

Assignment_2_KNN

Yaswanth kumar Golla

2023-09-12

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

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

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

```
library(e1071)
```

```
universal.df <- read.csv("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"
```

```
universal.df <- universal.df[, -c(1,5)]
```

Split Data into 60% training and 40% validation.

```

# 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_m.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_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"

```

#Second approach

```

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)

# 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"
```

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

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

```

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])
```

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
)

new.cust.norm <- new_customer
new.cust.norm <- predict(norm.values, new.cust.norm)
```

```
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?

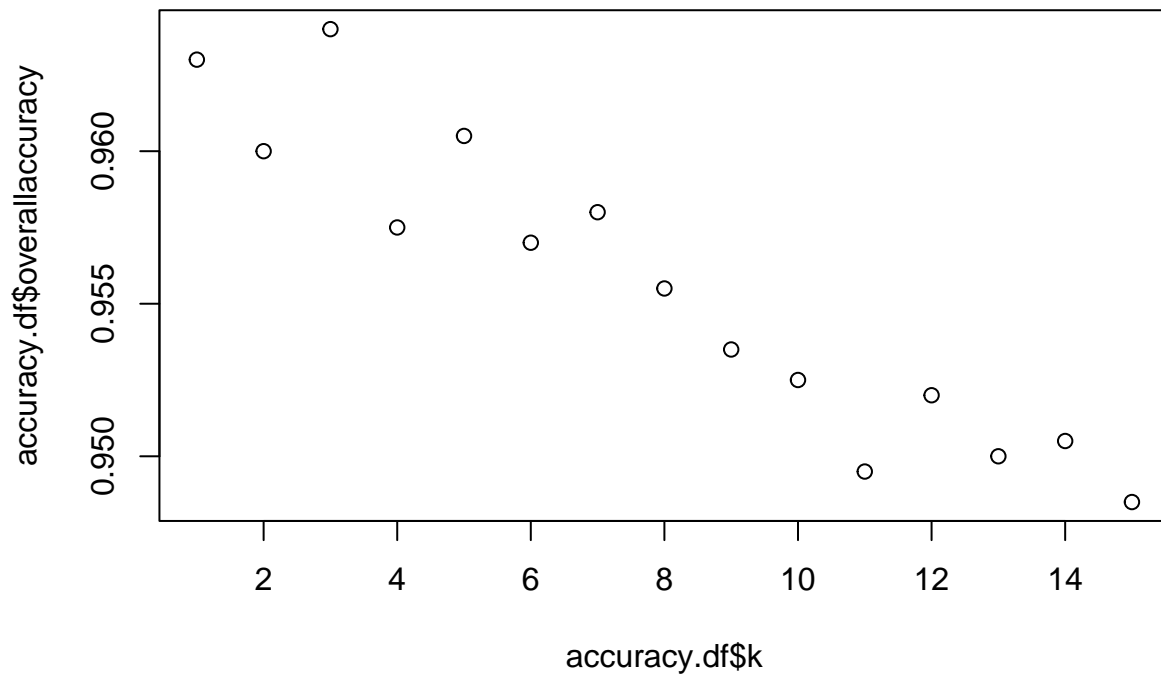
```
# Calculate the accuracy for each value of k
# Set the range of k values to consider

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]
}
```

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

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

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



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

```
knn.prediction.1 <- class::knn(train = train.norm.df,
                                test = valid.norm.df,
                                cl = train.df$Personal.Loan, k = 3)

confusionMatrix(knn.prediction.1,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.

```
new_customer2<-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,
  CDAccount = 0,
  Online = 1,
  CreditCard = 1)
```

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

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

```
print("This customer is classified as: Loan Rejected")
```

```
## [1] "This customer is classified as: Loan Rejected"
```

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?

