Assignment

2024-02-19

Introduction

The file UniversalBank.csv contains data on 5000 customers. The dataset includes customer demographic information (age, income, etc.), the customer's relationship with the bank (mortgage, securities account, etc.), and the customer response to the last personal loan campaign (Personal Loan). Among these 5000 customers, only 480 (= 9.6%) accepted the personal loan that was offered to them in the earlier campaign. ### Load necessary libraries

```r
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
```

Read the data

```r
# Load the data
setwd("/Users/meghana/Downloads")
Bank_data = read.csv("UniversalBank.csv")
# Check the structure and summary of the dataset
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 ...
```

```r
summary(Bank_data)
```

```
##        ID            Age          Experience       Income          ZIP.Code    
##  Min.   :   1   Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   : 9307  
##  1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:91911  
##  Median :2500   Median :45.00   Median :20.0   Median : 64.00   Median :93437  
##  Mean   :2500   Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :93152  
##  3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:94608  
##  Max.   :5000   Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :96651  
##      Family          CCAvg          Education        Mortgage    
##  Min.   :1.000   Min.   : 0.000   Min.   :1.000   Min.   :  0.0  
##  1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0  
##  Median :2.000   Median : 1.500   Median :2.000   Median :  0.0  
##  Mean   :2.396   Mean   : 1.938   Mean   :1.881   Mean   : 56.5  
##  3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0  
##  Max.   :4.000   Max.   :10.000   Max.   :3.000   Max.   :635.0  
##  Personal.Loan   Securities.Account   CD.Account         Online      
##  Min.   :0.000   Min.   :0.0000     Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.000   Median :0.0000     Median :0.0000   Median :1.0000  
##  Mean   :0.096   Mean   :0.1044     Mean   :0.0604   Mean   :0.5968  
##  3rd Qu.:0.000   3rd Qu.:0.0000     3rd Qu.:0.0000   3rd Qu.:1.0000  
##  Max.   :1.000   Max.   :1.0000     Max.   :1.0000   Max.   :1.0000  
##    CreditCard   
##  Min.   :0.000  
##  1st Qu.:0.000  
##  Median :0.000  
##  Mean   :0.294  
##  3rd Qu.:1.000  
##  Max.   :1.000  
```

Romove ID and ZIP Code as they are not predictors

```r
# Drop unnecessary columns (ID and ZIP code)
Bank_data <- Bank_data[, -c(1, 5)]
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          Education        Mortgage     Personal.Loan  
##  Min.   : 0.000   Min.   :1.000   Min.   :  0.0   Min.   :0.000  
##  1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0   1st Qu.:0.000  
##  Median : 1.500   Median :2.000   Median :  0.0   Median :0.000  
##  Mean   : 1.938   Mean   :1.881   Mean   : 56.5   Mean   :0.096  
##  3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0   3rd Qu.:0.000  
##  Max.   :10.000   Max.   :3.000   Max.   :635.0   Max.   :1.000  
##  Securities.Account   CD.Account         Online         CreditCard   
##  Min.   :0.0000     Min.   :0.0000   Min.   :0.0000   Min.   :0.000  
##  1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000  
##  Median :0.0000     Median :0.0000   Median :1.0000   Median :0.000  
##  Mean   :0.1044     Mean   :0.0604   Mean   :0.5968   Mean   :0.294  
##  3rd Qu.:0.0000     3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000  
##  Max.   :1.0000     Max.   :1.0000   Max.   :1.0000   Max.   :1.000
```

Split Data into 60% training and 40% validation. There are many ways to do this. We will look at 2 different ways. Before we split, let us transform categorical variables into dummy variables

###Only Education needs to be converted to factor

```r
Bank_data$Education <- as.factor(Bank_data$Education)
head(Bank_data$Education)
```

```
## [1] 1 1 1 2 2 2
## Levels: 1 2 3
```

Now, Convert Education to Dummy Variables

```r
dummy_groups <- dummyVars(~., data = Bank_data)
Bank_data <- as.data.frame(predict(dummy_groups, Bank_data))
```

