64060_Assignment2

Durgaprasad Gandi

2023-09-29

Summary

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

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 Importing and Cleaning:

Load the required libraries

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

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

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

```r
library(e1071)
```

Read the dataset

```r
universal.dm <- read.csv("C:/Users/gdurg/Desktop/rhistory/UniversalBank.csv")
dim(universal.dm)
```

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

```r
t(t(names(universal.dm)))
```

```
##       [,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

```r
universal.dm <- universal.dm[,-c(1,5)]
```

Now split Data to 60% training and 40% validation.And then transform categorical variables into dummy variables

```r
universal.dm$Education <- as.factor(universal.dm$Education)

groups <- dummyVars(~., data = universal.dm) #  it will create dummy groups
universal_m.dm <- as.data.frame(predict(groups,universal.dm))


set.seed(1)
train.index <- sample(row.names(universal_m.dm), 0.6*dim(universal_m.dm)[1])
valid.index <- setdiff(row.names(universal_m.dm), train.index)
train.dm <- universal_m.dm[train.index,]
valid.dm <- universal_m.dm[valid.index,]
t(t(names(train.dm)))
```

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

```r
#Secondary approach

library(caTools)
set.seed(1)
split <- sample.split(universal_m.dm, SplitRatio = 0.6)
training_set <- subset(universal_m.dm, split == TRUE)
validation_set <- subset(universal_m.dm, split == FALSE)

# To print size of training set  and size of validation set:

print(paste("The size of the training set is:", nrow(training_set)))
```

```
## [1] "The size of the training set is: 2858"
```

```r
print(paste("The size of the validation set is:", nrow(validation_set)))
```

```
## [1] "The size of the validation set is: 2142"
```

Now, let us normalize the data

```r
train.norm.dm <- train.dm[,-10]
valid.norm.dm <- valid.dm[,-10]

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

Question

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?

```r
#  categorical variables to dummy variables Conversion completed
# 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)
```

prediction using knn

```r
knn.pred1 <- class::knn(train = train.norm.dm,
                       test = new.cust.norm,
                       cl = train.dm$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?

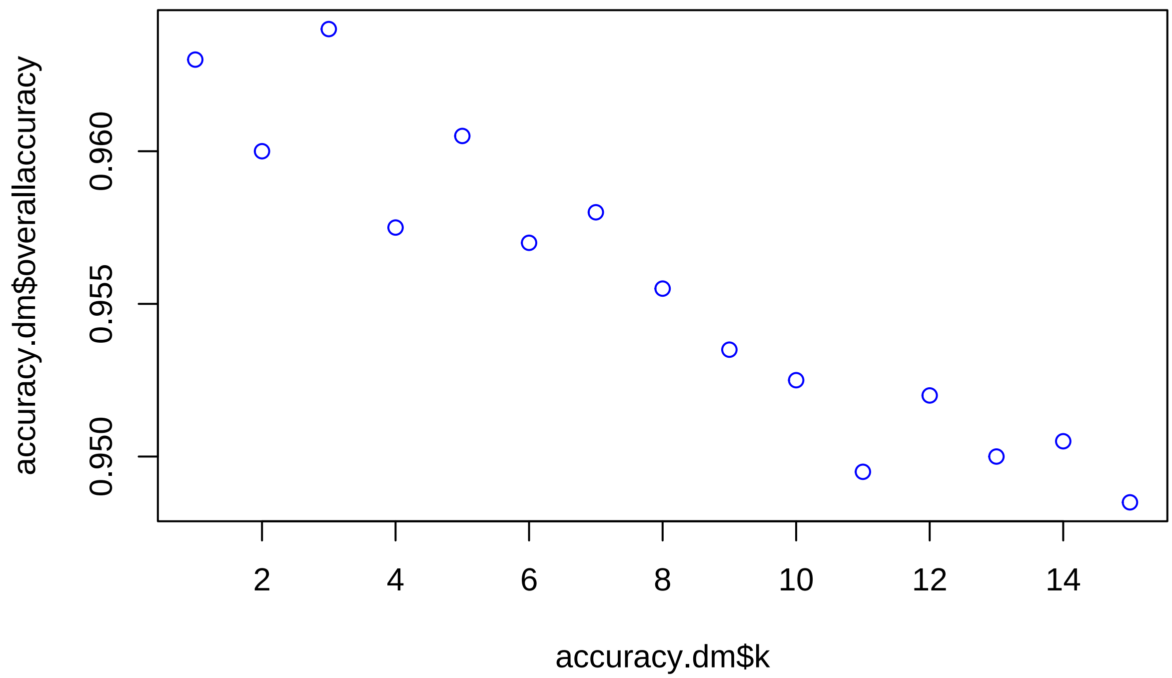
```r
# Calculate the accuracy for each value of k
# Set the range of k values to consider

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

which(accuracy.dm[,2] == max(accuracy.dm[,2]))
```

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

```r
plot(accuracy.dm$k,accuracy.dm$overallaccuracy,col = "blue")
```



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

```r
knn.pred_result <- class::knn(train = train.norm.dm,
test = valid.norm.dm,
cl = train.dm$Personal.Loan, k = 3)

# Creating the confusion matrix

confusion_matrix <- confusionMatrix(knn.pred_result,
                              as.factor(valid.dm$Personal.Loan),positive = "1")
confusion_matrix
```

```
## 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 : 0.6927
##             Specificity : 0.9950
##          Pos Pred Value : 0.9404
##          Neg Pred Value : 0.9659
##              Prevalence : 0.1025
##          Detection Rate : 0.0710
##    Detection Prevalence : 0.0755
##       Balanced Accuracy : 0.8438
##
##        'Positive' Class : 1
##
```

4. Consider the following customer: 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. Classify the customer using the best k.

