Assignment_2

Ruthvick Bulagakula

2023-09-14

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 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

Questions and Answers

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?

Answer: 0(Customer Loan Rejected)

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

Answer: K = 3

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

Answer:

```
matrix(c(1769, 178, 50, 3), nrow = 2, byrow = TRUE,
                            dimnames = list(Prediction = c(0, 1), Reference = c(0, 1)))
```

```
##            Reference
## Prediction    0   1
##          0 1769 178
##          1   50   3
```

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.

Answer: 0(Customer Loan Rejected)

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.

Answer:

Training Set Confusion Matrix:

```
matrix(c(2263, 54, 5, 178), nrow = 2, byrow = TRUE,
                            dimnames = list(Prediction = c(0, 1), Reference = c(0, 1)))
```

```
##            Reference
## Prediction    0   1
##          0 2263  54
##          1    5 178
```

Validation Set Confusion Matrix:

```
matrix(c(1358, 42 , 6 , 94), nrow = 2, byrow = TRUE,
                            dimnames = list(Prediction = c(0, 1), Reference = c(0, 1)))
```

```
##            Reference
## Prediction    0  1
##          0 1358 42
##          1    6 94
```

Testing Set Confusion Matrix:

```
matrix(c(884, 35, 4, 77), nrow = 2, byrow = TRUE,
                            dimnames = list(Prediction = c(0, 1), Reference = c(0, 1)))
```

```
##            Reference
## Prediction   0  1
##          0 884 35
##          1   4 77
```

Differences

Test VS Train

Accuracy: Accuracy of Training Confusion matrix(0.9764) is slightly higher than Test Confusion matrix(0.961).

Sensitivity(True Positive Rate): Training Confusion Matrix has a higher sensitivity (0.7672) compared to Test Confusion Matrix (0.6875).

Specificity(True Negative Rate): Both matrices have similar specificity values and they are both very high. Training Confusion Matrix has a slightly higher specificity (0.9978) compared to Test Confusion Matrix (0.9955).

Precision: The precision in Training Confusion Matrix (0.9727) has a higher value that Test Confusion Matrix (0.9506).

Test VS Validation

Accuracy: Accuracy of Validation Confusion matrix(0.968) is slightly higher than Test Confusion matrix(0.961).

Sensitivity(True Positive Rate): Both matrices have similar sensitivity values but Validation Confusion Matrix has a slightly higher sensitivity (0.69118) compared to Test Confusion Matrix (0.6875).

Specificity(True Negative Rate): Both matrices have similar specificity values and they are both very high. Validation Confusion Matrix has a slightly higher specificity (0.99560) compared to Test Confusion Matrix (0.9955).

Precision: The precision in Test Confusion Matrix (0.9506) is higher Validation Confusion Matrix (0.94000).

Reasons

Dataset: Since, we used different dataset size for Training, validation and test.Also, higher values in confusion matrix for Training data is because we used Same dataset in test argument for Train KNN.

Code

Data Import and Cleaning

Calling pre-installed libraries.

```
library(class)

library(caret)
```

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

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

```
library(e1071)
```

Reading data using read.csv

```r
data.df = read.csv("UniversalBank.csv")

dim(data.df)
```

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

```r
t(t(names(data.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

```r
data.df = data.df[,-c(1,5)]
```

Converting Education Column to factor

```r
data.df$Education = as.factor(data.df$Education)
```

Converting Education to dummy variable

```r
groups = dummyVars(~., data = data.df) # This creates the dummy groups

data_m.df = as.data.frame(predict(groups,data.df))

length(data_m.df)
```

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

To ensure we get sample if we re run the code

```r
set.seed(1)
```

Now splitting Data into 60% for training and 40% for validation.

```r
train.split = sample(row.names(data_m.df), 0.6*dim(data_m.df)[1])

valid.split = setdiff(row.names(data_m.df), train.split)

train.df = data_m.df[train.split,]

valid.df = data_m.df[valid.split,]

