BA 64060 Assignment 2

Kyle Breeden

2025-09-28

## R Markdown

#rm(list = ls())  
#Use readr package for faster and more efficient reading  
#install.packages("readr") #not needed if already installed  
library(readr)

## Warning: package 'readr' was built under R version 4.5.1

assign2\_dataset <- read\_csv("C:/Users/kylej/OneDrive/Documents/BA 64060/Assignment 2/Data/UniversalBank.CSV")

## Rows: 5000 Columns: 14  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## dbl (14): ID, Age, Experience, Income, ZIP Code, Family, CCAvg, Education, M...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

#Use Spec function to see column specifications  
spec(assign2\_dataset)

## cols(  
## ID = col\_double(),  
## Age = col\_double(),  
## Experience = col\_double(),  
## Income = col\_double(),  
## `ZIP Code` = col\_double(),  
## Family = col\_double(),  
## CCAvg = col\_double(),  
## Education = col\_double(),  
## Mortgage = col\_double(),  
## `Personal Loan` = col\_double(),  
## `Securities Account` = col\_double(),  
## `CD Account` = col\_double(),  
## Online = col\_double(),  
## CreditCard = col\_double()  
## )

#partition data into training (60%) and validation (40%) sets  
#use code from BA 6460 github BUT note the example code is for training, validation, & test sets  
#I must portion into training & validation ONLY to start, I must change example code  
#install.packages("caret")  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

#assigning the personal loan and education variables as factors was crucial to getting the transformation in the next chunk to work  
assign2\_dataset$`Personal Loan` <-factor(assign2\_dataset$`Personal Loan`, levels = c(0,1))   
assign2\_dataset$Education <- factor(assign2\_dataset$Education, levels = c(1,2,3))  
set.seed(15) #can be any integer but must stay consistent so I can reproduce results  
Train\_Index = createDataPartition(assign2\_dataset$`Personal Loan`,p = 0.6, list=FALSE) # 60% reserved for Train  
Train\_Data = assign2\_dataset[Train\_Index,]  
Validation\_Data = assign2\_dataset[-Train\_Index,] #rest as validation

#for number 1, I need to transform education into dummy variables  
#dummy variables are provided in the assignment: Education\_1, Education\_2, & Education\_3  
#use dummyVars function of caret package; it will look for categorical variables with more than 2 categories & prepare to split  
#library(caret) #already called caret library, no need to repeat  
dummies <- dummyVars(`Personal Loan` ~ ., data = Train\_Data, fullRank = FALSE)  
Train\_predictors <- predict(dummies, newdata = Train\_Data)

## Warning in model.frame.default(Terms, newdata, na.action = na.action, xlev =  
## object$lvls): variable 'Personal Loan' is not a factor

Valid\_predictors <- predict(dummies, newdata = Validation\_Data)

## Warning in model.frame.default(Terms, newdata, na.action = na.action, xlev =  
## object$lvls): variable 'Personal Loan' is not a factor

#change new education variable headings  
colnames(Train\_predictors) <- sub("^Education[ .\_]([123])$", "Education\_\\1",colnames(Train\_predictors))  
colnames(Valid\_predictors) <- sub("^Education[ .\_]([123])$", "Education\_\\1",colnames(Valid\_predictors))  
#convert matrices to data.frames  
Train\_transformed <- as.data.frame(Train\_predictors)  
Valid\_transformed <- as.data.frame(Valid\_predictors)  
#drop ID and zip code using dplyr  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

