FML Assignment 2

2023-10-01

Summary

The necessary libraries (class, caret, e1071) are loaded.

The UniversalBank.csv dataset, containing customer information, is read into the variable universal.df.

The code removes unnecessary columns like ID and ZIP code from the dataset.

Categorical variables (specifically, Education) are converted into dummy variables to facilitate further analysis.

The dataset is split into a training set (60%) and a validation set (40%).

The data are normalized using the mean and standard deviation.

A new customer's data is provided, and k-NN classification with k=1 is performed to predict whether this customer will accept a personal loan.The result indicates that this customer would be classified as "0," meaning they would not take the personal loan.

Accuracy for different values of k from 1 to 15 is calculated using k-NN on the validation set.A plot is generated to help choose an appropriate k that balances between overfitting and underfitting. In this case, the best k is determined to be 3.

The confusion matrix for the validation data is computed using k-NN with the best k (k=3).

Another customer's data is provided, and k-NN classification with the best k (k=3) is performed to predict whether this customer will accept a personal loan.The result indicates that this customer would be classified as "0," meaning they would not take the personal loan.

The dataset is repartitioned into training (50%), validation (30%), and test (20%) sets.k-NN classification is applied to each set using the best k (k=3), and confusion matrices are calculated for the training, validation, and test sets.The analysis suggests that the model tends to perform well on the training set but struggles to generalize to unseen data, as it is evident by discrepancies in performance between training, validation, and test sets.

Questions - Answers

1. How would this customer be classified? This new customer would be classified as 0, does not take the personal loan.
2. The best K is 3.
3. The confusion matrix for the validation data is computed using k-NN with the best k (k=3).
4. How would this customer be classified using k=3 ? This new customer would be classified as 0, does not take the personal loan.
5. The model is overfitted as the accuracy of training dataset is higher than validation and test set.

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("/Users/akhilchintu/Downloads/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.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.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.df), 0.6*dim(universal.df)[1])
valid.index <- setdiff(row.names(universal.df), train.index)
train.df <- universal.df[train.index,]
valid.df <- universal.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"
```

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

```
## [1] "The size of the training set is: 3000"
```

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

```
## [1] "The size of the validation set is: 2000"
```

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
```

The customer is classified as "0",that means no personal loan.

