FML2

HARI VINAYAK

2023-10-01

Questions - Answers

1. How would this customer be classified? This new customer would be classified as 0, does not take the personal loan
2. The best K is 3

Problem Statement

Universal bank is a young bank growing rapidly in terms of overall customer acquisition. The majority of these customers are liability customers (depositors) with varying sizes of relationship with the bank. The customer base of asset customers (borrowers) is quite small, and the bank is interested in expanding this base rapidly in more loan business. In particular, it wants to explore ways of converting its liability customers to personal loan customers.

A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. This has encouraged the retail marketing department to devise smarter campaigns with better target marketing. The goal is to use k-NN to predict whether a new customer will accept a loan offer. This will serve as the basis for the design of a new campaign.

The file UniversalBank.csv contains data on 5000 customers. The data include 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.

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

---

Data Import and Cleaning

First, load the required libraries

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

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

```
library(e1071)
```

Read the data.

```
Uni.df <- read.csv("UniversalBank.csv")
dim(Uni.df)
```

```
## [1] 5000   14
```

```
t(t(names(Uni.df))) # The t function creates a transpose of the dataframe
```

```
##       [,1]
##  [1,] "ID"
##  [2,] "Age"
##  [3,] "Experience"
##  [4,] "Income"
##  [5,] "ZIP.Code"
##  [6,] "Family"
##  [7,] "CCAvg"
##  [8,] "Education"
##  [9,] "Mortgage"
## [10,] "Personal.Loan"
## [11,] "Securities.Account"
## [12,] "CD.Account"
## [13,] "Online"
## [14,] "CreditCard"
```

Drop ID and ZIP

```
Uni.df <- Uni.df[,-c(1,5)]
```

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
Uni.df$Education <- as.factor(Uni.df$Education)

# Now, convert Education to Dummy Variables

G <- dummyVars(~., data = Uni.df) # This creates the dummy G
U.m.df <- as.data.frame(predict(G,Uni.df))


set.seed(1)  # Important to ensure that we get the same sample if we rerun the code
train.index <- sample(row.names(U.m.df), 0.6*dim(U.m.df)[1])
valid.index <- setdiff(row.names(U.m.df), train.index)
train.df <- U.m.df[train.index,]
valid.df <- U.m.df[valid.index,]
```

Now, let us normalize the data

```
TN.df <- train.df[,-10] # Note that Personal Income is the 10th variable
VN.df <- valid.df[,-10]

norm.values <- preProcess(train.df[, -10], method=c("center", "scale"))
TN.df <- predict(norm.values, train.df[, -10])
VN.df <- predict(norm.values, valid.df[, -10])
```

Questions

Consider the following customer:

1. 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, and Credit Card = 1. Perform a k-NN classification with all predictors except ID and ZIP code using k = 1. Remember to transform categorical predictors with more than two categories into dummy variables first. Specify the success class as 1 (loan acceptance), and use the default cutoff value of 0.5. How would this customer be classified?

```
# We have converted all categorical variables to dummy variables
# Let's create a new sample
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,
  CreditCard = 1
)

# Normalize the new customer
new.cust.norm <- new_customer
new.cust.norm <- predict(norm.values, new.cust.norm)
```

Now, let us predict using knn