#next 2 lines would not work despise matching col name, "error in select(): Can't select columns that don't exist"  
#Subset\_Train <- Train\_transformed %>% select(-ID, -`ZIP Code`)  
#subset\_Valid <- Valid\_transformed %>% select(-ID, -`ZIP Code`)  
Subset\_Train <- Train\_transformed %>% select(-ID, -contains("ZIP Code"))  
Subset\_Valid <- Valid\_transformed %>% select(-ID, -contains("ZIP Code"))  
#use preProcess() from the caret package to normalize data  
#normalize only continuous variables  
#use/adapt github example  
cols\_to\_scale <- c("Age", "Experience", "Income", "Family", "CCAvg", "Mortgage")  
norm.values <- preProcess(Subset\_Train[, cols\_to\_scale, drop = FALSE], method=c("center", "scale"))  
#make copies then apply norm to both subsets  
Subset\_Train\_scaled <- Subset\_Train  
Subset\_Valid\_scaled <- Subset\_Valid  
Subset\_Train\_scaled[, cols\_to\_scale] <- predict(norm.values, Subset\_Train[, cols\_to\_scale, drop = FALSE])  
Subset\_Valid\_scaled[, cols\_to\_scale] <- predict(norm.values, Subset\_Valid[, cols\_to\_scale, drop = FALSE])  
#getting the variance for this exercise is more difficult (for me) than what is shown in the github example  
summary(Subset\_Train\_scaled)

## Age Experience Income Family   
## Min. :-1.9325 Min. :-1.997167 Min. :-1.4435 Min. :-1.2237   
## 1st Qu.:-0.8857 1st Qu.:-0.864443 1st Qu.:-0.7619 1st Qu.:-1.2237   
## Median :-0.0134 Median : 0.006883 Median :-0.2341 Median :-0.3482   
## Mean : 0.0000 Mean : 0.000000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.8589 3rd Qu.: 0.878210 3rd Qu.: 0.5355 3rd Qu.: 0.5273   
## Max. : 1.9057 Max. : 2.010934 Max. : 3.3061 Max. : 1.4028   
## CCAvg Education\_1 Education\_2 Education\_3   
## Min. :-1.1014 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:-0.7024 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :-0.2465 Median :0.0000 Median :0.0000 Median :0.0000   
## Mean : 0.0000 Mean :0.4163 Mean :0.2873 Mean :0.2963   
## 3rd Qu.: 0.3234 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. : 4.5978 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Mortgage `Securities Account` `CD Account` Online   
## Min. :-0.5591 Min. :0.000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:-0.5591 1st Qu.:0.000 1st Qu.:0.00000 1st Qu.:0.0000   
## Median :-0.5591 Median :0.000 Median :0.00000 Median :1.0000   
## Mean : 0.0000 Mean :0.103 Mean :0.05467 Mean :0.5873   
## 3rd Qu.: 0.4322 3rd Qu.:0.000 3rd Qu.:0.00000 3rd Qu.:1.0000   
## Max. : 5.6581 Max. :1.000 Max. :1.00000 Max. :1.0000   
## CreditCard   
## Min. :0.0000   
## 1st Qu.:0.0000   
## Median :0.0000   
## Mean :0.2907   
## 3rd Qu.:1.0000   
## Max. :1.0000

train\_vars <- Subset\_Train\_scaled %>% summarise(across(all\_of(cols\_to\_scale), ~ var(.x, na.rm = TRUE)))  
summary(Subset\_Valid\_scaled)

## Age Experience Income Family   
## Min. :-1.93249 Min. :-1.997167 Min. :-1.443519 Min. :-1.223658   
## 1st Qu.:-0.79849 1st Qu.:-0.777310 1st Qu.:-0.783849 1st Qu.:-1.223658   
## Median : 0.07383 Median : 0.006883 Median :-0.212135 Median :-0.348157   
## Mean : 0.04029 Mean : 0.039994 Mean : 0.006974 Mean :-0.002772   
## 3rd Qu.: 0.94614 3rd Qu.: 0.878210 3rd Qu.: 0.557480 3rd Qu.: 1.402844   
## Max. : 1.90568 Max. : 2.010934 Max. : 2.888313 Max. : 1.402844   
## CCAvg Education\_1 Education\_2 Education\_3   
## Min. :-1.101360 Min. :0.0000 Min. :0.0000 Min. :0.000   
## 1st Qu.:-0.702417 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.000   
## Median :-0.189490 Median :0.0000 Median :0.0000 Median :0.000   
## Mean : 0.007767 Mean :0.4235 Mean :0.2705 Mean :0.306   
## 3rd Qu.: 0.380428 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.000   
## Max. : 4.597824 Max. :1.0000 Max. :1.0000 Max. :1.000   
## Mortgage `Securities Account` `CD Account` Online   
## Min. :-0.55909 Min. :0.0000 Min. :0.000 Min. :0.000   
## 1st Qu.:-0.55909 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:0.000   
## Median :-0.55909 Median :0.0000 Median :0.000 Median :1.000   
## Mean :-0.01481 Mean :0.1065 Mean :0.069 Mean :0.611   
## 3rd Qu.: 0.41999 3rd Qu.:0.0000 3rd Qu.:0.000 3rd Qu.:1.000   
## Max. : 5.48183 Max. :1.0000 Max. :1.000 Max. :1.000   
## CreditCard   
## Min. :0.000   
## 1st Qu.:0.000   
## Median :0.000   
## Mean :0.299   
## 3rd Qu.:1.000   
## Max. :1.000

