k-NN for classification - Assignment

Gaurav Kudeshia

2023-09-30

Summary

- Data Splitting:

Divided the “UniversalBank.csv” dataset into a training set containing 60% (3000 observations) and a validation set comprising 40% (2000 observations) out of the total 5000 observations. Additionally, introduced two new variables, expanding the total number of variables to 14.

- Validation Confusion Matrix:

Generated a confusion matrix for the validation data using the optimal k value, resulting in an accuracy of 96.5%. The sensitivity was found to be 69.3%, indicating the model’s ability to correctly identify positive cases, while specificity stood at 99.5%, reflecting its proficiency in recognizing negative cases.

- New Customer Prediction:

Applied the model to predict whether a new customer would accept a personal loan or not. The prediction outcome was “0”, denoting the customer’s decision not to take the loan. The best k value used for this prediction was 3, indicating the nearest neighbors considered for the classification.

- Data Splitting and Confusion Matrix:

Split the data into training, validation, and test sets, allocating 50%, 30%, and 20% of the data, respectively. Constructed a confusion matrix for the test set and further compared its performance with the training and validation sets, evaluating the model’s consistency across different datasets.

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 coNCerting its liability customers to personal loan customers. A campaign that the bank ran last year for liability customers showed a healthy coNCersion 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.

Loaded datasets from the necessary libraries.

Loaded the necessary libraries

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

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

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

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

Now reading the UniversalBank.csv file to A variable

```
A <- read.csv("/Users/gauravkudeshia/Downloads/UniversalBank.csv")
```

Dropped unneccery columns ID and ZIP

```
A <- A[,-c(1,5)]
```

Only Education needs to be converted to factor

```
A$Education <- as.factor(A$Education)
```

Created Dummy Variables

```
F <- dummyVars(~., data = A)
A.df <- as.data.frame(predict(F,A))
```

- Splitted the Data into Training (60%) and Validation (40%) Sets
- Set Seed is used getting the same value when we rerun the code

```
set.seed(1)
T <- sample(row.names(A.df), 0.6*dim(A.df)[1])
V <- setdiff(row.names(A.df), T)
T.df <- A.df[T,]
V.df <- A.df[V,]
```

Normalized

```
T.n.df <- T.df[,-10]
V.n.df <- V.df[,-10]

N.v <- preProcess(T.df[, -10], method=c("center", "scale"))
T.n.df <- predict(N.v, T.df[, -10])
V.n.df <- predict(N.v, V.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?

Firstly, Generated a new data set from the above question for analysis.

Normalized the new customer

```
N_c_n <- N_c
N_c_n <- predict(N.v, N_c_n)
N_c_n
```

```
##          Age Experience    Income     Family     CCAvg Education.1 Education.2
## 1 -0.4774216 -0.8953121 0.2389084 -0.3368482 0.0492415  -0.8461728    1.583646
##   Education.3   Mortgage Securities.Account CD.Account    Online CreditCard
## 1  -0.6509102 -0.5679457         -0.3338946 -0.2380992 0.8426977   1.554365
```

Predicted kNN its “0” that means the above new customer will not take the loan

