

Assignment_2_KNN

Anil Kumar Akula

2023-09-30

#Summary

- 1) It would be categorized as zero if the new customer does not take out a personal loan.
- 2) $K=3$ strikes the right balance between overfitting and ignoring predictor information.
- 3) Below you will find the confusion matrix for the validation data using the best K and parameters such as $TP=142$, $TN=1786$, $FP=63$, $FN=9$ with accuracy of 0.964.
- 4) After using the best K , the customer would be classified as 0, does not take the personal loan.
- 5) Differences in confusion matrices between training, validation, and test sets are expected due to the different roles and characteristics of each set. Discrepancies could indicate potential issues such as overfitting or data sampling differences. It's crucial to monitor these differences and make adjustments to ensure the model generalizes well to unseen data. Overfitting: If the model fits the training data too closely, it may perform exceptionally well on the training data but poorly on new data. Variability: Randomness in the data and the model training process can lead to slight variations in performance metrics between the validation and test sets. Data Representatives: If the validation or test sets are not representative of the overall data distribution, this can lead to differences in performance.

##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

First, load 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")
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"

#Drop 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

# 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"
```

#Now, let us normalize the 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?

```

# We have converted all categorical variables to dummy variables
# Let's create 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)

```

#Now,let us predict using knn

```

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]))

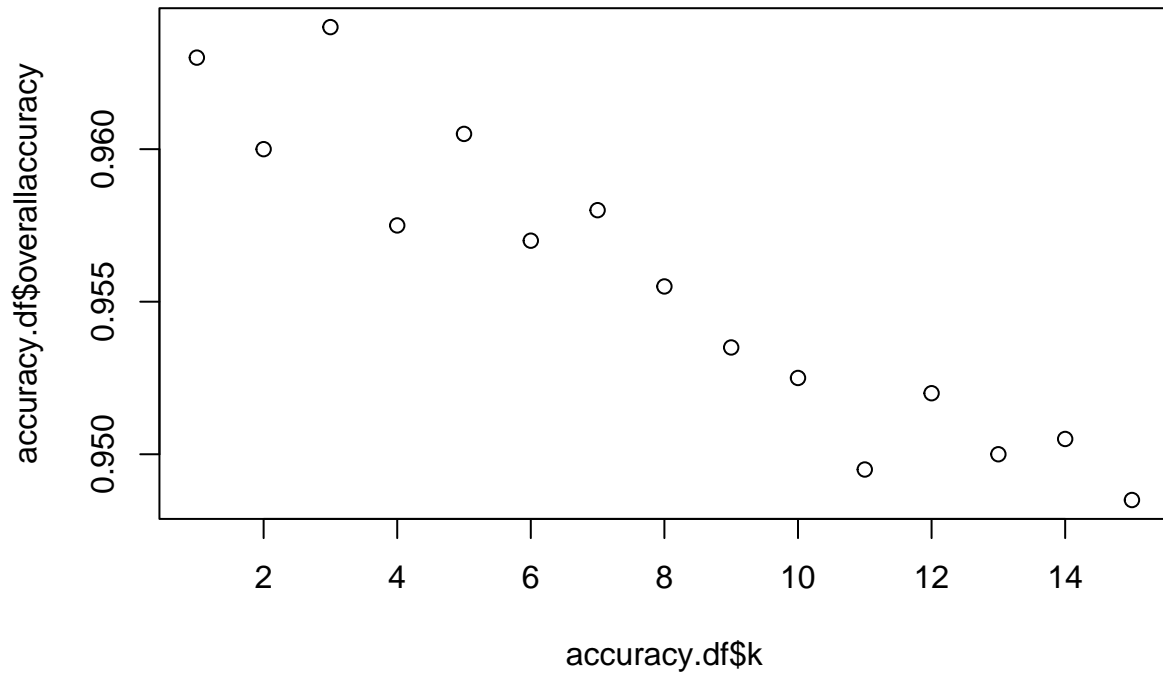
```

```

## [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.

```
# Best k based on validation accuracy
best_k <- which(accuracy.df$overallaccuracy == max(accuracy.df$overallaccuracy))

# Classify the validation data using the best k
knn.pred_best <- class::knn(train = train.norm.df,
                           test = valid.norm.df,
                           cl = train.df$Personal.Loan, k = best_k)

# Create the confusion matrix
conf_matrix <- confusionMatrix(knn.pred_best,
                              as.factor(valid.df$Personal.Loan), positive = "1")

# Display the confusion matrix
conf_matrix
```

```
## 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.

