ASSIGNMENT 2 FML ESWAR DUMPA

Eswar Dumpa

2024-02-25

#Overview

#Questions - Responses

1. How would this customer be classified? A. Since the new client does not take out a personal loan, they would be categorized as 0.
2. What is a choice of k that balances between overfitting and ignoring the predictor information? A. With an overall efficiency of 0, the optimal value of K is 3.
3. Show the confusion matrix for the validation data that results from using the best k. A. By using the best value for K as 3, and at set.seed(159) the confusion matrix was

* Reference
* Prediction 0 1 0 1811 61 1 7 121
* True positive = 121 True Negative = 1811 False Positive = 7 False Negative = 61

4.Classify the customer using the best k? A. Based on the best value of K, which is K=3, the client would be categorized as 0. Thus, the client declines the personal loan.

1. 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.

A. Compared to test and validation data sets, the training set has higher accuracy (97.4%), sensitivity (75.93%), and specificity (99.7%). It was caused by a number of things, including sample size, data leaking, and overfitting. The primary cause was overfitting, which occurs when a model is given permission to commit training data to memory in order to capture all of the training data’s headlines. Consequently, the model’s performance on training data will be remarkably higher than that of the other two data sets.

For Testing data: Accuracy was 95.60% Sensitivity was 60.64% Specificity was 99.23%

For Validation data: Accuracy was 96.13% Sensitivity was 65.51% Specificity was 99.41%

For Training data: Accuracy was 97.44% Sensitivity was 75.93% Specificity was 99.73%

# loaded the required libraries

library(class)  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(e1071)  
library(ggplot2)  
library(lattice)

#Data import   
universal\_bank <- read.csv("C:/Users/eshwa/Documents/Fundamentals of Machine Learning/Assignment 2/UniversalBank.csv")  
dim(universal\_bank)

## [1] 5000 14

# t function creates the transpose of the dataframe  
t(t(names(universal\_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"

#PUT ID AND ZIP

#here 1 and 5 are the indexes for the columns ID and ZIP  
universal\_bank <- universal\_bank[,-c(1,5)]   
dim(universal\_bank)

## [1] 5000 12

#education only need to be converted into factor  
universal\_bank$Education <- as.factor(universal\_bank$Education)  
  
# converting education level to dummy variables  
groups <- dummyVars(~.,data=universal\_bank)  
universal\_B\_bank <- as.data.frame(predict(groups,universal\_bank))

#gives us same sample if we return the code  
set.seed(159)   
  
#60% training data  
train\_index <- sample(row.names(universal\_B\_bank), 0.6\*dim(universal\_B\_bank)[1])  
train\_bank <- universal\_B\_bank[train\_index,]  
  
#40% validation data  
valid\_index <- setdiff(row.names(universal\_B\_bank), train\_index)  
valid\_bank <- universal\_B\_bank[valid\_index,]  
  
# Prints the dims of the datasets  
cat("Training data dimensions:", dim(train\_bank), "\n")

## Training data dimensions: 3000 14

cat("Validation data dimensions:", dim(valid\_bank), "\n")

## Validation data dimensions: 2000 14

# 10th variable of the data frame is personal loan  
train\_norm\_bank <- train\_bank[,-10] # Personal loan is the 10th variable in data frame  
valid\_norm\_bank <- valid\_bank[,-10]  
  
norm.values <- preProcess(train\_bank[, -10], method=c("center", "scale"))  
  
#Normalization of training dataset and validation dataset  
train\_norm\_bank <- predict(norm.values, train\_bank[, -10])  
valid\_norm\_bank <- predict(norm.values, valid\_bank[, -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?

#creating a new customer input  
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  
)  
  
# Normalizing  
new\_cust\_norm <- new\_customer  
new\_cust\_norm <- predict(norm.values, new\_cust\_norm)

#Assuming k=1  
knn.pred1 <- class::knn(train = train\_norm\_bank,   
 test = new\_cust\_norm,   
 cl = train\_bank$Personal.Loan, k = 1)  
  
