FML_Assignment2

Bhargav

2023-09-15

Summary

Questions - Answers

1. How would this customer be classified? A. This new customer would be classified as 0, does not take the personal loan

****

2. What is a choice of k that balances between overfitting and ignoring the predictor information? A. The best value of K is 3, with the overall efficiency of 0.966.

****

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

****

4. Classify the customer using the best k? A. 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.

****

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. A. The training set has

more accuracy(97.4%), sensitivity(75.93%) and specificity(99.7%) than test and validation data sets. it was due to some factors such as over fitting, sample size,data leakage etc. the main reason was over fitting, the model is allowed to memorize the training data. so that it will captures all the out liners of training data. As a result the model will perform exceptionally high on training data compared to that of remaining two datas.

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%

****

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

Load the Required Libraries

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

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

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

```r
library(e1071)
```

data import which was in .csv format

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

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

```r
t(t(names(universal.bank)))# 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 the ID and ZIP as mentioned in the question

```r
universal.bank <- universal.bank[,-c(1,5)]# 1 and 5 are the indexes for columns ID and ZIP
dim(universal.bank)
```

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

transforming categorical variables into dummy variables

```r
# only education need to be converted to factor
universal.bank$Education <- as.factor(universal.bank$Education)

# converting education levels to dummy variables
groups <- dummyVars(~.,data=universal.bank) # 'dummyVars' function creates dummy variables for factors
universal.B.bank <- as.data.frame(predict(groups,universal.bank))
```

Splitting the Data into 60% training and 40% validation.

```r
set.seed(159) # Important to ensure that we get the same sample if we rerun the code

train.index <- sample(row.names(universal.B.bank), 0.6*dim(universal.B.bank)[1]) # 60% training data
train.bank <- universal.B.bank[train.index,]

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

# Print the dimensions of the split datasets
cat("Training data dimensions:", dim(train.bank), "\n")
```

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

```
cat("Validation data dimensions:", dim(valid.bank), "\n")
```

```
## Validation data dimensions: 2000 14
```

Normalizing the training and validation data

```
train.norm.bank <- train.bank[,-10] # Personal loan is the 10th variable in data frame
valid.norm.bank <- valid.bank[,-10]

#Preprocessing of data
norm.values <- preProcess(train.bank[, -10], method=c("center", "scale"))

#Normalization of training and validation data
train.norm.bank <- predict(norm.values, train.bank[, -10])
valid.norm.bank <- predict(norm.values, valid.bank[, -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 the dataset and NormaliZing the data

