

FML Assignment -2

Ajay Shadam

2023-10-21

summary

Problem statement:

Summary 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

```
#firstly install and load the pacakges "class","caret","e1071"
library(class)
library(caret)
```

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

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

##Reading the data

```
A_data <- read.csv("C:/Users/Ajay Reddy/Downloads/UniversalBank (1).csv")
dim(A_data)
```

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

```
head(A_data)
```

```
##   ID Age Experience Income ZIP.Code Family CCAvg Education Mortgage
## 1  1  25         1     49   91107      4   1.6         1         0
## 2  2  45        19     34   90089      3   1.5         1         0
## 3  3  39        15     11   94720      1   1.0         1         0
```

```
## 4 4 35      9    100    94112      1 2.7      2      0
## 5 5 35      8     45    91330      4 1.0      2      0
## 6 6 37     13     29    92121      4 0.4      2    155
##   Personal.Loan Securities.Account CD.Account Online CreditCard
## 1           0           1           0      0      0
## 2           0           1           0      0      0
## 3           0           0           0      0      0
## 4           0           0           0      0      0
## 5           0           0           0      0      1
## 6           0           0           0      1      0
```

```
tail(A_data)
```

```
##           ID Age Experience Income ZIP.Code Family CCAvg Education Mortgage
## 4995 4995 64      40      75    94588      3 2.0      3      0
## 4996 4996 29      3      40    92697      1 1.9      3      0
## 4997 4997 30      4      15    92037      4 0.4      1     85
## 4998 4998 63     39      24    93023      2 0.3      3      0
## 4999 4999 65     40      49    90034      3 0.5      2      0
## 5000 5000 28      4      83    92612      3 0.8      1      0
##   Personal.Loan Securities.Account CD.Account Online CreditCard
## 4995           0           0           0      1      0
## 4996           0           0           0      1      0
## 4997           0           0           0      1      0
## 4998           0           0           0      0      0
## 4999           0           0           0      1      0
## 5000           0           0           0      1      1
```

```
t(t(names(A_data))) ## The dataframe is transposed using the t function.
```

```
##      [,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"
```

```
getwd()
```

```
## [1] "C:/Users/Ajay Reddy/OneDrive/Desktop"
```

```
#Drop ID and ZIP
Anew_data <- A_data[,-c(1,5)]
dim(Anew_data)
```

```
## [1] 5000 12
```

#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

#changing the education attribute's int value to char

```
Anew_data$Education <- as.factor(Anew_data$Education)
```

#constructing the dummy variables for the attribute "education"

```
dums<- dummyVars(~.,data=Anew_data)
A_data <- as.data.frame(predict(dums,Anew_data))
```

