684 0.3427673  
## 0.005586592 0.7841183 0.3451674  
## 0.006703911 0.7844024 0.3445264  
## 0.024581006 0.7753267 0.3038560  
## 0.027374302 0.7707877 0.3028661  
## 0.060335196 0.7560303 0.1766183  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.006703911.

### 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 1024 214  
## 1 90 147  
##   
## Accuracy : 0.7939   
## 95% CI : (0.7723, 0.8143)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 0.0002488   
##   
## Kappa : 0.3693   
##   
## Mcnemar's Test P-Value : 1.732e-12   
##   
## Sensitivity : 0.9192   
## Specificity : 0.4072   
## Pos Pred Value : 0.8271   
## Neg Pred Value : 0.6203   
## Prevalence : 0.7553   
## Detection Rate : 0.6942   
## Detection Prevalence : 0.8393   
## Balanced Accuracy : 0.6632   
##   
## '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=cv\_train\_control)

## # weights: 14  
## initial value 1957.831661   
## iter 10 value 1470.734611  
## iter 20 value 1393.238325  
## iter 30 value 1372.230110  
## iter 40 value 1342.635366  
## iter 50 value 1299.645907  
## iter 60 value 1283.948399  
## iter 70 value 1283.601491  
## iter 80 value 1281.273801  
## iter 90 value 1280.892055  
## iter 100 value 1280.886772  
## final value 1280.886400   
## converged  
## # weights: 14  
## initial value 2457.375004   
## iter 10 value 1403.867718  
## iter 20 value 1313.856473  
## iter 30 value 1291.856301  
## iter 40 value 1290.082376  
## iter 50 value 1286.375591  
## iter 60 value 1285.636133  
## iter 70 value 1285.619953  
## iter 80 value 1285.599991  
## iter 90 value 1285.451576  
## iter 100 value 1285.424930  
## iter 100 value 1285.424918  
## final value 1285.424327   
## converged  
## # weights: 14  
## initial value 2135.919813   
## iter 10 value 1467.162983  
## iter 20 value 1367.729173  
## iter 30 value 1333.809176  
## iter 40 value 1293.242411  
## iter 50 value 1273.523033  
## iter 60 value 1270.188117  
## iter 70 value 1270.107126  
## iter 80 value 1269.874927  
## final value 1269.854275   
## converged  
## # weights: 14  
## initial value 1912.504258   
## iter 10 value 1484.341341  
## iter 20 value 1312.499654  
## iter 30 value 1294.849732  
## iter 40 value 1287.379750  
## iter 50 value 1281.662293  
## iter 60 value 1280.295441  
## iter 70 value 1280.263944  
## iter 80 value 1280.137462  
## final value 1280.116518   
## converged  
## # weights: 14  
## initial value 2712.402101   
## iter 10 value 1634.986172  
## iter 20 value 1432.102402  
## iter 30 value 1368.763087  
## iter 40 value 1334.902458  
## iter 50 value 1301.638022  
## iter 60 value 1296.836091  
## iter 70 value 1296.710499  
## final value 1296.678765   
## converged  
## # weights: 14  
## initial value 2027.675308   
## iter 10 value 1460.912864  
## iter 20 value 1380.191113  
## iter 30 value 1343.305293  
## iter 40 value 1308.820613  
## iter 50 value 1287.920442  
## iter 60 value 1282.450425  
## iter 70 value 1282.030123  
## iter 80 value 1281.064854  
## iter 90 value 1280.847716  
## iter 100 value 1280.843617  
## final value 1280.843217   
## converged  
## # weights: 14  
## initial value 3067.836186   
## iter 10 value 1477.031599  
## iter 20 value 1372.641152  
## iter 30 value 1319.035272  
## iter 40 value 1312.295634  
## iter 50 value 1306.681402  
## iter 60 value 1305.656208  
## iter 70 value 1305.627085  
## iter 80 value 1305.322532  
## iter 90 value 1305.240048  
## final value 1305.238766   
## converged  
## # weights: 14  
## initial value 1870.856482   
## iter 10 value 1411.886341  
## iter 20 value 1373.136030  
## iter 30 value 1313.845701  
## iter 40 value 1296.538758  
## iter 50 value 1283.919622  
## iter 60 value 1280.768395  
## iter 70 value 1280.693740  
## iter 80 value 1279.920838  
## iter 90 value 1279.731647  
## iter 100 value 1279.729881  
## iter 110 value 1279.602928  
## iter 120 value 1279.562852  
## final value 1279.562718   
## converged  
## # weights: 14  
## initial value 3177.112625   
## iter 10 value 1344.027518  
## iter 20 value 1303.825604  
## iter 30 value 1292.214739  
## iter 40 value 1291.443104  
## iter 50 value 1290.950096  
## iter 60 value 1290.726340  
## iter 70 value 1290.723518  
## final value 1290.723408   
## converged  
## # weights: 14  
## initial value 2029.802031   
## iter 10 value 1376.355909  
## iter 20 value 1342.960010  
## iter 30 value 1293.354191  
## iter 40 value 1288.558534  
## iter 50 value 1280.379811  
## iter 60 value 1279.381284  
## iter 70 value 1279.371294  
## iter 80 value 1279.170828  
## iter 90 value 1279.143356  
## final value 1279.143198   
## converged  
## # weights: 14  
## initial value 2158.620352   
## iter 10 value 1570.467865  
## iter 20 value 1490.030616  
## iter 30 value 1443.848678  
## iter 40 value 1435.027116  
## iter 50 value 1427.119878  
## iter 60 value 1425.182317  
## iter 70 value 1425.165167  
## iter 80 value 1424.982588  
## iter 90 value 1424.930021  
## final value 1424.929836   
## 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 1036 191  
## 1 78 170  
##   
## Accuracy : 0.8176   
## 95% CI : (0.797, 0.837)  
## No Information Rate : 0.7553   
## P-Value [Acc > NIR] : 5.382e-09   
##   
## Kappa : 0.4483   
##   
## Mcnemar's Test P-Value : 8.565e-12   
##   
## Sensitivity : 0.9300   
## Specificity : 0.4709   
## Pos Pred Value : 0.8443   
## Neg Pred Value : 0.6855   
## Prevalence : 0.7553   
## Detection Rate : 0.7024   
## Detection Prevalence : 0.8319   
## Balanced Accuracy : 0.7004   
##   
## 'Positive' Class : 0   
##