FML Assignment 2

Harshith Kumar Yadav Temura

10-01-2023

**loading the libraries**

library(class)  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(e1071)

**loading the data**

Bank = read.csv("C:\\Users\\Harshith Kumar\\OneDrive\\Desktop\\fml\\UniversalBank (1).csv")  
dim(Bank)

## [1] 5000 14

t(t(names(Bank)))

## [,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**

Bank = Bank[,-c(1,5)]

**conversion of factor(Education)**

#Only Education needs to be converted into Factor in dataset  
Bank$Education = as.factor(Bank$Education)  
levels(Bank$Education)

## [1] "1" "2" "3"

#Now, Convert Education to Dummy Variables  
  
groups = dummyVars(~.,data = Bank) #This created a dummy variable   
  
Bank.Mod = as.data.frame(predict(groups,Bank))

**To have a consistent random selection we are setting up the value of set seed to 5**

set.seed(5)  
  
training.dif = sample(row.names(Bank.Mod),0.6\*dim(Bank.Mod)[1])  
validation.dif = setdiff(row.names(Bank.Mod),training.dif)  
train.diff = Bank.Mod[training.dif,]  
valid.diff = Bank.Mod[validation.dif,]  
t(t(names(train.diff)))

## [,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  
  
library(caTools)  
set.seed(1)  
split <- sample.split(Bank.Mod, SplitRatio = 0.6)  
train\_set <- subset(Bank.Mod, split == TRUE)  
valid\_set <- subset(Bank.Mod, split == FALSE)  
  
# Printing the sizes of the training and validation datasets.  
print(paste("The size of the training set is:", nrow(train\_set)))

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

print(paste("The size of the validation set is:", nrow(valid\_set)))

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

**Normalization of the dataset**

train.normal.diff <- train.diff[,-10] # Note that Personal Income is the 10th variable  
valid.normal.diff <- valid.diff[,-10]  
  
normal.values <- preProcess(train.diff[, -10], method=c("center", "scale"))  
train.normal.diff <- predict(normal.values, train.diff[, -10])  
valid.normal.diff <- predict(normal.values, valid.diff[, -10])

**Question No:1**

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\_CustomerX <- 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.normal <- New\_CustomerX  
New.Cust.normal <- predict(normal.values, New.Cust.normal)

**prediction using KNN**

KNN.Prediction1 <- class::knn(train = train.normal.diff,   
 test = New.Cust.normal,   
 cl = train.diff$Personal.Loan, k = 1)  
KNN.Prediction1

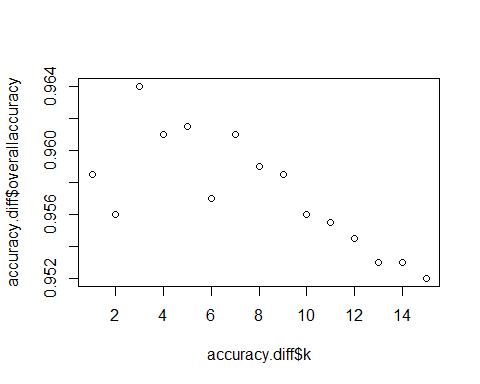
## [1] 0  
## Levels: 0 1

1. What is a choice of K that balances between over-fitting and ignoring the predictor information?

#Calculating the accuracy for each value of k  
#Set the range of k values   
  
accuracy.diff <- data.frame(k = seq(1, 15, 1), overallaccuracy = rep(0, 15))  
for(i in 1:15) {  
 KNN.Predct <- class::knn(train = train.normal.diff,   
 test = valid.normal.diff,   
 cl = train.diff$Personal.Loan, k = i)  
 accuracy.diff[i, 2] <- confusionMatrix(KNN.Predct,   
 as.factor(valid.diff$Personal.Loan),positive = "1")$overall[1]  
}  
  
which(accuracy.diff[,2] == max(accuracy.diff[,2]))

## [1] 3

plot(accuracy.diff$k,accuracy.diff$overallaccuracy)



**Question 3**

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

KNN.Prediction2 <- class::knn(train = train.normal.diff,   
 test = valid.normal.diff,   
 cl = train.diff$Personal.Loan, k = 3)  
  
confusionMatrix(KNN.Prediction2,as.factor(valid.diff$Personal.Loan))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1815 67  
## 1 5 113  
##   
## Accuracy : 0.964   
## 95% CI : (0.9549, 0.9717)  
## No Information Rate : 0.91   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7398   
##   
## Mcnemar's Test P-Value : 6.531e-13   
##   
## Sensitivity : 0.9973   
## Specificity : 0.6278   
## Pos Pred Value : 0.9644   
## Neg Pred Value : 0.9576   
## Prevalence : 0.9100   
## Detection Rate : 0.9075   
## Detection Prevalence : 0.9410   
## Balanced Accuracy : 0.8125   
##   
## 'Positive' Class : 0   
##

**Question 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**

#Classifying the customer using the best K.  
  