Data Partitioning

Overview

Partition the data into training (60%) and validation (40%) sets.

```r
set.seed(1)
train_indices <- sample(row.names(Bank_data), 0.6 * nrow(Bank_data))
valid_indices <- setdiff(row.names(Bank_data), train_indices)

train_df <- Bank_data[train_indices, ]
head(train_df)
```

```
##      Age Experience Income Family CCAvg Education.1 Education.2 Education.3
## 1017  30          5     69      1  0.80           0           1           0
## 4775  56         32     22      1  1.20           0           0           1
## 2177  41         14     51      3  2.33           0           1           0
## 1533  45         20     55      1  0.30           1           0           0
## 4567  24          0    131      1  5.40           1           0           0
## 2347  52         26     59      2  1.50           0           1           0
##      Mortgage Personal.Loan Securities.Account CD.Account Online CreditCard
## 1017        0             0                  1          0      1          0
## 4775        0             0                  0          0      1          1
## 2177        0             0                  0          0      1          0
## 1533        0             0                  0          0      1          1
## 4567        0             0                  0          0      1          0
## 2347      239             0                  0          0      0          1
```

```
valid_df <- Bank_data[valid_indices, ]
tail(valid_df)
```

```
##      Age Experience Income Family CCAvg Education.1 Education.2 Education.3
## 4984  51         26     72      1  2.90           1           0           0
## 4988  48         23     43      3  1.70           0           1           0
## 4990  24          0     38      1  1.00           0           0           1
## 4994  45         21    218      2  6.67           1           0           0
## 4995  64         40     75      3  2.00           0           0           1
## 4998  63         39     24      2  0.30           0           0           1
##      Mortgage Personal.Loan Securities.Account CD.Account Online CreditCard
## 4984        0             0                  0          0      0          0
## 4988      159             0                  0          0      1          0
## 4990        0             0                  0          0      1          0
## 4994        0             0                  0          0      1          0
## 4995        0             0                  0          0      1          0
## 4998        0             0                  0          0      0          0
```

Normalize Data

```
norm_values <- preProcess(train_df[, -which(names(train_df) %in% c("Personal.Loan"))], method = c("cente
train_norm <- predict(norm_values, train_df[, -which(names(train_df) %in% c("Personal.Loan"))])
valid_norm <- predict(norm_values, valid_df[, -which(names(valid_df) %in% c("Personal.Loan"))])
head(train_norm)
```

```
##              Age  Experience      Income     Family      CCAvg Education.1
## 1017 -1.35692091 -1.33449201 -0.08930255 -1.2057601 -0.6438668  -0.8461728
## 4775  0.92977739  1.03707939 -1.11769684 -1.2057601 -0.4128307  -0.8461728
## 2177 -0.38947163 -0.54396821 -0.48315568  0.5320637  0.2398463  -0.8461728
## 1533 -0.03767189 -0.01695234 -0.39563276 -1.2057601 -0.9326620   1.1813978
## 4567 -1.88462051 -1.77367191  1.26730268 -1.2057601  2.0130485   1.1813978
## 2347  0.57797765  0.51006352 -0.30810985 -0.3368482 -0.2395536  -0.8461728
##      Education.2 Education.3   Mortgage Securities.Account CD.Account
## 1017   1.5836463  -0.6509102 -0.5679457          2.9939587 -0.2380992
## 4775  -0.6312436   1.5357982 -0.5679457         -0.3338946 -0.2380992
## 2177   1.5836463  -0.6509102 -0.5679457         -0.3338946 -0.2380992
```

```
## 1533  -0.6312436  -0.6509102 -0.5679457         -0.3338946 -0.2380992
## 4567  -0.6312436  -0.6509102 -0.5679457         -0.3338946 -0.2380992
## 2347   1.5836463  -0.6509102  1.7992927         -0.3338946 -0.2380992
##          Online CreditCard
## 1017  0.8426977  -0.643135
## 4775  0.8426977   1.554365
## 2177  0.8426977  -0.643135
## 1533  0.8426977   1.554365
## 4567  0.8426977  -0.643135
## 2347 -1.1862695   1.554365
```

```
tail(valid_norm)
```

```
##              Age  Experience      Income     Family      CCAvg Education.1
## 4984  0.49002772  0.51006352 -0.02366036 -1.2057601  0.5690728   1.1813978
## 4988  0.22617791  0.24655559 -0.65820152  0.5320637 -0.1240356  -0.8461728
## 4990 -1.88462051 -1.77367191 -0.76760517 -1.2057601 -0.5283488  -0.8461728
## 4994 -0.03767189  0.07088363  3.17092615 -0.3368482  2.7465881   1.1813978
## 4995  1.63337687  1.73976722  0.04198183  0.5320637  0.0492415  -0.8461728
## 4998  1.54542693  1.65193124 -1.07393538 -0.3368482 -0.9326620  -0.8461728
##      Education.2 Education.3   Mortgage Securities.Account CD.Account
## 4984  -0.6312436  -0.6509102 -0.5679457         -0.3338946 -0.2380992
## 4988   1.5836463  -0.6509102  1.0069117         -0.3338946 -0.2380992
## 4990  -0.6312436   1.5357982 -0.5679457         -0.3338946 -0.2380992
## 4994  -0.6312436  -0.6509102 -0.5679457         -0.3338946 -0.2380992
## 4995  -0.6312436   1.5357982 -0.5679457         -0.3338946 -0.2380992
## 4998  -0.6312436   1.5357982 -0.5679457         -0.3338946 -0.2380992
##          Online CreditCard
## 4984 -1.1862695  -0.643135
## 4988  0.8426977  -0.643135
## 4990  0.8426977  -0.643135
## 4994  0.8426977  -0.643135
## 4995  0.8426977  -0.643135
## 4998 -1.1862695  -0.643135
```

