R Notebook

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*. #Random Forest analysis of the credit data

#https://archive.ics.uci.edu/ml/machine-learning-databases/00350/default%20of%20credit%20card%20clients.xls  
#Step 1: Collect data  
credit <- read.csv("default\_of\_credit\_card\_clients\_1.csv")  
set.seed(124)

In the csv file used- default\_of\_credit\_card\_clients\_1.csv, column names X1, X2… are discarded to keep only the more meaningful names given in row2- LIMIT\_BAL, SEX.. present in the second row remove the ID variable to avoid overfitting and remove factor names additionally present in row

credit<-credit[,-1]  
#Step 2: exploring the data  
str(credit)

## 'data.frame': 30000 obs. of 24 variables:  
## $ LIMIT\_BAL : int 20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...  
## $ SEX : int 2 2 2 2 1 1 1 2 2 1 ...  
## $ EDUCATION : int 2 2 2 2 2 1 1 2 3 3 ...  
## $ MARRIAGE : int 1 2 2 1 1 2 2 2 1 2 ...  
## $ AGE : int 24 26 34 37 57 37 29 23 28 35 ...  
## $ PAY\_0 : int 2 -1 0 0 -1 0 0 0 0 -2 ...  
## $ PAY\_2 : int 2 2 0 0 0 0 0 -1 0 -2 ...  
## $ PAY\_3 : int -1 0 0 0 -1 0 0 -1 2 -2 ...  
## $ PAY\_4 : int -1 0 0 0 0 0 0 0 0 -2 ...  
## $ PAY\_5 : int -2 0 0 0 0 0 0 0 0 -1 ...  
## $ PAY\_6 : int -2 2 0 0 0 0 0 -1 0 -1 ...  
## $ BILL\_AMT1 : int 3913 2682 29239 46990 8617 64400 367965 11876 11285 0 ...  
## $ BILL\_AMT2 : int 3102 1725 14027 48233 5670 57069 412023 380 14096 0 ...  
## $ BILL\_AMT3 : int 689 2682 13559 49291 35835 57608 445007 601 12108 0 ...  
## $ BILL\_AMT4 : int 0 3272 14331 28314 20940 19394 542653 221 12211 0 ...  
## $ BILL\_AMT5 : int 0 3455 14948 28959 19146 19619 483003 -159 11793 13007 ...  
## $ BILL\_AMT6 : int 0 3261 15549 29547 19131 20024 473944 567 3719 13912 ...  
## $ PAY\_AMT1 : int 0 0 1518 2000 2000 2500 55000 380 3329 0 ...  
## $ PAY\_AMT2 : int 689 1000 1500 2019 36681 1815 40000 601 0 0 ...  
## $ PAY\_AMT3 : int 0 1000 1000 1200 10000 657 38000 0 432 0 ...  
## $ PAY\_AMT4 : int 0 1000 1000 1100 9000 1000 20239 581 1000 13007 ...  
## $ PAY\_AMT5 : int 0 0 1000 1069 689 1000 13750 1687 1000 1122 ...  
## $ PAY\_AMT6 : int 0 2000 5000 1000 679 800 13770 1542 1000 0 ...  
## $ default.payment.next.month: int 1 1 0 0 0 0 0 0 0 0 ...

class(credit)

## [1] "data.frame"

#preprocessing: convert nominal variables to factors  
credit2 <- credit  
credit2[,c(2:4,6:11,24)] <- lapply(credit[,c(2:4,6:11,24)], as.factor)

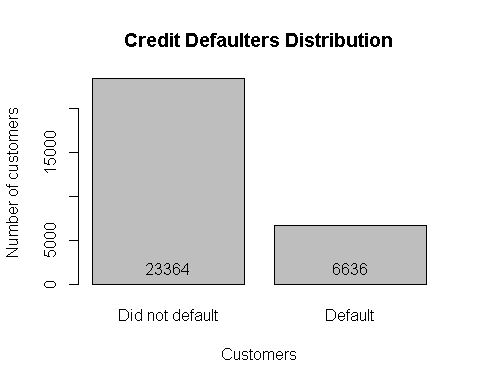
#2. Class Distribution  
  
summary(credit2)

