FML_Assignment_2

Aditya Ashish Kulkarni

2024-02-24

Directions

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 required and cleaning of data

```r
#loading the library

library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(class)

#Loading required package: ggplot2
#Loading required package: lattice
library(ggplot2)
library(lattice)
```

Importing and Reading data

```r
#setting working directory
setwd("/Users/aditya/Documents/Spring 24 Classes /FML/Assignments/Assignment_2")

#importing csv file

library(readr)
bank.df <- read_csv("UniversalBank.csv")
```

```
## Rows: 5000 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): ID, Age, Experience, Income, ZIP Code, Family, CCAvg, Education, M...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#using dim
dim(bank.df)
```

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

to make classification with all predictors except ID and ZIP code, Dropping ID and ZIP code…

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

For partitioning data in to 60% training and 40% testing, we need to make dummy variable, hare I am making education as dummy because it has three different categorical value - 1, 2 & 3…

```
#transform categorical predictors with more than two categories into dummy variables
# Education and Family need to be converted to factor

bank.df$Education <- as.factor(bank.df$Education)
bank.df$Family <- as.factor(bank.df$Family)

#converting both into dummy variables

dummy <-  dummyVars(~., data = bank.df)

#above will create dummy groups...

bank_dummy.df <-  as.data.frame(predict(dummy, bank.df))

set.seed(69)

training.index <- sample(row.names(bank_dummy.df), 0.6*dim(bank_dummy.df)[1])
validate.index <- setdiff(row.names(bank_dummy.df), training.index)

training.df <- bank_dummy.df[training.index,]
validate.df <- bank_dummy.df[validate.index,]


cat("Size of Training set:", nrow(training.df))
```

```
## Size of Training set: 3000
```

```
cat("Size of Validation set:", nrow(validate.df))
```

```
## Size of Validation set: 2000
```

```r
#Now Dropping 13th attribute since it is need to be predicted

normalization <- preProcess(training.df[, -13], method = c("center", "scale"))
training.normalization.df <- predict(normalization, training.df[, -13])
validation.normalization.df <-  predict(normalization, validate.df[, -13])
```

Questions :

Question - 1 : Age = 40, Experience = 10, Income = 84, 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 new data and dataframe and normalizing it...

new.customer.df <- data.frame(
  Age = 40,
  Experience = 10,
  Income = 84,
  Family.1 = 0,
  Family.2 = 1,
  Family.3 = 0,
  Family.4 = 0,
  CCAvg = 2,
  Education.1 = 0,
  Education.2 = 1,
  Education.3 = 0,
  Mortgage = 0,
  '`Securities Account`' = 0,
  '`CD Account`' = 0,
  Online = 1,
  CreditCard = 1,
check.names = F
)


#Normalizing the new bank customer

new.customer.df <- predict(normalization, new.customer.df)
```

Predict Using Knn

```r
#knn


p <- class::knn(train = training.normalization.df,
                test = new.customer.df,
                cl = training.df$`\`Personal Loan\``,
                k =1)

p
```

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

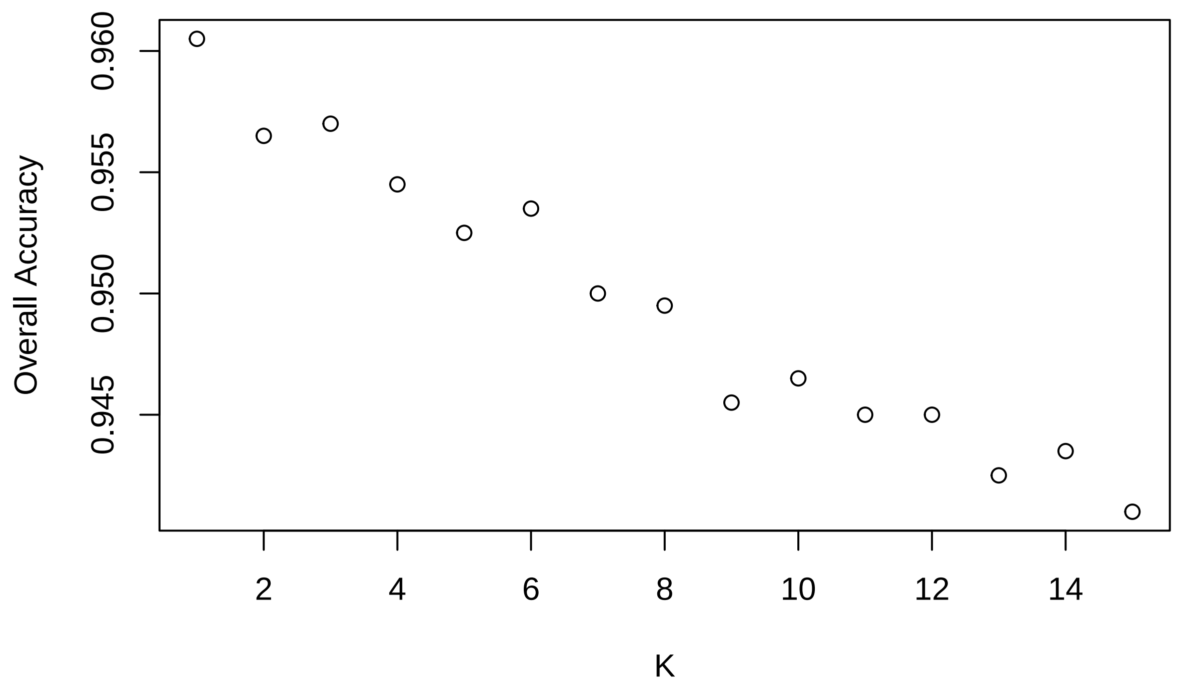
Here, Customer will not take personal loan…

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

```
#calculate the accuracy for different values of k ...

