Assignment 2

Siddhartha CBS

2023-09-12

Summary

1. The given dataframe(new_customer) where k=1 classified as 0, Hence he does not take the personal loan
2. Best k value is 3 that balances between overfitting and ignoring the predictor information
3. The confusion matrix for validation set using k=3 results: Miscalculations:48, Accuracy : 0.964 , Sensitivity : 0.9950, Specificity : 0.6927
4. The given dataframe(new_customer) where k=3 classified as 0, Hence he does not take the personal loan
5. Training set confusion matrix results: Miscalculations: 59, Accuracy : 0.9764, Sensitivity : 0.9978, Specificity : 0.7672

   Validation set confusion matrix results: Miscalculations: 48, Accuracy : 0.968, Sensitivity : 0.9956, Specificity : 0.6912

   Testing set confusion matrix results : Miscalculations:39, Accuracy : 0.961, Sensitivity : 0.9955, Specificity : 0.6875

In summary, the model exhibits good overall performance across the training, validation, and test sets, with slight variations in metrics. Notably, there is a noticeable decrease in specificity as we move from the training set to the validation and test sets. This drop suggests that the model may have a higher tendency to make false positive predictions (classifying instances as class 1 when they are actually class 0) on unseen data.

Questions - Answers

1. How would this customer be classified? when k=1

ANS) This new customer is classified as 0, does not take the personal loan

2. What is a choice of k?

ANS)The best K is 3

3. what are the results of confusion matrix for the validation data with k=3

ANS) Miscalculations:48 Accuracy : 0.964 , Sensitivity : 0.9950, Specificity : 0.6927

4. How would this customer be classified? when k=3

ANS) This new customer is classified as 0, does not take the personal loan

5. Compare the confusion matrix of the test set with that of the training and validation sets.

ANS)Miscalculation: M=false positive + false negative

Training set=59, Validation set=48, Testing set=39

Accuracy: Training set=0.9764, Validation set=0.968, Testing set=0.961

The accuracy on the training set is slightly higher than on the validation and test sets, but all sets have high accuracy, indicating that the model performs well in terms of overall correctness.

Sensitivity: Training set=0.9978, Validation set=0.9956, Testing set=0.9955

Sensitivity measures the models ability to detect the positive class (in this case, class 1) properly. All of the sets have extremely high sensitivity, suggesting that the model is exceptionally good at spotting class 1 cases.

Specificity: Training set=0.7672, Validation set=0.6912, Testing set=0.6875

Specificity measures the models ability to accurately identify the negative class (class 0 in this example). The training set has the maximum specificity, but the test and validation sets have lower specificity values, implying that the model is less accurate at recognizing class 0 occurrences.

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

loaded the required libraries

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

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

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

```
library(e1071)
```

Read the data.

```
universal.df <- read.csv("UniversalBank.csv") #Assigning data to the variable
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"
```

Droped ID and ZIP

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

Split Data into 60% training and 40% validation. There are many ways to do this. We will look at 2 different ways. Before we split, let us transform categorical variables into dummy variables

```
# Education converted into factor
universal.df$Education <- as.factor(universal.df$Education)

# converting Education into Dummy Variables

groups <- dummyVars(~., data = universal.df) # This creates the dummy groups
universal_m.df <- as.data.frame(predict(groups,universal.df))

#Split Data into 60% training and 40% validation

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
#install.packages("caTools")
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"
```

normalizing the train data & validation 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?

```
# creating 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)
```

predicting using knn

```
knn.pred1 <- class::knn(train = train.norm.df,
                       test = new.cust.norm,
                       cl = train.df$Personal.Loan, k = 1, prob = TRUE)
knn.pred1
```

```
## [1] 0
## attr(,"prob")
## [1] 1
## Levels: 0 1
```