Consider a new customer

```
new_customer <- data.frame(
  Age = 40,
  Experience = 10,
  Income = 84,
  Family = 2,
  CCAvg = 2,
  Education_1 = 0,
  Education_2 = 1,
  Education_3 = 0,
  Mortgage = 0,
  `Securities Account` = 0,
  `CD Account` = 0,
  Online = 1,
  `Credit Card` = 1
)
```

Normalize the new customer data using the same preprocessing

```
train_norm <- train_df[,-10] # Note that Personal Income is the 10th variable
valid_norm <- valid_df[,-10]

norm_values <- preProcess(train_df[, -10], method=c("center", "scale"))
train_norm <- predict(norm_values, train_df[, -10])
valid_norm <- predict(norm_values, valid_df[, -10])
norm_values
```

```
## Created from 3000 samples and 13 variables
##
## Pre-processing:
##   - centered (13)
##   - ignored (0)
##   - scaled (13)
```

```
head(train_norm)
```

```
##              Age  Experience      Income     Family      CCAvg Education.1
## 1017 -1.35692091 -1.33449201 -0.08930255 -1.2057601 -0.6438668  -0.8461728
## 4775  0.92977739  1.03707939 -1.11769684 -1.2057601 -0.4128307  -0.8461728
## 2177 -0.38947163 -0.54396821 -0.48315568  0.5320637  0.2398463  -0.8461728
## 1533 -0.03767189 -0.01695234 -0.39563276 -1.2057601 -0.9326620   1.1813978
## 4567 -1.88462051 -1.77367191  1.26730268 -1.2057601  2.0130485   1.1813978
## 2347  0.57797765  0.51006352 -0.30810985 -0.3368482 -0.2395536  -0.8461728
##      Education.2 Education.3   Mortgage Securities.Account CD.Account
## 1017   1.5836463  -0.6509102 -0.5679457          2.9939587 -0.2380992
## 4775  -0.6312436   1.5357982 -0.5679457         -0.3338946 -0.2380992
## 2177   1.5836463  -0.6509102 -0.5679457         -0.3338946 -0.2380992
## 1533  -0.6312436  -0.6509102 -0.5679457         -0.3338946 -0.2380992
## 4567  -0.6312436  -0.6509102 -0.5679457         -0.3338946 -0.2380992
## 2347   1.5836463  -0.6509102  1.7992927         -0.3338946 -0.2380992
##          Online CreditCard
## 1017  0.8426977  -0.643135
## 4775  0.8426977   1.554365
## 2177  0.8426977  -0.643135
## 1533  0.8426977   1.554365
## 4567  0.8426977  -0.643135
## 2347 -1.1862695   1.554365
```

```
tail(valid_norm)
```

```
##              Age  Experience      Income     Family      CCAvg Education.1
## 4984  0.49002772  0.51006352 -0.02366036 -1.2057601  0.5690728   1.1813978
## 4988  0.22617791  0.24655559 -0.65820152  0.5320637 -0.1240356  -0.8461728
## 4990 -1.88462051 -1.77367191 -0.76760517 -1.2057601 -0.5283488  -0.8461728
## 4994 -0.03767189  0.07088363  3.17092615 -0.3368482  2.7465881   1.1813978
## 4995  1.63337687  1.73976722  0.04198183  0.5320637  0.0492415  -0.8461728
## 4998  1.54542693  1.65193124 -1.07393538 -0.3368482 -0.9326620  -0.8461728
##      Education.2 Education.3   Mortgage Securities.Account CD.Account
```