#Setting the seed and partitioning the data into training (60%) and validation (40%) sets is necessary since we must run the function again.

```
set.seed(1)
trains_data <- sample(row.names(A_data), 0.6*dim(A_data)[1])
valida_data <- setdiff(row.names(A_data),trains_data)
train <- A_data[trains_data,]
valid <- A_data[valida_data,]
t(t(names(train)))
```

```
##      [,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"
```

```
summary(train)
```

```
##      Age      Experience      Income      Family
## Min.   :23.00  Min.   : -3.00  Min.    :  8.00  Min.    :1.000
## 1st Qu.:36.00  1st Qu.:10.00  1st Qu.: 39.00  1st Qu.:1.000
## Median :45.00  Median :20.00  Median : 63.00  Median :2.000
## Mean   :45.43  Mean   :20.19  Mean    : 73.08  Mean    :2.388
```

```
## 3rd Qu.:55.00 3rd Qu.:30.00 3rd Qu.: 98.00 3rd Qu.:3.000
## Max. :67.00 Max. :43.00 Max. :224.00 Max. :4.000
## CCAvg Education.1 Education.2 Education.3
## Min. : 0.000 Min. :0.0000 Min. :0.000 Min. :0.0000
## 1st Qu.: 0.700 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:0.0000
## Median : 1.500 Median :0.0000 Median :0.000 Median :0.0000
## Mean : 1.915 Mean :0.4173 Mean :0.285 Mean :0.2977
## 3rd Qu.: 2.500 3rd Qu.:1.0000 3rd Qu.:1.000 3rd Qu.:1.0000
## Max. :10.000 Max. :1.0000 Max. :1.000 Max. :1.0000
## Mortgage Personal.Loan Securities.Account CD.Account
## Min. : 0.00 Min. :0.00000 Min. :0.0000 Min. :0.00000
## 1st Qu.: 0.00 1st Qu.:0.00000 1st Qu.:0.0000 1st Qu.:0.00000
## Median : 0.00 Median :0.00000 Median :0.0000 Median :0.00000
## Mean : 57.34 Mean :0.09167 Mean :0.1003 Mean :0.05367
## 3rd Qu.:102.00 3rd Qu.:0.00000 3rd Qu.:0.0000 3rd Qu.:0.00000
## Max. :635.00 Max. :1.00000 Max. :1.0000 Max. :1.00000
## Online CreditCard
## Min. :0.0000 Min. :0.0000
## 1st Qu.:0.0000 1st Qu.:0.0000
## Median :1.0000 Median :0.0000
## Mean :0.5847 Mean :0.2927
## 3rd Qu.:1.0000 3rd Qu.:1.0000
## Max. :1.0000 Max. :1.0000
```

```
cat("The size of the training dataset is:",nrow(train))
```

```
## The size of the training dataset is: 3000
```

```
summary(valid)
```

```
## Age Experience Income Family
## Min. :23.0 Min. : -3.00 Min. : 8.00 Min. :1.000
## 1st Qu.:35.0 1st Qu.:10.00 1st Qu.: 39.00 1st Qu.:1.000
## Median :45.0 Median :20.00 Median : 64.00 Median :2.000
## Mean :45.2 Mean :19.97 Mean : 74.81 Mean :2.409
## 3rd Qu.:55.0 3rd Qu.:30.00 3rd Qu.: 99.00 3rd Qu.:3.000
## Max. :67.0 Max. :43.00 Max. :218.00 Max. :4.000
## CCAvg Education.1 Education.2 Education.3
## Min. : 0.000 Min. :0.000 Min. :0.000 Min. :0.000
## 1st Qu.: 0.700 1st Qu.:0.000 1st Qu.:0.000 1st Qu.:0.000
## Median : 1.600 Median :0.000 Median :0.000 Median :0.000
## Mean : 1.973 Mean :0.422 Mean :0.274 Mean :0.304
## 3rd Qu.: 2.600 3rd Qu.:1.000 3rd Qu.:1.000 3rd Qu.:1.000
## Max. :10.000 Max. :1.000 Max. :1.000 Max. :1.000
## Mortgage Personal.Loan Securities.Account CD.Account
## Min. : 0.00 Min. :0.0000 Min. :0.0000 Min. :0.0000
## 1st Qu.: 0.00 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000
## Median : 0.00 Median :0.0000 Median :0.0000 Median :0.0000
## Mean : 55.24 Mean :0.1025 Mean :0.1105 Mean :0.0705
## 3rd Qu.: 97.25 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:0.0000
## Max. :617.00 Max. :1.0000 Max. :1.0000 Max. :1.0000
## Online CreditCard
## Min. :0.000 Min. :0.000
```

```
## 1st Qu.:0.000 1st Qu.:0.000
## Median :1.000 Median :0.000
## Mean :0.615 Mean :0.296
## 3rd Qu.:1.000 3rd Qu.:1.000
## Max. :1.000 Max. :1.000
```

```
cat("The size of the validation dataset is:",nrow(valid))
```

```
## The size of the validation dataset is: 2000
```

```
##Now, let us normalize the data
```

```
trains_norm <- train[,-10]
valida_norm <- valid[,-10]
norm <- preProcess(train[,-10],method=c("center","scale"))
5
```

```
## [1] 5
```

```
trains_norm <- predict(norm,train[,-10])
valida_norm <- predict(norm,valid[,-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?

Creating new customer data

