UNIVERSAL-BANK-PROBLEM-ASSIGNMENT-2-FML

ANURODH-SINGH

2023-10-05

****SUMMARY**** Summarizing it all according to the questions:

Q1. How would this customer be classified? Ans.This new customer is classified as 0 or non-potential customer as this customer will not take the loan

---

Q2.What is a choice of k that balances between overfitting and ignoring the predictor information? Ans.The best value of K is coming out to be 3, and the overall efficiency of 0.966

Q3.Show the confusion matrix for the validation data that results from using the best k. Ans.By using the best value of K as 3, and at set.seed(159) the confusion matrix was

```
      Reference
```

Prediction 0 1 0 1811 61 1 7 121

True positive = 121 True Negative = 1811 False Positive = 7 False Negative = 61

Q4.Classify the customer using the best k? Ans. Using Best value of K i.e K=3, the customer would be classified as 0. So, the customer does not take the personal loan.

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

```
install.packages("caret")
```

```
## Installing package into 'C:/Users/ASUS/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'caret' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'caret'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Users\ASUS\AppData\Local\R\win-library\4.3\00LOCK\caret\libs\x64\caret.dll
## to C:\Users\ASUS\AppData\Local\R\win-library\4.3\caret\libs\x64\caret.dll:
## Permission denied

## Warning: restored 'caret'

##
## The downloaded binary packages are in
##  C:\Users\ASUS\AppData\Local\Temp\RtmpI93Juc\downloaded_packages
```

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

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(e1071)
```

```
bank.universal <- read.csv("C:/Users/ASUS/Downloads/UniversalBank.csv")
dim(bank.universal)
```

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

```
bank.universal <- bank.universal[,-c(1,5)]
dim(bank.universal)
```

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

```
bank.universal$Education <- as.factor(bank.universal$Education)
union <- dummyVars(~.,data=bank.universal) #here I used 'dummyVars' function to create dummy variables
bank.universal <- as.data.frame(predict(union,bank.universal))
```

```r
set.seed(159)
train.index <- sample(row.names(bank.universal), 0.6*dim(bank.universal)[1]) # Now 60% training data is
train.bank <- bank.universal[train.index,]

valid.index <- setdiff(row.names(bank.universal), train.index) # 40% validation data
valid.bank <- bank.universal[valid.index,]

cat("Training  the data dimensions:", dim(train.bank), "\n")
```

```
## Training  the data dimensions: 3000 14
```

```r
cat("Validating the data dimensions:", dim(valid.bank), "\n")
```

```
## Validating the data dimensions: 2000 14
```

```r
train.normalize.bank <- train.bank[,-10] #Now we see that Personal loan is the 10th variable in data fr
valid.normalize.bank <- valid.bank[,-10]
norm.values <- preProcess(train.bank[, -10], method=c("center", "scale"))


train.normalize.bank <- predict(norm.values, train.bank[, -10])
valid.normalize.bank <- predict(norm.values, valid.bank[, -10])
```

Now as per the question 01: Consider the following customer:

Q1. 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
#Now updating a new customer(entry):


new_entry <- 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
)
```

```
new.entry.norm <- new_entry
new.entry.norm <- predict(norm.values, new.entry.norm)
```

Performing knn clasifucation with k=1 and Predicting the new customer class

```
#Now Assuming value of k=1
knn.prediction1 <- class::knn(train = train.normalize.bank,
                       test = new.entry.norm,
                       cl = train.bank$Personal.Loan, k = 1)

# Printing the knn prediction now
knn.prediction1
```

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

Q2. What is a choice of k that balances between overfitting and ignoring the predictor information?

