FML_Assignment_2

Atshaya Suresh

2023-09-30

```
options(repos = "https://cran.stat.ucla.edu/")
```

Summary

(1) The customer specified will not accept the loan since the output is 0 (which is false) for a cut off value of 0.5 and when k=1.However, the results might change when we find the optimum value of k.

(2) The accuracy of the prediction when k=3 is very high i.e., more than 0.96.Hence, the choice of k that avoids overfitting and not ignoring the Predictor information is 3.

(3) The model has an accuracy of 96.4%, with a Kappa of 0.7785, indicating good predictive performance. It excels at identifying non-loan acceptors (class 0) with a sensitivity of 99.5%. However, its ability to correctly identify loan acceptors (class 1) is lower at 69.27% specificity. The Positive Predictive Value and Negative Predictive Value are 96.59% and 94.04% respectively. Balanced accuracy stands at 84.38%. Overall, the model's strength lies in predicting non-loan acceptors, but improvements are needed for loan acceptors.

(4) The customer specified will not accept the loan since the output is 0 (which is false) for a cut off value of 0.5 and when k=3, which is considered the best value of k (from the question above).

(5) The k-NN model with k=3 shows a consistent accuracy of approximately 95-98% across training, validation, and test data sets, indicating it generalizes well to new data. However, while the model's sensitivity is high (close to 100% for class 0), its specificity is lower, especially in the validation set. This suggests the model predicts non-loan acceptors (class 0) well but is less adept at identifying actual loan acceptors (class 1). Given the context of identifying loan acceptance, this is a concern since class 1 is more critical. The lower specificity in validation indicates potential false positives: predicting loan acceptance when it wouldn't occur. Comparing to the initial analysis, the findings reinforce the value of k=3 but also emphasize the need for enhancing prediction accuracy for the positive class, possibly through model refinement or incorporating additional features.

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

First, load the required libraries

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

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

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

```
library(e1071)
```

Read the data.

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

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

```
t(t(names(universal.df))) # 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 ID and ZIP

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

Split Data into 60% training and 40% validation. There are many ways to do this. We will look at 2 different ways. Before we split, let us transform categorical variables into dummy variables

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

# Now, convert Education to Dummy Variables

groups <- dummyVars(~., data = universal.df) # This creates the dummy groups
universal_m.df <- as.data.frame(predict(groups,universal.df))


set.seed(1)  # Important to ensure that we get the same sample if we rerun the code
train.index <- sample(row.names(universal_m.df), 0.6*dim(universal_m.df)[1])
valid.index <- setdiff(row.names(universal_m.df), train.index)
train.df <- universal_m.df[train.index,]
valid.df <- universal_m.df[valid.index,]
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"
```

```
#Second approach
install.packages("caTools")
```

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

```
## Warning: unable to access index for repository https://cran.stat.ucla.edu/src/contrib:
##   cannot open URL 'https://cran.stat.ucla.edu/src/contrib/PACKAGES'
```

```
## Warning: package 'caTools' is not available for this version of R
##
## A version of this package for your version of R might be available elsewhere,
## see the ideas at
## https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
```

```
## Warning: unable to access index for repository https://cran.stat.ucla.edu/bin/windows/contrib/4.3:
##   cannot open URL 'https://cran.stat.ucla.edu/bin/windows/contrib/4.3/PACKAGES'
```

```
library(caTools)
set.seed(1)
split <- sample.split(universal_m.df, SplitRatio = 0.6)
training_set <- subset(universal_m.df, split == TRUE)
validation_set <- subset(universal_m.df, split == FALSE)

# Print the sizes of the training and validation sets
print(paste("The size of the training set is:", nrow(training_set)))
```

```
## [1] "The size of the training set is: 2858"
```

```
print(paste("The size of the validation set is:", nrow(validation_set)))
```

```
## [1] "The size of the validation set is: 2142"
```

##Now, let us normalize 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

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?

```
# We have converted all categorical variables to dummy variables
# Let's create a new sample
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)
```

