DillonSteiger\_Assignment2

Dillon Steiger

2025-09-23

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

library(tidyverse)

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

## Warning: package 'ggplot2' was built under R version 4.5.1

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

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

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

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

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

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 4.0.0 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.4 ✔ tidyr 1.3.1  
## ✔ purrr 1.1.0   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(caret)

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

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(class)  
library(fastDummies)

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

set.seed(42)  
  
bank<- read.csv("C:/Users/steig/Downloads/UniversalBank.csv")  
  
#drop ID, ZIP Code  
bank2<- bank %>% select(-ID, -ZIP.Code)  
  
#target var  
y\_full<- factor(bank2$Personal.Loan, levels = c(0,1))  
  
#make dummy cols for education (3 levels: 1,2,3)  
bank2<- fastDummies::dummy\_cols(bank2, select\_columns = "Education",   
 remove\_selected\_columns = TRUE)  
  
#remove Personal.Loan from predictors  
X\_full<- bank2 %>% select(-Personal.Loan)  
  
num\_cols<- sapply(X\_full, is.numeric)  
  
stopifnot(nrow(X\_full) == length(y\_full)) #quick check  
  
#60% train, 40% validation  
idx\_60<- caret::createDataPartition(y\_full, p = 0.60, list = FALSE)  
  
X\_train\_60<- X\_full[idx\_60, , drop = FALSE]  
y\_train\_60<- y\_full[idx\_60]  
X\_val\_40<- X\_full[-idx\_60, , drop = FALSE]  
y\_val\_40<- y\_full[-idx\_60]  
  
train\_center\_60<- sapply(X\_train\_60[, num\_cols, drop = FALSE], mean)  
train\_scale\_60<- sapply(X\_train\_60[, num\_cols, drop = FALSE], sd)  
train\_scale\_60[train\_scale\_60 == 0]<- 1 #to guard against zero variance  
  
Xtr\_60<- X\_train\_60  
Xva\_40<- X\_val\_40  
Xtr\_60[, num\_cols]<- scale(X\_train\_60[, num\_cols, drop = FALSE],  
 center = train\_center\_60, scale = train\_scale\_60)  
Xva\_40[, num\_cols]<- scale(X\_val\_40[, num\_cols, drop = FALSE],   
 center = train\_center\_60, scale = train\_scale\_60)  
  
  
#Q1  
#customer features (Education\_2 = 1 so Education\_1 = 0, Education\_3 = 0)  
cust<- tibble(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2,   
 Mortgage = 0, Securities.Account = 0, CD.Account = 0, Online = 1,  
 CreditCard = 1, Education\_1 = 0, Education\_2 = 1, Education\_3 = 0  
 ) %>% select(colnames(X\_full))  
  
#scaling cust w/ training stats  
cust\_scaled<- cust   
cust\_scaled[, num\_cols]<- scale(cust[, num\_cols, drop = FALSE],   
 center = train\_center\_60, scale = train\_scale\_60)  
  
#k = 1 prediction  
pred\_k1<- class::knn(train = Xtr\_60, test = cust\_scaled, cl = y\_train\_60, k = 1)  
cat("Q1 - Prediction with k = 1 (1 = accept loan):", as.character(pred\_k1), "\n\n")

## Q1 - Prediction with k = 1 (1 = accept loan): 0

#Q2  
ks<- 1:50  
val\_acc<- numeric(length(ks))  
  
for (i in seq\_along(ks)) {  
 pred\_val\_i<- class::knn(train = Xtr\_60, test = Xva\_40, cl = y\_train\_60, k = ks[i])  
 val\_acc[i]<- mean(pred\_val\_i == y\_val\_40)}  
  
acc\_tbl<- tibble(k = ks, val\_accuracy = val\_acc) %>% arrange(desc(val\_accuracy))  
print(head(acc\_tbl, 10))

## # A tibble: 10 × 2  
## k val\_accuracy  
## <int> <dbl>  
## 1 5 0.960  
## 2 7 0.958  
## 3 1 0.958  
## 4 3 0.958  
## 5 8 0.957  
## 6 4 0.956  
## 7 6 0.955  
## 8 9 0.954  
## 9 10 0.954  
## 10 2 0.954

best\_k<- acc\_tbl$k[1]  
cat("Q2 - Best k by validation accuracy:", best\_k, "\n\n")