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

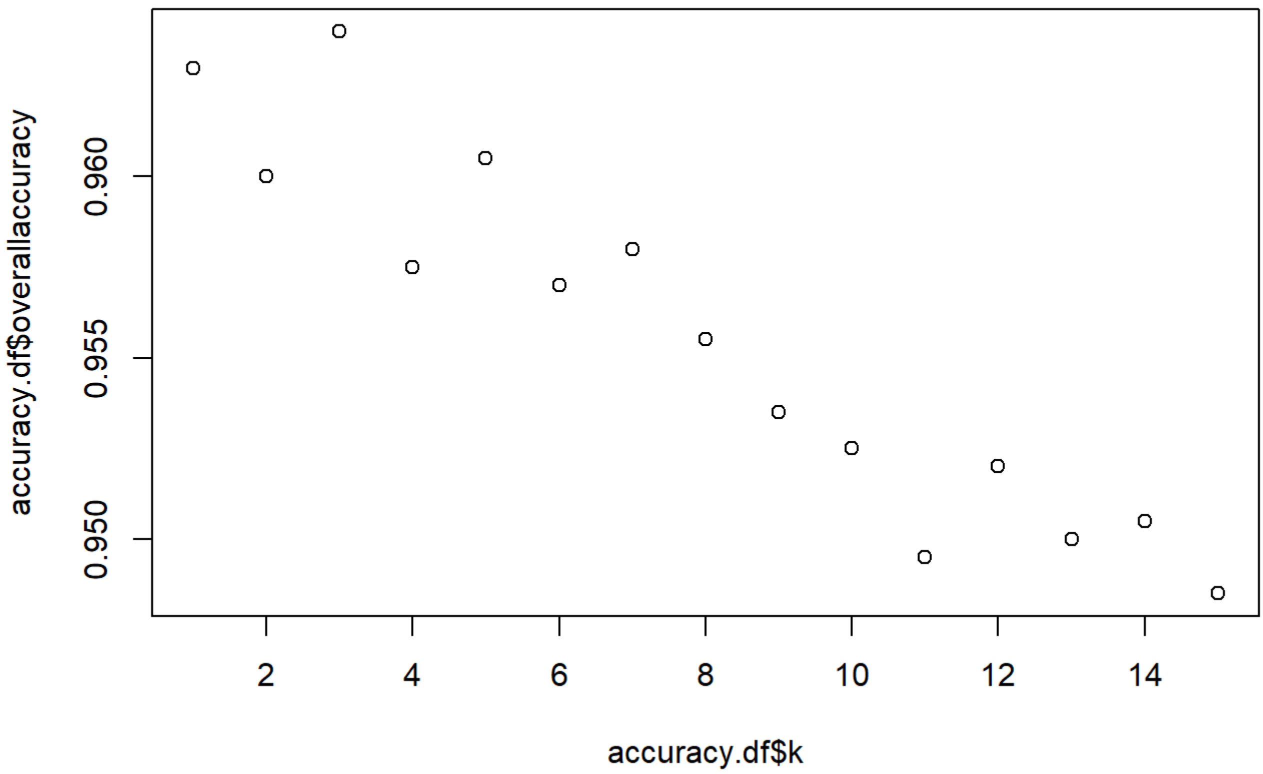
```
# Calculating the accuracy for each value of k
# Setting the range of k values to consider

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")$over
all[1]
}
accuracy.df
```

```
##     k overallaccuracy
## 1   1          0.9630
## 2   2          0.9600
## 3   3          0.9640
## 4   4          0.9575
## 5   5          0.9605
## 6   6          0.9570
## 7   7          0.9580
## 8   8          0.9555
## 9   9          0.9535
## 10 10          0.9525
## 11 11          0.9495
## 12 12          0.9520
## 13 13          0.9500
## 14 14          0.9505
## 15 15          0.9485
```

```
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.

#creating confusion matrix for the validation data using k value as 3

```
prediction_knn <- class::knn(train = train.norm.df, test = valid.norm.df,cl = train.df$Person
al.Loan, k = 3, prob=TRUE)
confusionMatrix(prediction_knn,as.factor(valid.df$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.

```
# creating 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)

knn.pred1 <- class::knn(train = train.norm.df,
 test = new.cust.norm,
cl = train.df$Personal.Loan, k = 3)
knn.pred1
```

```
## [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

```
# Repartition the training, validation and test sets to 50,30, and 20 percents.
set.seed(1)
train.index = sample(row.names(universal_m.df), 0.5*dim(universal_m.df)[1])
remaining.index = setdiff(row.names(universal_m.df),train.index)
valid.index = sample(remaining.index,0.3*dim(universal_m.df)[1])
test.index = setdiff(remaining.index,valid.index)

#Loading the partitioned data into the dataframe.
train.df = universal_m.df[train.index,]
valid.df= universal_m.df[valid.index,]
test.df = universal_m.df[test.index,]

#Normalizing the data after repartitioning accordingly.
train.norm.df <- train.df[, -10]
valid.norm.df <- valid.df[, -10]
test.norm.df <- test.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])
test.norm.df <- predict(norm.values, test.df[, -10])

#Applying the k-NN method to all the sets that we have using k value as 3.

#Training set
knn_train <- class::knn(train = train.norm.df,test = train.norm.df, cl = train.df$Personal.Lo
an, k =3)

#Validation set
knn_valid <- class::knn(train = train.norm.df,test = valid.norm.df,cl = train.df$Personal.Loa
n, k =3)

#Test set
knn_test <- class::knn(train = train.norm.df,test = test.norm.df, cl = train.df$Personal.Loa
n, k =3)

#Confusion Matrix
#Training set confusion matrix
confusionMatrix(knn_train, as.factor(train.df$Personal.Loan))
```

```
## 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 : 0.9978
##             Specificity : 0.7672
##          Pos Pred Value : 0.9767
##          Neg Pred Value : 0.9727
##              Prevalence : 0.9072
##          Detection Rate : 0.9052
##    Detection Prevalence : 0.9268
##       Balanced Accuracy : 0.8825
##
##        'Positive' Class : 0
##
```

```
#Validation set confusion matrix
confusionMatrix(knn_valid, as.factor(valid.df$Personal.Loan))
```

```
## 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 : 0.9956
##             Specificity : 0.6912
##          Pos Pred Value : 0.9700
##          Neg Pred Value : 0.9400
##              Prevalence : 0.9093
##          Detection Rate : 0.9053
##    Detection Prevalence : 0.9333
##       Balanced Accuracy : 0.8434
##
##        'Positive' Class : 0
##
```

```r
#Testing set confusion matrix
confusionMatrix(knn_test, as.factor(test.df$Personal.Loan))
```

```
## 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 : 0.9955
##             Specificity : 0.6875
##          Pos Pred Value : 0.9619
##          Neg Pred Value : 0.9506
##              Prevalence : 0.8880
##          Detection Rate : 0.8840
##    Detection Prevalence : 0.9190
##       Balanced Accuracy : 0.8415
## 
##        'Positive' Class : 0
## 
```