Now, let us predict using knn

```
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
```

---

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

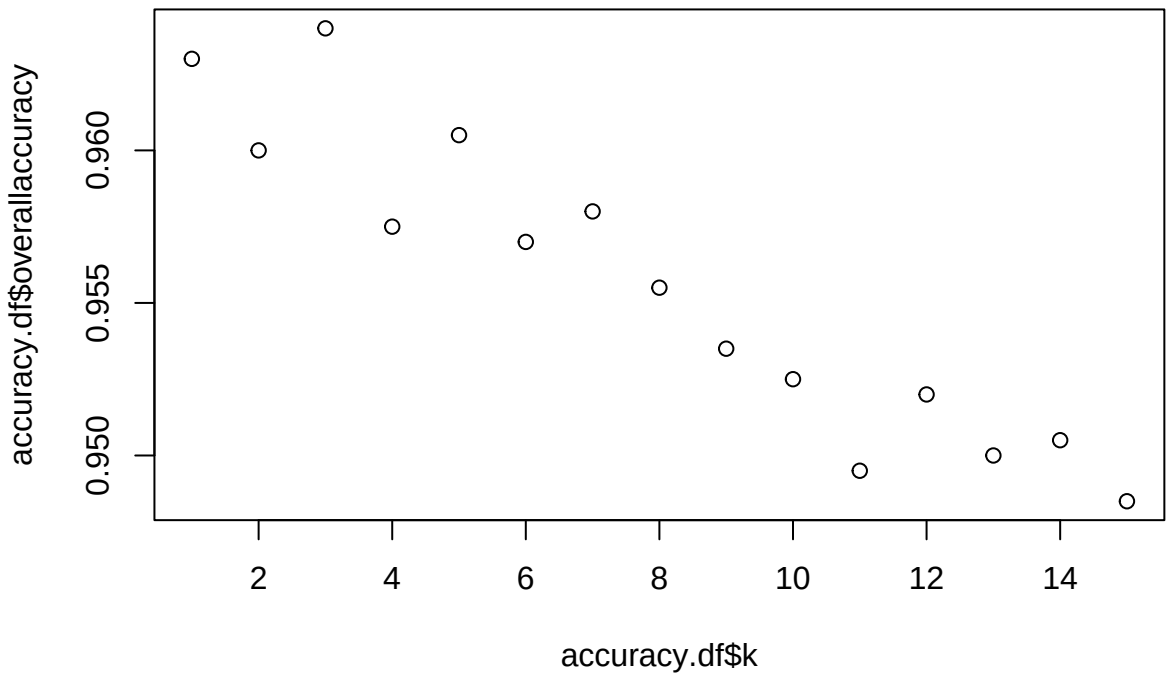
```
# Calculate the accuracy for each value of k
# Set the range of k values to consider

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.

```
# Using best k for validation data
knn.pred_validation <- class::knn(
  train = train.norm.df,
  test = valid.norm.df,
  cl = train.df$Personal.Loan,
  k = 3
)

confusionMatrix(knn.pred_validation, as.factor(valid.df$Personal.Loan))
```

```
## 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 : 0.9950
##             Specificity : 0.6927
##          Pos Pred Value : 0.9659
##          Neg Pred Value : 0.9404
##              Prevalence : 0.8975
##          Detection Rate : 0.8930
##    Detection Prevalence : 0.9245
##       Balanced Accuracy : 0.8438
##
##        'Positive' Class : 0
##
```

---

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.