```
Predict.knn <- class::knn(train = T.n.df,
                      test = N_c_n,
                      cl = T.df$Personal.Loan, k = 1)
Predict.knn
```

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

---

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

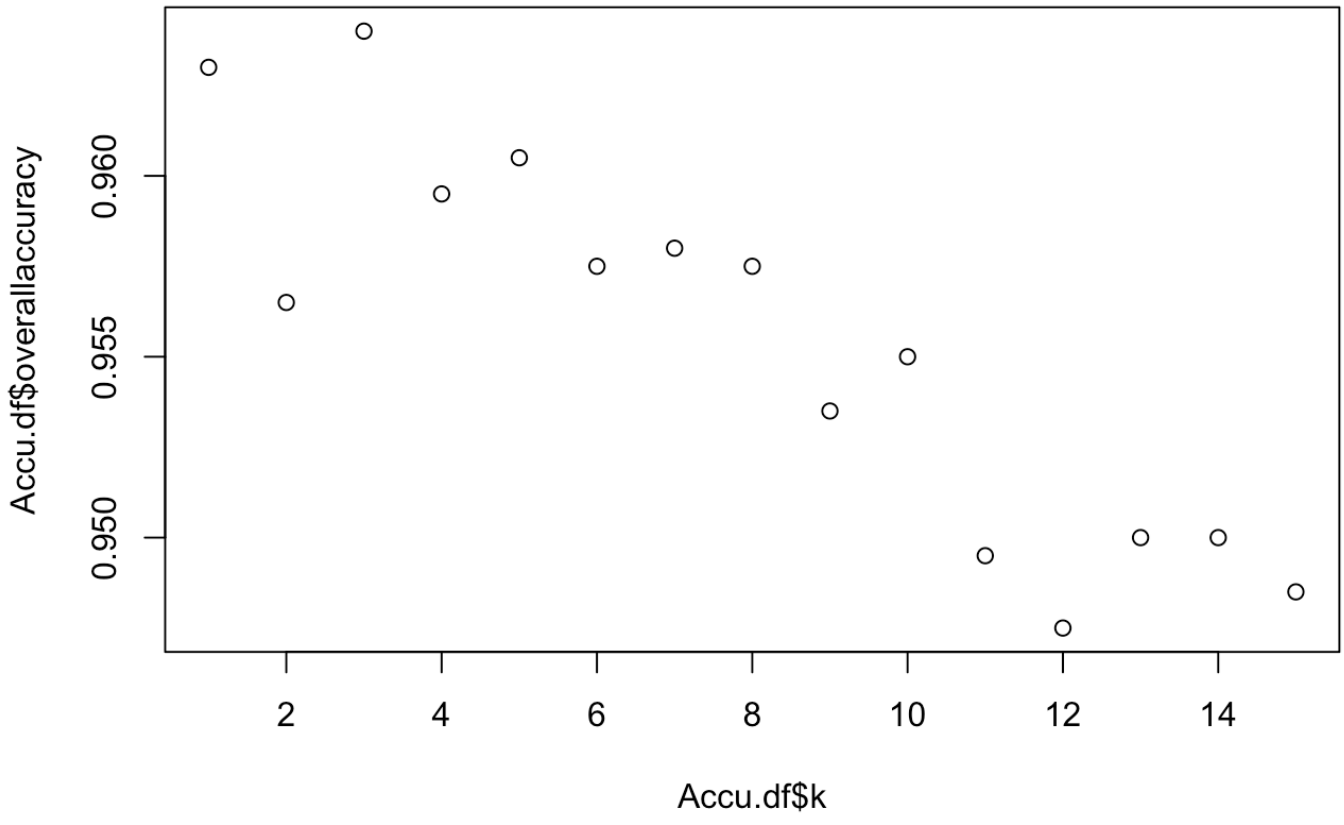
- Calculated the accuracy for each value of k
- Set the range of k values to consider

```
Accu.df <- data.frame(k = seq(1, 15, 1), overallaccuracy = rep(0, 15))
for(i in 1:15) {
  knn.pred <- class::knn(train = T.n.df,
                         test = V.n.df,
                         cl = T.df$Personal.Loan, k = i)
  Accu.df[i, 2] <- confusionMatrix(knn.pred,
                                   as.factor(V.df$Personal.Loan),positive = "1
")$overall[1]
}

k <- which(Accu.df[,2] == max(Accu.df[,2]))
k
```

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

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



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

Calculated the confusion matrix for the validation set

```
Confusion_va <- confusionMatrix(class::knn(train = T.n.df, test = V.n.df, cl = T.d
f$Personal.Loan, k = 3), as.factor(V.df$Personal.Loan), positive = "1")

Confusion_va
```

```
## 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. ***

Firstly, Generated a new data set from the given question for analysis.

