Assignment-2_FML

Spandana Sodadasi

2023-09-14

```
knitr::opts_chunk$set(echo = TRUE, comment = NULL)
```

Summary:-

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: 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?

Answer: The choice of k that balances between overfitting and ignoring the predictor information is '3' as it yields the highest overall accuracy on the validation data.

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

Answer: The confusion matrix shows different measures like accuracy, specificity, and sensitivity. From using the best k which is k=3, the accuracy is really high at 96.4%, which means the model is mostly correct. Specificity is even higher at 99.5%, showing it's good at identifying the negative classes. But the Sensitivity is at 69.2%, which means that the model is less effective at identifying the positive classes.

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: This customer would be classified as 0, 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.

Answer: By comparing the confusion matrix of the test set with that of training and validation sets we can notice that the training set shows higher sensitivity of 66.67%, which might mean it's fitting too closely to the data and possibly overfitting. In contrast, the test and validation sets have lower sensitivity around of 40.54% and 35.66%, but they seem to generalize better, which means when the model is k=3 it strikes balance between overfitting and underfitting.

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 Importing and Cleaning:

1. Loading the Required Libraries.

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

```
Loading required package: ggplot2
```

```
Loading required package: lattice
```

```
library(tinytex)
library(e1071)
```

2. Read the data.

```
library(readr)
UniversalBank.df <- read.csv("C:/Users/spand/Downloads/UniversalBank.csv")
dim(UniversalBank.df)
```

```
[1] 5000   14
```

3. Drop ID and ZIP variables.

```
UniversalBank.df <- UniversalBank.df[ ,-c(1,5)]
```

4. Transforming the categorical variables into dummy variables.

```
# Only education needs to be converted to factor
UniversalBank.df$Education <- as.factor(UniversalBank.df$Education)

# now, convert education to dummy variables

groups <- dummyVars(~., data = UniversalBank.df)

# Create Dummy variable.names
UniversalBank.df <- as.data.frame(predict(groups, UniversalBank.df))
```

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

```
set.seed(1) # Important to ensure that we get the same sample if we return the code

train.index <- sample(row.names(UniversalBank.df),0.6*dim(UniversalBank.df)[1])
valid.index <- setdiff(row.names(UniversalBank.df),train.index)
train.df <- UniversalBank.df[train.index,]
valid.df <- UniversalBank.df[valid.index,]
```

6. Normalizing the data.

```
train.norm.df <- train.df[,-10] # Note that Personal Income is the 10th variable
valid.norm.df <- valid.df[,-10]

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

Questions:-

1.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. 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?

```
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)
```

Predicting whether the new customer will accept or decline the loan using kNN where k=1.

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

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

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?

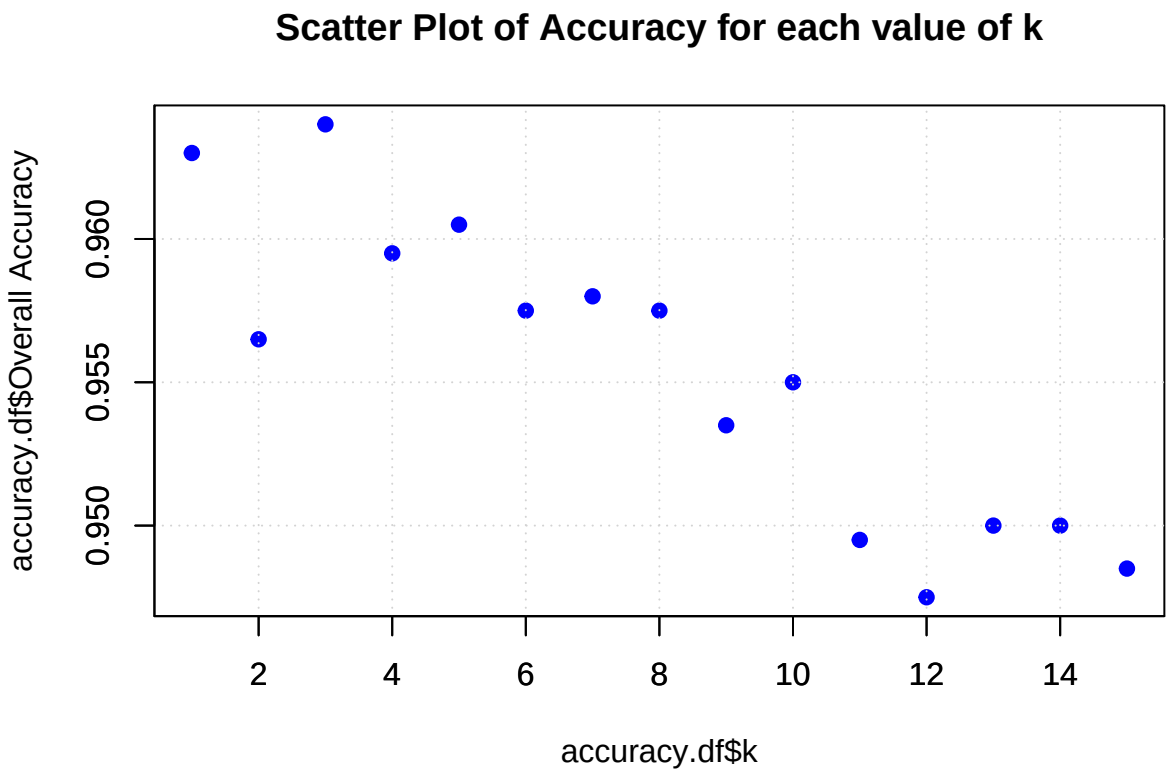
```
# Calculating the Accuracy for each value of k.
accuracy.df <- data.frame(k = seq(1, 15, 1), overallaccuracy = rep(0, 15))
for(i in 1:15) {
  knn.pred <- class::knn(train = train.norm.df,
                         test = valid.norm.df,
                         cl = train.df$Personal.Loan, k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn.pred,
                                       as.factor(valid.df$Personal.Loan),positive = "1")$overall[1]
}

best_k <- accuracy.df[which.max(accuracy.df$overallaccuracy), "k"]
print(best_k)
```