## LIMIT\_BAL SEX EDUCATION MARRIAGE AGE   
## Min. : 10000 1:11888 0: 14 0: 54 Min. :21.00   
## 1st Qu.: 50000 2:18112 1:10585 1:13659 1st Qu.:28.00   
## Median : 140000 2:14030 2:15964 Median :34.00   
## Mean : 167484 3: 4917 3: 323 Mean :35.49   
## 3rd Qu.: 240000 4: 123 3rd Qu.:41.00   
## Max. :1000000 5: 280 Max. :79.00   
## 6: 51   
## PAY\_0 PAY\_2 PAY\_3 PAY\_4   
## 0 :14737 0 :15730 0 :15764 0 :16455   
## -1 : 5686 -1 : 6050 -1 : 5938 -1 : 5687   
## 1 : 3688 2 : 3927 -2 : 4085 -2 : 4348   
## -2 : 2759 -2 : 3782 2 : 3819 2 : 3159   
## 2 : 2667 3 : 326 3 : 240 3 : 180   
## 3 : 322 4 : 99 4 : 76 4 : 69   
## (Other): 141 (Other): 86 (Other): 78 (Other): 102   
## PAY\_5 PAY\_6 BILL\_AMT1 BILL\_AMT2   
## 0 :16947 0 :16286 Min. :-165580 Min. :-69777   
## -1 : 5539 -1 : 5740 1st Qu.: 3559 1st Qu.: 2985   
## -2 : 4546 -2 : 4895 Median : 22382 Median : 21200   
## 2 : 2626 2 : 2766 Mean : 51223 Mean : 49179   
## 3 : 178 3 : 184 3rd Qu.: 67091 3rd Qu.: 64006   
## 4 : 84 4 : 49 Max. : 964511 Max. :983931   
## (Other): 80 (Other): 80   
## BILL\_AMT3 BILL\_AMT4 BILL\_AMT5 BILL\_AMT6   
## Min. :-157264 Min. :-170000 Min. :-81334 Min. :-339603   
## 1st Qu.: 2666 1st Qu.: 2327 1st Qu.: 1763 1st Qu.: 1256   
## Median : 20089 Median : 19052 Median : 18105 Median : 17071   
## Mean : 47013 Mean : 43263 Mean : 40311 Mean : 38872   
## 3rd Qu.: 60165 3rd Qu.: 54506 3rd Qu.: 50191 3rd Qu.: 49198   
## Max. :1664089 Max. : 891586 Max. :927171 Max. : 961664   
##   
## PAY\_AMT1 PAY\_AMT2 PAY\_AMT3 PAY\_AMT4   
## Min. : 0 Min. : 0 Min. : 0 Min. : 0   
## 1st Qu.: 1000 1st Qu.: 833 1st Qu.: 390 1st Qu.: 296   
## Median : 2100 Median : 2009 Median : 1800 Median : 1500   
## Mean : 5664 Mean : 5921 Mean : 5226 Mean : 4826   
## 3rd Qu.: 5006 3rd Qu.: 5000 3rd Qu.: 4505 3rd Qu.: 4013   
## Max. :873552 Max. :1684259 Max. :896040 Max. :621000   
##   
## PAY\_AMT5 PAY\_AMT6 default.payment.next.month  
## Min. : 0.0 Min. : 0.0 0:23364   
## 1st Qu.: 252.5 1st Qu.: 117.8 1: 6636   
## Median : 1500.0 Median : 1500.0   
## Mean : 4799.4 Mean : 5215.5   
## 3rd Qu.: 4031.5 3rd Qu.: 4000.0   
## Max. :426529.0 Max. :528666.0   
##

count <-table(credit2$default.payment.next.month)#0=did not default, 1= defaulted  
prop.table(count) #percentage of each class type

