Assignment 2

Fundamentals of Machine Learning

Hollister Victor

6-8-2025

library(readr) # Loading of required libraries readr- for csv files  
library(dplyr) # for data manipulation

##   
## 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(caret) # for dummy variable creation and data partition

## Loading required package: ggplot2

## Loading required package: lattice

library(class) # for KNN   
library(ggplot2) # for plotting model accuracy  
  
#Read and Clean The Data Set   
bank <- read\_csv("/Users/hollyvictor/Downloads/UniversalBank.csv") %>%  
select(-ID, -`ZIP Code`) # remove ID and ZIP predictors

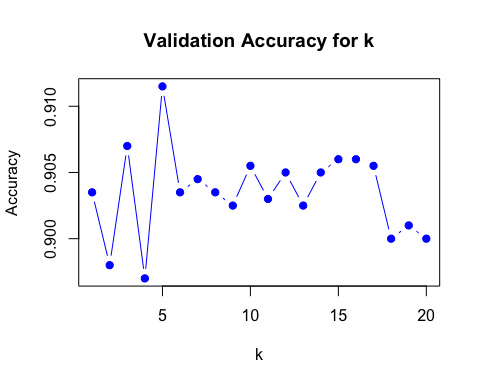
## Rows: 5000 Columns: 14

## ── 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.

dummies <- dummyVars(`Personal Loan` ~ ., data = bank) #Transform categorical predictors >two categories into dummy variables  
bank\_transformed <- data.frame(predict(dummies, newdata = bank))  
bank\_transformed$Personal\_Loan <- bank$`Personal Loan`  
  
x\_names <- setdiff(names(bank\_transformed), "Personal\_Loan")  
  
set.seed(123) #Partition the data into training (60%) and validation (40%) sets.  
train\_index <- createDataPartition(bank\_transformed$Personal\_Loan, p = 0.6, list = FALSE)  
train\_data <- bank\_transformed[train\_index, ]  
valid\_data <- bank\_transformed[-train\_index, ]  
  
  
x\_train <- train\_data[, x\_names] # Prepares the predictor matrix (X) and target vector (y) for modeling.  
y\_train <- train\_data$Personal\_Loan  
x\_valid <- valid\_data[, x\_names]  
y\_valid <- valid\_data$Personal\_Loan  
  
#\_\_\_\_ NEW CUSTOMER PREDICTION\_\_\_\_\_  
  
new\_customer <- as.data.frame(matrix(0, nrow = 1, ncol = ncol(x\_train))) #creates new customer input row  
colnames(new\_customer) <- colnames(x\_train)  
  
safe\_set <- function(df, column, value) {  
 if (column %in% colnames(df)) df[[column]] <- value  
 return(df)  
}  
  
new\_customer <- safe\_set(new\_customer, "Age", 40)  
new\_customer <- safe\_set(new\_customer, "Experience", 10)  
new\_customer <- safe\_set(new\_customer, "Income", 84)  
new\_customer <- safe\_set(new\_customer, "Family", 2)  
new\_customer <- safe\_set(new\_customer, "CCAvg", 2)  
new\_customer <- safe\_set(new\_customer, "Mortgage", 0)  
new\_customer <- safe\_set(new\_customer, "Securities.Account", 0)  
new\_customer <- safe\_set(new\_customer, "CD.Account", 0)  
new\_customer <- safe\_set(new\_customer, "Online", 1)  
new\_customer <- safe\_set(new\_customer, "CreditCard", 1)  
new\_customer <- safe\_set(new\_customer, "Education.1", 0)  
new\_customer <- safe\_set(new\_customer, "Education.2", 1)  
new\_customer <- safe\_set(new\_customer, "Education.3", 0)  
  
stopifnot(identical(names(new\_customer), names(x\_train))) #added this to ensure that new\_customer has the same structure as x\_train.   
  
   
prediction\_k1 <- knn(train = x\_train, test = new\_customer, cl = y\_train, k = 1) #Perform a k-NN classification … using k = 1  
cat("Prediction for new customer with k=1:", prediction\_k1, "\n")

## Prediction for new customer with k=1: 1

accuracy <- c()  
for (k in 1:20) {  
 pred <- knn(train = x\_train, test = x\_valid, cl = y\_train, k = k) #What is a choice of k that balances between overfitting and ignoring predictor information?”  
 acc <- mean(pred == y\_valid)  
 accuracy <- c(accuracy, acc)  
}  
  
  
plot(1:20, accuracy, type = "b", pch = 19, col = "blue",  
 xlab = "k", ylab = "Accuracy", main = "Validation Accuracy for k") #visual respresenation to select best K



best\_k <- which.max(accuracy)  
cat("Best k is:", best\_k, "with accuracy:", round(accuracy[best\_k], 4), "\n") #Selects the value of k with the highest validation accuracy.

## Best k is: 5 with accuracy: 0.9115

pred\_bestk <- knn(train = x\_train, test = x\_valid, cl = y\_train, k = best\_k)  
conf\_matrix <- confusionMatrix(pred\_bestk, as.factor(y\_valid), positive = "1") #Show the confusion matrix for the validation data that results from using the best k.  
print(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1745 123  
## 1 53 79  
##   
## Accuracy : 0.912   
## 95% CI : (0.8987, 0.9241)  
## No Information Rate : 0.899   
## P-Value [Acc > NIR] : 0.02746   
##   
## Kappa : 0.4273   
##   
## Mcnemar's Test P-Value : 1.981e-07   
##   
## Sensitivity : 0.3911   
## Specificity : 0.9705   
## Pos Pred Value : 0.5985   
## Neg Pred Value : 0.9342   
## Prevalence : 0.1010   
## Detection Rate : 0.0395   
## Detection Prevalence : 0.0660   
## Balanced Accuracy : 0.6808   
##   
## 'Positive' Class : 1   
##

prediction\_bestk <- knn(train = x\_train, test = new\_customer, cl = y\_train, k = best\_k)  
cat("Prediction for new customer with best k =", best\_k, ":", prediction\_bestk, "\n") #Classify the customer using the best k

