KNN\_Lab\_Clancy

Joey Clancy

9/8/2020

# Setup and Preprocessing

# Import libraries

library(caTools)  
library(caret)  
library(FNN)  
library(class)

# Import data and remove unwanted columns

df = read.csv("UniversalBank.csv")  
df <- df[ , -c(1, 5)]  
  
# Make Personal Loan Variable Neat  
df$Personal.Loan <- factor(df$Personal.Loan,  
 levels = c("0", "1"),  
 labels = c("No", "Yes"))  
names(df)

## [1] "Age" "Experience" "Income"   
## [4] "Family" "CCAvg" "Education"   
## [7] "Mortgage" "Personal.Loan" "Securities.Account"  
## [10] "CD.Account" "Online" "CreditCard"

# Checking data type for each variable

str(df)

## 'data.frame': 5000 obs. of 12 variables:  
## $ Age : int 25 45 39 35 35 37 53 50 35 34 ...  
## $ Experience : int 1 19 15 9 8 13 27 24 10 9 ...  
## $ Income : int 49 34 11 100 45 29 72 22 81 180 ...  
## $ Family : int 4 3 1 1 4 4 2 1 3 1 ...  
## $ CCAvg : num 1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...  
## $ Education : int 1 1 1 2 2 2 2 3 2 3 ...  
## $ Mortgage : int 0 0 0 0 0 155 0 0 104 0 ...  
## $ Personal.Loan : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 2 ...  
## $ Securities.Account: int 1 1 0 0 0 0 0 0 0 0 ...  
## $ CD.Account : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Online : int 0 0 0 0 0 1 1 0 1 0 ...  
## $ CreditCard : int 0 0 0 0 1 0 0 1 0 0 ...

# Data types are as expected.

# Train test split

set.seed(666)  
split <- sample.split(df$Personal.Loan, SplitRatio = 0.6)  
training\_set <- subset(df, split == TRUE)  
test\_set <- subset(df, split == FALSE)

# Feature scaling

norm\_values <- preProcess(training\_set[, -8],  
 method = c("center",  
 "scale"))  
  
training\_set[, -8] <- predict(norm\_values,  
 training\_set[, -8])  
head(training\_set)

## Age Experience Income Family CCAvg Education  
## 2 -0.01008703 -0.07829267 -0.8692464 0.5031342 -0.2564507 -1.0426967  
## 4 -0.88722006 -0.95699826 0.5660986 -1.2408319 0.4325984 0.1390525  
## 5 -0.88722006 -1.04486882 -0.6300222 1.3751173 -0.5435545 0.1390525  
## 8 0.42847948 0.36106013 -1.1302182 -1.2408319 -0.9454998 1.3208017  
## 9 -0.88722006 -0.86912770 0.1528932 0.5031342 -0.7732375 0.1390525  
## 10 -0.97493336 -0.95699826 2.3059107 -1.2408319 3.9926856 1.3208017  
## Mortgage Personal.Loan Securities.Account CD.Account Online CreditCard  
## 2 -0.5512914 No 2.9347083 -0.2518562 -1.1977423 -0.6420996  
## 4 -0.5512914 No -0.3406358 -0.2518562 -1.1977423 -0.6420996  
## 5 -0.5512914 No -0.3406358 -0.2518562 -1.1977423 1.5568716  
## 8 -0.5512914 No -0.3406358 -0.2518562 -1.1977423 1.5568716  
## 9 0.4737924 No -0.3406358 -0.2518562 0.8346258 -0.6420996  
## 10 -0.5512914 Yes -0.3406358 -0.2518562 -1.1977423 -0.6420996

# Normalize the test set

test\_set[, -8] <- predict(norm\_values,  
 test\_set[, -8])  
  
head(test\_set)

## Age Experience Income Family CCAvg Education  
## 1 -1.7643531 -1.6599627 -0.54303160 1.3751173 -0.1990299 -1.0426967  
## 3 -0.5363668 -0.4297749 -1.36944236 -1.2408319 -0.5435545 -1.0426967  
## 6 -0.7117935 -0.6055160 -0.97798463 1.3751173 -0.8880791 0.1390525  
## 7 0.6916194 0.6246718 -0.04283562 -0.3688488 -0.2564507 0.1390525  
## 12 -1.4134999 -1.3084805 -0.63002221 0.5031342 -1.0603413 0.1390525  
## 17 -0.6240801 -0.5176455 1.21852815 1.3751173 1.5810137 1.3208017  
## Mortgage Personal.Loan Securities.Account CD.Account Online CreditCard  
## 1 -0.5512914 No 2.9347083 -0.2518562 -1.1977423 -0.6420996  
## 3 -0.5512914 No -0.3406358 -0.2518562 -1.1977423 -0.6420996  
## 6 0.9764777 No -0.3406358 -0.2518562 0.8346258 -0.6420996  
## 7 -0.5512914 No -0.3406358 -0.2518562 0.8346258 -0.6420996  
## 12 -0.5512914 No -0.3406358 -0.2518562 0.8346258 -0.6420996  
## 17 0.7694896 Yes -0.3406358 -0.2518562 -1.1977423 -0.6420996