##   
## 0 1   
## 0.7788 0.2212

bp <- barplot(count, main = "Credit Defaulters Distribution",  
xlab = "Customers", names = c("Did not default", "Default"),  
ylab = "Number of customers")   
text(bp, 0, round(count, 1),cex=1,pos=3)

 #preparing the data:

#3 create a 75% training sample and use the rest for testing  
# Stratified sampling. Select rows to be based on Class variable as strata  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

TrainingDataIndex <- createDataPartition(credit2$default.payment.next.month, p=0.75, list = FALSE)  
  
# Create Training Data as subset of credit dataset with row index numbers as identified above and all columns  
trainingData <- credit2[TrainingDataIndex,]

prop.table(table(trainingData$default.payment.next.month))

##   
## 0 1   
## 0.7788 0.2212

# Distribution of training data is the same as that of the original dataset

#create test data  
testData <- credit2[-TrainingDataIndex,]

prop.table(table(testData$default.payment.next.month))

##   
## 0 1   
## 0.7788 0.2212

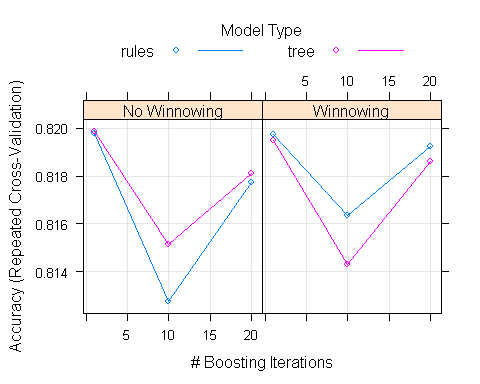
# Distribution of test data is the same as that of the original dataset

#Basic C5.0 Decision Tree Model  
  
# 5 fold cross validation is used to train and evaluate model  
  
TrainingParameters <- trainControl(method = "repeatedcv", number = 5, repeats = 1)   
  
# Train the model  
DecTreeC5 <- train(default.payment.next.month ~ ., data = trainingData,   
  
 method = "C5.0",  
 trControl= TrainingParameters,  
 na.action = na.omit  
 )

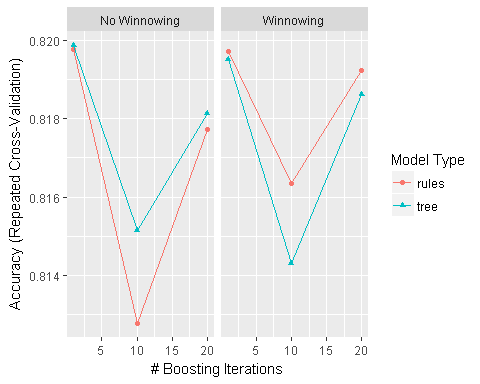
#View the results  
library(ggplot2)  
DecTreeC5

## C5.0   
##   
## 22500 samples  
## 23 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold, repeated 1 times)   
## Summary of sample sizes: 17999, 18000, 18001, 18000, 18000   
## Resampling results across tuning parameters:  
##   
## model winnow trials Accuracy Kappa   
## rules FALSE 1 0.8197777 0.3554920  
## rules FALSE 10 0.8127554 0.3330072  
## rules FALSE 20 0.8177332 0.3514024  
## rules TRUE 1 0.8197333 0.3522322  
## rules TRUE 10 0.8163555 0.3483743  
## rules TRUE 20 0.8192445 0.3584911  
## tree FALSE 1 0.8198666 0.3615090  
## tree FALSE 10 0.8151554 0.3477928  
## tree FALSE 20 0.8181332 0.3583382  
## tree TRUE 1 0.8195111 0.3519587  
## tree TRUE 10 0.8143109 0.3259600  
## tree TRUE 20 0.8186225 0.3540363  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were trials = 1, model = tree  
## and winnow = FALSE.