valid\_vars <- Subset\_Valid\_scaled %>% summarise(across(all\_of(cols\_to\_scale), ~ var(.x, na.rm = TRUE)))  
#add outcome variable back in  
#this was avoided in github example, although I am unsure how   
#I believe the code below will work because the rows remained in the same order when transformed and scaled  
Subset\_Train\_scaled$`Personal Loan` <- Train\_Data$`Personal Loan`  
Subset\_Valid\_scaled$`Personal Loan` <- Validation\_Data$`Personal Loan`

#Apply knn using FNN package  
# # install if needed  
#use kNN Implementation in R slides as a guide  
library(FNN)

## Warning: package 'FNN' was built under R version 4.5.1

Train\_Predictors <- Subset\_Train\_scaled[,1:13]  
Valid\_Predictors <- Subset\_Valid\_scaled [,1:13]  
  
Train\_labels <- factor(Subset\_Train\_scaled[,14], levels = c(0,1))  
Valid\_labels <- factor(Subset\_Valid\_scaled[,14], levels = c(0,1))  
  
#train a knn model where k=1  
Predicted\_Valid\_labels <- knn(train = Train\_Predictors, test = Valid\_Predictors, cl = Train\_labels, k = 1)  
head(Predicted\_Valid\_labels)

## [1] 0 0 0 0 1 0  
## Levels: 0 1

#create data frame with fields dictated bu the assignment directions  
Random\_customer\_df <- data.frame(  
 Age = 40,  
 Experience = 10,  
 Income = 84,  
 Family = 2,  
 CCAvg = 2,  
 Education\_1 = 0,  
 Education\_2 = 1,  
 Education\_3 = 0,  
 Mortgage = 0,  
 `Securities Account` = 0,  
 `CD Account` = 0,  
 Online = 1,  
 CreditCard = 1,  
 `Personal Loan` = NA)  
#scale new row  
cols\_to\_scale <- c("Age", "Experience", "Income", "Family", "CCAvg", "Mortgage")  
Random\_customer\_df[, cols\_to\_scale] <- predict(norm.values, Random\_customer\_df[, cols\_to\_scale, drop = FALSE])

#Predict classification of customer  
library(FNN)  
one\_pred\_fnn <- knn(train = as.matrix(Train\_Predictors),  
 test = as.matrix(Random\_customer\_df[,1:13]),  
 cl = Train\_labels,  
 k = 1)  
one\_pred\_fnn

## [1] 0  
## attr(,"nn.index")  
## [,1]  
## [1,] 2666  
## attr(,"nn.dist")  
## [,1]  
## [1,] 0.4977541  
## Levels: 0

#Based on the above k-NN prediction, the customer from question 1 would NOT accept the personal loan.

#What is a choice of k that balances between overfitting and ignoring the predictor information?  
#To determine k, we use the performance on the validation set  
#use/modify code from github  
library(caret)  
accuracy.df <- data.frame(k = seq(1, 14, 1), accuracy = rep(0, 14))  
# compute knn for different k on validation.  
for(i in 1:14) {  
 knn.pred <- knn(train = Train\_Predictors,test = Valid\_Predictors, cl = Train\_labels, k = i)  
 cm <- confusionMatrix(knn.pred, Valid\_labels)  
 accuracy.df[i, "accuracy"] <- cm$overall["Accuracy"]  
}   
accuracy.df

## k accuracy  
## 1 1 0.9540  
## 2 2 0.9540  
## 3 3 0.9645  
## 4 4 0.9580  
## 5 5 0.9620  
## 6 6 0.9595  
## 7 7 0.9610  
## 8 8 0.9545  
## 9 9 0.9580  
## 10 10 0.9550  
## 11 11 0.9570  
## 12 12 0.9520  
## 13 13 0.9535  
## 14 14 0.9480

#Based on the accuracy data frame above, 3 is the choice of k that balances between overfitting and ignoring the predictor information.