```

set.seed(1)
Training_Index1 <- sample(row.names(universal_m.df), 0.5*dim(universal_m.df)[1])
Validation_Index1 <- sample(setdiff(row.names(universal_m.df), Training_Index1), 0.3*dim(universal_m.df)[1])
Testing_Index1 <- setdiff(row.names(universal_m.df), union(Training_Index1, Validation_Index1))
Train_Data <- universal_m.df[Training_Index1,]
Validation_Data <- universal_m.df[Validation_Index1,]
Test_Data <- universal_m.df[Testing_Index1,]

```

```

training.norm <- Train_Data[,-10]
validation.norm <- Validation_Data[,-10]
Test.norm.df1 <- Test_Data[,-10]

norm.values1 <- preProcess(Train_Data[, -10], method=c("center", "scale"))
train.norm.df1 <- predict(norm.values1, Train_Data[,-10])
valid.norm.df1 <- predict(norm.values1, Validation_Data[,-10])
Test.norm.df1 <- predict(norm.values1, Test_Data[,-10])

```

```

validate_knn = class::knn(train = training.norm,
                          test = validation.norm,
                          cl = Train_Data$Personal.Loan,
                          k = 3)

```

```

testing_knn = class::knn(train = training.norm,
                          test = Test.norm.df1,
                          cl = Train_Data$Personal.Loan,
                          k = 3)

```

```

Training_knn = class::knn(train = training.norm,
                           test = train.norm.df1,
                           cl = Train_Data$Personal.Loan,
                           k = 3)

```

```

validate_confusion = confusionMatrix(validate_knn,
                                     as.factor(Validation_Data$Personal.Loan),
                                     positive = "1")

```

```

validate_confusion

```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 1315   79
##           1   49   57
##
##           Accuracy : 0.9147
##           95% CI : (0.8994, 0.9283)
##           No Information Rate : 0.9093
##           P-Value [Acc > NIR] : 0.25216
##
##           Kappa : 0.4254
##

```

```
## McNemar's Test P-Value : 0.01037
##
##      Sensitivity : 0.41912
##      Specificity : 0.96408
##      Pos Pred Value : 0.53774
##      Neg Pred Value : 0.94333
##      Prevalence : 0.09067
##      Detection Rate : 0.03800
##      Detection Prevalence : 0.07067
##      Balanced Accuracy : 0.69160
##
##      'Positive' Class : 1
##
```

```
test_confusion = confusionMatrix(testing_knn,
                                  as.factor(Test_Data$Personal.Loan),
                                  positive = "1")
```

```
test_confusion
```

```
## Confusion Matrix and Statistics
##
##      Reference
## Prediction  0   1
##      0 888 112
##      1   0   0
##
##      Accuracy : 0.888
##      95% CI : (0.8668, 0.9069)
##      No Information Rate : 0.888
##      P-Value [Acc > NIR] : 0.5251
##
##      Kappa : 0
##
## McNemar's Test P-Value : <2e-16
##
##      Sensitivity : 0.000
##      Specificity : 1.000
##      Pos Pred Value : NaN
##      Neg Pred Value : 0.888
##      Prevalence : 0.112
##      Detection Rate : 0.000
##      Detection Prevalence : 0.000
##      Balanced Accuracy : 0.500
##
##      'Positive' Class : 1
##
```

```
Training_confusion = confusionMatrix(Training_knn,
                                      as.factor(Train_Data$Personal.Loan),
                                      positive = "1")
```

```
Training_confusion
```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 2268  232
##           1    0    0
##
##           Accuracy : 0.9072
##           95% CI : (0.8951, 0.9183)
##           No Information Rate : 0.9072
##           P-Value [Acc > NIR] : 0.5175
##
##           Kappa : 0
##
## Mcnemar's Test P-Value : <2e-16
##
##           Sensitivity : 0.0000
##           Specificity : 1.0000
##           Pos Pred Value :    NaN
##           Neg Pred Value : 0.9072
##           Prevalence : 0.0928
##           Detection Rate : 0.0000
##           Detection Prevalence : 0.0000
##           Balanced Accuracy : 0.5000
##
##           'Positive' Class : 1
##

```