#?C5.0 #model- whether to use decision tree or rules, winnow- pruning, trials- boosting is used too. Tree selected- the one with the largest accuracy  
# Plot performance  
plot.train(DecTreeC5)



ggplot(DecTreeC5)



# Making predictions on test set  
DTPredictionsC5 <-predict(DecTreeC5, testData, na.action = na.pass)

# Print confusion matrix and results  
cm <-confusionMatrix(DTPredictionsC5, testData$default.payment.next.month)  
cm$overall

## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 8.188000e-01 3.608522e-01 8.098926e-01 8.274589e-01 7.788000e-01   
## AccuracyPValue McnemarPValue   
## 7.521114e-18 1.302305e-109

cm

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 5572 1090  
## 1 269 569  
##   
## Accuracy : 0.8188   
## 95% CI : (0.8099, 0.8275)  
## No Information Rate : 0.7788   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3609   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9539   
## Specificity : 0.3430   
## Pos Pred Value : 0.8364   
## Neg Pred Value : 0.6790   
## Prevalence : 0.7788   
## Detection Rate : 0.7429   
## Detection Prevalence : 0.8883   
## Balanced Accuracy : 0.6485   
##   
## 'Positive' Class : 0   
##

#Try to improve model performance (z-score standardization)   
trainingData\_standardized<-trainingData  
trainingData\_standardized[,c(1,5,12:22)] <-scale(trainingData\_standardized[,c(1,5,12:22)])  
summary(trainingData\_standardized) #check whether it is standardized

## LIMIT\_BAL SEX EDUCATION MARRIAGE AGE   
## Min. :-1.2066 1: 8907 0: 11 0: 40 Min. :-1.5684   
## 1st Qu.:-0.8997 2:13593 1: 7933 1:10218 1st Qu.:-0.8099   
## Median :-0.2091 2:10544 2:11993 Median :-0.1597   
## Mean : 0.0000 3: 3676 3: 249 Mean : 0.0000   
## 3rd Qu.: 0.5582 4: 90 3rd Qu.: 0.5988   
## Max. : 4.8550 5: 209 Max. : 4.7163   
## 6: 37   
## PAY\_0 PAY\_2 PAY\_3 PAY\_4   
## 0 :11136 0 :11865 0 :11849 0 :12371   
## -1 : 4202 -1 : 4530 -1 : 4451 -1 : 4251   
## 1 : 2770 2 : 2898 -2 : 3054 -2 : 3237   
## -2 : 2054 -2 : 2828 2 : 2856 2 : 2370   
## 2 : 1996 3 : 248 3 : 171 3 : 132   
## 3 : 233 4 : 67 4 : 59 4 : 58   
## (Other): 109 (Other): 64 (Other): 60 (Other): 81   
## PAY\_5 PAY\_6 BILL\_AMT1 BILL\_AMT2   
## 0 :12734 0 :12237 Min. :-2.9472 Min. :-1.1586   
## -1 : 4123 -1 : 4270 1st Qu.:-0.6477 1st Qu.:-0.6486   
## -2 : 3396 -2 : 3672 Median :-0.3916 Median :-0.3922   
## 2 : 1976 2 : 2084 Mean : 0.0000 Mean : 0.0000   
## 3 : 141 3 : 133 3rd Qu.: 0.2223 3rd Qu.: 0.2091   
## 4 : 65 4 : 39 Max. : 9.4596 Max. : 9.7472   
## (Other): 65 (Other): 65   
## BILL\_AMT3 BILL\_AMT4 BILL\_AMT5 BILL\_AMT6   
## Min. :-2.9283 Min. :-3.3058 Min. :-1.9982 Min. :-6.3539   
## 1st Qu.:-0.6366 1st Qu.:-0.6350 1st Qu.:-0.6341 1st Qu.:-0.6307   
## Median :-0.3878 Median :-0.3772 Median :-0.3667 Median :-0.3671   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.1882 3rd Qu.: 0.1763 3rd Qu.: 0.1637 3rd Qu.: 0.1742   
## Max. :23.1634 Max. :10.2789 Max. :12.8515 Max. :11.1000   
##   
## PAY\_AMT1 PAY\_AMT2 PAY\_AMT3   
## Min. :-0.35053 Min. :-0.26594 Min. :-0.31958   
## 1st Qu.:-0.28920 1st Qu.:-0.22742 1st Qu.:-0.29511   
## Median :-0.22174 Median :-0.17482 Median :-0.20834   
## Mean : 0.00000 Mean : 0.00000 Mean : 0.00000   
## 3rd Qu.:-0.04313 3rd Qu.:-0.03938 3rd Qu.:-0.04025   
## Max. :30.62247 Max. :76.04936 Max. :25.48620   
##   
## PAY\_AMT4 PAY\_AMT5 PAY\_AMT6   
## Min. :-0.31567 Min. :-0.31898 Min. : 0   
## 1st Qu.:-0.29582 1st Qu.:-0.30153 1st Qu.: 122   
## Median :-0.21640 Median :-0.21753 Median : 1500   
## Mean : 0.00000 Mean : 0.00000 Mean : 5185   
## 3rd Qu.:-0.04995 3rd Qu.:-0.04809 3rd Qu.: 4000   
## Max. :34.68831 Max. :28.52885 Max. :528666   
##   
## default.payment.next.month  
## 0:17523   
## 1: 4977   
##   
##   
##   
##   
##