```
## 4984  -0.6312436  -0.6509102 -0.5679457         -0.3338946 -0.2380992
## 4988   1.5836463  -0.6509102  1.0069117         -0.3338946 -0.2380992
## 4990  -0.6312436   1.5357982 -0.5679457         -0.3338946 -0.2380992
## 4994  -0.6312436  -0.6509102 -0.5679457         -0.3338946 -0.2380992
## 4995  -0.6312436   1.5357982 -0.5679457         -0.3338946 -0.2380992
## 4998  -0.6312436   1.5357982 -0.5679457         -0.3338946 -0.2380992
##          Online CreditCard
## 4984 -1.1862695  -0.643135
## 4988  0.8426977  -0.643135
## 4990  0.8426977  -0.643135
## 4994  0.8426977  -0.643135
## 4995  0.8426977  -0.643135
## 4998 -1.1862695  -0.643135
```

Perform k-NN classification with k=1 for the new customer

```r
# Perform k-NN classification with k=1 for the new customer
knn_pred_new_customer <- knn(train = train_norm, test = new_customer, cl = train_df$Personal.Loan, k = 
knn_pred_new_customer
```

```
## [1] 1
## Levels: 0 1
```

what is a choice of k that balances between overfitting and ignoring the predictor information?

```r
accuracy <- rep(0, 15)
for (i in 1:15) {
  knn_pred <- knn(train = train_norm, test = valid_norm, cl = train_df$Personal.Loan, k = i)
  accuracy[i] <- confusionMatrix(knn_pred, as.factor(valid_df$Personal.Loan), positive = "1")$overall[1
}
best_k <- which.max(accuracy)
best_k
```

```
## [1] 3
```

Validation Confusion Matrix

Show the confusion matrix for the validation data that results from using the best k.

```r
knn_pred_valid_best_k <- knn(train = train_norm, test = valid_norm, cl = train_df$Personal.Loan, k = bes
conf_matrix_valid <- confusionMatrix(knn_pred_valid_best_k, as.factor(valid_df$Personal.Loan), positive
```

Repartition the data into training, validation, and test sets (50% : 30% : 20%)

```r
train_indices <- sample(1:nrow(Bank_data), 0.5 * nrow(Bank_data))
valid_test_indices <- setdiff(1:nrow(Bank_data), train_indices)
valid_indices <- sample(valid_test_indices, 0.3 * length(valid_test_indices))
test_indices <- setdiff(valid_test_indices, valid_indices)

train_df <- Bank_data[train_indices, ]
valid_df <- Bank_data[valid_indices, ]
test_df <- Bank_data[test_indices, ]
```

