FML_ASSIGNMENT 2

Krishna Krupa Singamshetty

2023-10-01

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

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(e1071)
```

#Question and answers

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?

Ans: 0

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

Ans: K = 3

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

Ans:

```
matrix(c(1786, 63,9,142), ncol=2, byrow=TRUE, dimnames = list(prediction=c(0,1),reference=c(0,1)))
```

```
##            reference
## prediction    0   1
##          0 1786  63
##          1    9 142
```

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.

Ans: 0

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.

Ans:

confusion matrix for train, validation and test are below(refer to the code at the end)

#Differences between accuracy, sensitivity,Pos Pred Value and Neg Pred Value for training, validation and test data sets

#Comparing training with validation

Accuracy: Train data set has a higher accuracy (0.9764) than the validation (0.968).

Sensitivity:Train has higher sensitivity (0.7672) than validation (0.69118).

Specificity :Train has higher specificity (0.9978) than validation (0.99560).

Positive Predictive Value : Train has a higher positive predictive value (0.9727) than validation (0.94000)

Negative predictive value : Train has a higher negative predictive value (0.9767) than validation (0.97000)

##Comparing test with validation

Accuracy: Validation has a higher accuracy (0.968) than Test (0.961).

Sensitivity : Validation has higher sensitivity (0.69118) than Test (0.6875).

Specificity : Validation has higher specificity (0.99560)than Test (0.9955).

Positive Predictive Value: Test has a higher positive predictive value (0.9506) than validation (0.9400).

Negative predictive value : validation data set has a higher negative predictive value (0.97000) than test (0.9619)

Comparing test with train

Accuracy: Training (0.9764) has a slighly higher value than testing (0.961)

sensitivity: Training (0.7672) has a higher sensitivity than testing(0.6875)

specificity: specificity of training (0.9978) is higher than testing (0.9955)

positive predictive value: Training (0.9727) has a higher value than testing (0.9506)

negative predicitive value: training (0.9767) has a higher value than testing (0.9619)

##Reasons

Reasons why validation set and training set has higher accuracy, sensitivity, specificity, positive predictive value and negative predictive value than test set can be:

1. The random data split can lead to unequal data distribution and may cause the model to perform better in some places where there are easier samples and it might not perform better in some cases.
2. The size of the data might be one more reason for differences in the the values. smaller data set and larger dataset may have different values. 3.This can happen when the data of testing is overfitted.

#loading the required libraries

```
library(class)
library(caret)
library(e1071)
```

#Reading the data.

```r
universalbank_df <- read.csv("UniversalBank.csv")
dim(universalbank_df)
```

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

```r
t(t(names(universalbank_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"
```

#Droping ID and ZIP code

```r
universalbank_df <- universalbank_df[,-c(1,5)]
```

Splitting the Data into 60% training and 40% validation. Transform categorical variables into dummy variables

Education is only the categorical variable, so change it to factor

```r
universalbank_df$Education <- as.factor(universalbank_df$Education)
```

convertion of Education to Dummy Variables

```r
groups <- dummyVars(~., data = universalbank_df) # This creates the dummy groups
universal_modification.df <- as.data.frame(predict(groups,universalbank_df))
```

```r
set.seed(1)  # Important to ensure that we get the same sample if we rerun the code
training_index <- sample(row.names(universal_modification.df), 0.6*dim(universal_modification.df)[1])
validate_index <- setdiff(row.names(universal_modification.df), training_index)
train_set <- universal_modification.df[training_index,]
valid_set<- universal_modification.df[validate_index,]
t(t(names(train_set)))
```

```
##       [,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"
```

normalizing the data

```r
training_norm <- train_set[,-10] # Note that Personal Income is the 10th variable
validation_norm <- valid_set[,-10]

normalized_values <- preProcess(train_set[, -10], method=c("center", "scale"))
training_norm <- predict(normalized_values, train_set[, -10])
validation_norm <- predict(normalized_values, valid_set[, -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?

```r
# creating a sample
new_cust <- 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 new customer

```
new.cust<- predict(normalized_values, new_cust)
```

prediction using k-NN

```
knn.pred <- class::knn(train = training_norm,
                       test = new.cust,
                       cl = train_set$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?

