Assignment 2

2023-09-28

#Summary

Questions - Answers

1. This new customer would be classified as 0, does not take the personal loan
2. The best K is 3
3. The confusion matrix of validation has TN = 1786, TP = 142, FN = 63, FP = 9
4. With k=3, the customer is classified as 0, means that he/she declines the loan.
5. Compare the three confusion matrix we can see the Specificity rate of three data sets giving nearly the same percentage at 99,78%, which can correctly identify the true decline customer cases. Beside that, the training set has the highest sensitivity rate among other data sets. So it should be said that, training set is the more reliable data.

#General steps

```r
#Step 1: Loading the Libraries.
library(class)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(e1071)

#Step 2: Read the csv file.
universal.df <- read.csv("C:/Users/ADMIN/Downloads/UniversalBank.csv")
dim(universal.df)
```

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

```r
t(t(names(universal.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"
```

```
#Step 3: Drop ID and ZIP variables.
universal.df <- universal.df[,-c(1,5)]

#Step 4: Transforming the categorical variables into dummy variables.
universal.df$Education <- as.factor(universal.df$Education)
groups <- dummyVars(~., data = universal.df)
universal_m.df <- as.data.frame(predict(groups,universal.df))
```

```
#Step 5: Splitting the Data into 60% training and 40% validation set.
set.seed(1)
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"
```

```
#Step 6: Normalizing the data.
train.norm.df <- train.df[,-10]
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])
```

#Question: 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?

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

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

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

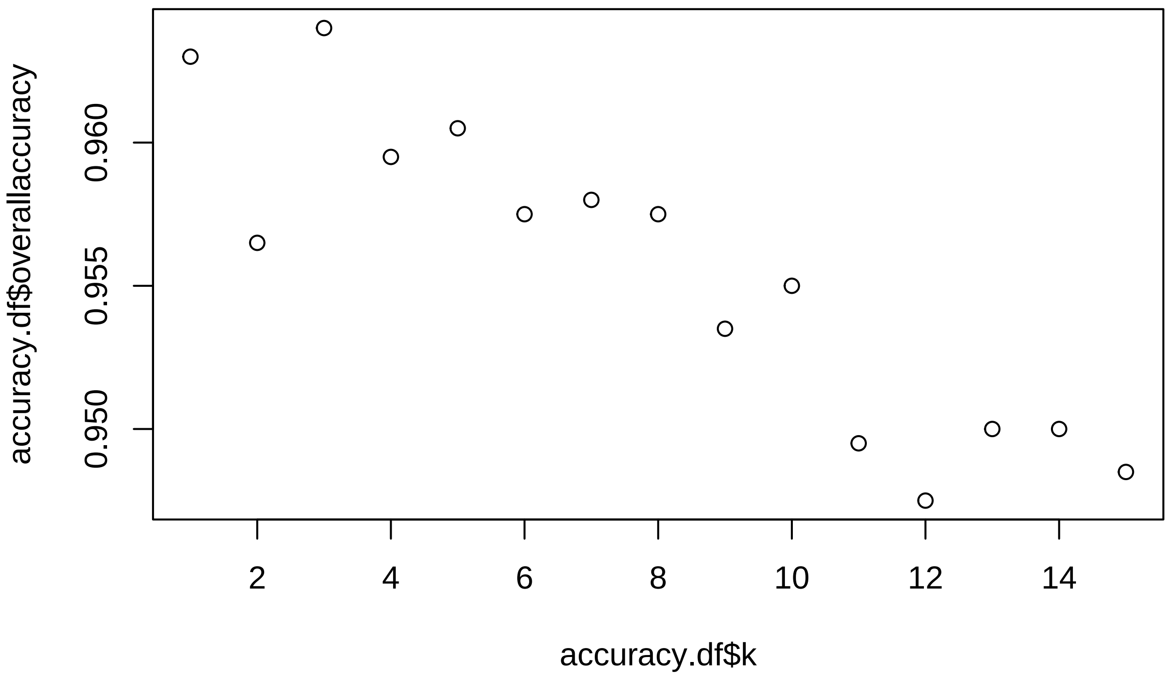
```
# Calculate the accuracy for each value of k, set the range of k values to consider
accuracy.df <- data.frame(k = seq(1, 15, 1), overallaccuracy = rep(0, 15))
for(i in 1:15) {
  knn.pred <- class::knn(train = train.norm.df,
                         test = valid.norm.df,
                         cl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

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

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

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

```r
plot(accuracy.df$k,accuracy.df$overallaccuracy)
```



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

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

print(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.

```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)

#Predicting whether the new customer will accept or decline the loan using kNN as 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
```

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.

```r
#Partition the data into training (50%), validation (30%) and Test set (20%)
set.seed(1)
train.index <- sample(row.names(universal_m.df), 0.5*dim(universal_m.df)[1])
valid.index <- sample(row.names(universal_m.df), 0.3*dim(universal_m.df)[1])
test_set.index <- sample(row.names(universal_m.df), 0.2*dim(universal_m.df)[1])
train.df <- universal_m.df[train.index,]
valid.df <- universal_m.df[valid.index,]
test.df<- universal_m.df[test_set.index,]

print(paste("The size of the training set is:", nrow(train.df)))
```

```
## [1] "The size of the training set is: 2500"
```

```r
print(paste("The size of the validation set is:", nrow(valid.df)))
```

```
## [1] "The size of the validation set is: 1500"
```

```r
print(paste("The size of the validation set is:", nrow(test.df)))
```

```
## [1] "The size of the validation set is: 1000"
```

```r
#Normalizing 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"))
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])

#Create confusion matrix and Statistics for each data set

#Confusion matrix for test set
  knn.pred <- class::knn(train = train.norm.df,
                         test = test.norm.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
##          0 901  33
##          1   2  64
##
##                Accuracy : 0.965
##                  95% CI : (0.9517, 0.9755)
##     No Information Rate : 0.903
```

```
##     P-Value [Acc > NIR] : 4.737e-14
##
##                   Kappa : 0.767
##
##  Mcnemar's Test P-Value : 3.959e-07
##
##             Sensitivity : 0.6598
##             Specificity : 0.9978
##          Pos Pred Value : 0.9697
##          Neg Pred Value : 0.9647
##              Prevalence : 0.0970
##          Detection Rate : 0.0640
##    Detection Prevalence : 0.0660
##       Balanced Accuracy : 0.8288
##
##        'Positive' Class : 1
##
```

```
#Confusion matrix for training set
  knn.pred <- class::knn(train = train.norm.df,
                         test = train.norm.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 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
##
```

```r
#Confusion matrix for valid set
  knn.pred <- class::knn(train = train.norm.df,
                         test = valid.norm.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 1347   43
##          1    3  107
##
##                Accuracy : 0.9693
##                  95% CI : (0.9593, 0.9775)
##     No Information Rate : 0.9
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8067
##
##  Mcnemar's Test P-Value : 8.912e-09
##
##             Sensitivity : 0.71333
##             Specificity : 0.99778
##          Pos Pred Value : 0.97273
##          Neg Pred Value : 0.96906
##              Prevalence : 0.10000
##          Detection Rate : 0.07133
##    Detection Prevalence : 0.07333
##       Balanced Accuracy : 0.85556
##
##        'Positive' Class : 1
##
```