Normalize the data for each set

```r
norm_values <- preProcess(train_df[, -which(names(train_df) %in% c("Personal.Loan"))], method = c("cent
train_norm <- predict(norm_values, train_df[, -which(names(train_df) %in% c("Personal.Loan"))])
valid_norm <- predict(norm_values, valid_df[, -which(names(valid_df) %in% c("Personal.Loan"))])
test_norm <- predict(norm_values, test_df[, -which(names(test_df) %in% c("Personal.Loan"))])
```

Perform k-NN classification with the best k for the test set

```r
knn_pred_test_best_k <- knn(train = train_norm, test = test_norm, cl = train_df$Personal.Loan, k = best_
knn_pred_test_best_k
```

```
##    [1] 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0
##   [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
##   [75] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
##  [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [186] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
##  [223] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [260] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
##  [297] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [334] 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
##  [371] 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [408] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
##  [445] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [482] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0
##  [519] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
##  [556] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0
##  [593] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [630] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
##  [667] 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [704] 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
##  [741] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
##  [778] 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [815] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
##  [852] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0
##  [889] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
##  [926] 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0
##  [963] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
```

```
## [1000] 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
## [1037] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
## [1074] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1111] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1148] 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1185] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
## [1222] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
## [1259] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0
## [1296] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1333] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0
## [1370] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0
## [1407] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1444] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
## [1481] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0
## [1518] 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1555] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1592] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0
## [1629] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0
## [1666] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1703] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1740] 0 0 0 0 0 0 0 0 0 0 0
## Levels: 0 1
```

Create confusion matrices for each set

```r
conf_matrix_train <- confusionMatrix(knn(train = train_norm, test = train_norm, cl = train_df$Personal.L
conf_matrix_valid <- confusionMatrix(knn(train = train_norm, test = valid_norm, cl = train_df$Personal.L
conf_matrix_test <- confusionMatrix(knn_pred_test_best_k, as.factor(test_df$Personal.Loan), positive =
```

Display the confusion matrices

```r
conf_matrix_train
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2243   46
##          1    8  203
##
##                Accuracy : 0.9784
##                  95% CI : (0.9719, 0.9837)
##     No Information Rate : 0.9004
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8708
##
##  Mcnemar's Test P-Value : 4.777e-07
##
##             Sensitivity : 0.8153
```

```
##             Specificity : 0.9964
##          Pos Pred Value : 0.9621
##          Neg Pred Value : 0.9799
##              Prevalence : 0.0996
##          Detection Rate : 0.0812
##    Detection Prevalence : 0.0844
##       Balanced Accuracy : 0.9059
##
##        'Positive' Class : 1
##
```

```
conf_matrix_valid
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 677  19
##          1   8  46
##
##                Accuracy : 0.964
##                  95% CI : (0.9481, 0.9761)
##     No Information Rate : 0.9133
##     P-Value [Acc > NIR] : 2.827e-08
##
##                   Kappa : 0.7537
##
##  Mcnemar's Test P-Value : 0.05429
##
##             Sensitivity : 0.70769
##             Specificity : 0.98832
##          Pos Pred Value : 0.85185
##          Neg Pred Value : 0.97270
##              Prevalence : 0.08667
##          Detection Rate : 0.06133
##    Detection Prevalence : 0.07200
##       Balanced Accuracy : 0.84801
##
##        'Positive' Class : 1
##
```

```
conf_matrix_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1570   55
##          1   14  111
##
##                Accuracy : 0.9606
##                  95% CI : (0.9504, 0.9692)
##     No Information Rate : 0.9051
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7418
##
##  Mcnemar's Test P-Value : 1.469e-06
##
##             Sensitivity : 0.66867
##             Specificity : 0.99116
##          Pos Pred Value : 0.88800
##          Neg Pred Value : 0.96615
##              Prevalence : 0.09486
##          Detection Rate : 0.06343
##    Detection Prevalence : 0.07143
##       Balanced Accuracy : 0.82992
##
##        'Positive' Class : 1
##
```

``‘