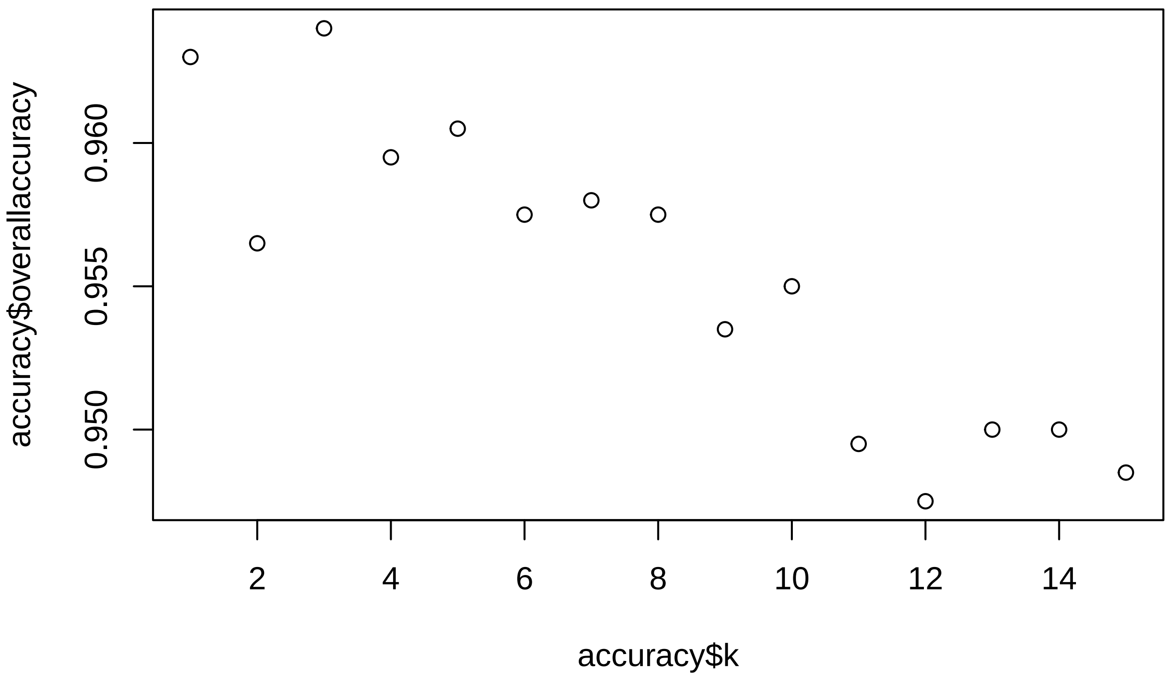
```
# Calculation of accuracy for each value of k
# Set the range of k values to consider

accuracy <- data.frame(k = seq(1, 15, 1), overallaccuracy = rep(0, 15))
for(i in 1:15) {
  knn.pred <- class::knn(train = training_norm,
                         test = validation_norm,
                         cl = train_set$Personal.Loan, k = i)
  accuracy[i, 2] <- confusionMatrix(knn.pred,
                                    as.factor(valid_set$Personal.Loan),positive = "1")$overall[1]
}

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

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

```
plot(accuracy$k,accuracy$overallaccuracy)
```



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

#Creating a Confusion Matrix for best K=3

```
knn.pred <- class::knn(train = training_norm,
                         test = validation_norm,
                         cl = train_set$Personal.Loan, k = 3)
```

```
confusionMatrix(knn.pred, as.factor(valid_set$Personal.Loan), positive = "1")
```

```
## 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.

#The new customer is already normalized in question 1, so using the same data

```
knn.prediction <- class::knn(train = training_norm,
                      test = new.cust,
                      cl = train_set$Personal.Loan, k = 3)
knn.prediction
```

```
## [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.

Spliting the data into 50% training, 30% Validation and 20% Testing

```
set.seed(1)
Training.1 <- sample(row.names(universal_modification.df), 0.5*dim(universal_modification.df)[1])
validation1 <- sample(setdiff(row.names(universal_modification.df),Training.1),0.3*dim(universal_modifi
Testing1 <-setdiff(row.names(universal_modification.df),union(Training.1,validation1))
Training_data_set <- universal_modification.df[Training.1,]
Valid_Data_set <- universal_modification.df[validation1,]
Testing_Data_set <- universal_modification.df[Testing1,]
```

Normalizing the data

```
normalized_train <- Training_data_set[,-10]
norm.valid <- Valid_Data_set[,-10]
norm.test  <-Testing_Data_set[,-10]

normalized.values <- preProcess(Training_data_set[, -10], method=c("center", "scale"))
```