testData\_standardized<-testData  
testData\_standardized[,c(1,5,12:22)] <-scale(testData\_standardized[,c(1,5,12:22)])

#6. Create a C5.0 decision tree model using standardized data  
  
DecTreeModel\_s <- train(default.payment.next.month ~ ., data = trainingData\_standardized,  
 method = "C5.0",  
 trControl= TrainingParameters,  
 preProcess=c("center", "scale"),  
 na.action = na.omit  
 )

## Warning in preProcess.default(method = c("center", "scale"), x =  
## structure(c(-1.12987114520506, : These variables have zero variances:  
## PAY\_28

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28, PAY\_58  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28, PAY\_58  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28, PAY\_58  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28, PAY\_58

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28, PAY\_41  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28, PAY\_41  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28, PAY\_41  
  
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28, PAY\_41

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut  
## = 10, : These variables have zero variances: PAY\_28

DTPredictions\_s <-predict(DecTreeModel\_s, testData\_standardized, na.action = na.pass)

cm\_s <-confusionMatrix(DTPredictions\_s, testData\_standardized$default.payment.next.month)  
cm\_s$overall

## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 8.158667e-01 3.387130e-01 8.069059e-01 8.245812e-01 7.788000e-01   
## AccuracyPValue McnemarPValue   
## 1.564932e-15 3.411087e-126

cm\_s

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 5595 1135  
## 1 246 524  
##   
## Accuracy : 0.8159   
## 95% CI : (0.8069, 0.8246)  
## No Information Rate : 0.7788   
## P-Value [Acc > NIR] : 1.565e-15   
##   
## Kappa : 0.3387   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9579   
## Specificity : 0.3159   
## Pos Pred Value : 0.8314   
## Neg Pred Value : 0.6805   
## Prevalence : 0.7788   
## Detection Rate : 0.7460   
## Detection Prevalence : 0.8973   
## Balanced Accuracy : 0.6369   
##   
## 'Positive' Class : 0   
##

# we observe no improvement in accuracy value due to standardization

# train model with neural network  
#TrainingParameters3 <- trainControl(method = "repeatedcv", number = 5, sampling='up')  
NoTrainingParameters <- trainControl(method = "none")  
NNModel <- train(default.payment.next.month ~ ., data = trainingData\_standardized,  
 method = "nnet", #models available- caret documentiation section6  
 trControl= NoTrainingParameters,  
 tuneGrid = data.frame(size = 5, #not autotune. specify list of values to consider- in nnet documentation  
 decay = 0  
 )  
   