```
knn.pred1 <- class::knn(train = TN.df,
                        test = new.cust.norm,
                        cl = train.df$Personal.Loan, k = 1)
knn.pred1
```

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

---

2. What is a choice of k that balances between overfitting and ignoring the predictor information?

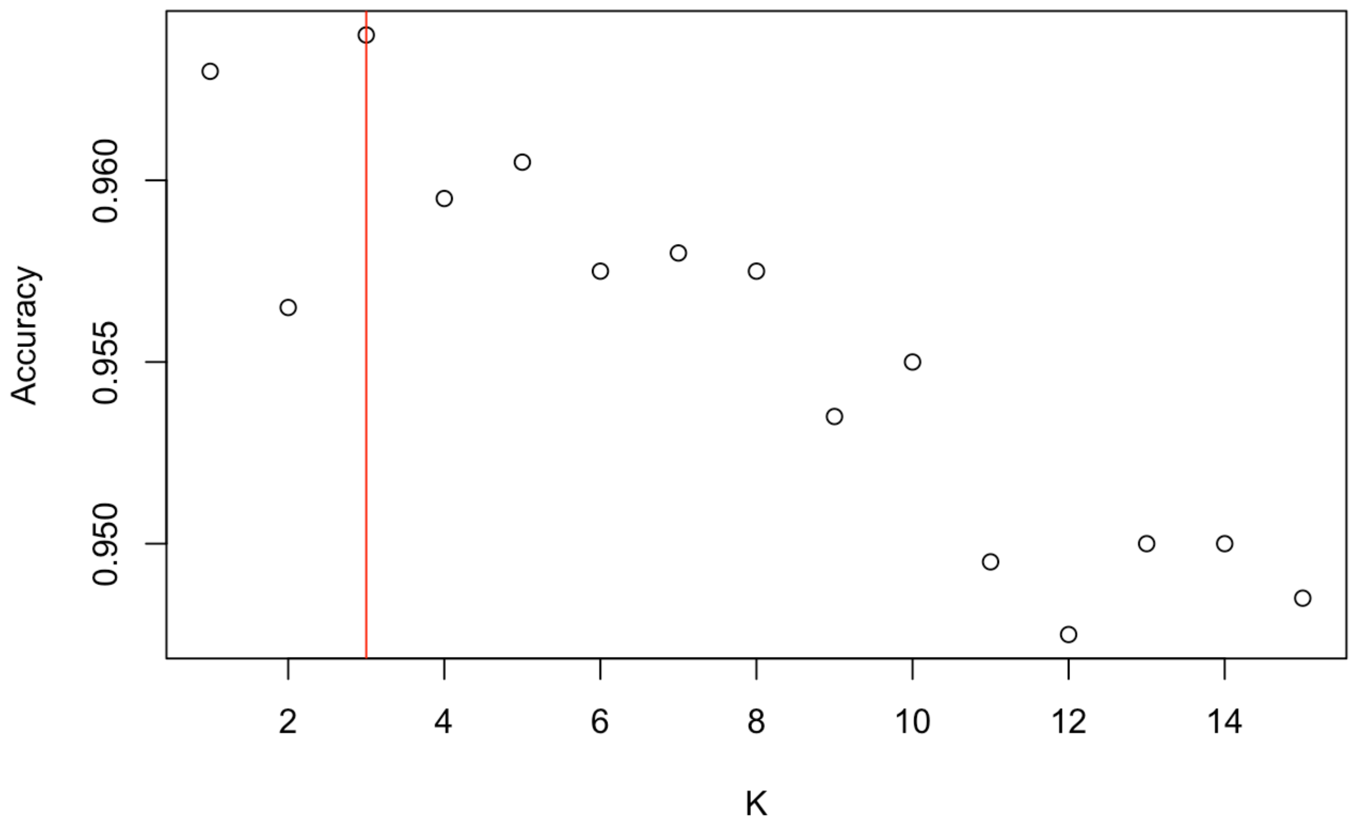
```
# Calculate the accuracy for each value of k
# Set the range of k values to consider

accuracy.df <- data.frame(k = seq(1, 15, 1), overallaccuracy = rep(0, 15))
for(i in 1:15) {
  knn.pred <- class::knn(train = TN.df,
                         test = VN.df,
                         cl = train.df$Personal.Loan, k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn.pred,
                                       as.factor(valid.df$Personal.Loan), positive =
"1")$overall[1]
}

best_k <- which.max(accuracy.df[,2])
accuracy.df[best_k,]
```

```
##   k overallaccuracy
## 3 3           0.964
```

```
plot(accuracy.df$k, accuracy.df$overallaccuracy, xlab = "K", ylab = "Accuracy")
abline(v = best_k, col = "red")
```



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

