64060 Assignment 2

Melissa Dennis

2025-09-24

## Load libraries  
library(caret)

## Warning: package 'caret' was built under R version 4.3.3

## Loading required package: ggplot2

## Loading required package: lattice

library(ISLR)

## Warning: package 'ISLR' was built under R version 4.3.3

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.3.3

##   
## 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

library(ggplot2)  
library(FNN)

## Warning: package 'FNN' was built under R version 4.3.3

library(class)

## Warning: package 'class' was built under R version 4.3.3

##   
## Attaching package: 'class'

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

library(crayon)

## Warning: package 'crayon' was built under R version 4.3.3

##   
## Attaching package: 'crayon'

## The following object is masked from 'package:ggplot2':  
##   
## %+%

## Load dataset  
df <- read.csv("C:/Users/m\_den/OneDrive/Documents/UniversalBank.csv")

## Explore dataset  
head(df)

## ID Age Experience Income ZIP.Code Family CCAvg Education Mortgage  
## 1 1 25 1 49 91107 4 1.6 1 0  
## 2 2 45 19 34 90089 3 1.5 1 0  
## 3 3 39 15 11 94720 1 1.0 1 0  
## 4 4 35 9 100 94112 1 2.7 2 0  
## 5 5 35 8 45 91330 4 1.0 2 0  
## 6 6 37 13 29 92121 4 0.4 2 155  
## Personal.Loan Securities.Account CD.Account Online CreditCard  
## 1 0 1 0 0 0  
## 2 0 1 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 1  
## 6 0 0 0 1 0

summary(df)

## 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

cat("1. 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, and Credit Card = 1. Perform a k-NN classification with all predictors except ID and ZIP code using k = 1.   
  
Remember to transform categorical predictors with more than two categories into dummy variables first. Specify the success class as 1 (loan acceptance), and use the default cutoff value of 0.5.   
  
How would this customer be classified?")

## 1. 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, and Credit Card = 1. Perform a k-NN classification with all predictors except ID and ZIP code using k = 1.   
##   
## Remember to transform categorical predictors with more than two categories into dummy variables first. Specify the success class as 1 (loan acceptance), and use the default cutoff value of 0.5.   
##   
## How would this customer be classified?

## Remove Zip.Code and ID Columns and transform data into dummy variables  
  
df\_bank <- data.frame(select(df,-c(ZIP.Code,ID)) %>%   
 mutate(Education\_1 = ifelse(Education == 1,1,0),  
 Education\_2 = ifelse(Education == 2,1,0),  
 Education\_3 = ifelse(Education == 3,1,0)))  
   
df\_bank <- df\_bank %>% select(-Education)

## Data configuration sets  
set.seed(123)  
  
train.index <- createDataPartition(df$Personal.Loan,p = 0.6, list = FALSE)  
train.df <- df[train.index, ]  
valid.df <- df[-train.index, ]  
train.labels <- train.df$Personal.Loan  
valid.labels <- valid.df$Personal.Loan

## Check dimensions of new data partitions  
  
cat("The first value denotes the row count and the second represents the column count of the partitioned data, validating the accuracy of the 60/40 data split.", "\n", "\n")

## The first value denotes the row count and the second represents the column count of the partitioned data, validating the accuracy of the 60/40 data split.   
##

cat(bold("Train Data Dimensions:"), dim(train.df), "\n")

## Train Data Dimensions: 3000 14

cat(bold("Valid Data Dimensions:"), dim(valid.df), "\n")

## Valid Data Dimensions: 2000 14

## Add new customer information  
  
new.customer <- data.frame(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Education1 = 0, Education2 = 1, Education3 = 0, Mortgage = 0, Securities.Account = 0, CD.Account = 0, Online = 1, CreditCard = 1)  
  
## Confirm new data structure  
  
new.cust.form <- setdiff(names(train.df),names(new.customer))  
new.customer[new.cust.form] <- 0  
  
## Reorder columns  
  
new.customer <- new.customer[,names(train.df)]

## Normalize data  
  
train.norm.df <- train.df  
valid.norm.df <- valid.df  
norm.df <- df  
  
norm.values <- preProcess(train.df[, 1:2], method=c("center", "scale"))  
train.norm.df[, 1:2] <- predict(norm.values, train.df[, 1:2])  
valid.norm.df[, 1:2] <- predict(norm.values, valid.df[, 1:2])  
norm.df[, 1:2] <- predict(norm.values, df[, 1:2])  
new.norm.df <- predict(norm.values, new.customer)