# Build KNN Function

knnBuild <- function(name, Knum) {  
 name <- knn(train = training\_set[, -8],  
 test = test\_set[, -8],  
 cl = training\_set[, 8],  
 k = Knum)  
 confusionMatrix(name, as.factor(test\_set[, 8]))  
}  
  
# K == 3 turned out to perform the best

# KNN Model with K == 3

knnBuild(K3, 3)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1794 79  
## Yes 14 113  
##   
## Accuracy : 0.9535   
## 95% CI : (0.9433, 0.9623)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6843   
##   
## Mcnemar's Test P-Value : 3.212e-11   
##   
## Sensitivity : 0.9923   
## Specificity : 0.5885   
## Pos Pred Value : 0.9578   
## Neg Pred Value : 0.8898   
## Prevalence : 0.9040   
## Detection Rate : 0.8970   
## Detection Prevalence : 0.9365   
## Balanced Accuracy : 0.7904   
##   
## 'Positive' Class : No   
##

# KNN Model with K == 5

knnBuild(K5, 5)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1799 91  
## Yes 9 101  
##   
## Accuracy : 0.95   
## 95% CI : (0.9395, 0.9591)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 1.904e-14   
##   
## Kappa : 0.644   
##   
## Mcnemar's Test P-Value : 5.496e-16   
##   
## Sensitivity : 0.9950   
## Specificity : 0.5260   
## Pos Pred Value : 0.9519   
## Neg Pred Value : 0.9182   
## Prevalence : 0.9040   
## Detection Rate : 0.8995   
## Detection Prevalence : 0.9450   
## Balanced Accuracy : 0.7605   
##   
## 'Positive' Class : No   
##

# KNN Model with K == 7

knnBuild(K5, 7)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1800 92  
## Yes 8 100  
##   
## Accuracy : 0.95   
## 95% CI : (0.9395, 0.9591)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 1.904e-14   
##   
## Kappa : 0.6419   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9956   
## Specificity : 0.5208   
## Pos Pred Value : 0.9514   
## Neg Pred Value : 0.9259   
## Prevalence : 0.9040   
## Detection Rate : 0.9000   
## Detection Prevalence : 0.9460   
## Balanced Accuracy : 0.7582   
##   
## 'Positive' Class : No   
##

# KNN Model with K == 9

knnBuild(K9, 9)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1803 99  
## Yes 5 93  
##   
## Accuracy : 0.948   
## 95% CI : (0.9373, 0.9573)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 2.971e-13   
##   
## Kappa : 0.6165   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9972   
## Specificity : 0.4844   
## Pos Pred Value : 0.9479   
## Neg Pred Value : 0.9490   
## Prevalence : 0.9040   
## Detection Rate : 0.9015   
## Detection Prevalence : 0.9510   
## Balanced Accuracy : 0.7408   
##   
## 'Positive' Class : No   
##

# Create new customer

newCustomer <- data.frame(Age = 40,  
 Experience = 10,  
 Income = 84,  
 Family = 2,  
 CCAvg = 2,  
 Education = 2,  
 Mortgage = 0,  
 Securities.Account = 0,  
 CD.Account = 0,  
 Online = 1,  
 CreditCard = 1)  
newCustomer

## Age Experience Income Family CCAvg Education Mortgage Securities.Account  
## 1 40 10 84 2 2 2 0 0  
## CD.Account Online CreditCard  
## 1 0 1 1

# Normalize new customer data

newCustomer <- predict(norm\_values, newCustomer)  
newCustomer

## Age Experience Income Family CCAvg Education Mortgage  
## 1 -0.4486535 -0.8691277 0.2181362 -0.3688488 0.03065312 0.1390525 -0.5512914  
## Securities.Account CD.Account Online CreditCard  
## 1 -0.3406358 -0.2518562 0.8346258 1.556872

# Run KNN on new customer data

customKNN <- knn(train = training\_set[, -8],  
 test = newCustomer,  
 cl = training\_set[, 8],  
 k = 3)  
customKNN

## [1] No  
## Levels: No Yes

# The KNN predicts that the new customer will not accept the loan offer