#Show the confusion matrix for the validation data that results from using the best k.  
Pred\_best\_k <- knn(train = as.matrix(Train\_Predictors),  
 test = as.matrix(Valid\_Predictors),  
 cl = Train\_labels,  
 k = 3)  
library(caret)  
cm\_best\_k<- caret::confusionMatrix(data = Pred\_best\_k,  
 reference = Valid\_labels,  
 positive = "1")  
cm\_best\_k

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1804 67  
## 1 4 125  
##   
## Accuracy : 0.9645   
## 95% CI : (0.9554, 0.9722)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7603   
##   
## Mcnemar's Test P-Value : 1.866e-13   
##   
## Sensitivity : 0.6510   
## Specificity : 0.9978   
## Pos Pred Value : 0.9690   
## Neg Pred Value : 0.9642   
## Prevalence : 0.0960   
## Detection Rate : 0.0625   
## Detection Prevalence : 0.0645   
## Balanced Accuracy : 0.8244   
##   
## 'Positive' Class : 1   
##

length(Pred\_best\_k)

## [1] 2000

length(Valid\_labels)

## [1] 2000

is.factor(Pred\_best\_k); is.factor(Valid\_labels)

## [1] TRUE

## [1] TRUE

#another way  
#install.packages("gmodels") # install if necessary  
library("gmodels")

## Warning: package 'gmodels' was built under R version 4.5.1

CrossTable(x=Valid\_labels,y=Pred\_best\_k, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 2000   
##   
##   
## | Pred\_best\_k   
## Valid\_labels | 0 | 1 | Row Total |   
## -------------|-----------|-----------|-----------|  
## 0 | 1804 | 4 | 1808 |   
## | 0.998 | 0.002 | 0.904 |   
## | 0.964 | 0.031 | |   
## | 0.902 | 0.002 | |   
## -------------|-----------|-----------|-----------|  
## 1 | 67 | 125 | 192 |   
## | 0.349 | 0.651 | 0.096 |   
## | 0.036 | 0.969 | |   
## | 0.034 | 0.062 | |   
## -------------|-----------|-----------|-----------|  
## Column Total | 1871 | 129 | 2000 |   
## | 0.935 | 0.064 | |   
## -------------|-----------|-----------|-----------|  
##   
##

#Customer in question 4 has same caharteristics as question 1  
#library(FNN)  
two\_pred\_fnn <- knn(train = as.matrix(Train\_Predictors),  
 test = as.matrix(Random\_customer\_df[,1:13]),  
 cl = Train\_labels,  
 k = 3)  
two\_pred\_fnn

## [1] 0  
## attr(,"nn.index")  
## [,1] [,2] [,3]  
## [1,] 2666 2082 1004  
## attr(,"nn.dist")  
## [,1] [,2] [,3]  
## [1,] 0.4977541 0.6383293 0.7080997  
## Levels: 0

#Based on the above k-NN prediction, the customer from question 4 would NOT accept the personal loan.

#Repartition into training (50%), validation (30%), and test (20%) sets  
set.seed(15)  
Test\_Index\_two = createDataPartition(assign2\_dataset$`Personal Loan`,p = 0.2, list=FALSE) #20% reserved for test  
Test\_Data\_two = assign2\_dataset[Test\_Index\_two,]  
TraVal\_Data\_two = assign2\_dataset[-Test\_Index\_two,] #Validation and Training is the rest  
  
Train\_Index\_two = createDataPartition(TraVal\_Data\_two$`Personal Loan`,p = 0.5, list=FALSE) #50% reserved for training  
Train\_Data\_two = TraVal\_Data\_two[Train\_Index\_two,]  
Validation\_Data\_two = TraVal\_Data\_two[-Train\_Index\_two,] #rest as validation

