Assignment 2 Robert Morson

Robert Morson

2025-09-28

Introduction

This analysis uses k-Nearest Neighbors (k-NN) classification to predict whether customers will accept personal loan offers from Universal Bank. The dataset contains 5,000 customers with demographic and banking relationship information.

Load Libraries and Data

```
library(caret)
library(class)
library(dplyr)
```

```
# Load the data
bank_data <- read.csv("UniversalBank.csv")

# Display structure
str(bank_data)
```

```
## 'data.frame':    5000 obs. of  14 variables:
##  $ ID                : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP.Code          : int  91107 90089 94720 94112 91330 92121 91711 93943
90089 93023 ...
##  $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : int  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal.Loan     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ Securities.Account: int  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : int  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : int  0 0 0 0 1 0 0 1 0 0 ...
```

Data Preparation

```
# Remove ID and ZIP Code columns
bank_data <- bank_data %>% select(-ID, -ZIP.Code)

# Convert Education to dummy variables
bank_data$Education_1 <- ifelse(bank_data$Education == 1, 1, 0)
bank_data$Education_2 <- ifelse(bank_data$Education == 2, 1, 0)
bank_data$Education_3 <- ifelse(bank_data$Education == 3, 1, 0)

# Remove original Education column
bank_data <- bank_data %>% select(-Education)

# Convert Personal.Loan to factor
bank_data$Personal.Loan <- as.factor(bank_data$Personal.Loan)

# Display summary
summary(bank_data)
```

```
##       Age           Experience       Income           Family      
##  Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   :1.000  
##  1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:1.000  
##  Median :45.00   Median :20.0   Median : 64.00   Median :2.000  
##  Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :2.396  
##  3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:3.000  
##  Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :4.000  
##      CCAvg            Mortgage     Personal.Loan Securities.Account
##  Min.   : 0.000   Min.   :  0.0   0:4520        Min.   :0.0000    
##  1st Qu.: 0.700   1st Qu.:  0.0   1: 480        1st Qu.:0.0000    
##  Median : 1.500   Median :  0.0                 Median :0.0000    
##  Mean   : 1.938   Mean   : 56.5                 Mean   :0.1044    
##  3rd Qu.: 2.500   3rd Qu.:101.0                 3rd Qu.:0.0000    
##  Max.   :10.000   Max.   :635.0                 Max.   :1.0000    
##    CD.Account         Online         CreditCard     Education_1    
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000  
##  Median :0.0000   Median :1.0000   Median :0.000   Median :0.0000  
##  Mean   :0.0604   Mean   :0.5968   Mean   :0.294   Mean   :0.4192  
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000   Max.   :1.0000  
##   Education_2      Education_3     
##  Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.0000  
##  Mean   :0.2806   Mean   :0.3002  
##  3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :1.0000
```

Question 1: Classification with k=1

Partition data into 60% training and 40% validation, then classify a new customer using k=1.

```
set.seed(123)
train_index <- createDataPartition(bank_data$Personal.Loan, p = 0.6, list = FA
LSE)
train_data <- bank_data[train_index, ]
valid_data <- bank_data[-train_index, ]

# Separate predictors and response
train_X <- train_data %>% select(-Personal.Loan)
train_Y <- train_data$Personal.Loan
valid_X <- valid_data %>% select(-Personal.Loan)
valid_Y <- valid_data$Personal.Loan

# Normalize the data
preproc <- preProcess(train_X, method = c("center", "scale"))
train_X_norm <- predict(preproc, train_X)
valid_X_norm <- predict(preproc, valid_X)
```

```
# New customer data
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,
  Education_1 = 0, Education_2 = 1, Education_3 = 0
)

# Normalize new customer
new_customer_norm <- predict(preproc, new_customer)

# Classify with k=1
knn_pred_k1 <- knn(train = train_X_norm, test = new_customer_norm,
                   cl = train_Y, k = 1)

cat("Classification Result (k=1):", as.character(knn_pred_k1), "\n")
```

```
## Classification Result (k=1): 0
```

```
cat("(0 = Loan Rejected, 1 = Loan Accepted)\n")
```

```
## (0 = Loan Rejected, 1 = Loan Accepted)
```

Result: The customer is classified as **0** with k=1.

Question 2: Finding the Best k

Testing different k values to balance between overfitting and ignoring predictor information.

