Knn\_Classification

Osama Bin Zahir

2023-10-01

#Importing required packages

library(psych)  
library(FNN)  
library(ISLR)  
library(class)

##   
## Attaching package: 'class'

## The following objects are masked from 'package:FNN':  
##   
## knn, knn.cv

library(caret)

## Loading required package: ggplot2

##   
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':  
##   
## %+%, alpha

## Loading required package: lattice

library(caTools)

#Importing dataset

universalbank <- read.csv("C:\\Users\\Osama Zahir\\Downloads\\UniversalBank.csv")  
summary(universalbank)

## ID Age Experience Income ZIP.Code   
## Min. : 1 Min. :23.00 Min. :-3.0 Min. : 8.00 Min. : 9307   
## 1st Qu.:1251 1st Qu.:35.00 1st Qu.:10.0 1st Qu.: 39.00 1st Qu.:91911   
## Median :2500 Median :45.00 Median :20.0 Median : 64.00 Median :93437   
## Mean :2500 Mean :45.34 Mean :20.1 Mean : 73.77 Mean :93153   
## 3rd Qu.:3750 3rd Qu.:55.00 3rd Qu.:30.0 3rd Qu.: 98.00 3rd Qu.:94608   
## Max. :5000 Max. :67.00 Max. :43.0 Max. :224.00 Max. :96651   
## Family CCAvg Education Mortgage   
## Min. :1.000 Min. : 0.000 Min. :1.000 Min. : 0.0   
## 1st Qu.:1.000 1st Qu.: 0.700 1st Qu.:1.000 1st Qu.: 0.0   
## Median :2.000 Median : 1.500 Median :2.000 Median : 0.0   
## Mean :2.396 Mean : 1.938 Mean :1.881 Mean : 56.5   
## 3rd Qu.:3.000 3rd Qu.: 2.500 3rd Qu.:3.000 3rd Qu.:101.0   
## Max. :4.000 Max. :10.000 Max. :3.000 Max. :635.0   
## Personal.Loan Securities.Account CD.Account Online   
## Min. :0.000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :0.000 Median :0.0000 Median :0.0000 Median :1.0000   
## Mean :0.096 Mean :0.1044 Mean :0.0604 Mean :0.5968   
## 3rd Qu.:0.000 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:1.0000   
## Max. :1.000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## CreditCard   
## Min. :0.000   
## 1st Qu.:0.000   
## Median :0.000   
## Mean :0.294   
## 3rd Qu.:1.000   
## Max. :1.000

#Eliminating ZIP code and ID from the dataset

ds=subset(universalbank, select=-c(ID, ZIP.Code ))  
summary(ds)

## Age Experience Income Family   
## Min. :23.00 Min. :-3.0 Min. : 8.00 Min. :1.000   
## 1st Qu.:35.00 1st Qu.:10.0 1st Qu.: 39.00 1st Qu.:1.000   
## Median :45.00 Median :20.0 Median : 64.00 Median :2.000   
## Mean :45.34 Mean :20.1 Mean : 73.77 Mean :2.396   
## 3rd Qu.:55.00 3rd Qu.:30.0 3rd Qu.: 98.00 3rd Qu.:3.000   
## Max. :67.00 Max. :43.0 Max. :224.00 Max. :4.000   
## CCAvg Education Mortgage Personal.Loan   
## Min. : 0.000 Min. :1.000 Min. : 0.0 Min. :0.000   
## 1st Qu.: 0.700 1st Qu.:1.000 1st Qu.: 0.0 1st Qu.:0.000   
## Median : 1.500 Median :2.000 Median : 0.0 Median :0.000   
## Mean : 1.938 Mean :1.881 Mean : 56.5 Mean :0.096   
## 3rd Qu.: 2.500 3rd Qu.:3.000 3rd Qu.:101.0 3rd Qu.:0.000   
## Max. :10.000 Max. :3.000 Max. :635.0 Max. :1.000   
## Securities.Account CD.Account Online CreditCard   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.000   
## Median :0.0000 Median :0.0000 Median :1.0000 Median :0.000   
## Mean :0.1044 Mean :0.0604 Mean :0.5968 Mean :0.294   
## 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:1.0000 3rd Qu.:1.000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.000

#converting education into factor

ds$Education = as.factor(ds$Education)