New\_CustomerY = 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  
)  
  
KNN.Prediction3 <- class::knn(train = train.normal.diff,   
 test = New\_CustomerY,   
 cl = train.diff$Personal.Loan, k = 3)  
  
KNN.Prediction3

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

#The customer has been classified as approved for personal loan

**Question5**

**Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%). Apply**

set.seed(5)  
#Let's take 50% of the entire modified data as Training data   
train.diff2 = sample(row.names(Bank.Mod), 0.5\*dim(Bank.Mod)[1])  
  
#Let's take 30% of the data from the remaining 50% as Validation Data   
valid.diff2 = sample(setdiff(row.names(Bank.Mod), train.diff2), 0.3\*dim(Bank.Mod)[1])  
  
#Let's take remaining 20% of the modified data as Test Data  
test.diff2 = setdiff(row.names(Bank.Mod), union(train.diff2,valid.diff2))  
  
train.normal.diff2 = Bank.Mod[train.diff2,]  
valid.normal.diff2 = Bank.Mod[valid.diff2,]  
test.normal.diff2 = Bank.Mod[test.diff2,]  
  
#transporting the data  
t(t(names(train.normal.diff2)))

## [,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"

# Applying the k-NN method with the chosen value K.  
  
trainknn2 = knn(train = train.normal.diff2[,-8], test = train.normal.diff2[,-8], cl = train.normal.diff2[,8], k=3)  
  
validknn2 = knn(train = train.normal.diff2[,-8], test = valid.normal.diff2[,-8], cl = train.normal.diff2[,8], k=3)  
  
testknn2 = knn(train = train.normal.diff2[,-8], test = test.normal.diff2[,-8], cl = train.normal.diff2[,8], k=3)

**Comparing the confusion matrix of the training set, validation sets and test set**

Confusionmatrix\_trainknn2 = confusionMatrix(trainknn2, as.factor(train.normal.diff2$Personal.Loan),positive = "1")  
  
Confusionmatrix\_trainknn2

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1678 205  
## 1 563 54  
##   
## Accuracy : 0.6928   
## 95% CI : (0.6743, 0.7108)  
## No Information Rate : 0.8964   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : -0.0265   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.20849   
## Specificity : 0.74877   
## Pos Pred Value : 0.08752   
## Neg Pred Value : 0.89113   
## Prevalence : 0.10360   
## Detection Rate : 0.02160   
## Detection Prevalence : 0.24680   
## Balanced Accuracy : 0.47863   
##   
## 'Positive' Class : 1   
##

Confusionmatrix\_validknn2 = confusionMatrix(validknn2, as.factor(valid.normal.diff2$Personal.Loan),positive = "1")  
  
Confusionmatrix\_trainknn2

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1678 205  
## 1 563 54  
##   
## Accuracy : 0.6928   
## 95% CI : (0.6743, 0.7108)  
## No Information Rate : 0.8964   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : -0.0265   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.20849   
## Specificity : 0.74877   
## Pos Pred Value : 0.08752   
## Neg Pred Value : 0.89113   
## Prevalence : 0.10360   
## Detection Rate : 0.02160   
## Detection Prevalence : 0.24680   
## Balanced Accuracy : 0.47863   
##   
## 'Positive' Class : 1   
##

Confusionmatrix\_testknn2 = confusionMatrix(testknn2, as.factor(test.normal.diff2$Personal.Loan),positive = "1")  
  
Confusionmatrix\_trainknn2

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1678 205  
## 1 563 54  
##   
## Accuracy : 0.6928   
## 95% CI : (0.6743, 0.7108)  
## No Information Rate : 0.8964   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : -0.0265   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.20849   
## Specificity : 0.74877   
## Pos Pred Value : 0.08752   
## Neg Pred Value : 0.89113   
## Prevalence : 0.10360   
## Detection Rate : 0.02160   
## Detection Prevalence : 0.24680   
## Balanced Accuracy : 0.47863   
##   
## 'Positive' Class : 1   
##