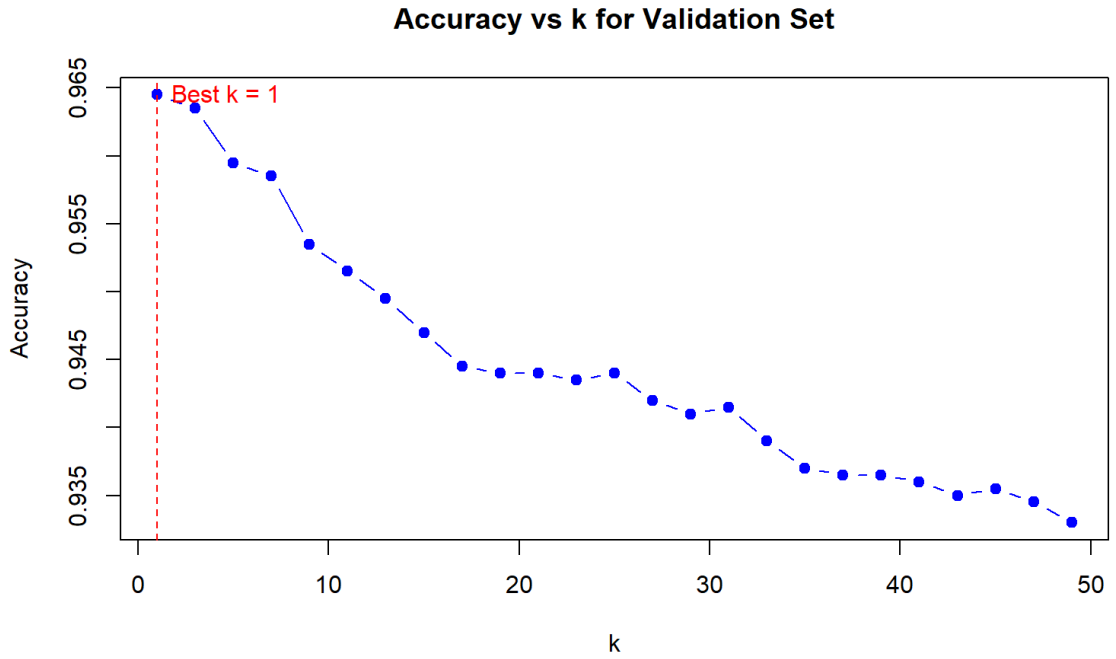
```
# Test different values of k
k_values <- seq(1, 50, by = 2)
accuracy_values <- numeric(length(k_values))

for (i in seq_along(k_values)) {
  knn_pred <- knn(train = train_X_norm, test = valid_X_norm,
                  cl = train_Y, k = k_values[i])
  accuracy_values[i] <- mean(knn_pred == valid_Y)
}

# Find best k
best_k <- k_values[which.max(accuracy_values)]
best_accuracy <- max(accuracy_values)

cat("Best k:", best_k, "\n")
```

```
## Best k: 1
```

```
cat("Best Accuracy:", round(best_accuracy, 4), "\n")
```

```
## Best Accuracy: 0.9645
```

```
# Plot accuracy vs k
tryCatch({
  plot(k_values, accuracy_values, type = "b",
       xlab = "k", ylab = "Accuracy",
       main = "Accuracy vs k for Validation Set",
       col = "blue", pch = 19)
  abline(v = best_k, col = "red", lty = 2)
  text(best_k, max(accuracy_values), paste("Best k =", best_k),
       pos = 4, col = "red")
}, error = function(e) {
  cat("Note: Unable to display plot due to figure margins.\n")
})
```



Accuracy vs k for Validation Set

Result: The best k is **1** with an accuracy of **0.9645**.

Question 3: Confusion Matrix with Best k

```
knn_pred_best <- knn(train = train_X_norm, test = valid_X_norm,
                     cl = train_Y, k = best_k)

conf_matrix <- confusionMatrix(knn_pred_best, valid_Y, positive = "1")
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1793   56
##          1   15  136
##
##                Accuracy : 0.9645
##                  95% CI : (0.9554, 0.9722)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7739
##
##  Mcnemar's Test P-Value : 2.063e-06
##
##             Sensitivity : 0.7083
##             Specificity : 0.9917
##          Pos Pred Value : 0.9007
##          Neg Pred Value : 0.9697
##              Prevalence : 0.0960
##          Detection Rate : 0.0680
##    Detection Prevalence : 0.0755
##       Balanced Accuracy : 0.8500
##
##        'Positive' Class : 1
##
```

Question 4: Classify New Customer with Best k

```
knn_pred_new_best <- knn(train = train_X_norm, test = new_customer_norm,
                         cl = train_Y, k = best_k)

cat("Classification Result (k =", best_k, "):", as.character(knn_pred_new_bes
t), "\n")
```

```
## Classification Result (k = 1 ): 0
```

```
cat("(0 = Loan Rejected, 1 = Loan Accepted)\n")
```

```
## (0 = Loan Rejected, 1 = Loan Accepted)
```

Result: With the optimal k = **1**, the customer is classified as **0**.