```
#Confusion Matrix using best K=3
knn.pred <- class::knn(train = TN.df,
                         test = VN.df,
                         cl = train.df$Personal.Loan, k = 3)

confusionMatrix(knn.pred,as.factor(valid.df$Personal.Loan))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1786   63
##          1    9  142
##
##                Accuracy : 0.964
##                  95% CI : (0.9549, 0.9717)
##     No Information Rate : 0.8975
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7785
##
##  Mcnemar's Test P-Value : 4.208e-10
##
##             Sensitivity : 0.9950
##             Specificity : 0.6927
##          Pos Pred Value : 0.9659
##          Neg Pred Value : 0.9404
##              Prevalence : 0.8975
##          Detection Rate : 0.8930
##    Detection Prevalence : 0.9245
##       Balanced Accuracy : 0.8438
##
##        'Positive' Class : 0
##
```

```
#Part 4 of Question
#New customer profile
new_customer2 <-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,
  CDAccount = 0,
  Online = 1,
  CreditCard = 1)
```

```
knn.pred1 <- class::knn(train = TN.df,
                       test = new_customer2,
                       cl = train.df$Personal.Loan, k = 3)
knn.pred1
```

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

```
#Print the predicted class (1 for loan acceptance, 0 for loan rejection)
print("This customer is classified as: Loan Rejected")
```

```
## [1] "This customer is classified as: Loan Rejected"
```

5,Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%). Apply the k-NN method with the k chosen above. Compare the confusion matrix of the test set with that of the training and validation sets. Comment on the differences and their reason.

```
# Set the seed
set.seed(123)

# Split the data into training, validation, and test sets
train_index <- sample(row.names(Uni.df), 0.5*dim(Uni.df)[1])
validation_index <- sample(setdiff(row.names(Uni.df), train_index), 0.3*dim(Uni.df)[1
])
test_index <- setdiff(row.names(Uni.df), union(train_index, validation_index))

# Create the training, validation, and test sets
train_data <- Uni.df[train_index, ]
validation_data <- Uni.df[validation_index, ]
test_data <- Uni.df[test_index, ]

# Normalize the training, validation, and test data
NV <- preProcess(train_data[, -10], method = c("center", "scale"))
train_data_normalized <- predict(NV, train_data[, -10])
validation_data_normalized <- predict(NV, validation_data[, -10])
test_data_normalized <- predict(NV, test_data[, -10])


# Calculated the confusion matrix for the training set
T_confu <- confusionMatrix(class::knn(train = train_data_normalized, test = train_dat
a_normalized, cl = train_data$Personal.Loan, k = 3), as.factor(train_data$Personal.Lo
an), positive = "1")

# Calculated the confusion matrix for the validation set
Va_confu <- confusionMatrix(class::knn(train = train_data_normalized, test = validati
on_data_normalized, cl = train_data$Personal.Loan, k = 3), as.factor(validation_data$
Personal.Loan), positive = "1")

# Calculated the confusion matrix for the test set
Te_confu <- confusionMatrix(class::knn(train = train_data_normalized, test = test_dat
a_normalized,cl = train_data$Personal.Loan, k = 3), as.factor(test_data$Personal.Loan
), positive = "1")


print(T_confu)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 2271    0
##          1    0  229
## 
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.9084
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 1
## 
##  Mcnemar's Test P-Value : NA
## 
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.0916
##          Detection Rate : 0.0916
##    Detection Prevalence : 0.0916
##       Balanced Accuracy : 1.0000
## 
##        'Positive' Class : 1
## 
```

```
print(Va_confu)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1357    0
##          1    0  143
##
##                Accuracy : 1
##                  95% CI : (0.9975, 1)
##     No Information Rate : 0.9047
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.00000
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 1.00000
##              Prevalence : 0.09533
##          Detection Rate : 0.09533
##    Detection Prevalence : 0.09533
##       Balanced Accuracy : 1.00000
##
##        'Positive' Class : 1
##
```

```
print(Te_confu)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 892   0
##          1   0 108
##
##                Accuracy : 1
##                  95% CI : (0.9963, 1)
##     No Information Rate : 0.892
##     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 : 0.108
##          Detection Rate : 0.108
##    Detection Prevalence : 0.108
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 1
##
```