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

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

```
# Assuming value of k=1
knn.pred1 <- class::knn(train = train.norm.bank,
                        test = new.cust.norm,
                        cl = train.bank$Personal.Loan, k = 1)

# Print the knn prediction
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

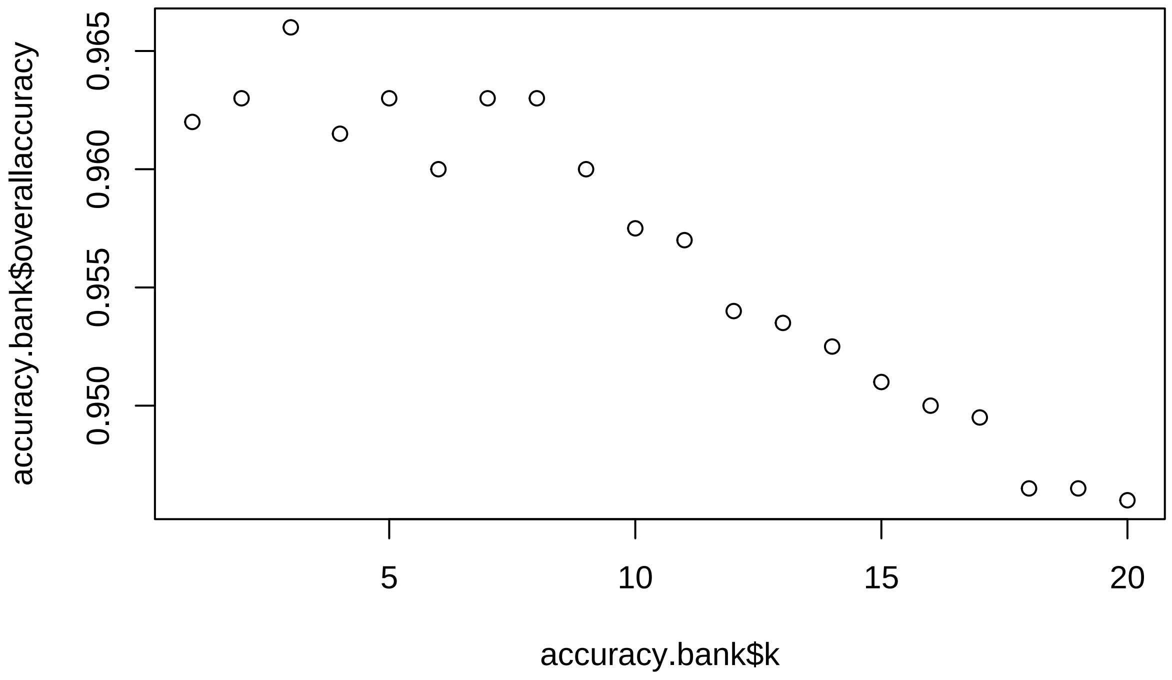
```
# Set the range of k values to consider 1 to 15
accuracy.bank <- data.frame(k = seq(1, 20, 1), overallaccuracy = rep(0, 20))

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

accuracy.bank[i, 2] <- confusionMatrix(knn.pred, as.factor(valid.bank$Personal.Loan),
                                        positive = "1")$overall[1] # overall[1] gives the accuracy
}


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

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

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

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



***

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

Creating the confusion matrix for the validation data for best value of k

```
#taking best value of k for prediction
knn.pred2 <- class::knn(train = train.norm.bank,
                        test = valid.norm.bank,
                        cl = train.bank$Personal.Loan, k = bestValueofk)

#confusion matrix for data
confusion_matrix <- confusionMatrix(knn.pred2,
                                    as.factor(valid.bank$Personal.Loan), positive = "1")

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

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

```
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
##
```

***

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.

Creating the new dataset and NormaliZing the data

```
# creating a new customer1
new_customer1 <- 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.cust.norm1 <- new_customer1
new.cust.norm1 <- predict(norm.values, new.cust.norm1)
```

Predict using knn Algorithm

```
#taking best value of k, as it has maximum accuracy.
knn.pred3 <- class::knn(train = train.norm.bank,
                        test = new.cust.norm1,
                        cl = train.bank$Personal.Loan, k = bestValueofk)

#print the prediction
knn.pred3
```

```
## [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.

split the already cleaned data again to training, validation and testing

```
set.seed(159) # Important to ensure that we get the same sample if we rerun the code

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

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

# Print 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
```

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

#Preprocessing of data
norm.values1 <- preProcess(trainData1[ ,-10], method=c("center", "scale"))
train.norm.bank1 <- predict(norm.values1, trainData1[ ,-10])
valid.norm.bank1 <- predict(norm.values1, validData1[ ,-10])
test.norm.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.norm.bank1,
                             test = train.norm.bank1,
                             cl = trainData1$Personal.Loan, k = 3)

#confusion matrix for training data
```

```r
confusion_matrix.train <- confusionMatrix(knn.pred.train,
                                          as.factor(trainData1$Personal.Loan), positive = "1")

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

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

```r
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

```r
#knn prediction for validation data at best value of k
knn.pred.valid <- class::knn(train = train.norm.bank1,
                             test = valid.norm.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")

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

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

```r
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
##
```

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

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

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

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

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

```r
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
##
```