```r
Train1 <- predict(normalized.values, normalized_train)
valid1 <- predict(normalized.values, norm.valid)
test1 <-predict(normalized.values,norm.test)
```

prediction using K-NN(k- Nearest neighbors)

```r
knn_train = class::knn(train = Train1,
                     test = Train1,
                     cl = Training_data_set$Personal.Loan,
                     k = 3)

knn_validation = class::knn(train = Train1,
                           test = valid1,
                           cl = Training_data_set$Personal.Loan,
                           k = 3)

knn_testing = class::knn(train = Train1,
                     test = test1,
                     cl = Training_data_set$Personal.Loan,
                     k = 3)
```

#confusion matrix for training data set

```r
Train.confusion.matrix = confusionMatrix(knn_train,
                                          as.factor(Training_data_set$Personal.Loan),
                                          positive = "1")

Train.confusion.matrix
```

```
## 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
##
```

###confusion Matrix for validation data set

```
validation.confusion.matrix = confusionMatrix(knn_validation,
                                              as.factor(Valid_Data_set$Personal.Loan),
                                              positive = "1")

validation.confusion.matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1358   42
##          1    6   94
##
##                Accuracy : 0.968
##                  95% CI : (0.9578, 0.9763)
##     No Information Rate : 0.9093
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7797
##
##  Mcnemar's Test P-Value : 4.376e-07
##
##             Sensitivity : 0.69118
##             Specificity : 0.99560
##          Pos Pred Value : 0.94000
##          Neg Pred Value : 0.97000
##              Prevalence : 0.09067
##          Detection Rate : 0.06267
##    Detection Prevalence : 0.06667
##       Balanced Accuracy : 0.84339
##
##        'Positive' Class : 1
##
```

Test confusion Matrix

```
test.confusion.matrix = confusionMatrix(knn_testing,
                                        as.factor(Testing_Data_set$Personal.Loan),
                                        positive = "1")



test.confusion.matrix
```

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

```
##
##           Reference
## Prediction   0   1
##          0 884  35
##          1   4  77
##
##                Accuracy : 0.961
##                  95% CI : (0.9471, 0.9721)
##     No Information Rate : 0.888
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.777
##
##  Mcnemar's Test P-Value : 1.556e-06
##
##             Sensitivity : 0.6875
##             Specificity : 0.9955
##          Pos Pred Value : 0.9506
##          Neg Pred Value : 0.9619
##              Prevalence : 0.1120
##          Detection Rate : 0.0770
##    Detection Prevalence : 0.0810
##       Balanced Accuracy : 0.8415
##
##        'Positive' Class : 1
##
```

#Differences between accuracy, sensitivity,Pos Pred Value and Neg Pred Value for training, validation and test data sets

#Comparing training with validation

Accuracy: Train data set has a higher accuracy (0.9764) than the validation (0.968).

Sensitivity:Train has higher sensitivity (0.7672) than validation (0.69118).

Specificity :Train has higher specificity (0.9978) than validation (0.99560).

Positive Predictive Value : Train has a higher positive predictive value (0.9727) than validation (0.94000)

Negative predictive value : Train has a higher negative predictive value (0.9767) than validation (0.97000)

##Comparing test with validation

Accuracy: Validation has a higher accuracy (0.968) than Test (0.961).

Sensitivity : Validation has higher sensitivity (0.69118) than Test (0.6875).

Specificity : Validation has higher specificity (0.99560)than Test (0.9955).

Positive Predictive Value: Test has a higher positive predictive value (0.9506) than validation (0.9400).

Negative predictive value : validation data set has a higher negative predictive value (0.97000) than test (0.9619)

Comparing test with train

Accuracy: Training (0.9764) has a slighly higher value than testing (0.961)

sensitivity: Training (0.7672) has a higher sensitivity than testing(0.6875) specificity: specificity of training (0.9978) is higher than testing (0.9955)

positive predictive value: Training (0.9727) has a higher value than testing (0.9506)

negative predicitive value: training (0.9767) has a higher value than testing (0.9619)

##Reasons

Reasons why validation set and training set has higher accuracy, sensitivity, specificity, positive predictive value and negative predictive value than test set can be:

1. The random data split can lead to unequal data distribution and may cause the model to perform better in some places where there are easier samples and it might not perform better in some cases.
2. The size of the data might be one more reason for differences in the the values. smaller data set and larger dataset may have different values. 3.This can happen when the data of testing is overfitted.