64060_Assignment 2

Niharika Matsa

2023-10-02

Summary

Questions - 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

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 :

```r
matrix(c(1786,63,9,142), ncol = 2 , byrow = TRUE, dimnames = list(prediction=c(0,1), Reference = c(0,1)
```

```
##            Reference
## prediction    0   1
##           0 1786  63
##           1    9 142
```

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

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:

comparing Test with Train:

Accuracy: Train has a higher accuracy (0.9772) compared to Test (0.9507).

Sensitivity (True Positive Rate):Train has higher sensitivity (0.7589) compared to Test (0.5875).

Specificity (True Negative Rate): Train has higher specificity (0.9987) compared to Test (0.99403).

Positive Predictive Value (Precision): Train has a higher positive predictive value (0.9827) compared to Test (0.92157).

comparing Train with Validation:

Accuracy: Train has a higher accuracy (0.9772) compared to Validation (0.958).

Sensitivity (True Positive Rate): Train has higher sensitivity (0.7589) compared to Validation (0.625).

Specificity (True Negative Rate): Train has higher specificity (0.9987) compared to Validation (0.9934).

Positive Predictive Value (Precision): Train still has a higher positive predictive value (0.9827) compared to Validation (0.9091).

comapring test with validation:

Accuracy : validation has a higher accuracy (0.986) than the test(0.961).

sensitivity: validation has a higher sensitivity(0.69118) than the test(0.6875)

Specificity : validation has a higher specificity(0.99560) than the test(0.9955)

positive predictive rate: test has higher positive predictive value(0.9506) than validation(0.94000)

Potential Reasons for Differences:

Data set Differences:

Variations in the composition and distribution of data between different sets can significantly impact model performance. For illustration, one data set may be more imbalanced, making it harder to predict rare events.

Sample Variability:

Performance requirements may be impacted in small data sets due to changes in the specific samples used for the confirmation and test sets.

Randomness:

There's an inherent randomness, especially when using techniques like cross-validation. Different random splits or initializations may lead to variations in performance on different datasets. Some models, similar as neural networks, involve randomness in their optimization process, leading to slight variations.

Hyper-parameter Tuning:

If hyper-parameter tuning or model selection was performed based on the validation set, the model might be specifically optimized for that set, leading to better performance on it.

Problem

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 Importing and Cleaning:

Load the required libraries

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

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(e1071)
```

Read the dataset

```
universal <- read.csv("UniversalBank.csv")
dim(universal)
```

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

```
t(t(names(universal)))
```

```
##       [,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 ID and ZIP

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

Split the Data to 60% training and 40% validation.And then transform categorical variables into dummy variables

```r
universal$Education <- as.factor(universal$Education)
groups <- dummyVars(~., data = universal) #  it will create dummy groups
universal.dm <- as.data.frame(predict(groups,universal))


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

```
##       [,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, let us normalize the data*

```r
train.normalization <- train.dm[,-10]
valid.normalization <- valid.dm[,-10]

norm.values <- preProcess(train.dm[, -10], method=c("center", "scale"))
train.normalization <- predict(norm.values, train.dm[, -10])
valid.normalization <- predict(norm.values, valid.dm[, -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?

```r
#  categorical variables to dummy variables Conversion completed
# Let's create a new sample
```

```r
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
)

# Normalize the new customer

new.cust.norm <- new_customer
new.cust.norm <- predict(norm.values, new.cust.norm)
```

prediction using knn

```r
knn.pred1 <- class::knn(train = train.normalization,
                       test = new.cust.norm,
                       cl = train.dm$Personal.Loan, k = 1)
knn.pred1
```

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

---

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

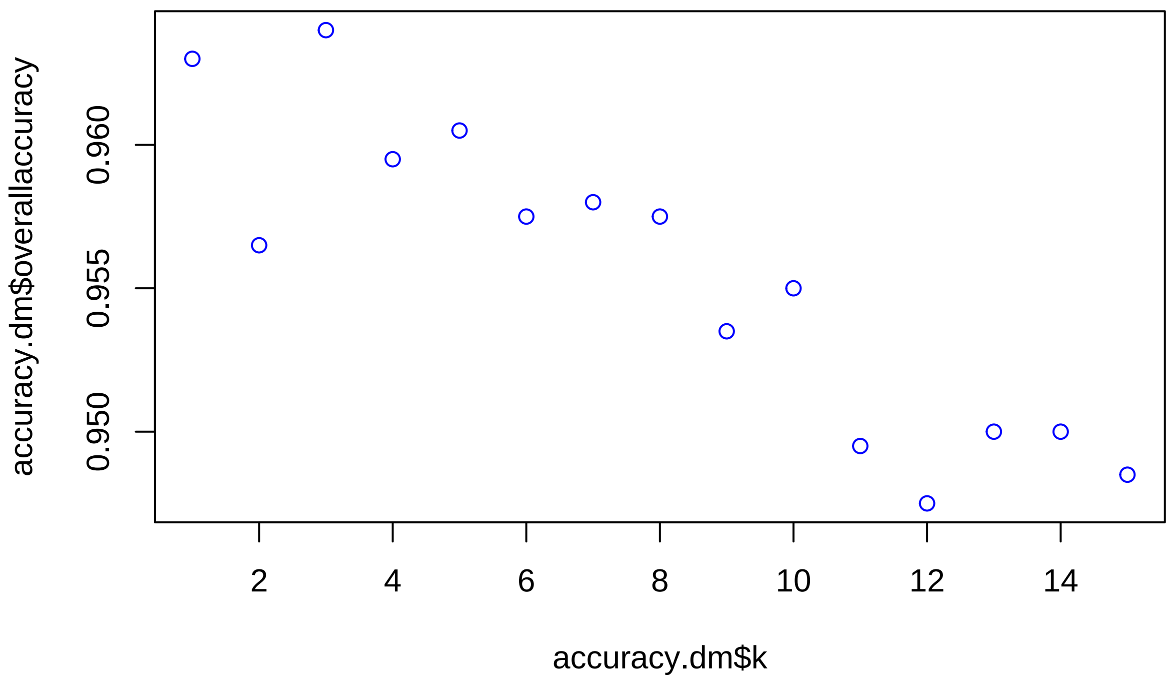
```r
# Calculate the accuracy for each value of k
# Set the range of k values to consider

