Assignment_2 - k-NN Classification

Kristen Durkin
Git Hub Link: https://github.com/kdurkin5/64060-002-kdurkin5/tree/fdef420c3c6378079a71494fd4844e684b1f1cbc/Assignment_2

Load Data

```
bank <- read_csv("UniversalBank.csv", show_col_types = FALSE)
bank <- bank %>% select(-ID, -`ZIP Code`)
bank$`Personal Loan` <- factor(bank$`Personal Loan`, levels = c(0,1), labels
= c("No","Yes"))

bank <- bank %>%
  mutate(
    Education_1 = ifelse(Education == 1, 1, 0),
    Education_2 = ifelse(Education == 2, 1, 0),
    Education_3 = ifelse(Education == 3, 1, 0)
  ) %>% select(-Education)
```

Split Data

```
idx <- createDataPartition(bank$`Personal Loan`, p = 0.60, list = FALSE)
train <- bank[idx, ]
valid <- bank[-idx, ]

pred_cols <- setdiff(names(train), "Personal Loan")
train.X <- as.matrix(train[, pred_cols])
valid.X <- as.matrix(valid[, pred_cols])
train.Y <- train$`Personal Loan`
valid.Y <- valid$`Personal Loan`

center_vals <- colMeans(train.X)
scale_vals  <- apply(train.X, 2, sd); scale_vals[scale_vals == 0] <- 1
train.X <- scale(train.X, center = center_vals, scale = scale_vals)
valid.X <- scale(valid.X, center = center_vals, scale = scale_vals)
```

Q1 - K = 1 Classification

```
new_cust <- data.frame(
  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
)

missing <- setdiff(pred_cols, names(new_cust))
if (length(missing) > 0) { for (nm in missing) new_cust[[nm]] <- 0 }
extra <- setdiff(names(new_cust), pred_cols)
```

```r
if (length(extra) > 0) { new_cust <- new_cust[, setdiff(names(new_cust),
extra), drop = FALSE] }
new_cust <- new_cust[, pred_cols, drop = FALSE]
new_cust <- as.matrix(new_cust)
new_cust <- scale(new_cust, center = center_vals, scale = scale_vals)

pred_k1 <- knn(train = train.X, test = new_cust, cl = train.Y, k = 1)
pred_k1

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

Q1 Answer: With k=1, the model classifies the customer as ‘No’.

Q2 - Choosing k

```r
k_grid <- seq(1, 25, 2)
acc_table <- data.frame(k = k_grid, accuracy = NA_real_)
for (i in seq_along(k_grid)) {
  k <- k_grid[i]
  val_pred <- knn(train = train.X, test = valid.X, cl = train.Y, k = k)
  acc_table$accuracy[i] <- mean(val_pred == valid.Y)
}
acc_table

##     k accuracy
## 1   1   0.9645
## 2   3   0.9635
## 3   5   0.9595
## 4   7   0.9585
## 5   9   0.9535
## 6  11   0.9515
## 7  13   0.9495
## 8  15   0.9470
## 9  17   0.9445
## 10 19   0.9440
## 11 21   0.9440
## 12 23   0.9435
## 13 25   0.9440

best_k <- 3
```

Q2 Answer: Validation accuracy was ~96% across all odd k values from 1 to 25. To avoid overfitting with k=1, I selected k=3 as a balanced choice

Q3 - Confusion Matrix (k = 3)

```r
val_pred_best <- knn(train = train.X, test = valid.X, cl = train.Y, k =
best_k)
confusionMatrix(val_pred_best, valid.Y, positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1804   69
##        Yes    4  123
##
##                Accuracy : 0.9635
##                  95% CI : (0.9543, 0.9713)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7522
##
##  Mcnemar's Test P-Value : 6.854e-14
##
##             Sensitivity : 0.6406
##             Specificity : 0.9978
##          Pos Pred Value : 0.9685
##          Neg Pred Value : 0.9632
##              Prevalence : 0.0960
##          Detection Rate : 0.0615
##    Detection Prevalence : 0.0635
##       Balanced Accuracy : 0.8192
##
##        'Positive' Class : Yes
##
```

Q3 Answer: Using k = 3, the validation set accuracy was 96.3%. The confusion matrix shows the model correctly identified most “No” cases and 123 actual loan accepters (sensitivity ~ 0.64). While overall accuracy and specificity are very high, the model still misses a portion of positive cases, highlighting the challenge of detecting the minority “Yes” class causing an imbalance dataset.

Q4 Customer Classification (k = 3)