```r
# To find the customer having best k value.
knn.pred2 <- class::knn(train = train.norm.dm,
test = new.cust.norm,
cl = train.dm$Personal.Loan, k = 3)
knn.pred2
```

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

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.

```r
set.seed(1)

training_set = sample(nrow(universal_m.dm), 0.5 * nrow(universal_m.dm))

validation_set = sample(setdiff(1:nrow(universal_m.dm), training_set),
                        0.3 * nrow(universal_m.dm))

test_set = setdiff(1:nrow(universal_m.dm), union(training_set, validation_set))

train.dm = universal_m.dm[training_set,]

valid.dm = universal_m.dm[validation_set,]

test.dm = universal_m.dm[test_set,]
```

```r
train.norm.dm = train.dm[,-10]

valid.norm.dm = valid.dm[,-10]

test.norm.dm = test.dm[,-10]

norm.values = preProcess(train.dm[, -10], method=c("center", "scale"))
# Z Normalize

train.norm.dm = predict(norm.values, train.norm.dm)

valid.norm.dm = predict(norm.values, valid.norm.dm)

test.norm.dm = predict(norm.values, test.norm.dm)

train_knn_pred = class::knn(train = train.norm.dm,
                            test = train.norm.dm,
                            cl = train.dm$Personal.Loan,
                            k = 3)

validation_knn_pred = class::knn(train = train.norm.dm,
                            test = valid.norm.dm,
                            cl = train.dm$Personal.Loan,
                            k = 3)

test_knn_pred = class::knn(train = train.norm.dm,
                      test = test.norm.dm,
                      cl = train.dm$Personal.Loan,
                      k = 3)

#Creating the confusion matrix for training set:
train_confusion_matrix = confusionMatrix(train_knn_pred,
                              as.factor(train.dm$Personal.Loan),positive = "1")

train_confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2263   54
##          1    5  178
##
##                Accuracy : 0.9764
##                  95% CI : (0.9697, 0.982)
##     No Information Rate : 0.9072
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8452
##
##  Mcnemar's Test P-Value : 4.129e-10
##
##             Sensitivity : 0.7672
```

```
##             Specificity : 0.9978
##          Pos Pred Value : 0.9727
##          Neg Pred Value : 0.9767
##              Prevalence : 0.0928
##          Detection Rate : 0.0712
##    Detection Prevalence : 0.0732
##       Balanced Accuracy : 0.8825
##
##        'Positive' Class : 1
##
```

```
#Creating the confusion matrix for validation set:


validation_confusion_matrix = confusionMatrix(validation_knn_pred,
                              as.factor(valid.dm$Personal.Loan),positive = "1")

validation_confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1358   42
##          1    6   94
##
##                Accuracy : 0.968
##                  95% CI : (0.9578, 0.9763)
##     No Information Rate : 0.9093
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7797
##
##  Mcnemar's Test P-Value : 4.376e-07
##
##             Sensitivity : 0.69118
##             Specificity : 0.99560
##          Pos Pred Value : 0.94000
##          Neg Pred Value : 0.97000
##              Prevalence : 0.09067
##          Detection Rate : 0.06267
##    Detection Prevalence : 0.06667
##       Balanced Accuracy : 0.84339
##
##        'Positive' Class : 1
##
```

```
#Creating the confusion matrix for test set:
testing_confusion_matrix = confusionMatrix(test_knn_pred,
                              as.factor(test.dm$Personal.Loan),positive = "1")

testing_confusion_matrix
```

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

```
##
##           Reference
## Prediction   0   1
##          0 884  35
##          1   4  77
##
##                Accuracy : 0.961
##                  95% CI : (0.9471, 0.9721)
##     No Information Rate : 0.888
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.777
##
##  Mcnemar's Test P-Value : 1.556e-06
##
##             Sensitivity : 0.6875
##             Specificity : 0.9955
##          Pos Pred Value : 0.9506
##          Neg Pred Value : 0.9619
##              Prevalence : 0.1120
##          Detection Rate : 0.0770
##    Detection Prevalence : 0.0810
##       Balanced Accuracy : 0.8415
##
##        'Positive' Class : 1
##
```

Differences and their reasons:

By comparing the above Confusion matrix for the validation and training with test set:

1.Accuracy : Accuracy for the Validation set is 0.968 and for training set is 0.9764 which are bit more than accuracy of test set 0.961.

2.Sensitivity: Sensitivity in the validation set is 0.7672 is slightly more than Training set 0.69118 which amounts to 7% approximately .Interestingly Validation's sensitivity has far more than test set sensitivity accounting for 0.6875.

3.Precision : Precision in the validation set is 0.94000,test set is 0.9506 So,training set has the greater precision 0.9727 than remaining sets.

4.Specificity: Specificity of Training,validation,Test set are 0.9978>0.99560>0.9955.

Reason

The reason in differences for the datset is because the Sizes of dataset are different for each sets. Some of misclassification also happens in datasets and performance aslo the causes for the differences in datasets.