#setting range of K values to caonsider

accuracy.df <-  data.frame(k = seq(1,15,1), overallaccuracy = rep(0,15))
for (i in 1:15) {
  predict <- class::knn(train = training.normalization.df,
                        test = validation.normalization.df,
                        cl = training.df$`\`Personal Loan\``,
                        k = i
                          )
  accuracy.df[i,2] <- confusionMatrix(predict,
                                       as.factor(validate.df$`\`Personal Loan\``), positive = "1")$overal


}

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

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

```
plot(accuracy.df$k, accuracy.df$overallaccuracy,xlab = "K", ylab = "Overall Accuracy")
```



As per plot, the best value o K is 1.

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

Confusion matrix for k = 1 can be shown below

```
prediction_2 <- class::knn(train = training.normalization.df,
                           test = validation.normalization.df,
                           cl = training.df$`\`Personal Loan\``, k = 1)

confusionMatrix(prediction_2, as.factor(validate.df$`\`Personal Loan\``), positive = "1")$table
```

```
##           Reference
## Prediction    0    1
##          0 1800   60
##          1   19  121
```

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.

- Making dataframe for seconf customer

```
second.customer.df <- data.frame(
  Age = 40,
  Experience = 10,
  Income = 84,
  Family.1 = 0,
  Family.2 = 1,
  Family.3 = 0,
  Family.4 = 0,
  CCAvg = 2,
  Education.1 = 0,
  Education.2 = 1,
  Education.3 = 0,
  Mortgage = 0,
  '`Securities Account`' = 0,
  '`CD Account`' = 0,
  Online = 1,
  CreditCard = 1,
check.names = F
)

#normalizing second customer

second.customer.df <- predict(normalization, second.customer.df)

#predicting data using knn...

pred_second_customer <- class::knn(train = training.normalization.df,
                                   test = second.customer.df,
                                   cl = training.df$`\`Personal Loan\``, k = 1)

pred_second_customer
```

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

Here, new customer will als not take personal loan…

Question.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.

Making Sets:

```
#Making indexes of data

training_2.index <- sample(row.names(bank_dummy.df), 0.5*dim(bank_dummy.df)[1])
subset.index <- setdiff(row.names(bank_dummy.df), training.index)
subset.df <- bank_dummy.df[subset.index,]
valid_2.index <- sample(row.names(subset.df), 0.6*dim(subset.df)[1])
test_5.index = setdiff(row.names(subset.df), valid_2.index)



#Making Dataset
```

```r
training_2.df <- bank_dummy.df[training_2.index,]
valid_2.df <- bank_dummy.df[valid_2.index,]
test_5.df <- bank_dummy.df[test_5.index,]

#Checking size of set to confirm correct splitting of data

cat("Size of new Training set is:", nrow(training_2.df),"\n")
```

```
## Size of new Training set is: 2500
```

```r
cat("Size of new validation set is:", nrow(valid_2.df), "\n")
```

```
## Size of new validation set is: 1200
```

```r
cat("size of new Test set is:", nrow(test_5.df))
```

```
## size of new Test set is: 800
```

Normalizing datasets-

```r
norm <- preProcess(training_2.df[, -13], method=c("center", "scale"))
training_2.norm.df <- predict(norm, training_2.df[,-13])
valid_2.norm.df <- predict(norm, valid_2.df[,-13])
test_5.norm.df <- predict(norm,test_5.df[,-13])
```

Hence, our splitting data is right.. Now, applying Knn to test and training set:

```r
pred_1 <- class::knn(train = training_2.df,
                     test = test_5.df,
                     cl = training_2.df$`\`Personal Loan\``,
                     k = 1)
confusionMatrix(pred_1, as.factor(test_5.df$`\`Personal Loan\``), positive = "1")$table
```

```
##           Reference
## Prediction   0   1
##          0 709  24
##          1  14  53
```

Applying knn algorithm to train and valid set:

```r
pred_2 <- class::knn(train = valid_2.norm.df,
                     test = test_5.norm.df,
                     cl = valid_2.df$`\`Personal Loan\``,
                     k=1
                     )
confusionMatrix(pred_2, as.factor(test_5.df$`\`Personal Loan\``), positive = "1")$table
```

```
##           Reference
## Prediction   0   1
##          0 713  43
##          1  10  34
```

Applying Knn training set itself

```r
pred_0 <- class::knn(train = training_2.norm.df,
                     test = training_2.norm.df,
                     cl = training_2.df$`\`Personal Loan\``,
                     k = 1)
confusionMatrix(pred_0, as.factor(training_2.df$`\`Personal Loan\``), positive = "1")$table
```

```
##           Reference
## Prediction    0    1
##          0 2279    0
##          1    0  221
```

Conclusion - In k = 1 and Models accuracy is very high around 96%..