Assignment 2

2023-09-29

Summary

The goal of this code is to predict if a customer would accept a personal loan offer by doing k-NN (k-Nearest Neighbors) classification on a dataset containing customer information. The code includes parts for model training, evaluation, and data preprocessing.

The dataset is read from a CSV file. Columns "ID" and "ZIP Code" are dropped as they are not relevant for modeling. Categorical variables, specifically the "Education" column, are converted into dummy variables.

The dataset is split into a training set (60%) and a validation set (40%). The sample function is used for this purpose.Normalization is applied to both the training and validation sets using the mean and standard deviation from the training set.

We normalize the data for an imaginary customer using the given attributes.

The accuracy is used to determine the choice of k for the k-NN model, which is trained on the training data.The model is used to predict this customer's choice about a personal loan offer. The result in this case is "0," meaning the customer is categorized as not taking a personal loan.

A range of k values (from 1 to 15) are considered to choose the best k value.

Accuracy is calculated for each k and also a plot is generated , and the best k is selected based on the highest accuracy. In this case, k = 3 is chosen.

The confusion matrix for the validation data using the best k (k = 3) is displayed. It provides information about true positives, true negatives, false positives, and false negatives.

Then the data is repartitioned into training (50%), validation (30%), and test (20%) sets and k-NN method is applied with k = 3 to each of them.

Confusion matrices for the training, validation, and test sets are calculated and displayed.We note that the training set has the highest accuracy, likely due to overfitting. The validation set has lower accuracy, and the test set accuracy is higher than that of the validation set, possibly because it contains examples more similar to the training set.

Questions - Answers

1. How would this customer be classified? This new customer would be classified as 0, does not take the personal loan.
2. The best K is 3
3. The confusion matrix for validation data with k=3 was shown with accuracy of 0.964
4. Classify the customer using the best k? The new customer would be classified as 0, does not take the personal loan

5. Comparing confusion matrices of training,validation and testing : The accuracy of the training set is high, the accuracy of the validation set is low, and the accuracy of the test set is higher than the accuracy of the validation set,which occurs when a model is overfit to the training data.

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("/Users/ritikakalyani/Downloads/FML/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.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_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_df),0.6*dim(universal_df)[1])
valid.index <- setdiff(row.names(universal_df),train.index)
train.df <- universal_df[train.index,]
valid.df <- universal_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"
```

```
# Print the sizes of the training and validation sets
print(paste("The size of the training set is:", nrow(train.df)))
```

```
## [1] "The size of the training set is: 3000"
```

```
print(paste("The size of the validation set is:", nrow(valid.df)))
```

```
## [1] "The size of the validation set is: 2000"
```

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
```

The value of prediction is '0' which means that the new customer is classified as '0', does not take personal loan

---

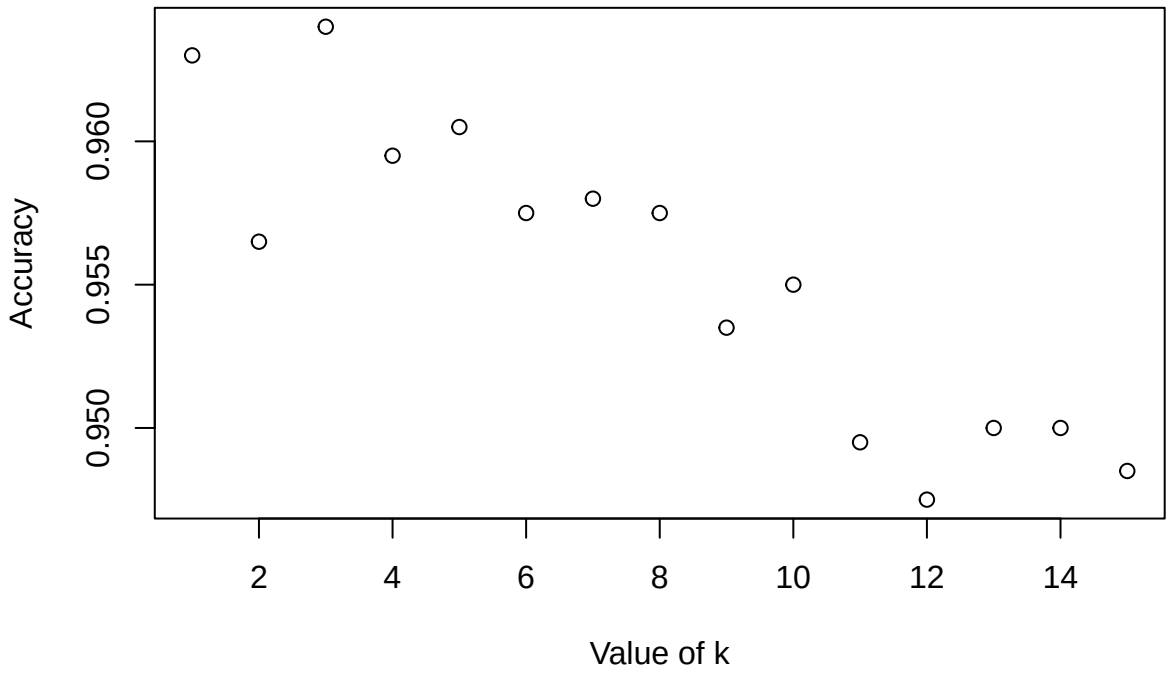
2. What is a choice of k that balances between overfitting and ignoring the predictor information?

```
# 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,xlab = "Value of k",ylab = "Accuracy")
```



The best value of k is "3"

---

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

```
knn.pred2 <- class::knn(train = train.norm.df,
                         test = valid.norm.df,
                         cl = train.df$Personal.Loan, k = 3)

accuracy.matrix <- confusionMatrix(knn.pred2,
                                       as.factor(valid.df$Personal.Loan),positive = "1")
accuracy.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.