```
new_custo <- 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 dataset
cust.norm <- predict(norm, new_custo)
```

Performing kNN classification

```
prediction <- class::knn(train = trains_norm,
test = cust_norm,
cl = train$Personal.Loan, k = 1)
prediction
```

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

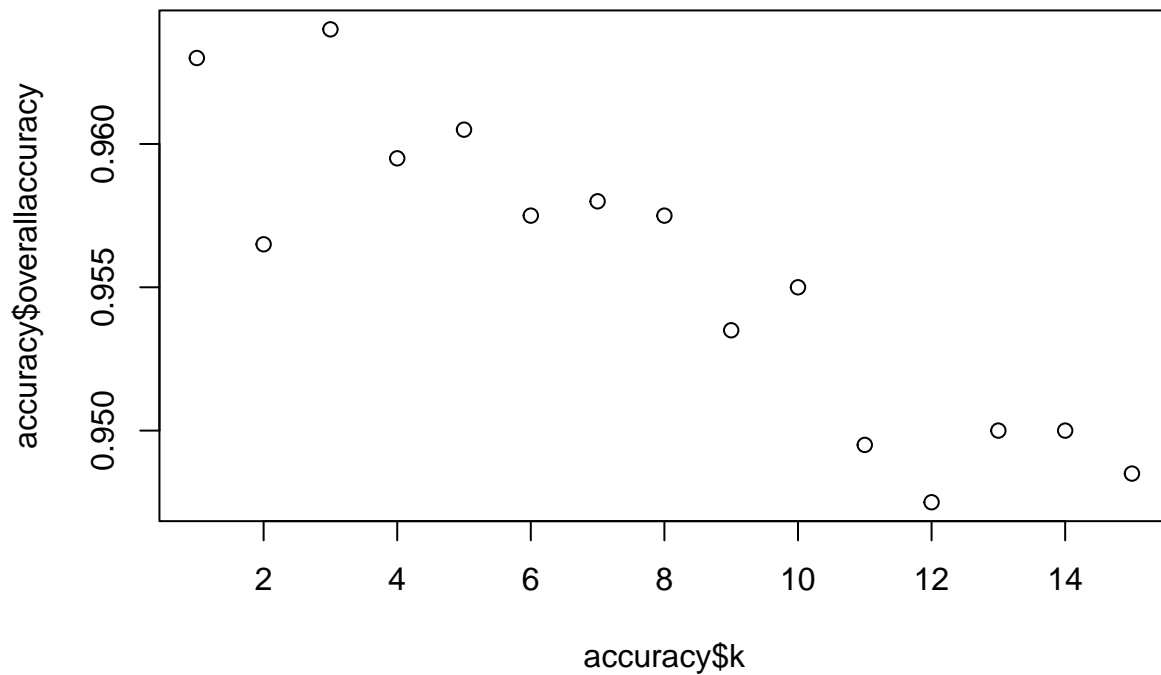
```
## [1] 3
```

```
accuracy
```

```
##      k overallaccuracy
## 1    1          0.9630
## 2    2          0.9565
## 3    3          0.9640
## 4    4          0.9595
## 5    5          0.9605
## 6    6          0.9575
## 7    7          0.9580
## 8    8          0.9575
## 9    9          0.9535
## 10  10          0.9550
## 11  11          0.9495
## 12  12          0.9475
## 13  13          0.9500
## 14  14          0.9500
## 15  15          0.9485
```

Among the k values from 1 to 15, 3 is the best performer. The most accurate for 3 is this k, which strikes a compromise between overfitting and disregarding forecasts.

```
plot(accuracy$k, accuracy$overallaccuracy)
```



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

confusion matrix

```
pred <- class::knn(train = trains_norm,
test = valida_norm,
cl = train$Personal.Loan, k=3)
confusionMatrix(pred,as.factor(valid$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
##
```

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 CreditCard = 1. Classify the customer using the best k.

#Make the same column names in a data frame for the client 2 .