```
knn.pred1 <- class::knn(train = train.norm.df,
                        test = new.cust.norm,
                        cl = train.df$Personal.Loan, k = 3)
knn.pred1
```

```
## [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)
split1 <- sample.split(universal_m.df, SplitRatio = 0.50)  # 50% for training
training_set <- subset(universal_m.df, split1 == TRUE)

remaining_set <- subset(universal_m.df, split1 == FALSE)
split2 <- sample.split(remaining_set, SplitRatio = 0.60)  # 60% of the remaining 50% (i.e., 30% of the
validation_set <- subset(remaining_set, split2 == TRUE)
test_set <- subset(remaining_set, split2 == FALSE)  # the remaining for test

print(paste("Size of training set:", nrow(training_set)))
```

```
## [1] "Size of training set: 2501"
```

```
print(paste("Size of validation set:", nrow(validation_set)))
```

```
## [1] "Size of validation set: 1426"
```

```r
print(paste("Size of test set:", nrow(test_set)))
```

```
## [1] "Size of test set: 1073"
```

```r
norm.values <- preProcess(training_set[, -10], method=c("center", "scale"))

train.norm.df <- predict(norm.values, training_set[, -10])
valid.norm.df <- predict(norm.values, validation_set[, -10])
test.norm.df <- predict(norm.values, test_set[, -10])

# Using k=3 as found earlier
knn.pred_train <- class::knn(
  train = train.norm.df,
  test = train.norm.df,
  cl = training_set$Personal.Loan,
  k = 3)

knn.pred_validation <- class::knn(
  train = train.norm.df,
  test = valid.norm.df,
  cl = training_set$Personal.Loan,
  k = 3)

knn.pred_test <- class::knn(
  train = train.norm.df,
  test = test.norm.df,
  cl = training_set$Personal.Loan,
  k = 3)

# Print the confusion matrices
print("Confusion Matrix for Training Set")
```

```
## [1] "Confusion Matrix for Training Set"
```

```r
print(confusionMatrix(knn.pred_train, as.factor(training_set$Personal.Loan)))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2265   54
##          1    2  180
##
##                Accuracy : 0.9776
##                  95% CI : (0.971, 0.983)
##     No Information Rate : 0.9064
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8534
##
##  Mcnemar's Test P-Value : 9.416e-12
```

```
##
##             Sensitivity : 0.9991
##             Specificity : 0.7692
##          Pos Pred Value : 0.9767
##          Neg Pred Value : 0.9890
##              Prevalence : 0.9064
##          Detection Rate : 0.9056
##    Detection Prevalence : 0.9272
##       Balanced Accuracy : 0.8842
##
##        'Positive' Class : 0
##
```

```
print("Confusion Matrix for Validation Set")
```

```
## [1] "Confusion Matrix for Validation Set"
```

```
print(confusionMatrix(knn.pred_validation, as.factor(validation_set$Personal.Loan)))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1283   56
##          1    6   81
##
##                Accuracy : 0.9565
##                  95% CI : (0.9446, 0.9665)
##     No Information Rate : 0.9039
##     P-Value [Acc > NIR] : 6.289e-14
##
##                   Kappa : 0.7009
##
##  Mcnemar's Test P-Value : 4.877e-10
##
##             Sensitivity : 0.9953
##             Specificity : 0.5912
##          Pos Pred Value : 0.9582
##          Neg Pred Value : 0.9310
##              Prevalence : 0.9039
##          Detection Rate : 0.8997
##    Detection Prevalence : 0.9390
##       Balanced Accuracy : 0.7933
##
##        'Positive' Class : 0
##
```

```
print("Confusion Matrix for Test Set")
```

```
## [1] "Confusion Matrix for Test Set"
```

```r
print(confusionMatrix(knn.pred_test, as.factor(test_set$Personal.Loan)))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 961  40
##          1   3  69
##
##                Accuracy : 0.9599
##                  95% CI : (0.9464, 0.9708)
##     No Information Rate : 0.8984
##     P-Value [Acc > NIR] : 5.443e-14
##
##                   Kappa : 0.7415
##
##  Mcnemar's Test P-Value : 4.021e-08
##
##             Sensitivity : 0.9969
##             Specificity : 0.6330
##          Pos Pred Value : 0.9600
##          Neg Pred Value : 0.9583
##              Prevalence : 0.8984
##          Detection Rate : 0.8956
##    Detection Prevalence : 0.9329
##       Balanced Accuracy : 0.8150
##
##        'Positive' Class : 0
##
```