```
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
)

# Normalize the new customer

new.cust.norm1 <- new_customer1
new.cust.norm1 <- predict(norm.values, new.cust.norm1)
```

```
# Now, let us predict using knn
knn.pred3 <- class::knn(train = train.norm.df,
                        test = new.cust.norm1,
                        cl = train.df$Personal.Loan, k = 3)
knn.pred3
```

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

The value of prediction for $k = 3$ is "0" which means the given customer is classified as "0",does not take personal loan

---

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.

```
#Partitioning data into training,validation and testing(50% : 30% : 20%)
train.index1 <- sample(row.names(universal_df), 0.5*dim(universal_df)[1])
valid.index1 <- sample(setdiff(row.names(universal_df), train.index1), 0.3*dim(universal_df)[1])
test.index1 <- setdiff(setdiff(row.names(universal_df), train.index1), valid.index1)
train.df1 <- universal_df[train.index1,]
valid.df1 <- universal_df[valid.index1,]
test.df1 <- universal_df[test.index1,]
# Print the sizes of the training,validation and test sets
print(paste("The size of the training set is:", nrow(train.df1)))
```

```
## [1] "The size of the training set is: 2500"
```

```
print(paste("The size of the validation set is:", nrow(valid.df1)))
```

```
## [1] "The size of the validation set is: 1500"
```

```
print(paste("The size of the testing set is:", nrow(test.df1)))
```

```
## [1] "The size of the testing set is: 1000"
```

```
# Now, let us normalize the data
train.norm.df1 <- train.df1[,-10] # Note that Personal Income is the 10th variable
valid.norm.df1 <- valid.df1[,-10]
test.norm.df1 <- test.df1[-10]
norm.values1 <- preProcess(train.df1[, -10], method=c("center", "scale"))
train.norm.df1 <- predict(norm.values1, train.df1[, -10])
valid.norm.df1 <- predict(norm.values1, valid.df1[, -10])
test.norm.df1 <-  predict(norm.values1, test.df1[-10])
```

Confusion matrix for training set:

```
knn.pred4 <- class::knn(train = train.norm.df1,
                        test = train.norm.df1,
                        cl = train.df1$Personal.Loan, k = 3)
```

```
matrix4 <- confusionMatrix(knn.pred4,as.factor(train.df1$Personal.Loan),positive = "1")
matrix4
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2254   50
##          1    6  190
##
##                Accuracy : 0.9776
##                  95% CI : (0.971, 0.983)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8594
##
##  Mcnemar's Test P-Value : 9.132e-09
##
##             Sensitivity : 0.7917
##             Specificity : 0.9973
##          Pos Pred Value : 0.9694
##          Neg Pred Value : 0.9783
##              Prevalence : 0.0960
##          Detection Rate : 0.0760
##    Detection Prevalence : 0.0784
##       Balanced Accuracy : 0.8945
##
##        'Positive' Class : 1
##
```

Confusion matrix for validation set:

```
knn.pred5 <- class::knn(train = train.norm.df1,
                        test = valid.norm.df1,
                        cl = train.df1$Personal.Loan, k = 3)
matrix5 <- confusionMatrix(knn.pred5,as.factor(valid.df1$Personal.Loan),positive = "1")
matrix5
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1343   46
##          1   17   94
##
##                Accuracy : 0.958
##                  95% CI : (0.9466, 0.9676)
##     No Information Rate : 0.9067
##     P-Value [Acc > NIR] : 2.658e-14
##
##                   Kappa : 0.7264
##
##  Mcnemar's Test P-Value : 0.0004192
```

```
##
##             Sensitivity : 0.67143
##             Specificity : 0.98750
##          Pos Pred Value : 0.84685
##          Neg Pred Value : 0.96688
##              Prevalence : 0.09333
##          Detection Rate : 0.06267
##    Detection Prevalence : 0.07400
##       Balanced Accuracy : 0.82946
##
##        'Positive' Class : 1
##
```

Confusion matrix for test set:

```
knn.pred6 <- class::knn(train = train.norm.df1,
                        test = test.norm.df1,
                        cl = train.df1$Personal.Loan, k = 3)
matrix6 <- confusionMatrix(knn.pred6,as.factor(test.df1$Personal.Loan),positive = "1")
matrix6
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 896  34
##          1   4  66
##
##                Accuracy : 0.962
##                  95% CI : (0.9482, 0.973)
##     No Information Rate : 0.9
##     P-Value [Acc > NIR] : 1.383e-13
##
##                   Kappa : 0.7564
##
##  Mcnemar's Test P-Value : 2.546e-06
##
##             Sensitivity : 0.6600
##             Specificity : 0.9956
##          Pos Pred Value : 0.9429
##          Neg Pred Value : 0.9634
##              Prevalence : 0.1000
##          Detection Rate : 0.0660
##    Detection Prevalence : 0.0700
##       Balanced Accuracy : 0.8278
##
##        'Positive' Class : 1
##
```

The accuracy of the training set is high, the accuracy of the validation set is low, and the accuracy of the test set is higher than the accuracy of the validation set. This situation often occurs when a model is overfit to the training data.The model has learnt to fit the training data well.Consequently, it achieves a high accuracy on the training set because it memorizes the training examples.Since the model has overfit the training data, it fails to observe unseen data, which is represented by the validation set. As a result, the validation set accuracy is lower than the training set accuracy.When compared to the validation set, the test set could contain examples that are more similar to the training set, making it easier for the overfitted model to perform well there.