)

## # weights: 421  
## initial value 18125.177994   
## iter 10 value 11192.479307  
## iter 20 value 10495.446324  
## iter 30 value 10145.578211  
## iter 40 value 9792.040539  
## iter 50 value 9651.925874  
## iter 60 value 9579.858748  
## iter 70 value 9531.007685  
## iter 80 value 9485.888163  
## iter 90 value 9440.518797  
## iter 100 value 9409.700671  
## final value 9409.700671   
## stopped after 100 iterations

NNPredictions <-predict(NNModel, testData\_standardized)  
# Create confusion matrix  
cmNN <-confusionMatrix(NNPredictions, testData\_standardized$default.payment.next.month)  
cmNN$overall

## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 8.180000e-01 3.607127e-01 8.090780e-01 8.266741e-01 7.788000e-01   
## AccuracyPValue McnemarPValue   
## 3.364524e-17 5.370821e-105

#NaiveBayes model  
library(klaR)

## Loading required package: MASS

TrainingParameters <- trainControl(method = "repeatedcv", number = 5, repeats = 1)   
NBModel <- train(default.payment.next.month ~ ., data = trainingData\_standardized,   
 method = "nb",  
 trControl= TrainingParameters,  
 #tuneGrid = data.frame(fL=1, usekernel=FALSE),  
 na.action = na.omit  
 )

## Warning: model fit failed for Fold1.Rep1: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :   
## Zero variances for at least one class in variables: PAY\_28, PAY\_38, PAY\_41, PAY\_48, PAY\_58, PAY\_68

## Warning: model fit failed for Fold2.Rep1: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :   
## Zero variances for at least one class in variables: PAY\_28, PAY\_31, PAY\_41, PAY\_58, PAY\_68

## Warning: model fit failed for Fold3.Rep1: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :   
## Zero variances for at least one class in variables: PAY\_28, PAY\_41, PAY\_58, PAY\_68

## Warning: model fit failed for Fold4.Rep1: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :   
## Zero variances for at least one class in variables: PAY\_28, PAY\_31, PAY\_41, PAY\_56, PAY\_58, PAY\_68

## Warning: model fit failed for Fold5.Rep1: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :   
## Zero variances for at least one class in variables: PAY\_28, PAY\_38, PAY\_41, PAY\_48, PAY\_56, PAY\_58, PAY\_68

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =  
## trainInfo, : There were missing values in resampled performance measures.

## Warning in train.default(x, y, weights = w, ...): missing values found in  
## aggregated results

NBPredictions <- predict(NBModel,testData\_standardized)

# Create confusion matrix  
cmNB <-confusionMatrix(NBPredictions, testData\_standardized$default.payment.next.month)  
cmNB$overall

## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 0.7788000 0.0000000 0.7692325 0.7881503 0.7788000   
## AccuracyPValue McnemarPValue   
## 0.5065803 0.0000000

cmNB$byClass #precision, recall

## Sensitivity Specificity Pos Pred Value   
## 1.0000000 0.0000000 0.7788000   
## Neg Pred Value Precision Recall   
## NaN 0.7788000 1.0000000   
## F1 Prevalence Detection Rate   
## 0.8756465 0.7788000 0.7788000   
## Detection Prevalence Balanced Accuracy   
## 1.0000000 0.5000000

cmNB

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 5841 1659  
## 1 0 0  
##   
## Accuracy : 0.7788   
## 95% CI : (0.7692, 0.7882)  
## No Information Rate : 0.7788   
## P-Value [Acc > NIR] : 0.5066   
##   
## Kappa : 0   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.7788   
## Neg Pred Value : NaN   
## Prevalence : 0.7788   
## Detection Rate : 0.7788   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : 0   
##

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.