#convert education to dummy variables

groups = dummyVars(~.,data = ds) #this creates dummy groups  
ds\_df = as.data.frame(predict(groups, ds))  
summary(ds\_df)

## Age Experience Income Family   
## Min. :23.00 Min. :-3.0 Min. : 8.00 Min. :1.000   
## 1st Qu.:35.00 1st Qu.:10.0 1st Qu.: 39.00 1st Qu.:1.000   
## Median :45.00 Median :20.0 Median : 64.00 Median :2.000   
## Mean :45.34 Mean :20.1 Mean : 73.77 Mean :2.396   
## 3rd Qu.:55.00 3rd Qu.:30.0 3rd Qu.: 98.00 3rd Qu.:3.000   
## Max. :67.00 Max. :43.0 Max. :224.00 Max. :4.000   
## CCAvg Education.1 Education.2 Education.3   
## Min. : 0.000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.: 0.700 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median : 1.500 Median :0.0000 Median :0.0000 Median :0.0000   
## Mean : 1.938 Mean :0.4192 Mean :0.2806 Mean :0.3002   
## 3rd Qu.: 2.500 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :10.000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Mortgage Personal.Loan Securities.Account CD.Account   
## Min. : 0.0 Min. :0.000 Min. :0.0000 Min. :0.0000   
## 1st Qu.: 0.0 1st Qu.:0.000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median : 0.0 Median :0.000 Median :0.0000 Median :0.0000   
## Mean : 56.5 Mean :0.096 Mean :0.1044 Mean :0.0604   
## 3rd Qu.:101.0 3rd Qu.:0.000 3rd Qu.:0.0000 3rd Qu.:0.0000   
## Max. :635.0 Max. :1.000 Max. :1.0000 Max. :1.0000   
## Online CreditCard   
## Min. :0.0000 Min. :0.000   
## 1st Qu.:0.0000 1st Qu.:0.000   
## Median :1.0000 Median :0.000   
## Mean :0.5968 Mean :0.294   
## 3rd Qu.:1.0000 3rd Qu.:1.000   
## Max. :1.0000 Max. :1.000

#partitioning the data into training and validation

set.seed(123)  
split = sample.split(ds\_df, SplitRatio = 0.6)  
train.df = subset(ds\_df, split == TRUE)  
valid.df = subset(ds\_df, split == FALSE)  
  
# Print the sizes of the training and validation sets  
print(paste("The size of the training set is:", nrow(train.df)))

## [1] "The size of the training set is: 2857"

print(paste("The size of the Validation set is:", nrow(valid.df)))

## [1] "The size of the Validation set is: 2143"

# normalizing the data

train.norm.df = train.df[,-10] #note that personal income is the 10th variable  
valid.norm.df = valid.df[,-10]  
  
norm.values = preProcess(train.df[,-10], method=c("center", "scale"))  
train.norm.df = predict(norm.values, train.df[,-10])  
Valid.norm.df = predict(norm.values, valid.df[,-10])

#Question 1:

new\_cust = 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   
)  
new\_cust

## Age Experience Income Family CCAvg Education.1 Education.2 Education.3  
## 1 40 10 84 2 2 0 1 0  
## Mortgage Securities.Account CD.Account Online CreditCard  
## 1 0 0 0 1 1

# Normalize the new\_cust

new.cust.norm = new\_cust  
new.cust.norm = predict(norm.values, new.cust.norm)

knn1 = class::knn(train = train.norm.df, test = new.cust.norm, cl = train.df$Personal.Loan, k = 1)  
knn1

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

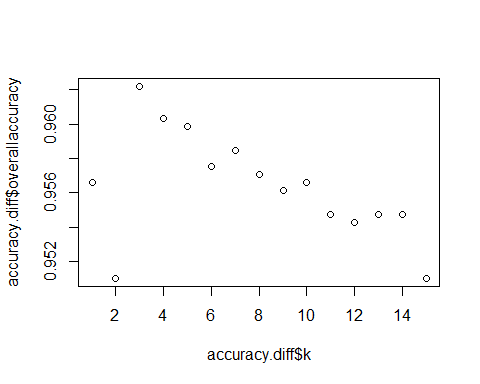
#Based on the kNN algorithm with a k value of 1 (i.e., considering only the nearest neighbor), the algorithm predicts that the new customer is in the class labeled “0.” which means loan is not accpeted.