```
custo2 <- 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)

#Normalizing the 2nd client dataset
cust2_norm <- predict(norm , custo2)
```

5. Repeating the process by partitioning the data into three parts - 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.seed(500)
Trains_Index <- sample(row.names(A_data), .5*dim(A_data)[1])#create train index
9
```

```
## [1] 9
```

```
#create validation index
Valid_Index <- sample(setdiff(row.names(A_data),Trains_Index),.3*dim(A_data)[1])
Test_Index =setdiff(row.names(A_data),union(Trains_Index,Valid_Index))#create test index
train.df <- A_data[Trains_Index,]
cat("The size of the new training dataset is:", nrow(train.df))
```

```
## The size of the new training dataset is: 2500
```



```
valid.df <- A_data[Valid_Index, ]
cat("The size of the new validation dataset is:", nrow(valid.df))
```

```
## The size of the new validation dataset is: 1500
```

```
test.df <- A_data[Test_Index, ]
cat("The size of the new test dataset is:", nrow(test.df))
```

```
## The size of the new test dataset is: 1000
```

Normalizing the data

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

Performing kNN and creating confusion matrix on training, testing, validation data

```
predic3 <- class::knn(train = train.df.norm,
test = test.df.norm,
cl = train.df$Personal.Loan, k=3)
confusionMatrix(predic3,as.factor(test.df$Personal.Loan))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0    1
##           0 898  36
##           1   6  60
##
##           Accuracy : 0.958
##           95% CI : (0.9436, 0.9696)
##           No Information Rate : 0.904
##           P-Value [Acc > NIR] : 9.200e-11
##
##           Kappa : 0.7187
##
##           McNemar's Test P-Value : 7.648e-06
##
##           Sensitivity : 0.9934
##           Specificity : 0.6250
##           Pos Pred Value : 0.9615
##           Neg Pred Value : 0.9091
##           Prevalence : 0.9040
##           Detection Rate : 0.8980
##           Detection Prevalence : 0.9340
##           Balanced Accuracy : 0.8092
##
##           'Positive' Class : 0
##
```

```

predic4 <- class::knn(train = train.df.norm,
test = valid.df.norm,
cl = train.df$Personal.Loan, k=3)
confusionMatrix(predic4,as.factor(valid.df$Personal.Loan))

```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 1332   65
##           1    8   95
##
##           Accuracy : 0.9513
##           95% CI : (0.9392, 0.9617)
##       No Information Rate : 0.8933
##       P-Value [Acc > NIR] : 6.496e-16
##
##           Kappa : 0.6971
##
##  McNemar's Test P-Value : 5.590e-11
##
##           Sensitivity : 0.9940
##           Specificity : 0.5938
##       Pos Pred Value : 0.9535
##       Neg Pred Value : 0.9223
##           Prevalence : 0.8933
##       Detection Rate : 0.8880
##       Detection Prevalence : 0.9313
##       Balanced Accuracy : 0.7939
##
##       'Positive' Class : 0
##

```

```

predic5 <- class::knn(train = train.df.norm,
test = train.df.norm,
cl = train.df$Personal.Loan, k=3)
confusionMatrix(predic5,as.factor(train.df$Personal.Loan))

```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 2273   53
##           1    3  171
##
##           Accuracy : 0.9776
##           95% CI : (0.971, 0.983)
##       No Information Rate : 0.9104
##       P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.8473
##

```

```
## McNemar's Test P-Value : 5.835e-11
##
##      Sensitivity : 0.9987
##      Specificity : 0.7634
##      Pos Pred Value : 0.9772
##      Neg Pred Value : 0.9828
##      Prevalence : 0.9104
##      Detection Rate : 0.9092
##      Detection Prevalence : 0.9304
##      Balanced Accuracy : 0.8810
##
##      'Positive' Class : 0
##
```