Assignment\_2

Ameer Abdlrasul

2023-10-01

Summary

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 baserapidly in more loan business. In particular, it wants to explore ways of converting its liability customers topersonal loan customers.

install “class”,“caret”,“e1071”

call the libraries “class”,“caret”,“e1071”

library(class)  
  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(e1071)

Read the bank csv file

dataset <- read.csv("~/Downloads/UniversalBank.csv", header=FALSE)  
head(dataset)

## V1 V2 V3 V4 V5 V6 V7 V8 V9  
## 1 ID Age Experience Income ZIP Code Family CCAvg Education Mortgage  
## 2 1 25 1 49 91107 4 1.60 1 0  
## 3 2 45 19 34 90089 3 1.50 1 0  
## 4 3 39 15 11 94720 1 1.00 1 0  
## 5 4 35 9 100 94112 1 2.70 2 0  
## 6 5 35 8 45 91330 4 1.00 2 0  
## V10 V11 V12 V13 V14  
## 1 Personal Loan Securities Account CD Account Online CreditCard  
## 2 0 1 0 0 0  
## 3 0 1 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 1

universal.df = read.csv("UniversalBank.csv")  
  
dim(universal.df)

## [1] 5000 14

t(t(names(universal.df)))

## [,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"

#Dropping ID and ZIP.Code Columns

universal.df = universal.df[,-c(1,5)]

#Converting Education Column to factor

universal.df$Education = as.factor(universal.df$Education)

#Converting Education to dummy variable

groups = dummyVars(~., data = universal.df) # This creates the dummy groups  
  
universal\_m.df = as.data.frame(predict(groups,universal.df))  
  
length(universal\_m.df)

## [1] 14

set.seed(1)

#Splitting Data into 60% for training and 40% for validation

train.index = sample(row.names(universal\_m.df), 0.6\*dim(universal\_m.df)[1])  
  
valid.index = setdiff(row.names(universal\_m.df), train.index)   
  
train.df = universal\_m.df[train.index,]  
  
valid.df = universal\_m.df[valid.index,]  
  
t(t(names(train.df)))

## [,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"

#Normalize data

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

# Questions 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?

#Now create new customer data based on above question\*

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  
)

#Normalizing new customer data

new.cust.norm = predict(norm.values, new\_customer)

#Now predict using KNN

knn.predict = class::knn(train = train.norm.df, test = new.cust.norm,   
 cl = train.df$Personal.Loan, k = 1)  
knn.predict

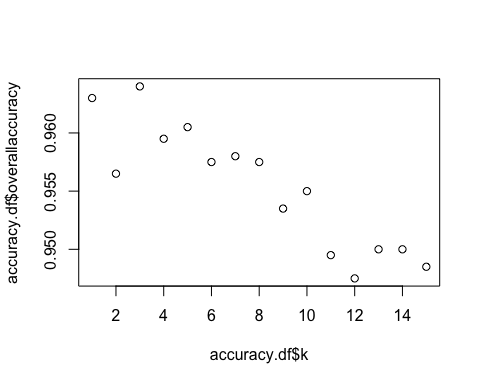
## [1] 0  
## Levels: 0 1

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

accuracy.df = data.frame(k = seq(1, 15, 1), overallaccuracy = rep(0, 15))  
for(i in 1:15) {  
 knn.pred = class::knn(train = train.norm.df,   
 test = valid.norm.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]  
}  
  
which(accuracy.df[,2] == max(accuracy.df[,2]))

## [1] 3

plot(accuracy.df$k,accuracy.df$overallaccuracy)



# Question 3. Show the confusion matrix for the validation data that results from using the best k.\*\*

best\_knn\_pred = class::knn(train = train.norm.df,   
 test = valid.norm.df,   
 cl = train.df$Personal.Loan, k = 3)

#Now create confusion matrix

confusion\_matrix = confusionMatrix(best\_knn\_pred,   
 as.factor(valid.df$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   
##

# Question 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.\*\*

knn.pred1 <- class::knn(train = train.norm.df,  
 test = new.cust.norm,  
 cl = train.df$Personal.Loan, k = 3)  
knn.pred1

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

# Question 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.\*\*

#To ensure we get sample if we re run the code

set.seed(1)

#Split the data

training\_set = sample(nrow(universal\_m.df), 0.5 \* nrow(universal\_m.df))  
  
validation\_set = sample(setdiff(1:nrow(universal\_m.df), training\_set), 0.3 \* nrow(universal\_m.df))  
  
test\_set = setdiff(1:nrow(universal\_m.df), union(training\_set, validation\_set))  
  
train.df = universal\_m.df[training\_set,]  
  
valid.df = universal\_m.df[validation\_set,]  
  
test.df = universal\_m.df[test\_set,]

#Normalize the data

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

#Predict using KNN

training\_knn\_pred = class::knn(train = train.norm.df,   
 test = train.norm.df,   
 cl = train.df$Personal.Loan,   
 k = 3)  
  
validation\_knn\_pred = class::knn(train = train.norm.df,   
 test = valid.norm.df,   
 cl = train.df$Personal.Loan,   
 k = 3)  
  
test\_knn\_pred = class::knn(train = train.norm.df,   
 test = test.norm.df,   
 cl = train.df$Personal.Loan,   
 k = 3)

#Confusion Matrix for Training set

training\_confusion\_matrix = confusionMatrix(training\_knn\_pred,   
 as.factor(train.df$Personal.Loan),   
 positive = "1")  
  
training\_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   
##

#Confusion Matrix for Validation set

validation\_confusion\_matrix = confusionMatrix(validation\_knn\_pred,   
 as.factor(valid.df$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   
##

#Confusion Matrix for Test set

test\_confusion\_matrix = confusionMatrix(test\_knn\_pred,   
 as.factor(test.df$Personal.Loan),   
 positive = "1")  
  
  
test\_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   
##