# Question 2

accuracy.diff <- data.frame(k = seq(1, 15, 1), overallaccuracy = rep(0, 15))  
for(i in 1:15) {  
 KNN.Pred <- class::knn(train = train.norm.df,   
 test = Valid.norm.df,   
 cl = train.df$Personal.Loan, k = i)  
 accuracy.diff[i, 2] <- confusionMatrix(KNN.Pred,   
 as.factor(valid.df$Personal.Loan),positive = "1")$overall[1]  
}  
  
which(accuracy.diff[,2] == max(accuracy.diff[,2]))

## [1] 3

plot(accuracy.diff$k,accuracy.diff$overallaccuracy)



# The best K is 3

# Question 3

# Best k value   
best\_k <- 3  
  
# Train the kNN model with the best k  
best\_knn <- class::knn(train = train.norm.df,   
 test = Valid.norm.df,   
 cl = train.df$Personal.Loan, k = best\_k)  
  
# Create the confusion matrix  
confusion\_matrix <- confusionMatrix(best\_knn, as.factor(valid.df$Personal.Loan))  
  
# Display the confusion matrix  
print("Confusion Matrix:")

## [1] "Confusion Matrix:"

print(confusion\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1947 70  
## 1 11 115  
##   
## Accuracy : 0.9622   
## 95% CI : (0.9532, 0.9699)  
## No Information Rate : 0.9137   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.72   
##   
## Mcnemar's Test P-Value : 1.16e-10   
##   
## Sensitivity : 0.9944   
## Specificity : 0.6216   
## Pos Pred Value : 0.9653   
## Neg Pred Value : 0.9127   
## Prevalence : 0.9137   
## Detection Rate : 0.9085   
## Detection Prevalence : 0.9412   
## Balanced Accuracy : 0.8080   
##   
## 'Positive' Class : 0   
##

#Question 4:

# Customer data  
new\_cust <- 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   
)  
  
# Normalize the customer data  
new\_cust.norm <- predict(norm.values, new\_cust)  
  
# Classify the customer using the best k (k = 3)  
customer\_classification <- class::knn(train = train.norm.df,   
 test = new\_cust.norm,   
 cl = train.df$Personal.Loan,   
 k = best\_k)  
  
# Display the classification result  
if (customer\_classification == 1) {  
 cat("The customer is classified as 'Accepted (1)' for a personal loan.\n")  
} else {  
 cat("The customer is classified as 'Not Accepted (0)' for a personal loan.\n")  
}

## The customer is classified as 'Not Accepted (0)' for a personal loan.

# Question 5

# Partition the data into training, validation, and test sets (50% : 30% : 20%)  
set.seed(123)  
split1 <- sample.split(ds\_df, SplitRatio = 0.5)  
train\_valid.df <- subset(ds\_df, split1 == TRUE)  
valid\_test.df <- subset(ds\_df, split1 == FALSE)  
  
# Further split the combined validation and test data into 30% validation and 20% test  
split2 <- sample.split(valid\_test.df, SplitRatio = 0.6)  
valid.df <- subset(valid\_test.df, split2 == TRUE)  
test.df <- subset(valid\_test.df, split2 == FALSE)  
  
# Print the sizes of the training, validation, and test sets  
print(paste("The size of the training set is:", nrow(train\_valid.df)))

## [1] "The size of the training set is: 2500"

print(paste("The size of the Validation set is:", nrow(valid.df)))

## [1] "The size of the Validation set is: 1428"

print(paste("The size of the Test set is:", nrow(test.df)))

## [1] "The size of the Test set is: 1072"

# Normalize the data  
norm.values <- preProcess(train\_valid.df[, -10], method = c("center", "scale"))  
train\_valid.norm.df <- predict(norm.values, train\_valid.df[, -10])  
valid.norm.df <- predict(norm.values, valid.df[, -10])  
test.norm.df <- predict(norm.values, test.df[, -10])  
  
# Define the best k value  
best\_k <- 3  
  
# Train the k-NN model with the best k using the training set  
best\_knn\_train <- class::knn(train = train\_valid.norm.df,  
 test = train\_valid.norm.df,  
 cl = train\_valid.df$Personal.Loan,  
 k = best\_k)  
  