# Prints knn prediction  
knn.pred1

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

#loan not granted for given test dataset for k=1

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

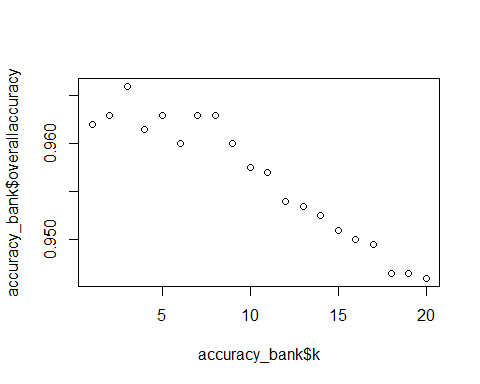
#set range of k 1 to 20  
accuracy\_bank <- data.frame(k = seq(1, 20, 1), overallaccuracy = rep(0, 20))  
  
for(i in 1:20) {  
knn.pred <- class::knn(train = train\_norm\_bank,   
 test = valid\_norm\_bank,   
 cl = train\_bank$Personal.Loan, k = i)  
  
  
accuracy\_bank[i, 2] <- confusionMatrix(knn.pred, as.factor(valid\_bank$Personal.Loan),  
 positive = "1")$overall[1]   
}  
  
#k value with max accuracy  
bestValueofk <- which(accuracy\_bank[,2] == max(accuracy\_bank[,2])) # gives the k value with maximum accuracy  
accuracy\_bank

## k overallaccuracy  
## 1 1 0.9620  
## 2 2 0.9630  
## 3 3 0.9660  
## 4 4 0.9615  
## 5 5 0.9630  
## 6 6 0.9600  
## 7 7 0.9630  
## 8 8 0.9630  
## 9 9 0.9600  
## 10 10 0.9575  
## 11 11 0.9570  
## 12 12 0.9540  
## 13 13 0.9535  
## 14 14 0.9525  
## 15 15 0.9510  
## 16 16 0.9500  
## 17 17 0.9495  
## 18 18 0.9465  
## 19 19 0.9465  
## 20 20 0.9460

#prints the best value of k  
cat("The Best Value of k is:", bestValueofk)

## The Best Value of k is: 3

# The Best Value of k is 3  
#Plotting graph between k value and accuracy  
plot(accuracy\_bank$k,accuracy\_bank$overallaccuracy)



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

# take the best k value for prediction  
knn.pred2 <- class::knn(train = train\_norm\_bank,   
 test = valid\_norm\_bank,   
 cl = train\_bank$Personal.Loan, k = bestValueofk)  
  
  
# confusion matrix for dataset  
confusion\_matrix <- confusionMatrix(knn.pred2,  
 as.factor(valid\_bank$Personal.Loan), positive = "1")  
  
cat("Confusion Matrix for validation data:", "\n")

## Confusion Matrix for validation data:

# Confusion Matrix of validation data  
print(confusion\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1811 61  
## 1 7 121  
##   
## Accuracy : 0.966   
## 95% CI : (0.9571, 0.9735)  
## No Information Rate : 0.909   
## P-Value [Acc > NIR] : < 2e-16   
##   
## Kappa : 0.7628   
##   
## Mcnemar's Test P-Value : 1.3e-10   
##   
## Sensitivity : 0.6648   
## Specificity : 0.9961   
## Pos Pred Value : 0.9453   
## Neg Pred Value : 0.9674   
## Prevalence : 0.0910   
## Detection Rate : 0.0605   
## Detection Prevalence : 0.0640   
## Balanced Accuracy : 0.8305   
##   
## 'Positive' Class : 1   
##

1. 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\_customer1 <- 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 new customer  
new\_cust\_norm1 <- new\_customer1  
new\_cust\_norm1 <- predict(norm.values, new\_cust\_norm1)

knn.pred3 <- class::knn(train = train\_norm\_bank,   
 test = new\_cust\_norm1,   
 cl = train\_bank$Personal.Loan, k = bestValueofk)  
  
#prints prediction  
knn.pred3

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

1. 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(159) # Ensures that we get the same sample if we rerun the code  
  
# Split the data to training (50%), validation (30%) and testing (20%) sets each  
train\_index1 <- sample(row.names(universal\_B\_bank), 0.5\*dim(universal\_B\_bank)[1])  
valid\_index1 <- sample(setdiff(row.names(universal\_B\_bank), train\_index1),  
 0.3\*dim(universal\_B\_bank)[1])   
test\_index1 <- setdiff(row.names(universal\_B\_bank), c(train\_index1,valid\_index1))  
  
train\_Data1 <- universal\_B\_bank[train\_index1,]  
valid\_Data1 <- universal\_B\_bank[valid\_index1,]  
test\_Data1 <- universal\_B\_bank[test\_index1,]  
  
# Print dimensions of split datasets  
cat("Training data dimensions:", dim(train\_Data1), "\n")

## Training data dimensions: 2500 14

cat("Validation data dimensions:", dim(valid\_Data1), "\n")

