FML Assignment2

Shreya Thodupunuri 811301506

2023-09-29

```r
#importing the required packages
library('caret')
```

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

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

```r
library('e1071')
library('ISLR')
library('class')
```

```r
universal.df <- read.csv("~/Shreya R documents/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"
```

```r
#Drop ID and ZIP
universal.df <- universal.df[,-c(1,5)]
```

```r
# Converting Education to a factor
universal.df$Education <- as.factor(universal.df$Education)
```

```r
#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))
```

```r
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"
```

```r
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)
```

```r
# 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"
```

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

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

```r
#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
```

---

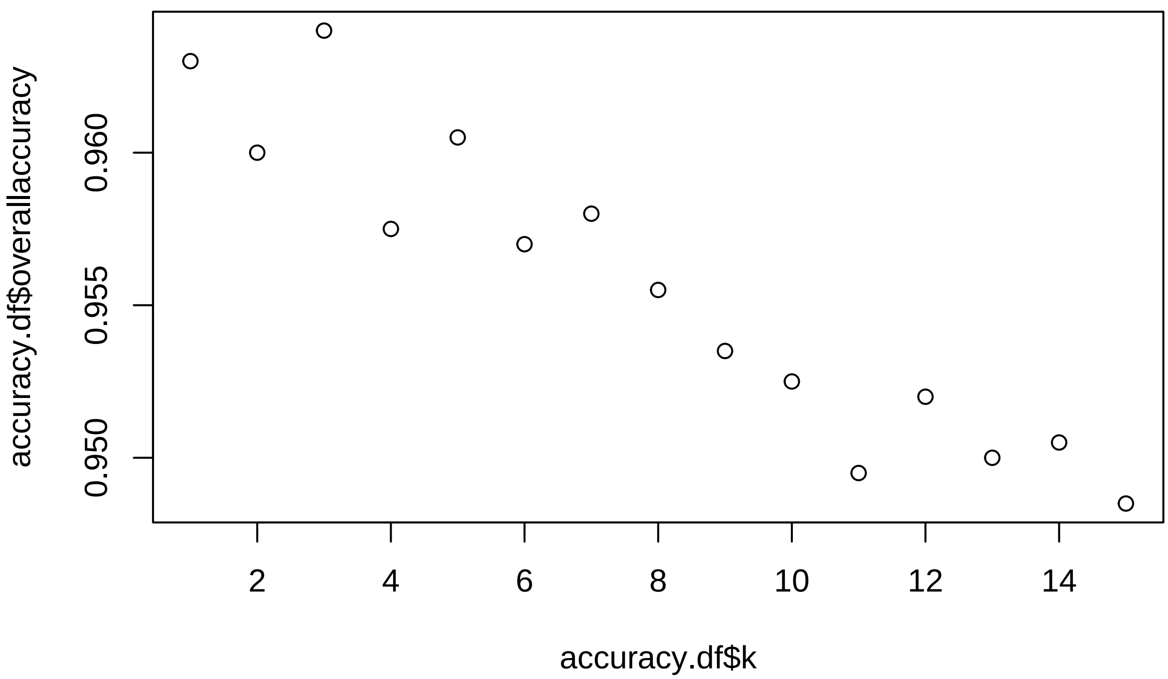
2. What is a choice of k that balances between overfitting and ignoring the predictor information?

```
# Calculate the accuracy for each value of k
# Setting the range of k values to be considered

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)
```




```
#The best performing k in the range of 1 to 15 is 3.This k balances overfitting and ignoring prediction
```

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

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

```
customer2.df <- 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 the 2nd customer dataset
custnorm2 <- predict(norm.values,customer2.df)
```

5.Repeating the process by partitioning the data into three parts - 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
# Split the data into training (50%), validation (30%), and test (20%) sets
set.seed(123)
Train.Index <- sample(row.names(universal_m.df), .5*dim(universal_m.df)[1])#create train index
```

```r
#creating validation index
Valid.Index <- sample(setdiff(row.names(universal_m.df),Train.Index),.3*dim(universal_m.df)[1])
Test.Index =setdiff(row.names(universal_m.df),union(Train.Index,Valid.Index))#create test index
train.df <- universal_m.df[Train.Index,]
cat("The size of the new training dataset is:", nrow(train.df))
```

```
## The size of the new training dataset is: 2500
```

```r
valid.df <- universal_m.df[Valid.Index, ]
cat("The size of the new validation dataset is:", nrow(valid.df))
```

```
## The size of the new validation dataset is: 1500
```

```r
test.df <- universal_m.df[Test.Index,]
cat("The size of the new test dataset is:",nrow(test.df))
```

```
## The size of the new test dataset is: 1000
```

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

#Performing kNN and creating confusion matrix on training, testing, validation data sets

```r
pred3 <- class::knn(train = train.df.norm,
test = test.df.norm,
cl = train.df$Personal.Loan, k=3)
confusionMatrix(pred3,as.factor(test.df$Personal.Loan))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##          0 890  38
##          1   2  70
##
##                Accuracy : 0.96
##                  95% CI : (0.9459, 0.9713)
##     No Information Rate : 0.892
##     P-Value [Acc > NIR] : 4.095e-15
##
##                   Kappa : 0.7568
##
##  Mcnemar's Test P-Value : 3.130e-08
```

```
##
##             Sensitivity : 0.9978
##             Specificity : 0.6481
##          Pos Pred Value : 0.9591
##          Neg Pred Value : 0.9722
##              Prevalence : 0.8920
##          Detection Rate : 0.8900
##    Detection Prevalence : 0.9280
##       Balanced Accuracy : 0.8230
##
##        'Positive' Class : 0
##
```

```r
pred4 <- class::knn(train = train.df.norm,
test = valid.df.norm,
cl = train.df$Personal.Loan, k=3)
confusionMatrix(pred4,as.factor(valid.df$Personal.Loan))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1350   58
##          1    7   85
##
##                Accuracy : 0.9567
##                  95% CI : (0.9451, 0.9664)
##     No Information Rate : 0.9047
##     P-Value [Acc > NIR] : 2.347e-14
##
##                   Kappa : 0.7011
##
##  Mcnemar's Test P-Value : 5.584e-10
##
##             Sensitivity : 0.9948
##             Specificity : 0.5944
##          Pos Pred Value : 0.9588
##          Neg Pred Value : 0.9239
##              Prevalence : 0.9047
##          Detection Rate : 0.9000
##    Detection Prevalence : 0.9387
##       Balanced Accuracy : 0.7946
##
##        'Positive' Class : 0
##
```

```r
pred5 <- class::knn(train = train.df.norm,
test = train.df.norm,
cl = train.df$Personal.Loan, k=3)
confusionMatrix(pred5,as.factor(train.df$Personal.Loan))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    0    1
##          0 2267   57
##          1    4  172
##
##                Accuracy : 0.9756
##                  95% CI : (0.9688, 0.9813)
##     No Information Rate : 0.9084
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8364
##
##  Mcnemar's Test P-Value : 2.777e-11
##
##             Sensitivity : 0.9982
##             Specificity : 0.7511
##          Pos Pred Value : 0.9755
##          Neg Pred Value : 0.9773
##              Prevalence : 0.9084
##          Detection Rate : 0.9068
##    Detection Prevalence : 0.9296
##       Balanced Accuracy : 0.8747
##
##        'Positive' Class : 0
##
```

The sets are not mutually exclusive.