## Q2 - Best k by validation accuracy: 5

#Q3  
pred\_val\_best<- class::knn(train = Xtr\_60, test = Xva\_40, cl = y\_train\_60, k = best\_k)  
caret::confusionMatrix(pred\_val\_best, y\_val\_40, positive = "1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1805 78  
## 1 3 114  
##   
## Accuracy : 0.9595   
## 95% CI : (0.9499, 0.9677)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7173   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.5938   
## Specificity : 0.9983   
## Pos Pred Value : 0.9744   
## Neg Pred Value : 0.9586   
## Prevalence : 0.0960   
## Detection Rate : 0.0570   
## Detection Prevalence : 0.0585   
## Balanced Accuracy : 0.7960   
##   
## 'Positive' Class : 1   
##

cat("Q3 - validation Confusion Matrix (k = ", best\_k, ")\n")

## Q3 - validation Confusion Matrix (k = 5 )

#Q4  
pred\_best\_cust<- class::knn(train = Xtr\_60, test = cust\_scaled, cl = y\_train\_60, k = best\_k)  
cat("\nQ4 - Customer prediction with best (k =", best\_k, ")\n")

##   
## Q4 - Customer prediction with best (k = 5 )

#Q5  
#50% training  
idx\_50<- caret::createDataPartition(y\_full, p = 0.50, list = FALSE)  
X\_tr\_50<- X\_full[idx\_50, , drop = FALSE]  
y\_tr\_50<- y\_full[idx\_50]  
X\_rest<- X\_full[-idx\_50, , drop = FALSE]  
y\_rest<- y\_full[-idx\_50]  
  
#from remaining 50%: 60% -> validation (30% overall), 40% -> test (20% overall)  
idx\_val\_from\_rest<- caret::createDataPartition(y\_rest, p = 0.60, list = FALSE)  
X\_va\_30<- X\_rest[idx\_val\_from\_rest, , drop = FALSE]  
y\_va\_30<- y\_rest[idx\_val\_from\_rest]  
X\_te\_20<- X\_rest[-idx\_val\_from\_rest, , drop = FALSE]  
y\_te\_20<- y\_rest[-idx\_val\_from\_rest]  
  
#scaling w/ only 50% training stats  
train\_center\_50<- sapply(X\_tr\_50[, num\_cols, drop = FALSE], mean)  
train\_scale\_50<- sapply(X\_tr\_50[, num\_cols, drop = FALSE], sd)  
train\_scale\_50[train\_scale\_50 == 0]<- 1  
  
Xtr\_50<- X\_tr\_50  
Xva\_30\_scaled<- X\_va\_30  
Xte\_20\_scaled<- X\_te\_20  
  
Xtr\_50[, num\_cols]<- scale(X\_tr\_50[, num\_cols, drop = FALSE],  
 center = train\_center\_50, scale = train\_scale\_50)  
Xva\_30\_scaled[, num\_cols]<- scale(X\_va\_30[, num\_cols, drop = FALSE],   
 center = train\_center\_50, scale = train\_scale\_50)  
Xte\_20\_scaled[, num\_cols]<- scale(X\_te\_20[, num\_cols]<- X\_te\_20[, num\_cols, drop = FALSE],   
 center = train\_center\_50, scale = train\_scale\_50)  
  
#eval with best\_k  
pred\_tr\_50<- class::knn(train = Xtr\_50, test = Xtr\_50, cl = y\_tr\_50, k = best\_k)  
pred\_va\_30<- class::knn(train = Xtr\_50, test = Xva\_30\_scaled, cl = y\_tr\_50, k = best\_k)  
pred\_te\_20<- class::knn(train = Xtr\_50, test = Xte\_20\_scaled, cl = y\_tr\_50, k = best\_k)  
  
cm\_tr\_50<- caret::confusionMatrix(pred\_tr\_50, y\_tr\_50, positive = "1")  
cm\_va\_30<- caret::confusionMatrix(pred\_va\_30, y\_va\_30, positive = "1")  
cm\_te\_20<- caret::confusionMatrix(pred\_te\_20, y\_te\_20, positive = "1")  
  
cat("Q5 - Train (50%) Confusion Matrix (k =", best\_k, ")\n"); print(cm\_tr\_50)