## Use k-NN  
  
k <- 1  
  
cat("Utilizing 1 for the k value, the prediction output of '0' suggests the customer will not accept the loan","\n", "\n")

## Utilizing 1 for the k value, the prediction output of '0' suggests the customer will not accept the loan   
##

nn <- knn(train = train.norm.df[,-10], test = new.norm.df[,-10],  
cl = train.norm.df[, 10], k = k)  
  
##nn.new.cust.pred <- knn(train = train.df[,-10],test = new.cust.df, cl = train.df[,10], k=k, prob=TRUE  
  
nn

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

cat(bold("Validation Data", "\n", "\n"))

## Validation Data   
##

k.conf.matrix <- knn(train = train.norm.df[,-10], test = valid.norm.df[,-10], cl = train.norm.df[,10], k = k, prob = TRUE)  
confusionMatrix(k.conf.matrix, as.factor(valid.norm.df[,10]))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1692 137  
## 1 106 65  
##   
## Accuracy : 0.8785   
## 95% CI : (0.8634, 0.8925)  
## No Information Rate : 0.899   
## P-Value [Acc > NIR] : 0.99866   
##   
## Kappa : 0.282   
##   
## Mcnemar's Test P-Value : 0.05429   
##   
## Sensitivity : 0.9410   
## Specificity : 0.3218   
## Pos Pred Value : 0.9251   
## Neg Pred Value : 0.3801   
## Prevalence : 0.8990   
## Detection Rate : 0.8460   
## Detection Prevalence : 0.9145   
## Balanced Accuracy : 0.6314   
##   
## 'Positive' Class : 0   
##

cat("2. What is a choice of k that balances between overfitting and ignoring the predictor  
information?")

## 2. What is a choice of k that balances between overfitting and ignoring the predictor  
## information?

# initialize a data frame with two columns: k, and accuracy.  
accuracy.df <- data.frame(k = seq(1, 14, 1), accuracy = rep(0, 14))  
  
# compute accuracy  
  
for(i in 1:14){  
knn.pred <- knn(train = train.norm.df[, -10], test = valid.norm.df[, -10],  
cl = train.norm.df[, 10], k = i, prob = TRUE)  
accuracy.df[i, 2] <- confusionMatrix(knn.pred, as.factor(valid.norm.df[, 10]))$overall[1]  
}  
  
accuracy.df

## k accuracy  
## 1 1 0.8785  
## 2 2 0.8745  
## 3 3 0.8845  
## 4 4 0.8880  
## 5 5 0.8895  
## 6 6 0.8945  
## 7 7 0.8915  
## 8 8 0.8915  
## 9 9 0.8935  
## 10 10 0.8950  
## 11 11 0.8960  
## 12 12 0.8970  
## 13 13 0.8960  
## 14 14 0.8985

cat("Using the training data to classify the records in the validation date to calculate the error rates for various choises of k, we have determined '6' is the best value for k.", "\n", "\n")

## Using the training data to classify the records in the validation date to calculate the error rates for various choises of k, we have determined '6' is the best value for k.   
##

best.k <- 6

cat("3. Show the confusion matrix for the validation data that results from using the best k.")

## 3. Show the confusion matrix for the validation data that results from using the best k.

## Rerun with best.k  
  
cat(bold("Best Fit Validation Data", "\n", "\n"))

## Best Fit Validation Data   
##

best.conf.matrix <- knn(train = train.norm.df[,-10], test = valid.norm.df[,-10], cl = train.norm.df[,10], k = best.k, prob = TRUE)  
confusionMatrix(best.conf.matrix, as.factor(valid.norm.df[,10]))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1744 159  
## 1 54 43  
##   
## Accuracy : 0.8935   
## 95% CI : (0.8791, 0.9067)  
## No Information Rate : 0.899   
## P-Value [Acc > NIR] : 0.804   
##   
## Kappa : 0.2377   
##   
## Mcnemar's Test P-Value : 1.034e-12   
##   
## Sensitivity : 0.9700   
## Specificity : 0.2129   
## Pos Pred Value : 0.9164   
## Neg Pred Value : 0.4433   
## Prevalence : 0.8990   
## Detection Rate : 0.8720   
## Detection Prevalence : 0.9515   
## Balanced Accuracy : 0.5914   
##   
## 'Positive' Class : 0   
##

## Re-run accuracy with best k  
  
cat("Selecting 9 as the best k value gives us a prediction accuracy of 89.35%","\n", "\n")

## Selecting 9 as the best k value gives us a prediction accuracy of 89.35%   
##

conf.matrix <- table(Predicted = knn.pred, Actual = valid.labels)  
  
best.accuracy <- sum(diag(conf.matrix))/sum(conf.matrix)  
  
best.accuracy

## [1] 0.8985

cat("4. Consider the following customer: 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 and Credit  
Card = 1. Classify the customer using the best k.")