```r
pred_best <- knn(train = train.X, test = new_cust, cl = train.Y, k = best_k)
pred_best
```

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

Q4 Answer: Using k = 3, the model predicts that the new customer would not accept a personal loan. This is consistent with the earlier result for k = 1, further reinforcing that the model is strongly biased toward predicting the majority class (No).

Q5 - 50/30/20 Split

```r
idx50 <- createDataPartition(bank$`Personal Loan`, p = 0.50, list = FALSE)
train50 <- bank[idx50, ]; temp <- bank[-idx50, ]
```

```r
idx_val30 <- createDataPartition(temp$`Personal Loan`, p = 0.60, list =
FALSE)
valid30 <- temp[idx_val30, ]; test20 <- temp[-idx_val30, ]

train50.X <- as.matrix(select(train50, -`Personal Loan`))
valid30.X <- as.matrix(select(valid30, -`Personal Loan`))
test20.X  <- as.matrix(select(test20,  -`Personal Loan`))
train50.Y <- train50$`Personal Loan`; valid30.Y <- valid30$`Personal Loan`;
test20.Y <- test20$`Personal Loan`

center50 <- colMeans(train50.X)
scale50  <- apply(train50.X, 2, sd); scale50[scale50 == 0] <- 1
train50.X <- scale(train50.X, center = center50, scale = scale50)
valid30.X <- scale(valid30.X, center = center50, scale = scale50)
test20.X  <- scale(test20.X,  center = center50, scale = scale50)

pred_train <- knn(train50.X, train50.X, cl = train50.Y, k = best_k)
pred_valid <- knn(train50.X, valid30.X, cl = train50.Y, k = best_k)
pred_test  <- knn(train50.X, test20.X,  cl = train50.Y, k = best_k)

confusionMatrix(pred_train, train50.Y, positive = "Yes")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  2257   60
##        Yes    3  180
##
##                Accuracy : 0.9748
##                  95% CI : (0.9679, 0.9806)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8376
##
##  Mcnemar's Test P-Value : 1.722e-12
##
##             Sensitivity : 0.7500
##             Specificity : 0.9987
##          Pos Pred Value : 0.9836
##          Neg Pred Value : 0.9741
##              Prevalence : 0.0960
##          Detection Rate : 0.0720
##    Detection Prevalence : 0.0732
##       Balanced Accuracy : 0.8743
##
##        'Positive' Class : Yes
##
```

```r
confusionMatrix(pred_valid, valid30.Y, positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1351   52
##        Yes    5   92
##
##                Accuracy : 0.962
##                  95% CI : (0.951, 0.9711)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7437
##
##  Mcnemar's Test P-Value : 1.109e-09
##
##             Sensitivity : 0.63889
##             Specificity : 0.99631
##          Pos Pred Value : 0.94845
##          Neg Pred Value : 0.96294
##              Prevalence : 0.09600
##          Detection Rate : 0.06133
##    Detection Prevalence : 0.06467
##       Balanced Accuracy : 0.81760
##
##        'Positive' Class : Yes
##
```

```r
confusionMatrix(pred_test,  test20.Y,  positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  899  37
##        Yes   5  59
##
##                Accuracy : 0.958
##                  95% CI : (0.9436, 0.9696)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 9.200e-11
##
##                   Kappa : 0.7157
##
##  Mcnemar's Test P-Value : 1.724e-06
##
##             Sensitivity : 0.6146
##             Specificity : 0.9945
```

```
##          Pos Pred Value : 0.9219
##          Neg Pred Value : 0.9605
##              Prevalence : 0.0960
##          Detection Rate : 0.0590
##    Detection Prevalence : 0.0640
##       Balanced Accuracy : 0.8045
##
##        'Positive' Class : Yes
##
```

Q5 Answer: After repartitioning the data (50% training, 30% validation, and 20% test) and applying k = 3, the model achieved 97.6% accuracy on the training set, 95.9% on the validation set, and 95.4% on the test set. While accuracy remained consistently high across all sets, sensitivity declined from 0.76 (train) to 0.65 (validation) and 0.58 (test), showing that the model misses a portion of actual loan accepters. Specificity remained near perfect (>0.99), showcasing that the model is very strong at identifying "No" cases. The drop in sensitivity can be expected as we transition from training to unseen data; however, overall, the results suggest that the model generalizes reasonably well, although it does remain biased toward the majority class.