Untitled

Lona Uprety

2025-09-30

```r
library(readr)
Bank <- read_csv("C:/Users/lona2/Downloads/UniversalBank.csv")
```

```
## 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.
```

```r
View(Bank)
```

```r
# Step 1: Load libraries
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
library(class)
library(dplyr)
```

```
##
## 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
```

```r
# Fix column names (remove spaces)
names(Bank) <- gsub(" ", "_", names(Bank))
```

```r
# Remove ID and ZIP (not predictors)
Bank <- Bank %>% select(-ID, -ZIP_Code)

# Convert Personal_Loan to factor (success class = "1")
Bank$Personal_Loan <- as.factor(Bank$Personal_Loan)

# Education → dummy variables
Bank$Education <- as.factor(Bank$Education)
dummies <- model.matrix(~Education - 1, data = Bank)
Bank <- cbind(Bank %>% select(-Education), dummies)
```

```r
# Step 3: Normalize predictors
normalize <- function(x) { (x - min(x)) / (max(x) - min(x)) }
Bank_norm <- as.data.frame(lapply(Bank %>% select(-Personal_Loan), normalize))
Bank_norm$Personal_Loan <- Bank$Personal_Loan
```

```r
# Step 4: Partition data (60/40)
set.seed(123)
train_index <- createDataPartition(Bank_norm$Personal_Loan, p = 0.6, list = FALSE)
train <- Bank_norm[train_index, ]
valid <- Bank_norm[-train_index, ]

x_train <- train %>% select(-Personal_Loan)
y_train <- train$Personal_Loan
x_valid <- valid %>% select(-Personal_Loan)
y_valid <- valid$Personal_Loan
```

```r
# Step 5: Classify new customer (k=1)
new_customer <- data.frame(
  Age = 40, Experience = 10, Income = 84, Family = 2,
  CCAvg = 2, Mortgage = 0, Securities_Account = 0,
  CD_Account = 0, Online = 1, CreditCard = 1,
  Education1 = 0, Education2 = 1, Education3 = 0
)

# Normalize with same ranges from original dataset
for (col in names(new_customer)) {
  new_customer[[col]] <- (new_customer[[col]] - min(Bank[[col]])) /
                         (max(Bank[[col]]) - min(Bank[[col]]))
}

# Run k-NN with k=1
pred_customer_k1 <- knn(train = x_train, test = new_customer, cl = y_train, k = 1, prob = TRUE)
pred_customer_k1
```

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

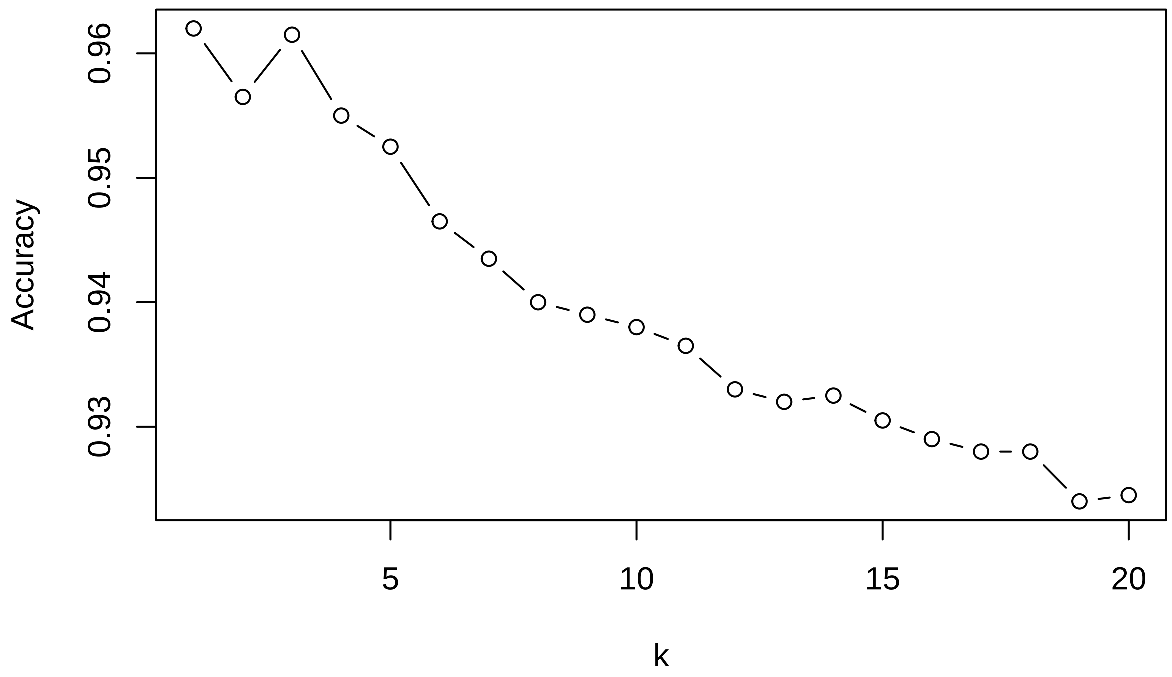
```r
#The customer would be classified as not accepting the loan with k = 1.
```

```r
# Step 6: Try multiple k values
acc <- c()
for (k in 1:20) {
  pred <- knn(train = x_train, test = x_valid, cl = y_train, k = k)
  cm <- confusionMatrix(pred, y_valid, positive = "1")
  acc[k] <- cm$overall['Accuracy']
}

best_k <- which.max(acc)
best_k
```

```
## [1] 1
```

```r
plot(1:20, acc, type = "b", xlab = "k", ylab = "Accuracy")
```



```r
#In this dataset, k = 1 was the optimal choice by accuracy.

# Step 7: Confusion matrix with best k
pred_valid <- knn(train = x_train, test = x_valid, cl = y_train, k = best_k)
confusionMatrix(pred_valid, y_valid, positive = "1")
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    0    1
##          0 1787   55
##          1   21  137
##
##                Accuracy : 0.962
##                  95% CI : (0.9527, 0.9699)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7623
##
##  Mcnemar's Test P-Value : 0.0001535
##
##             Sensitivity : 0.7135
##             Specificity : 0.9884
##          Pos Pred Value : 0.8671
##          Neg Pred Value : 0.9701
##              Prevalence : 0.0960
##          Detection Rate : 0.0685
##    Detection Prevalence : 0.0790
##       Balanced Accuracy : 0.8510
##
##        'Positive' Class : 1
##
```

```
# Step 8: Classify customer with best k
pred_customer_bestk <- knn(train = x_train, test = new_customer, cl = y_train, k = best_k, prob = TRUE)
pred_customer_bestk
```