## 4. Consider the following customer: 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 and Credit  
## Card = 1. Classify the customer using the best k.

## Classify new customer with best k  
  
new.cust.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)  
  
nn.new.cust.pred <- knn(train = train.df[,-10],test = new.cust.df, cl = train.df[,10], k=9, prob = TRUE)  
  
  
cat("Even with the most effective k value, the prediction output of '0' suggests the customer will not accept the loan with an accuracy of 92.86%","\n", "\n")

## Even with the most effective k value, the prediction output of '0' suggests the customer will not accept the loan with an accuracy of 92.86%   
##

nn.new.cust.pred

## [1] 0  
## attr(,"prob")  
## [1] 0.9285714  
## Levels: 0 1

cat("5. Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%). Apply the k-NN method with the k chosen above. Compare the confusion matrix of the test set  
with that of the training and validation sets. Comment on the differences and their reason.")

## 5. Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%). Apply the k-NN method with the k chosen above. Compare the confusion matrix of the test set  
## with that of the training and validation sets. Comment on the differences and their reason.

set.seed(123)  
  
## Set training data to 50%  
train.index50 <- createDataPartition(df$Personal.Loan,p = 0.5, list = FALSE)  
train.df50 <- df[train.index, ]  
temp.df <- df[-train.index50, ]  
  
## Set validation data to 30%  
  
valid.index30 <- createDataPartition(temp.df$Personal.Loan,p = 0.6, list = FALSE)  
valid.df30 <- df[valid.index30, ]  
test.df20 <- df[-valid.index30, ]  
  
## Check dimensions of new data partitions  
  
cat("The first value denotes the row count and the second represents the column count of the partitioned data, validating the accuracy of the 50/30/20 data split.", "\n", "\n")

## The first value denotes the row count and the second represents the column count of the partitioned data, validating the accuracy of the 50/30/20 data split.   
##

cat(bold("Train Data Dimensions:"), dim(train.df50), "\n")

## Train Data Dimensions: 3000 14

cat(bold("Valid Data Dimensions:"), dim(valid.df30), "\n")

## Valid Data Dimensions: 1500 14

cat(bold("Test Data Dimensions:"), dim(test.df20))

## Test Data Dimensions: 3500 14

## Train kNN   
  
train.labels50 <- train.df50$Personal.Loan  
valid.labels30 <- valid.df30$Personal.Loan  
test.labels20 <- test.df20$Personal.Loan  
  
## Normalize data  
  
train.norm.df50 <- train.df50  
valid.norm.df30 <- valid.df30  
test.norm.df20 <- test.df20  
norm.df <- df  
  
norm.values2 <- preProcess(train.df50[, 1:2], method=c("center", "scale"))  
train.norm.df50[, 1:2] <- predict(norm.values2, train.df50[, 1:2])  
valid.norm.df30[, 1:2] <- predict(norm.values2, valid.df30[, 1:2])  
test.norm.df20[, 1:2] <- predict(norm.values2, test.df20[, 1:2])  
norm.df[, 1:2] <- predict(norm.values2, df[, 1:2])  
  
## kNN on train data  
  
nn.train50 <- knn(train = train.norm.df50[,-10], test = train.norm.df50[,-10], cl = train.norm.df50[, 10], k = 9)  
  
cat(bold("Train Data:"), "\n", "\n") ##data title

## Train Data:   
##

conf.matrix.train50 <- knn(train = train.norm.df50[,-10], test = train.norm.df50[,-10], cl = train.norm.df50[,10], k = 9, prob = TRUE)  
confusionMatrix(conf.matrix.train50, as.factor(train.norm.df50[,10]))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2689 210  
## 1 33 68  
##   
## Accuracy : 0.919   
## 95% CI : (0.9087, 0.9285)  
## No Information Rate : 0.9073   
## P-Value [Acc > NIR] : 0.01369   
##   
## Kappa : 0.3255   
##   
## Mcnemar's Test P-Value : < 2e-16   
##   
## Sensitivity : 0.9879   
## Specificity : 0.2446   
## Pos Pred Value : 0.9276   
## Neg Pred Value : 0.6733   
## Prevalence : 0.9073   
## Detection Rate : 0.8963   
## Detection Prevalence : 0.9663   
## Balanced Accuracy : 0.6162   
##   
## 'Positive' Class : 0   
##