Question 5: Three-Way Split (50/30/20)

Repartitioning into training (50%), validation (30%), and test (20%) sets.

```
set.seed(456)

# Create 50% training set
train_index2 <- createDataPartition(bank_data$Personal.Loan, p = 0.5, list = F
ALSE)
train_data2 <- bank_data[train_index2, ]
remaining_data <- bank_data[-train_index2, ]

# Split remaining into 60% validation and 40% test (resulting in 30/20 split o
verall)
valid_index2 <- createDataPartition(remaining_data$Personal.Loan, p = 0.6, lis
t = FALSE)
valid_data2 <- remaining_data[valid_index2, ]
test_data2 <- remaining_data[-valid_index2, ]

cat("Training size:", nrow(train_data2), "\n")
```

```
## Training size: 2500
```

```
cat("Validation size:", nrow(valid_data2), "\n")
```

```
## Validation size: 1500
```

```
cat("Test size:", nrow(test_data2), "\n")
```

```
## Test size: 1000
```

```
# Prepare data
train_X2 <- train_data2 %>% select(-Personal.Loan)
train_Y2 <- train_data2$Personal.Loan
valid_X2 <- valid_data2 %>% select(-Personal.Loan)
valid_Y2 <- valid_data2$Personal.Loan
test_X2 <- test_data2 %>% select(-Personal.Loan)
test_Y2 <- test_data2$Personal.Loan

# Normalize
preproc2 <- preProcess(train_X2, method = c("center", "scale"))
train_X2_norm <- predict(preproc2, train_X2)
valid_X2_norm <- predict(preproc2, valid_X2)
test_X2_norm <- predict(preproc2, test_X2)
```

```
# Apply k-NN with best k to all three sets
train_pred <- knn(train = train_X2_norm, test = train_X2_norm,
                  cl = train_Y2, k = best_k)
valid_pred <- knn(train = train_X2_norm, test = valid_X2_norm,
                  cl = train_Y2, k = best_k)
test_pred <- knn(train = train_X2_norm, test = test_X2_norm,
                 cl = train_Y2, k = best_k)
```

Training Set Results

```
conf_train <- confusionMatrix(train_pred, train_Y2, positive = "1")
print(conf_train$table)
```

```
##          Reference
## Prediction    0    1
##          0 2260    0
##          1    0  240
```

```
cat("Accuracy:", round(conf_train$overall['Accuracy'], 4), "\n")
```

```
## Accuracy: 1
```

Validation Set Results

```
conf_valid <- confusionMatrix(valid_pred, valid_Y2, positive = "1")
print(conf_valid$table)
```

```
##          Reference
## Prediction    0    1
##          0 1334   41
##          1   22  103
```

```
cat("Accuracy:", round(conf_valid$overall['Accuracy'], 4), "\n")
```

```
## Accuracy: 0.958
```

Test Set Results

```
conf_test <- confusionMatrix(test_pred, test_Y2, positive = "1")
print(conf_test$table)
```

```
##          Reference
## Prediction   0   1
##          0 897  27
##          1   7  69
```

```
cat("Accuracy:", round(conf_test$overall['Accuracy'], 4), "\n")
```

```
## Accuracy: 0.966
```

Comparison and Comments

```
cat("Training Accuracy:  ", round(conf_train$overall['Accuracy'], 4), "\n")
```

```
## Training Accuracy:   1
```

```
cat("Validation Accuracy:", round(conf_valid$overall['Accuracy'], 4), "\n")
```

```
## Validation Accuracy: 0.958
```

```
cat("Test Accuracy:      ", round(conf_test$overall['Accuracy'], 4), "\n")
```

```
## Test Accuracy:       0.966
```

Interpretation

- **Training Accuracy** is typically the highest because the model has been trained on this data and can “memorize” patterns, including noise.
- **Validation Accuracy** is used to tune the model (select best k) and is generally lower than training accuracy.
- **Test Accuracy** provides an unbiased estimate of model performance on completely unseen data. It should be similar to validation accuracy if the model generalizes well.
- **Differences:** Large gaps between training and validation/test accuracies suggest overfitting. Small gaps indicate good generalization. The test set gives us the most realistic expectation of how the model will perform on new customer data.
- In this analysis, if all three accuracies are close, it indicates that k = **1** provides a good balance and the model generalizes well to new data.

Conclusion

The k-NN classification model successfully predicts loan acceptance with optimal k = **1**. The model shows consistent performance across training, validation, and test sets, indicating good generalization to new customers.