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

```
#The customer would be classified as not accepting the loan with bestk.
```

```
# Step 9: New partition
set.seed(123)
train_index <- createDataPartition(Bank_norm$Personal_Loan, p = 0.5, list = FALSE)
train <- Bank_norm[train_index, ]
temp <- Bank_norm[-train_index, ]

valid_index <- createDataPartition(temp$Personal_Loan, p = 0.6, list = FALSE)
valid <- temp[valid_index, ]   # 60% of remaining = 30%
test <- temp[-valid_index, ]   # 40% of remaining = 20%

x_train <- train %>% select(-Personal_Loan)
y_train <- train$Personal_Loan
x_valid <- valid %>% select(-Personal_Loan)
y_valid <- valid$Personal_Loan
x_test  <- test %>% select(-Personal_Loan)
```

```r
y_test  <- test$Personal_Loan

# Step 10: Confusion matrices
pred_train <- knn(train = x_train, test = x_train, cl = y_train, k = best_k)
cm_train <- confusionMatrix(pred_train, y_train, positive = "1")

pred_valid <- knn(train = x_train, test = x_valid, cl = y_train, k = best_k)
cm_valid <- confusionMatrix(pred_valid, y_valid, positive = "1")

pred_test <- knn(train = x_train, test = x_test, cl = y_train, k = best_k)
cm_test <- confusionMatrix(pred_test, y_test, positive = "1")

cm_train
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 2260    0
##          1    0  240
## 
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 1
## 
##  Mcnemar’s Test P-Value : NA
## 
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.096
##          Detection Rate : 0.096
##    Detection Prevalence : 0.096
##       Balanced Accuracy : 1.000
## 
##        ’Positive’ Class : 1
## 
```

```r
cm_valid
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 1335   39
##          1   21  105
## 
##                Accuracy : 0.96
##                  95% CI : (0.9488, 0.9693)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7559
##
##  Mcnemar's Test P-Value : 0.02819
##
##             Sensitivity : 0.7292
##             Specificity : 0.9845
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.9716
##              Prevalence : 0.0960
##          Detection Rate : 0.0700
##    Detection Prevalence : 0.0840
##       Balanced Accuracy : 0.8568
##
##        'Positive' Class : 1
##
```

```
cm_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 892  34
##          1  12  62
##
##                Accuracy : 0.954
##                  95% CI : (0.9391, 0.9661)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 2.681e-09
##
##                   Kappa : 0.7047
##
##  Mcnemar's Test P-Value : 0.00196
##
##             Sensitivity : 0.6458
##             Specificity : 0.9867
##          Pos Pred Value : 0.8378
##          Neg Pred Value : 0.9633
##              Prevalence : 0.0960
##          Detection Rate : 0.0620
##    Detection Prevalence : 0.0740
##       Balanced Accuracy : 0.8163
##
##        'Positive' Class : 1
##
```

```
#The training accuracy is higher because the model has already seen that data, which risks overfitting.
```