## kNN on validation data  
  
nn.valid30 <- knn(train = train.norm.df50[,-10], test = valid.norm.df30[,-10],  
cl = train.norm.df50[, 10], k = 9)  
  
cat(bold("Validation Data:"), "\n", "\n") ##data title

## Validation Data:   
##

conf.matrix.valid30 <- knn(train = train.norm.df50[,-10], test = valid.norm.df30[,-10], cl = train.norm.df50[,10], k = best.k, prob = TRUE)  
confusionMatrix(conf.matrix.valid30, as.factor(valid.norm.df30[,10]))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1318 100  
## 1 32 50  
##   
## Accuracy : 0.912   
## 95% CI : (0.8965, 0.9259)  
## No Information Rate : 0.9   
## P-Value [Acc > NIR] : 0.06403   
##   
## Kappa : 0.3878   
##   
## Mcnemar's Test P-Value : 5.49e-09   
##   
## Sensitivity : 0.9763   
## Specificity : 0.3333   
## Pos Pred Value : 0.9295   
## Neg Pred Value : 0.6098   
## Prevalence : 0.9000   
## Detection Rate : 0.8787   
## Detection Prevalence : 0.9453   
## Balanced Accuracy : 0.6548   
##   
## 'Positive' Class : 0   
##

## kNN on test data  
  
nn.test20 <- knn(train = train.norm.df50[,-10], test = test.norm.df20[,-10], cl = train.norm.df50[, 10], k = best.k)  
  
  
conf.matrix.test20 <- knn(train = train.norm.df50[,-10], test = test.norm.df20[,-10], cl = train.norm.df50[,10], k = best.k, prob = TRUE)  
  
cat(bold("Test Data:"), "\n", "\n") ##data title

## Test Data:   
##

confusionMatrix(conf.matrix.test20, as.factor(test.norm.df20[,10]))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 3098 237  
## 1 72 93  
##   
## Accuracy : 0.9117   
## 95% CI : (0.9018, 0.9209)  
## No Information Rate : 0.9057   
## P-Value [Acc > NIR] : 0.1172   
##   
## Kappa : 0.3339   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9773   
## Specificity : 0.2818   
## Pos Pred Value : 0.9289   
## Neg Pred Value : 0.5636   
## Prevalence : 0.9057   
## Detection Rate : 0.8851   
## Detection Prevalence : 0.9529   
## Balanced Accuracy : 0.6296   
##   
## 'Positive' Class : 0   
##

## Calculate train accuracy  
  
cat("The train data accuracy is the highest. This is expected since the model was trained on this data. However, the validation and test accuracies are only slightly lower. This small difference suggests that the model is generalizing well and not overfitting or underfitting the data.", "\n", "\n")

## The train data accuracy is the highest. This is expected since the model was trained on this data. However, the validation and test accuracies are only slightly lower. This small difference suggests that the model is generalizing well and not overfitting or underfitting the data.   
##

conf.matrix.train50 <- table(Predicted = nn.train50, Actual = train.labels50)  
conf.matrix.valid30 <- table(Predicted = nn.valid30, Actual = valid.labels30)  
conf.matrix.test20 <- table(Predicted = nn.test20, Actual = test.labels20)  
  
train.accuracy50 <- sprintf("%.1f%%", sum(diag(conf.matrix.train50))/sum(conf.matrix.train50)\*100)  
  
cat(bold("Train Accuracy:"), train.accuracy50, "\n")

## Train Accuracy: 91.9%

## Calculate valid accuracy  
  
valid.accuracy30 <- sprintf("%.1f%%", sum(diag(conf.matrix.valid30))/sum(conf.matrix.valid30)\*100)  
  
cat( bold("Valid Accuracy:"), valid.accuracy30, "\n")

## Valid Accuracy: 90.7%

## Calculate test accuracy  
  
test.accuracy20 <- sprintf("%.1f%%",sum(diag(conf.matrix.test20))/sum(conf.matrix.test20)\*100)  
  
cat(bold("Test Accuracy:"), test.accuracy20, "\n")

## Test Accuracy: 91.0%