Calculating the accuracy for each value of k

```
#Now Setting the range of k values to consider 1 to 15
accuracy.bank <- data.frame(k = seq(1, 20, 1), overallaccuracy = rep(0, 20))

for(i in 1:20) {
knn.prediction1 <- class::knn(train = train.normalize.bank,
                       test = valid.normalize.bank,
                       cl = train.bank$Personal.Loan, k = i)

accuracy.bank[i, 2] <- confusionMatrix(knn.prediction1, as.factor(valid.bank$Personal.Loan),
                                       positive = "1")$overall[1] #here we see that overall[1] gives th
}


bestValueofk <- which(accuracy.bank[,2] == max(accuracy.bank[,2])) #Here it gives the k value with maxi
accuracy.bank
```

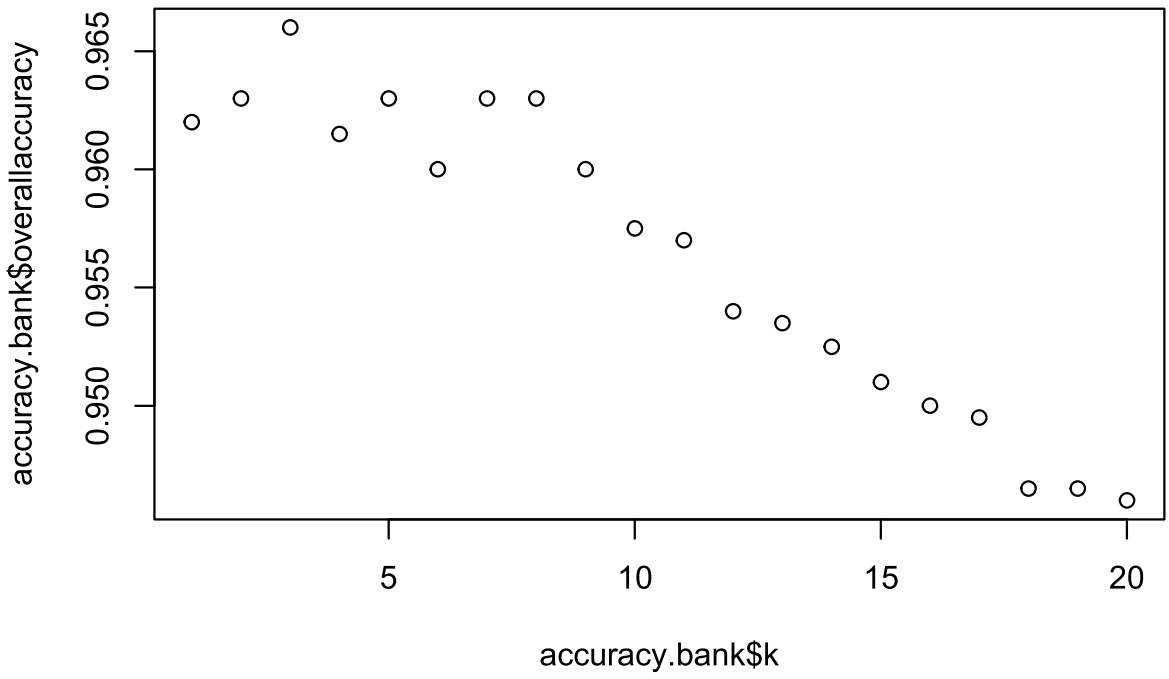
```
##     k overallaccuracy
## 1   1          0.9620
## 2   2          0.9630
## 3   3          0.9660
## 4   4          0.9615
## 5   5          0.9630
## 6   6          0.9600
```

```
## 7   7          0.9630
## 8   8          0.9630
## 9   9          0.9600
## 10 10          0.9575
## 11 11          0.9570
## 12 12          0.9540
## 13 13          0.9535
## 14 14          0.9525
## 15 15          0.9510
## 16 16          0.9500
## 17 17          0.9495
## 18 18          0.9465
## 19 19          0.9465
## 20 20          0.9460
```

```
#Now Printing the best value of k
cat("The Best Value of k is:", bestValueofk)
```

```
## The Best Value of k is: 3
```

```
#Now Plotting the graph between k values and accuracy
plot(accuracy.bank$k,accuracy.bank$overallaccuracy)
```



Q3. Show the confusion matrix for the validation data that results from using the best k.