Here output is zero that means the new customer did’nt took the loan, Knn Predict

```
Predict_knn <- class::knn(train = Trx.nor.df, test = NC_Normalized,cl = T.df$Perso
nal.Loan, k = 1)
Predict_knn
```

```
## [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. ***

Used “set.seed” for maintaining the same result all the time, Additionally we have trained training, validation, and test sets (50% : 30% : 20%)

```
set.seed(1)
Tr <- sample(row.names(A.df), 0.5*dim(A.df)[1])
Va <- sample(setdiff(row.names(A.df), Tr), 0.3*dim(A.df)[1])
Te <- setdiff(row.names(A.df),union(Tr,Va))
Tr1 <- A.df[Tr,]
Va1 <- A.df[Va,]
Te1 <- A.df[Te,]
```

Normalized data

```
Tr2 <- Tr1[,-10]
Va2 <- Va1[,-10]
Te2 <- Te1[,-10]

NormV <- preProcess(Tr1[, -10], method=c("center","scale"))
Tr2 <- predict(NormV, Tr1[, -10])
Va2 <- predict(NormV, Va1[, -10])
Te2 <- predict(NormV, Te1[, -10])
```

Size of the training and validation has been divided into 50:30:20 ratio

```
dim(Tr1)
```

```
## [1] 2500   14
```

```
dim(Va1)
```

```
## [1] 1500   14
```

```
dim(Te1)
```

```
## [1] 1000   14
```

Confusion Matrix - Calculated confusion matrix for training, validation sets & test set

```
# Calculated the confusion matrix for the training set
Confusion.Tr <- confusionMatrix(class::knn(train = Tr2, test = Tr2, cl = Tr1$Perso
nal.Loan, k = 3), as.factor(Tr1$Personal.Loan), positive = "1")

# Calculated the confusion matrix for the validation set
Confusion.Va <- confusionMatrix(class::knn(train = Tr2, test = Va2, cl = Tr1$Perso
nal.Loan, k = 3), as.factor(Va1$Personal.Loan), positive = "1")

# Calculated the confusion matrix for the test set
Confusion.Te <- confusionMatrix(class::knn(train = Tr2, test = Te2, cl = Tr1$Perso
nal.Loan, k = 3), as.factor(Te1$Personal.Loan), positive = "1")

# Printed the all of them
print(Confusion.Tr)
```

```
## 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
##
```

```
print(Confusion.Va)
```

```
## 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 : 0.69118
##             Specificity : 0.99560
##          Pos Pred Value : 0.94000
##          Neg Pred Value : 0.97000
##              Prevalence : 0.09067
##          Detection Rate : 0.06267
##    Detection Prevalence : 0.06667
##       Balanced Accuracy : 0.84339
##
##        'Positive' Class : 1
##
```

```
print(Confusion.Te)
```

```
## 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 : 0.6875
##             Specificity : 0.9955
##          Pos Pred Value : 0.9506
##          Neg Pred Value : 0.9619
##              Prevalence : 0.1120
##          Detection Rate : 0.0770
##    Detection Prevalence : 0.0810
##       Balanced Accuracy : 0.8415
##
##        'Positive' Class : 1
##
```

- Compared the confusion matrix test set with that of the training and validation sets and commented on the differences and their reason.

Comparison:

Sensitivity (True Positive Rate): Training Set > Validation Set > Test Set The training set has the highest sensitivity, indicating it performs best at correctly identifying positive cases. The validation set follows, and the test set has slightly lower sensitivity.

Specificity (True Negative Rate): Training Set > Validation Set > Test Set All sets have high specificity, meaning they are effective at correctly identifying negative cases. The training set has the highest specificity, followed by the validation set and then the test set.

Positive Predictive Value (Precision): Training Set > Test Set > Validation Set The training set has the highest precision, indicating that when it predicts a positive case, it is highly likely to be correct. The test set follows, and the validation set has slightly lower precision.

Accuracy: Training Set > Validation Set > Test Set All sets have high accuracy, but the training set has the highest accuracy, indicating it performs best overall. The validation set follows, and the test set has slightly lower accuracy.

The following are the comments differences and their reason:-

---

Sensitivity: The model is good at identifying positive cases, but slightly better on the training set than the validation or test sets.

Specificity: The model is good at identifying negative cases in all datasets.

Precision: The model is more likely to be correct when it predicts a positive case in the training set, but is still relatively high in the validation and test sets.

Accuracy: The model performs well overall in all data sets, but accuracy may not be the best metric to use if the classes are imbalanced.

Balanced Accuracy: The model balances sensitivity and specificity well in all data sets, but slightly better in the validation and test sets than the training set.