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"
```

Now normalize data

```r
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
```

---

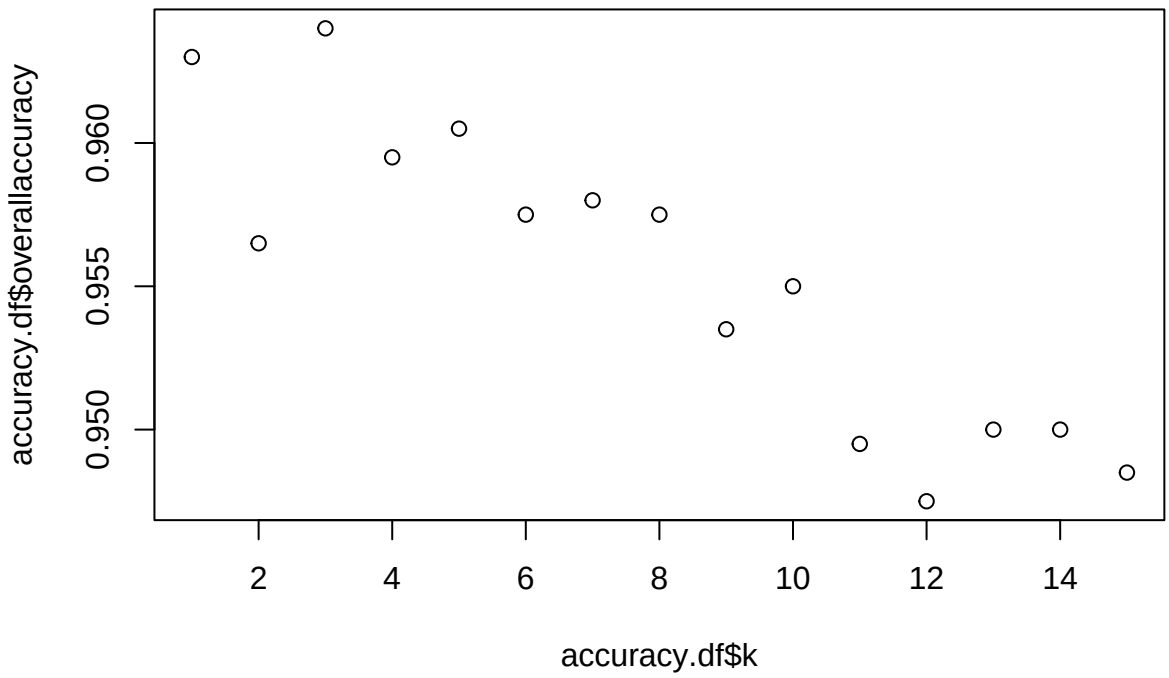
2. What is a choice of k that balances between overfitting and ignoring the predictor information?

```
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)
```



***

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
##
```

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.

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

new_customer_knn_pred
```

```
## [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.

To ensure we get sample if we re run the code

```
set.seed(1)
```

Now split the data

```
training_set = sample(nrow(data_m.df), 0.5 * nrow(data_m.df))

validation_set = sample(setdiff(1:nrow(data_m.df), training_set), 0.3 * nrow(data_m.df))

test_set = setdiff(1:nrow(data_m.df), union(training_set, validation_set))

train.df = data_m.df[training_set,]

valid.df = data_m.df[validation_set,]

test.df = data_m.df[test_set,]
```

Now 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)
```

Now 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

```r
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
##
```

Differences and Their Reasons for Validation and Test Confusion Matrix

Differences

Test VS Train

Accuracy: Accuracy of Training Confusion matrix(0.9764) is slightly higher than Test Confusion matrix(0.961).

Sensitivity(True Positive Rate): Training Confusion Matrix has a higher sensitivity (0.7672) compared to Test Confusion Matrix (0.6875).

Specificity(True Negative Rate): Both matrices have similar specificity values and they are both very high. Training Confusion Matrix has a slightly higher specificity (0.9978) compared to Test Confusion Matrix (0.9955).

Precision: The precision in Training Confusion Matrix (0.9727) has a higher value that Test Confusion Matrix (0.9506).

Test VS Validation

Accuracy: Accuracy of Validation Confusion matrix(0.968) is slightly higher than Test Confusion matrix(0.961).

Sensitivity(True Positive Rate): Both matrices have similar sensitivity values but Validation Confusion Matrix has a slightly higher sensitivity (0.69118) compared to Test Confusion Matrix (0.6875).

Specificity(True Negative Rate): Both matrices have similar specificity values and they are both very high. Validation Confusion Matrix has a slightly higher specificity (0.99560) compared to Test Confusion Matrix (0.9955).

Precision: The precision in Test Confusion Matrix (0.9506) is higher Validation Confusion Matrix (0.94000).

Reasons

Dataset: Since, we used different dataset size for Training, validation and test.Also, higher values in confusion matrix for Training data is because we used Same dataset in test argument for Train KNN.