#normalize data  
# Copy the original data per example in github  
train.norm.df <- Train\_Data\_two  
valid.norm.df <- Validation\_Data\_two  
traval.norm.df <- TraVal\_Data\_two   
test.norm.df <- Test\_Data\_two   
  
cols\_to\_scale\_two <- c("Age", "Experience", "Income", "Family", "CCAvg", "Mortgage", "Securities Account", "CD Account", "Online", "CreditCard")  
  
  
#use preProcess() from the caret package to normalize .  
norm.values\_two <- preProcess(Train\_Data\_two[, cols\_to\_scale\_two, drop = FALSE],   
 method = c("center", "scale"))  
  
train.norm.df[, cols\_to\_scale\_two] <- predict(norm.values\_two, Train\_Data\_two[, cols\_to\_scale\_two, drop = FALSE])  
valid.norm.df[, cols\_to\_scale\_two] <- predict(norm.values\_two, Validation\_Data\_two[, cols\_to\_scale\_two, drop = FALSE])  
traval.norm.df[, cols\_to\_scale\_two] <- predict(norm.values\_two, TraVal\_Data\_two[, cols\_to\_scale\_two, drop = FALSE])  
test.norm.df[, cols\_to\_scale\_two] <- predict(norm.values\_two, Test\_Data\_two[, cols\_to\_scale\_two, drop = FALSE])

#replace Education with dummy variables  
#I thought this might be easier than the way I did it last time; it wasn't  
#by far the most time consuming thing I had to do on this assignment; has to be a better way  
library(caret)  
dummy\_model <- dummyVars(~ Education, data = test.norm.df, fullRank = FALSE)   
edu\_dummies <- predict(dummy\_model, newdata = test.norm.df)  
colnames(edu\_dummies) <- sub("^Education[ .\_]([123])$", "Education\_\\1", colnames(edu\_dummies))  
test.norm.df <- cbind(test.norm.df, edu\_dummies)  
test.norm.df$Education <- NULL  
  
#repeat for train and valid  
train.norm.df$Education <- factor(as.character(train.norm.df$Education), levels = c("1","2","3"))  
valid.norm.df$Education <- factor(as.character(valid.norm.df$Education), levels = c("1","2","3"))  
  
  
edu\_model <- dummyVars(~ Education,  
 data = train.norm.df[, "Education", drop = FALSE],  
 fullRank = FALSE)  
  
  
edu\_train <- predict(edu\_model, newdata = train.norm.df[, "Education", drop = FALSE])  
edu\_valid <- predict(edu\_model, newdata = valid.norm.df[, "Education", drop = FALSE])  
  
  
fix\_names <- function(m) {  
 colnames(m) <- sub("^Education[ .\_]([123])$", "Education\_\\1", colnames(m))  
 m  
}  
edu\_train <- fix\_names(edu\_train)  
edu\_valid <- fix\_names(edu\_valid)  
  
  
train.norm.df <- cbind(train.norm.df, edu\_train); train.norm.df$Education <- NULL  
valid.norm.df <- cbind(valid.norm.df, edu\_valid); valid.norm.df$Education <- NULL

#remove ID & Zip Code  
Subset\_Train\_two <- train.norm.df %>% select(-ID, -contains("ZIP Code"))  
Subset\_Valid\_two <- valid.norm.df %>% select(-ID, -contains("ZIP Code"))  
Subset\_Test\_two <- test.norm.df %>% select(-ID, -contains("ZIP Code"))

#re-order data  
Train\_ready <- Subset\_Train\_two %>% relocate(starts\_with("Personal"), .after = "Education\_3")  
Valid\_ready <- Subset\_Valid\_two %>% relocate(starts\_with("Personal"), .after = "Education\_3")  
Test\_ready <- Subset\_Test\_two %>% relocate(starts\_with("Personal"), .after = "Education\_3")  
  
#Apply knn using FNN package  
# # install if needed  
#use kNN Implementation in R slides as a guide  
library(FNN)  
Train\_Predictors\_two <- Train\_ready[,1:13]  
Valid\_Predictors\_two <- Valid\_ready [,1:13]  
Test\_Predictors\_two <- Test\_ready [,1:13]  
  