#Now let's create the confusion matrix for the validation data for best value of k

```r
#taking the best value of k for prediction
knn.prediction2 <- class::knn(train = train.normalize.bank,
                        test = valid.normalize.bank,
                        cl = train.bank$Personal.Loan, k = bestValueofk)

#confusion matrix for data is now obtained below
confusion_matrix <- confusionMatrix(knn.prediction2,
                                    as.factor(valid.bank$Personal.Loan), positive = "1")

#printing the matrix now:
cat("Confusion Matrix for validation data:", "\n")
```

```
## Confusion Matrix for validation data:
```

```r
print(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1811   61
##          1    7  121
##
##                Accuracy : 0.966
##                  95% CI : (0.9571, 0.9735)
##     No Information Rate : 0.909
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7628
##
##  Mcnemar's Test P-Value : 1.3e-10
##
##             Sensitivity : 0.6648
##             Specificity : 0.9961
##          Pos Pred Value : 0.9453
##          Neg Pred Value : 0.9674
##              Prevalence : 0.0910
##          Detection Rate : 0.0605
##    Detection Prevalence : 0.0640
##       Balanced Accuracy : 0.8305
##
##        'Positive' Class : 1
##
```

Q4. 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.

#Now Creating the new dataset and NormaliZing the data

```r
# creating a new customer named: new_person2
new_person2 <- 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 new customer
new.person.norm2 <- new_person2
new.person.norm2 <- predict(norm.values, new.person.norm2)
```

Predicting using knn Algorithm

```
#taking best value of k, as it has maximum accuracy.
knn.prediction4 <- class::knn(train = train.normalize.bank,
                        test = new.person.norm2,
                        cl = train.bank$Personal.Loan, k = bestValueofk)

#printing  the prediction now
knn.prediction4
```

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

Q5. 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.

#Now splitting the already cleaned data again for training, validation and testing

```
set.seed(159)

#Now splitting the data into training (50%), validation (30%), and testing (20%) sets
train.index1 <- sample(row.names(bank.universal), 0.5*dim(bank.universal)[1])# 50% training data
valid.index1 <- sample(setdiff(row.names(bank.universal), train.index1),
                       0.3*dim(bank.universal)[1]) # 30% validation data
test.index1 <- setdiff(row.names(bank.universal), c(train.index1,valid.index1)) # 20% test data

trainData1 <- bank.universal[train.index1,]
validData1 <-bank.universal[valid.index1,]
testData1 <- bank.universal[test.index1,]

#now Printing the dimensions of the split datasets
cat("Training data dimensions:", dim(trainData1), "\n")
```

```
## Training data dimensions: 2500 14
```

```
cat("Validation data dimensions:", dim(validData1), "\n")
```

```
## Validation data dimensions: 1500 14
```

```
cat("Testing data dimensions:", dim(testData1), "\n")
```

```
## Testing data dimensions: 1000 14
```

```
#Normalizing the data for all the 3 sets
train.normalize.bank1 <- trainData1[ ,-10] #removing the 10th variable(personal loan)
valid.normalize.bank1 <- validData1[ ,-10]
test.normalize.bank1 <- testData1[ ,-10]

#Preprocessing of data
norm.values1 <- preProcess(trainData1[ ,-10], method=c("center", "scale"))
train.normalize.bank1 <- predict(norm.values1, trainData1[ ,-10])
valid.normalize.bank1 <- predict(norm.values1, validData1[ ,-10])
test.normalize.bank1 <- predict(norm.values1, testData1[ ,-10])
```

confusion matrix for the data at k=3 with training data

```
#knn prediction for validation data at best value of k
knn.pred.train <- class::knn(train = train.normalize.bank1,
                             test = train.normalize.bank1,
                             cl = trainData1$Personal.Loan, k = 3)

#confusion matrix for training data
confusion_matrix.train <- confusionMatrix(knn.pred.train,
                                          as.factor(trainData1$Personal.Loan), positive = "1")

#printing  the matrix now:
cat("Confusion Matrix for training data:", "\n")
```

```
## Confusion Matrix for training data:
```

```
print(confusion_matrix.train)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2253   58
##          1    6  183
##
##                Accuracy : 0.9744
##                  95% CI : (0.9674, 0.9802)
```

```
##     No Information Rate : 0.9036
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8374
##
##  Mcnemar's Test P-Value : 1.83e-10
##
##             Sensitivity : 0.7593
##             Specificity : 0.9973
##          Pos Pred Value : 0.9683
##          Neg Pred Value : 0.9749
##              Prevalence : 0.0964
##          Detection Rate : 0.0732
##    Detection Prevalence : 0.0756
##       Balanced Accuracy : 0.8783
##
##        'Positive' Class : 1
##
```