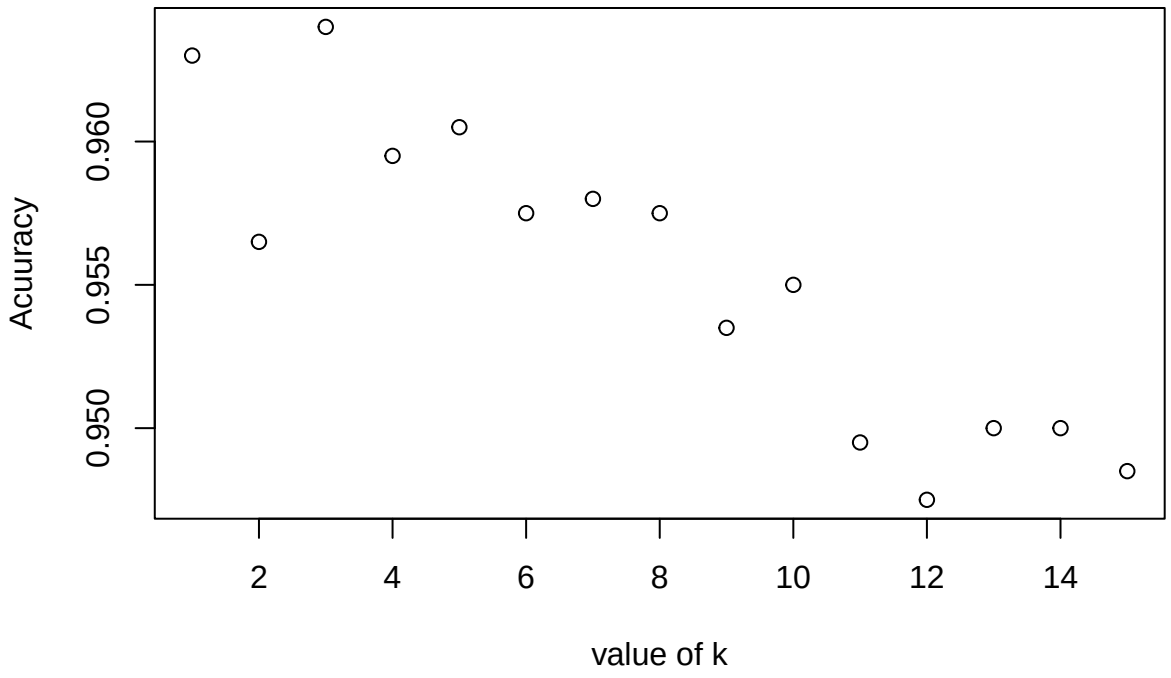
---

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

print(paste("The value that best fits is:",best.k))
```

```
## [1] "The value that best fits is: 3"
```

```
plot(accuracy.df$k,accuracy.df$overallaccuracy,xlab="value of k",ylab="Acuuracy")
```



---

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

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

```
best.k.matrix <- confusionMatrix(knn.pred_k,
                                 as.factor(valid.df$Personal.Loan),positive = "1")
best.k.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 CreditCard = 1. Classify the customer using the best k.

```
second_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
second.cust.norm <- second_customer
second.cust.norm <- predict(norm.values,second.cust.norm )
```

```
#let us predit through knn
knn.pred.cust <- class::knn(train = train.norm.df,
                        test = second.cust.norm,
                        cl = train.df$Personal.Loan, k = 3)
knn.pred.cust
```

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

The customer is classified as "0",that means no 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.

```
#Repartitioning the data.
train.index.new <- sample(row.names(universal.df), 0.5*dim(universal.df)[1])
valid.index.new <- sample(setdiff(row.names(universal.df), train.index.new), 0.3*dim(universal.df)[1])
test.index.new <- setdiff(setdiff(row.names(universal.df), train.index.new), valid.index.new)
train.set <- universal.df[train.index.new,]
valid.set <- universal.df[valid.index.new,]
test.set <- universal.df[test.index.new,]
```

Now, let us normalize the data

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

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

Confusion matrix for training,validation and testing set:

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

train.matrix <- confusionMatrix(knn.pred.train,as.factor(train.set$Personal.Loan),positive = "1")
train.matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2254   50
##          1    6  190
##
##                Accuracy : 0.9776
##                  95% CI : (0.971, 0.983)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8594
##
##  Mcnemar's Test P-Value : 9.132e-09
##
##             Sensitivity : 0.7917
##             Specificity : 0.9973
##          Pos Pred Value : 0.9694
##          Neg Pred Value : 0.9783
##              Prevalence : 0.0960
##          Detection Rate : 0.0760
##    Detection Prevalence : 0.0784
##       Balanced Accuracy : 0.8945
##
##        'Positive' Class : 1
##
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1343   46
##          1   17   94
##
##                Accuracy : 0.958
##                  95% CI : (0.9466, 0.9676)
##     No Information Rate : 0.9067
##     P-Value [Acc > NIR] : 2.658e-14
##
##                   Kappa : 0.7264
##
```

```
##  Mcnemar's Test P-Value : 0.0004192
##
##             Sensitivity : 0.67143
##             Specificity : 0.98750
##          Pos Pred Value : 0.84685
##          Neg Pred Value : 0.96688
##              Prevalence : 0.09333
##          Detection Rate : 0.06267
##    Detection Prevalence : 0.07400
##       Balanced Accuracy : 0.82946
##
##        'Positive' Class : 1
##
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 896  34
##          1   4  66
##
##                Accuracy : 0.962
##                  95% CI : (0.9482, 0.973)
##     No Information Rate : 0.9
##     P-Value [Acc > NIR] : 1.383e-13
##
##                   Kappa : 0.7564
##
##  Mcnemar's Test P-Value : 2.546e-06
##
##             Sensitivity : 0.6600
##             Specificity : 0.9956
##          Pos Pred Value : 0.9429
##          Neg Pred Value : 0.9634
##              Prevalence : 0.1000
##          Detection Rate : 0.0660
##    Detection Prevalence : 0.0700
##       Balanced Accuracy : 0.8278
##
##        'Positive' Class : 1
##
```

Here, we can observe that the model is overfitted.Overfitting appears when a model's accuracy on the test set is higher than on the validation set while the accuracy on the training set is higher than validation and testing.The model is unable to generalize to new input while learning the training set.