## Prediction for new customer with best k = 5 : 1

set.seed(123) #Repartition Data: 50/30/20 Train/Validation/Test Split  
train\_index2 <- createDataPartition(bank\_transformed$Personal\_Loan, p = 0.5, list = FALSE)  
train\_set <- bank\_transformed[train\_index2, ]  
temp\_set <- bank\_transformed[-train\_index2, ]  
  
  
valid\_index2 <- createDataPartition(temp\_set$Personal\_Loan, p = 0.6, list = FALSE)  
valid\_set <- temp\_set[valid\_index2, ]  
test\_set <- temp\_set[-valid\_index2, ]  
  
x\_train2 <- train\_set[, x\_names] #k-NN on All Three Sets with Best k.Compare the confusion matrix of the test set with that of the training and validation sets.”  
y\_train2 <- train\_set$Personal\_Loan  
x\_valid2 <- valid\_set[, x\_names]  
y\_valid2 <- valid\_set$Personal\_Loan  
x\_test2 <- test\_set[, x\_names]  
y\_test2 <- test\_set$Personal\_Loan  
  
  
pred\_train <- knn(train = x\_train2, test = x\_train2, cl = y\_train2, k = best\_k)  
pred\_valid <- knn(train = x\_train2, test = x\_valid2, cl = y\_train2, k = best\_k)  
pred\_test <- knn(train = x\_train2, test = x\_test2, cl = y\_train2, k = best\_k)  
  
print(confusionMatrix(pred\_train, as.factor(y\_train2), positive = "1"))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2235 113  
## 1 36 116  
##   
## Accuracy : 0.9404   
## 95% CI : (0.9304, 0.9494)  
## No Information Rate : 0.9084   
## P-Value [Acc > NIR] : 2.577e-09   
##   
## Kappa : 0.5781   
##   
## Mcnemar's Test P-Value : 4.780e-10   
##   
## Sensitivity : 0.5066   
## Specificity : 0.9841   
## Pos Pred Value : 0.7632   
## Neg Pred Value : 0.9519   
## Prevalence : 0.0916   
## Detection Rate : 0.0464   
## Detection Prevalence : 0.0608   
## Balanced Accuracy : 0.7453   
##   
## 'Positive' Class : 1   
##

print(confusionMatrix(pred\_valid, as.factor(y\_valid2), positive = "1"))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1306 98  
## 1 51 45  
##   
## Accuracy : 0.9007   
## 95% CI : (0.8844, 0.9153)  
## No Information Rate : 0.9047   
## P-Value [Acc > NIR] : 0.7188540   
##   
## Kappa : 0.3249   
##   
## Mcnemar's Test P-Value : 0.0001643   
##   
## Sensitivity : 0.31469   
## Specificity : 0.96242   
## Pos Pred Value : 0.46875   
## Neg Pred Value : 0.93020   
## Prevalence : 0.09533   
## Detection Rate : 0.03000   
## Detection Prevalence : 0.06400   
## Balanced Accuracy : 0.63855   
##   
## 'Positive' Class : 1   
##

print(confusionMatrix(pred\_test, as.factor(y\_test2), positive = "1"))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 864 74  
## 1 28 34  
##   
## Accuracy : 0.898   
## 95% CI : (0.8776, 0.9161)  
## No Information Rate : 0.892   
## P-Value [Acc > NIR] : 0.2908   
##   
## Kappa : 0.3487   
##   
## Mcnemar's Test P-Value : 8.363e-06   
##   
## Sensitivity : 0.3148   
## Specificity : 0.9686   
## Pos Pred Value : 0.5484   
## Neg Pred Value : 0.9211   
## Prevalence : 0.1080   
## Detection Rate : 0.0340   
## Detection Prevalence : 0.0620   
## Balanced Accuracy : 0.6417   
##   
## 'Positive' Class : 1   
##

PPV calculation:

Out of all customers the model predicted **would accept** the loan, **about 60% actually did**.

79/79+53=59.85%

A close-up of numbers

AI-generated content may be incorrect.

**Summary of Results**

The k-Nearest Neighbors (k-NN) algorithm was applied to the UniversalBank dataset to predict whether a customer would accept a personal loan offer. After evaluating values of k from 1 to 20, the optimal number of neighbors was determined to be k = 5, yielding the highest validation accuracy of 91.15%.

The model is very good at identifying customers who are not likely to accept a loan (high specificity). Its ability to find the customers who would accept a loan, however, is somewhat limited (i.e., low sensitivity). When it predicts that a customer will accept a loan, it is correct about 60% of the time (i.e., positive predictive value).

The model shows consistent accuracy and specificity when you look across the sets of training, validation, and testing. Yet, its sensitivity drops significantly from 50.66% in the training set to about 31.5% in both the validation and test sets and suggesting an ability to generalize that's just not there when it comes to detecting true loan acceptors.

This performance gap is likely due to the **class imbalance** in the dataset—only about **9.6%** of customers accepted a loan offer. As a result, the model tends to favor the majority class (loan not accepted). Additionally, since k-NN is a **local, instance-based method**, it is particularly sensitive to variations in the data distribution between training and unseen sets, which may further hinder its ability to consistently detect the minority class.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Sensitivity ( Loan accepted) | Specificity ( Loan declined) |
| Train | 94.04% | 50.66% | 98.41% |
| Validation | 90.07% | 31.47% | 96.24% |
| Test | 89.8% | 31.48% | 96.86% |