```
[1] 3
```

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

```
[1] 3
```

```
# Create a scatter plot
plot(accuracy.df$k, accuracy.df$overallaccuracy,
     xlab = "accuracy.df$k",
     ylab = "accuracy.df$Overall Accuracy",
     main = "Scatter Plot of Accuracy for each value of k",
     pch = 19,
     col = "blue"
)
grid()
axis(1, at = pretty(accuracy.df$k))
axis(2, at = pretty(accuracy.df$overallaccuracy))
```



The choice of k that balances between overfitting and ignoring the predictor information is '3' as it yields the highest overall accuracy on the validation data.

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

```
k<- 3
    knn.pred <- class::knn(train = train.norm.df,
                           test = valid.norm.df,
                           cl = train.df$Personal.Loan, k = 3)
conf_matrix <- confusionMatrix(knn.pred,
                               as.factor(valid.df$Personal.Loan), positive = "1")

print(conf_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
```

The confusion matrix shows different measures like accuracy, specificity, and sensitivity. From using the best k which is k=3, the accuracy is really high at 96.4%, which means the model is mostly correct. Specificity is even higher at 99.5%, showing it's good at identifying the negative classes. But the Sensitivity is at 69.2%, which means that the model is less effective at identifying the positive classes.

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

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

This customer would be classified as 0, 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.

Splitting the Data into 50% training, 30% validation and 20% testing.

```
set.seed(2)  # Important to ensure that we get the same sample if we rerun the code
train.index <- sample(row.names(UniversalBank.df), 0.5*dim(UniversalBank.df)[1])
valid.index <- sample(setdiff(row.names(UniversalBank.df), train.index),
                      0.3 * dim(UniversalBank.df)[1])
test.index <- setdiff(row.names(UniversalBank.df), c(train.index, valid.index))
train.df <- UniversalBank.df[train.index, ]
valid.df <- UniversalBank.df[valid.index, ]
test.df  <- UniversalBank.df[test.index, ]
```

Normalizing the data.

```
train.norm.df <- train.df[,-10] # Note that Personal Income is the 10th variable
valid.norm.df <- valid.df[,-10]
test.norm.df <- test.df[,-10]

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

The confusion matrix of the test set with that of the training and validation sets.

```
k<- 3
   knn.pred <- class::knn(train = train.df,
                          test = test.df,
                          cl = train.df$Personal.Loan, k = 3)
conf_matrix_test <- confusionMatrix(knn.pred,
                                    as.factor(test.df$Personal.Loan), positive = "1")
print(conf_matrix_test)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 893  44
         1  33  30

               Accuracy : 0.923
                 95% CI : (0.9047, 0.9388)
    No Information Rate : 0.926
    P-Value [Acc > NIR] : 0.6689

                  Kappa : 0.3969

 Mcnemar's Test P-Value : 0.2545

            Sensitivity : 0.4054
```

```
            Specificity : 0.9644
         Pos Pred Value : 0.4762
         Neg Pred Value : 0.9530
             Prevalence : 0.0740
         Detection Rate : 0.0300
   Detection Prevalence : 0.0630
      Balanced Accuracy : 0.6849

       'Positive' Class : 1
```

```
   knn.pred <- class::knn(train = train.df,
                          test = train.df,
                          cl = train.df$Personal.Loan, k = 3)
conf_matrix_train <- confusionMatrix(knn.pred,
                                as.factor(train.df$Personal.Loan), positive = "1")
print(conf_matrix_train)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2210   83
         1   41  166

               Accuracy : 0.9504
                 95% CI : (0.9411, 0.9586)
    No Information Rate : 0.9004
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.701

 Mcnemar's Test P-Value : 0.0002315

            Sensitivity : 0.6667
            Specificity : 0.9818
         Pos Pred Value : 0.8019
         Neg Pred Value : 0.9638
             Prevalence : 0.0996
         Detection Rate : 0.0664
   Detection Prevalence : 0.0828
      Balanced Accuracy : 0.8242

       'Positive' Class : 1
```

```
   knn.pred <- class::knn(train = train.df,
                          test = valid.df,
                          cl = train.df$Personal.Loan, k = 3)
conf_matrix_valid <- confusionMatrix(knn.pred,
                                as.factor(valid.df$Personal.Loan), positive = "1")
print(conf_matrix_valid)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1293  101
         1   50   56

               Accuracy : 0.8993
                 95% CI : (0.883, 0.9141)
    No Information Rate : 0.8953
    P-Value [Acc > NIR] : 0.3245

                  Kappa : 0.373

 Mcnemar's Test P-Value : 4.723e-05

            Sensitivity : 0.35669
            Specificity : 0.96277
         Pos Pred Value : 0.52830
         Neg Pred Value : 0.92755
             Prevalence : 0.10467
         Detection Rate : 0.03733
   Detection Prevalence : 0.07067
      Balanced Accuracy : 0.65973

       'Positive' Class : 1
```

By comparing the confusion matrix of the test set with that of training and validation sets we can notice that the training set shows higher sensitivity of 66.67%, which might mean it's fitting too closely to the data and possibly overfitting. In contrast, the test and validation sets have lower sensitivity around of 40.54% and 35.66%, but they seem to generalize better, which means when the model is k=3 it strikes balance between overfitting and underfitting.