## Validation data dimensions: 1500 14

cat("Testing data dimensions:", dim(test\_Data1), "\n")

## Testing data dimensions: 1000 14

#Normalize data for 3 sets  
train\_norm\_bank1 <- train\_Data1[ ,-10] #removing the 10th variable(personal loan)  
valid\_norm\_bank1 <- valid\_Data1[ ,-10]  
test\_norm\_bank1 <- test\_Data1[ ,-10]  
  
#Preprocessing  
norm.values1 <- preProcess(train\_Data1[ ,-10], method=c("center", "scale"))  
train\_norm\_bank1 <- predict(norm.values1, train\_Data1[ ,-10])  
valid\_norm\_bank1 <- predict(norm.values1, valid\_Data1[ ,-10])  
test\_norm\_bank1 <- predict(norm.values1, test\_Data1[ ,-10])

#knn prediction for best value of k  
knn.pred.train <- class::knn(train = train\_norm\_bank1,   
 test = train\_norm\_bank1,   
 cl = train\_Data1$Personal.Loan, k = 3)  
  
#confusion matrix of training data  
confusion\_matrix.train <- confusionMatrix(knn.pred.train,   
 as.factor(train\_Data1$Personal.Loan), positive = "1")  
  
#print matrix  
cat("Confusion Matrix for training data:", "\n")

## Confusion Matrix for training data:

#Confusion Matrix of training data:  
print(confusion\_matrix.train)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2253 58  
## 1 6 183  
##   
## Accuracy : 0.9744   
## 95% CI : (0.9674, 0.9802)  
## No Information Rate : 0.9036   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8374   
##   
## Mcnemar's Test P-Value : 1.83e-10   
##   
## Sensitivity : 0.7593   
## Specificity : 0.9973   
## Pos Pred Value : 0.9683   
## Neg Pred Value : 0.9749   
## Prevalence : 0.0964   
## Detection Rate : 0.0732   
## Detection Prevalence : 0.0756   
## Balanced Accuracy : 0.8783   
##   
## 'Positive' Class : 1   
##

knn.pred.valid <- class::knn(train = train\_norm\_bank1,   
 test = valid\_norm\_bank1,   
 cl = train\_Data1$Personal.Loan, k = bestValueofk)  
  
#confusion matrix   
confusion\_matrix.valid <- confusionMatrix(knn.pred.valid,   
 as.factor(valid\_Data1$Personal.Loan), positive = "1")  
  
#print matrix  
cat("Confusion Matrix for Validation data:", "\n")

## Confusion Matrix for Validation data:

#Confusion Matrix   
print(confusion\_matrix.valid)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1347 50  
## 1 8 95  
##   
## Accuracy : 0.9613   
## 95% CI : (0.9503, 0.9705)  
## No Information Rate : 0.9033   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7457   
##   
## Mcnemar's Test P-Value : 7.303e-08   
##   
## Sensitivity : 0.65517   
## Specificity : 0.99410   
## Pos Pred Value : 0.92233   
## Neg Pred Value : 0.96421   
## Prevalence : 0.09667   
## Detection Rate : 0.06333   
## Detection Prevalence : 0.06867   
## Balanced Accuracy : 0.82463   
##   
## 'Positive' Class : 1   
##

#knn prediction for best value of k  
knn.pred.test <- class::knn(train = train\_norm\_bank1,   
 test = test\_norm\_bank1,   
 cl = train\_Data1$Personal.Loan, k = bestValueofk)  
  
#confusion matrix   
confusion\_matrix.test <- confusionMatrix(knn.pred.test,   
 as.factor(test\_Data1$Personal.Loan), positive = "1")  
  
#print matrix  
cat("Confusion Matrix for Test data:", "\n")

## Confusion Matrix for Test data:

# Confusion Matrix   
print(confusion\_matrix.test)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 899 37  
## 1 7 57  
##   
## Accuracy : 0.956   
## 95% CI : (0.9414, 0.9679)  
## No Information Rate : 0.906   
## P-Value [Acc > NIR] : 1.733e-09   
##   
## Kappa : 0.6986   
##   
## Mcnemar's Test P-Value : 1.232e-05   
##   
## Sensitivity : 0.6064   
## Specificity : 0.9923   
## Pos Pred Value : 0.8906   
## Neg Pred Value : 0.9605   
## Prevalence : 0.0940   
## Detection Rate : 0.0570   
## Detection Prevalence : 0.0640   
## Balanced Accuracy : 0.7993   
##   
## 'Positive' Class : 1   
##