# Create the confusion matrix for the training set  
confusion\_matrix\_train <- confusionMatrix(best\_knn\_train, as.factor(train\_valid.df$Personal.Loan))  
  
# Train the k-NN model with the best k using the validation set  
best\_knn\_valid <- class::knn(train = train\_valid.norm.df,  
 test = valid.norm.df,  
 cl = train\_valid.df$Personal.Loan,  
 k = best\_k)  
  
# Create the confusion matrix for the validation set  
confusion\_matrix\_valid <- confusionMatrix(best\_knn\_valid, as.factor(valid.df$Personal.Loan))  
  
# Train the k-NN model with the best k using the test set  
best\_knn\_test <- class::knn(train = train\_valid.norm.df,  
 test = test.norm.df,  
 cl = train\_valid.df$Personal.Loan,  
 k = best\_k)  
  
# Create the confusion matrix for the test set  
confusion\_matrix\_test <- confusionMatrix(best\_knn\_test, as.factor(test.df$Personal.Loan))  
  
# Display the confusion matrices and their differences  
print("Confusion Matrix for Training Set:")

## [1] "Confusion Matrix for Training Set:"

print(confusion\_matrix\_train)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2235 48  
## 1 6 211  
##   
## Accuracy : 0.9784   
## 95% CI : (0.9719, 0.9837)  
## No Information Rate : 0.8964   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8747   
##   
## Mcnemar's Test P-Value : 2.414e-08   
##   
## Sensitivity : 0.9973   
## Specificity : 0.8147   
## Pos Pred Value : 0.9790   
## Neg Pred Value : 0.9724   
## Prevalence : 0.8964   
## Detection Rate : 0.8940   
## Detection Prevalence : 0.9132   
## Balanced Accuracy : 0.9060   
##   
## 'Positive' Class : 0   
##

print("Confusion Matrix for Validation Set:")

## [1] "Confusion Matrix for Validation Set:"

print(confusion\_matrix\_valid)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1301 41  
## 1 6 80  
##   
## Accuracy : 0.9671   
## 95% CI : (0.9565, 0.9757)  
## No Information Rate : 0.9153   
## P-Value [Acc > NIR] : 1.732e-15   
##   
## Kappa : 0.7557   
##   
## Mcnemar's Test P-Value : 7.071e-07   
##   
## Sensitivity : 0.9954   
## Specificity : 0.6612   
## Pos Pred Value : 0.9694   
## Neg Pred Value : 0.9302   
## Prevalence : 0.9153   
## Detection Rate : 0.9111   
## Detection Prevalence : 0.9398   
## Balanced Accuracy : 0.8283   
##   
## 'Positive' Class : 0   
##

print("Confusion Matrix for Test Set:")

## [1] "Confusion Matrix for Test Set:"

print(confusion\_matrix\_test)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 965 38  
## 1 7 62  
##   
## Accuracy : 0.958   
## 95% CI : (0.9442, 0.9692)  
## No Information Rate : 0.9067   
## P-Value [Acc > NIR] : 1.186e-10   
##   
## Kappa : 0.7118   
##   
## Mcnemar's Test P-Value : 7.744e-06   
##   
## Sensitivity : 0.9928   
## Specificity : 0.6200   
## Pos Pred Value : 0.9621   
## Neg Pred Value : 0.8986   
## Prevalence : 0.9067   
## Detection Rate : 0.9002   
## Detection Prevalence : 0.9356   
## Balanced Accuracy : 0.8064   
##   
## 'Positive' Class : 0   
##

# The model performs exceptionally well on the training set, with high accuracy and sensitivity.

# The model performs well on the validation set, with high sensitivity, although specificity has decreased compared to the training set. The model still maintains good accuracy and precision.

# The model performs well on the test set, with high sensitivity and accuracy. However, specificity has further decreased compared to both the training and validation sets.

# The high sensitivity suggests that the model is good at identifying customers who are likely to accept a personal loan, which is valuable for marketing purposes.

# The model shows a trend of decreasing specificity as it moves from the training set to the test set. This means that while the model is very good at correctly identifying customers who would not accept a personal loan (high sensitivity), it tends to produce more false positives in the test set, resulting in a lower specificity.