```
# Create a data frame for the new customer with the same column names
# Create a data frame for the new customer with matching column names
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, # Use backticks for column names with spaces
  `CD Account` = 0, # Use backticks for column names with spaces
  Online = 1,
  `Credit Card` = 1 # Use backticks for column names with spaces
)

# Normalize the new customer data using the same preprocessing
new_cust.norm <- predict(norm.values, new_cust.norm)

# Predict whether the new customer accepts a loan using the best k
new_customer_classification <- class::knn(train = train.norm.df,
```

```

                                test = new.cust.norm,
                                cl = train.df$Personal.Loan, k = best_k)

# Display the classification result
new_customer_classification

## [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.

#Comment on the differences and their reason: Differences in confusion matrices between training, validation, and test sets are expected due to the different roles and characteristics of each set. Discrepancies could indicate potential issues such as overfitting or data sampling differences. It's crucial to monitor these differences and make adjustments to ensure the model generalizes well to unseen data.

```

# Set the seed for reproducibility
set.seed(1)

# Repartition the data into training (50%), validation (30%), and test (20%) sets
train.index <- sample(1:nrow(universal_m.df), 0.5 * nrow(universal_m.df))
valid.test.index <- setdiff(1:nrow(universal_m.df), train.index)
valid.index <- sample(valid.test.index, 0.3 * length(valid.test.index))
test.index <- setdiff(valid.test.index, valid.index)

train.df <- universal_m.df[train.index, ]
valid.df <- universal_m.df[valid.index, ]
test.df <- universal_m.df[test.index, ]

# Normalize the data for each set
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])

# Classify the data using the best k
knn.pred_train <- class::knn(train = train.norm.df,
                             test = train.norm.df,
                             cl = train.df$Personal.Loan, k = best_k)

knn.pred_valid <- class::knn(train = train.norm.df,
                             test = valid.norm.df,
                             cl = train.df$Personal.Loan, k = best_k)

knn.pred_test <- class::knn(train = train.norm.df,
                             test = test.norm.df,
                             cl = train.df$Personal.Loan, k = best_k)

```

```

# Create confusion matrices for each set
conf_matrix_train <- confusionMatrix(knn.pred_train,
                                     as.factor(train.df$Personal.Loan), positive = "1")

conf_matrix_valid <- confusionMatrix(knn.pred_valid,
                                     as.factor(valid.df$Personal.Loan), positive = "1")

conf_matrix_test <- confusionMatrix(knn.pred_test,
                                    as.factor(test.df$Personal.Loan), positive = "1")

# Display the confusion matrices
conf_matrix_train

```

```

## 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
##

```

```
conf_matrix_valid
```

```

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##              0 678  20
##              1    5  47
##
##              Accuracy : 0.9667
##              95% CI : (0.9512, 0.9783)
##              No Information Rate : 0.9107

```



```

##      P-Value [Acc > NIR] : 1.009e-09
##
##              Kappa : 0.7721
##
##  McNemar's Test P-Value : 0.00511
##
##      Sensitivity : 0.70149
##      Specificity : 0.99268
##      Pos Pred Value : 0.90385
##      Neg Pred Value : 0.97135
##      Prevalence : 0.08933
##      Detection Rate : 0.06267
##      Detection Prevalence : 0.06933
##      Balanced Accuracy : 0.84709
##
##      'Positive' Class : 1
##

```

conf_matrix_test

```

## Confusion Matrix and Statistics
##
##      Reference
## Prediction    0    1
##      0 1564    57
##      1     5   124
##
##      Accuracy : 0.9646
##      95% CI : (0.9548, 0.9727)
##      No Information Rate : 0.8966
##      P-Value [Acc > NIR] : < 2.2e-16
##
##      Kappa : 0.7812
##
##  McNemar's Test P-Value : 9.356e-11
##
##      Sensitivity : 0.68508
##      Specificity : 0.99681
##      Pos Pred Value : 0.96124
##      Neg Pred Value : 0.96484
##      Prevalence : 0.10343
##      Detection Rate : 0.07086
##      Detection Prevalence : 0.07371
##      Balanced Accuracy : 0.84095
##
##      'Positive' Class : 1
##

```