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

which(accuracy.dm[,2] == max(accuracy.dm[,2]))
```

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

```r
plot(accuracy.dm$k,accuracy.dm$overallaccuracy,col = "blue")
```



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

```r
knn.pred_result <- class::knn(train = train.normalization,
test = valid.normalization,
cl = train.dm$Personal.Loan, k = 3)

# Creating the confusion matrix

confusion_matrix <- confusionMatrix(knn.pred_result,
                              as.factor(valid.dm$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.

```
# To find the customer having best k value.
knn.pred2 <- class::knn(train = train.normalization,
test = new.cust.norm,
cl = train.dm$Personal.Loan, k = 3)
knn.pred2
```

```
## [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.

```
set.seed(1)

training_set = sample(nrow(universal.dm), 0.5 * nrow(universal.dm))

validation_set = sample(setdiff(1:nrow(universal.dm), training_set),
                        0.3 * nrow(universal.dm))

test_set = setdiff(1:nrow(universal.dm), union(training_set, validation_set))

train.dm = universal.dm[training_set,]

valid.dm = universal.dm[validation_set,]

test.dm = universal.dm[test_set,]


train.normalization = train.dm[,-10]
```

```r
valid.normalization = valid.dm[,-10]

test.norm.dm = test.dm[,-10]

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

train.normalization = predict(norm.values, train.normalization)

valid.normalization = predict(norm.values, valid.normalization)

test.norm.dm = predict(norm.values, test.norm.dm)

train_pred = class::knn(train = train.normalization,
                           test = train.normalization,
                           cl = train.dm$Personal.Loan,
                           k = 3)

valid_pred = class::knn(train = train.normalization,
                           test = valid.normalization,
                           cl = train.dm$Personal.Loan,
                           k = 3)

test_knn_pred = class::knn(train = train.normalization,
                     test = test.norm.dm,
                     cl = train.dm$Personal.Loan,
                     k = 3)

#confusion matrix for training set:
train_confusion_matrix = confusionMatrix(train_pred,
                            as.factor(train.dm$Personal.Loan),positive = "1")

train_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:


confusion_matrix = confusionMatrix(valid_pred,
                              as.factor(valid.dm$Personal.Loan),positive = "1")

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:
confusion_matrix_valid = confusionMatrix(test_knn_pred,
                              as.factor(test.dm$Personal.Loan),positive = "1")

confusion_matrix_valid
```

```
## 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
##
```

comparing Test with Train:

Accuracy: Train has a higher accuracy (0.9772) compared to Test (0.9507).

Sensitivity (True Positive Rate):Train has higher sensitivity (0.7589) compared to Test (0.5875).

Specificity (True Negative Rate): Train has higher specificity (0.9987) compared to Test (0.99403).

Positive Predictive Value (Precision): Train has a higher positive predictive value (0.9827) compared to Test (0.92157).

comparing Train with Validation:

Accuracy: Train has a higher accuracy (0.9772) compared to Validation (0.958).

Sensitivity (True Positive Rate): Train has higher sensitivity (0.7589) compared to Validation (0.625).

Specificity (True Negative Rate): Train has higher specificity (0.9987) compared to Validation (0.9934).

Positive Predictive Value (Precision): Train still has a higher positive predictive value (0.9827) compared to Validation (0.9091).

comapring test with validation:

Accuracy : validation has a higher accuracy (0.986) than the test(0.961).

sensitivity: validation has a higher sensitivity(0.69118) than the test(0.6875).

Specificity : validation has a higher specificity(0.99560) than the test(0.9955).

positive predictive rate: test has higher positive predictive value(0.9506) than validation(0.94000).

Potential Reasons for Differences:

Data set Differences Variations in the composition and distribution of data between different sets can significantly impact model performance. For illustration, one data set may be more imbalanced, making it harder to predict rare events.

Sample Variability Performance requirements may be impacted in small data sets due to changes in the specific samples used for the confirmation and test sets.

Randomness There's an inherent randomness, especially when using techniques like cross-validation. Different random splits or initializations may lead to variations in performance on different datasets. Some models, similar as neural networks, involve randomness in their optimization process, leading to slight variations.

Hyper-parameter Tuning If hyperparameter tuning or model selection was performed based on the validation set, the model might be specifically optimized for that set, leading to better performance on it.