## Q5 - Train (50%) Confusion Matrix (k = 5 )

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2256 78  
## 1 4 162  
##   
## Accuracy : 0.9672   
## 95% CI : (0.9594, 0.9738)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7808   
##   
## Mcnemar's Test P-Value : 7.536e-16   
##   
## Sensitivity : 0.6750   
## Specificity : 0.9982   
## Pos Pred Value : 0.9759   
## Neg Pred Value : 0.9666   
## Prevalence : 0.0960   
## Detection Rate : 0.0648   
## Detection Prevalence : 0.0664   
## Balanced Accuracy : 0.8366   
##   
## 'Positive' Class : 1   
##

cat("\nQ5 - Validation (30%) Confusion Matrix (k =", best\_k, ")\n"); print(cm\_va\_30)

##   
## Q5 - Validation (30%) Confusion Matrix (k = 5 )

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1354 67  
## 1 2 77  
##   
## Accuracy : 0.954   
## 95% CI : (0.9421, 0.964)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 3.461e-13   
##   
## Kappa : 0.668   
##   
## Mcnemar's Test P-Value : 1.312e-14   
##   
## Sensitivity : 0.53472   
## Specificity : 0.99853   
## Pos Pred Value : 0.97468   
## Neg Pred Value : 0.95285   
## Prevalence : 0.09600   
## Detection Rate : 0.05133   
## Detection Prevalence : 0.05267   
## Balanced Accuracy : 0.76662   
##   
## 'Positive' Class : 1   
##

cat("\nQ5 - Test (20%) Confusion Matrix (k = ", best\_k, ")\n"); print(cm\_te\_20)

##   
## Q5 - Test (20%) Confusion Matrix (k = 5 )

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 901 54  
## 1 3 42  
##   
## Accuracy : 0.943   
## 95% CI : (0.9268, 0.9565)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 4.950e-06   
##   
## Kappa : 0.5694   
##   
## Mcnemar's Test P-Value : 3.528e-11   
##   
## Sensitivity : 0.4375   
## Specificity : 0.9967   
## Pos Pred Value : 0.9333   
## Neg Pred Value : 0.9435   
## Prevalence : 0.0960   
## Detection Rate : 0.0420   
## Detection Prevalence : 0.0450   
## Balanced Accuracy : 0.7171   
##   
## 'Positive' Class : 1   
##

# Comparison of Confusion Matrices  
#Training set 50%: accuracy is highest of the three sets, specificity is almost perfect, sensitivity is reasonably good but lower than specificity. This makes sense since k-NN tends to memorize local structure on the training data so predictions are strongest here.  
  
#Validation set 30%: accuracy drops slightly compared to training, specificity remains very high (few false positives), sensitivity decreases further which means more of the "loan acceptors" are missed. This is expected since moving from training to hold out data exposes the model to new patterns thereby reducing recall.  
  
#Test set 20%: accuracy is similar to validation, specificity is strong, sensitivity is slightly less than validation but still noticeably lower than training. This confirms the model generalization well as validation and test results are consistent with no major over-fitting.  
  
#Training vs Validation/Test  
#Training shows the best performance because k-NN is essentially "memorizing" the training points. The neighbors include the training points themselves. Validation and test accuracies are slightly lower which is normal and indicates some generalization error.  
  
#High specificity vs lower sensitivity  
#All splits show very high specificity but more modest sensitivity. This imbalance is due to the dataset's class distribution in which only 9.6% of customers accepted a loan in the campaign. k-NN tends to favor the majority class (0 = no loan) when classes are imbalanced.  
  
#Consistency between validation and test  
#Validation and test results are very close which means the choice of k = 3 gives stable, generalizable performance. k = 3 balances over-fitting and under-fitting, maintains high accuracy across training, validation, and test sets, and produces consistent confusion matrices. It uses enough neighbors to smooth out noise without losing important predictor information.