confusion matrix for the data at k=3 with validation data

```
#knn prediction for validation data at best value of k
knn.pred.valid <- class::knn(train = train.normalize.bank1,
                             test = valid.normalize.bank1,
                             cl = trainData1$Personal.Loan, k = bestValueofk)

#confusion matrix for validation data
confusion_matrix.valid <- confusionMatrix(knn.pred.valid,
                                          as.factor(validData1$Personal.Loan), positive = "1")

#printing the matrix
cat("Confusion Matrix for Validation data:", "\n")
```

```
## Confusion Matrix for Validation data:
```

```
print(confusion_matrix.valid)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1347   50
##          1    8   95
##
##                Accuracy : 0.9613
##                  95% CI : (0.9503, 0.9705)
##     No Information Rate : 0.9033
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7457
```

```
##
##  Mcnemar's Test P-Value : 7.303e-08
##
##             Sensitivity : 0.65517
##             Specificity : 0.99410
##          Pos Pred Value : 0.92233
##          Neg Pred Value : 0.96421
##              Prevalence : 0.09667
##          Detection Rate : 0.06333
##    Detection Prevalence : 0.06867
##       Balanced Accuracy : 0.82463
##
##        'Positive' Class : 1
##
```

#now creating the confusion matrix for the data at k=3 with test data

```
#knn prediction for test data at best value of k
knn.prediction.test <- class::knn(train = train.normalize.bank1,
                          test = test.normalize.bank1,
                          cl = trainData1$Personal.Loan, k = bestValueofk)

#obtaining the confusion matrix for test data
confusion_matrix.test <- confusionMatrix(knn.prediction.test,
                                         as.factor(testData1$Personal.Loan), positive = "1")

#printing  the matrix now
cat("Confusion Matrix for Test data:", "\n")
```

```
## Confusion Matrix for Test data:
```

```
print(confusion_matrix.test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 899  37
##          1   7  57
##
##                Accuracy : 0.956
##                  95% CI : (0.9414, 0.9679)
##     No Information Rate : 0.906
##     P-Value [Acc > NIR] : 1.733e-09
##
##                   Kappa : 0.6986
##
##  Mcnemar's Test P-Value : 1.232e-05
##
##             Sensitivity : 0.6064
##             Specificity : 0.9923
##          Pos Pred Value : 0.8906
##          Neg Pred Value : 0.9605
```

```
##              Prevalence : 0.0940
##          Detection Rate : 0.0570
##    Detection Prevalence : 0.0640
##       Balanced Accuracy : 0.7993
##
##        'Positive' Class : 1
##
```

Q5. 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. Ans. The training set has more accuracy(97.4%), sensitivity(75.93%) and specificity(99.7%) than test and validation data sets because of the several factors. Factors Contributing to Training Set superiority are:-

I. Overfitting: The primary reason behind the training set's superior performance is overfitting. Overfitting occurs when the model excessively adapts to the nuances and outliers present in the training data. It effectively memorizes the training data, including its noise and outliers.

II. Sample Size: The size of the training dataset might also contribute to this disparity. With a larger training dataset, the model has more instances to learn from, potentially allowing it to fit the data too closely.

III. Data Leakage: Data leakage, if present, can artificially boost the model's performance on the training data. Leakage occurs when information from the validation or test sets inadvertently influences the model's training process.

For Training data: Accuracy was 97.44% Sensitivity was 75.93% Specificity was 99.73%

For Validation data: Accuracy was 96.13% Sensitivity was 65.51% Specificity was 99.41%

For Training data: Accuracy was 95.60% Sensitivity was 60.64% Specificity was 99.23%