Train\_labels <- factor(Train\_ready[,14], levels = c(0,1))  
Valid\_labels <- factor(Valid\_ready[,14], levels = c(0,1))  
Test\_labels <- factor(Test\_ready[,14], levels = c(0,1))  
  
#train a knn model where k=3  
pred\_train <- knn(train = as.matrix(Train\_Predictors\_two),  
 test = as.matrix(Train\_Predictors\_two),  
 cl = Train\_labels,  
 k = 3)  
  
pred\_valid <- knn(train = as.matrix(Train\_Predictors\_two),  
 test = as.matrix(Valid\_Predictors\_two),  
 cl = Train\_labels,  
 k = 3)  
  
pred\_test <- knn(train = as.matrix(Train\_Predictors\_two),  
 test = as.matrix(Test\_Predictors\_two),  
 cl = Train\_labels,  
 k = 3)

#confusion matrices of test, valid, and train sets  
#Training  
library(gmodels)  
CrossTable(x = Train\_labels, y = pred\_train,  
 prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 2000   
##   
##   
## | pred\_train   
## Train\_labels | 0 | 1 | Row Total |   
## -------------|-----------|-----------|-----------|  
## 0 | 1804 | 4 | 1808 |   
## | 0.998 | 0.002 | 0.904 |   
## | 0.971 | 0.028 | |   
## | 0.902 | 0.002 | |   
## -------------|-----------|-----------|-----------|  
## 1 | 53 | 139 | 192 |   
## | 0.276 | 0.724 | 0.096 |   
## | 0.029 | 0.972 | |   
## | 0.026 | 0.070 | |   
## -------------|-----------|-----------|-----------|  
## Column Total | 1857 | 143 | 2000 |   
## | 0.928 | 0.071 | |   
## -------------|-----------|-----------|-----------|  
##   
##

#Test  
CrossTable(x = Test\_labels, y = pred\_test,  
 prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 1000   
##   
##   
## | pred\_test   
## Test\_labels | 0 | 1 | Row Total |   
## -------------|-----------|-----------|-----------|  
## 0 | 896 | 8 | 904 |   
## | 0.991 | 0.009 | 0.904 |   
## | 0.952 | 0.136 | |   
## | 0.896 | 0.008 | |   
## -------------|-----------|-----------|-----------|  
## 1 | 45 | 51 | 96 |   
## | 0.469 | 0.531 | 0.096 |   
## | 0.048 | 0.864 | |   
## | 0.045 | 0.051 | |   
## -------------|-----------|-----------|-----------|  
## Column Total | 941 | 59 | 1000 |   
## | 0.941 | 0.059 | |   
## -------------|-----------|-----------|-----------|  
##   
##

#Validation  
CrossTable(x = Valid\_labels, y = pred\_valid,  
 prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 2000   
##   
##   
## | pred\_valid   
## Valid\_labels | 0 | 1 | Row Total |   
## -------------|-----------|-----------|-----------|  
## 0 | 1791 | 17 | 1808 |   
## | 0.991 | 0.009 | 0.904 |   
## | 0.956 | 0.135 | |   
## | 0.895 | 0.008 | |   
## -------------|-----------|-----------|-----------|  
## 1 | 83 | 109 | 192 |   
## | 0.432 | 0.568 | 0.096 |   
## | 0.044 | 0.865 | |   
## | 0.042 | 0.054 | |   
## -------------|-----------|-----------|-----------|  
## Column Total | 1874 | 126 | 2000 |   
## | 0.937 | 0.063 | |   
## -------------|-----------|-----------|-----------|  
##   
##

#The differences are that the Training set has more correct predictions (1804 of 1808 for “0” and 139 of 192 for “1”). The Validation and Test sets also perform well when predicting “0”, but do not perform as well when predicting “1”. There are more false negatives in the Training set and more false positives in the validation set. There were much less “1” cases to learn from in the training set, so the accuracy of the “1” predictions decreased in the test and validation sets when new data was introduced. This is a sign that the model overfitted the training set.