k-NN for classification

Contents

Assignment 2 (k-NN Classification)

Use k-NN to help Universal bank explore ways of converting its liability customers to personal loan customers.

Code walkthrough

load given data.

Sample data

Filter out the attributes that are not needed i.e. ID and Zip Code

```
# display filtered data
#head(filtered_data)
```

Reference I followed on data splitting: https://topepo.github.io/caret/data-splitting.html Partition the data and split it into training, test and validation data sets.

```
set.seed(13)
train_index = createDataPartition(filtered_data$`Personal Loan`, p=0.6, list=FALSE) # 60% training data
# Train Data (60%)
train_data = filtered_data[train_index,]
```

```
## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
val_test_data = filtered_data[-train_index,] # rest of the data for validation and testing

test_index = createDataPartition(val_test_data$`Personal Loan`, p=0.5, list=FALSE) # 50% of the remaning
# Test Data (20%)
test_data = val_test_data[test_index,]
# Validation Data (20%)
validation_data = val_test_data[-test_index,]
```

Display Train / Validation / Test data

```r
summary(train_data)
```

```
##       Age          Experience        Income           Family     
##  Min.   :23.00   Min.   :-3.00   Min.   :  8.00   Min.   :1.000  
##  1st Qu.:35.00   1st Qu.:10.00   1st Qu.: 39.00   1st Qu.:1.000  
##  Median :45.00   Median :20.00   Median : 64.00   Median :2.000  
##  Mean   :45.47   Mean   :20.25   Mean   : 74.02   Mean   :2.381  
##  3rd Qu.:55.00   3rd Qu.:30.00   3rd Qu.: 99.00   3rd Qu.:3.000  
##  Max.   :67.00   Max.   :43.00   Max.   :218.00   Max.   :4.000  
##      CCAvg          Education        Mortgage       Personal Loan    
##  Min.   : 0.000   Min.   :1.000   Min.   :  0.00   Min.   :0.00000  
##  1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.00   1st Qu.:0.00000  
##  Median : 1.600   Median :2.000   Median :  0.00   Median :0.00000  
##  Mean   : 1.965   Mean   :1.887   Mean   : 56.16   Mean   :0.09533  
##  3rd Qu.: 2.600   3rd Qu.:3.000   3rd Qu.:101.00   3rd Qu.:0.00000  
##  Max.   :10.000   Max.   :3.000   Max.   :617.00   Max.   :1.00000  
##  Securities Account   CD Account          Online         CreditCard    
##  Min.   :0.0000     Min.   :0.00000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.0000     1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.0000     Median :0.00000   Median :1.0000   Median :0.0000  
##  Mean   :0.1053     Mean   :0.05767   Mean   :0.5917   Mean   :0.2987  
##  3rd Qu.:0.0000     3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :1.0000     Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
```

```r
summary(validation_data)
```

```
##       Age          Experience        Income           Family     
##  Min.   :23.00   Min.   :-3.00   Min.   :  8.00   Min.   :1.000  
##  1st Qu.:36.00   1st Qu.:11.00   1st Qu.: 40.00   1st Qu.:1.000  
##  Median :46.00   Median :21.00   Median : 65.00   Median :2.000  
##  Mean   :45.42   Mean   :20.22   Mean   : 75.11   Mean   :2.382  
##  3rd Qu.:55.00   3rd Qu.:30.00   3rd Qu.:102.00   3rd Qu.:3.000  
##  Max.   :67.00   Max.   :42.00   Max.   :205.00   Max.   :4.000  
##      CCAvg         Education        Mortgage       Personal Loan 
##  Min.   : 0.00   Min.   :1.000   Min.   :  0.00   Min.   :0.000  
##  1st Qu.: 0.60   1st Qu.:1.000   1st Qu.:  0.00   1st Qu.:0.000  
##  Median : 1.50   Median :2.000   Median :  0.00   Median :0.000  
##  Mean   : 1.94   Mean   :1.819   Mean   : 56.33   Mean   :0.111  
##  3rd Qu.: 2.60   3rd Qu.:3.000   3rd Qu.: 99.00   3rd Qu.:0.000  
##  Max.   :10.00   Max.   :3.000   Max.   :590.00   Max.   :1.000  
##  Securities Account   CD Account        Online        CreditCard   
##  Min.   :0.000      Min.   :0.000   Min.   :0.000   Min.   :0.000  
##  1st Qu.:0.000      1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000  
##  Median :0.000      Median :0.000   Median :1.000   Median :0.000  
##  Mean   :0.104      Mean   :0.073   Mean   :0.583   Mean   :0.291  
##  3rd Qu.:0.000      3rd Qu.:0.000   3rd Qu.:1.000   3rd Qu.:1.000  
##  Max.   :1.000      Max.   :1.000   Max.   :1.000   Max.   :1.000
```

```r
summary(test_data)
```

```
##       Age          Experience        Income          Family     
##  Min.   :23.00   Min.   :-2.00   Min.   :  8.0   Min.   :1.000  
##  1st Qu.:35.00   1st Qu.:10.00   1st Qu.: 38.0   1st Qu.:1.000  
##  Median :45.00   Median :20.00   Median : 61.0   Median :2.000  
```

```
##  Mean   :44.88   Mean   :19.56   Mean   : 71.7   Mean   :2.458
##  3rd Qu.:55.00   3rd Qu.:29.00   3rd Qu.: 94.0   3rd Qu.:4.000
##  Max.   :67.00   Max.   :42.00   Max.   :224.0   Max.   :4.000
##      CCAvg          Education        Mortgage       Personal Loan
##  Min.   :0.000   Min.   :1.000   Min.   :  0.00   Min.   :0.000
##  1st Qu.:0.700   1st Qu.:1.000   1st Qu.:  0.00   1st Qu.:0.000
##  Median :1.500   Median :2.000   Median :  0.00   Median :0.000
##  Mean   :1.854   Mean   :1.925   Mean   : 57.68   Mean   :0.083
##  3rd Qu.:2.400   3rd Qu.:3.000   3rd Qu.:102.25   3rd Qu.:0.000
##  Max.   :9.000   Max.   :3.000   Max.   :635.00   Max.   :1.000
##  Securities Account   CD Account        Online         CreditCard
##  Min.   :0.000      Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:0.000      1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000
##  Median :0.000      Median :0.000   Median :1.000   Median :0.000
##  Mean   :0.102      Mean   :0.056   Mean   :0.626   Mean   :0.283
##  3rd Qu.:0.000      3rd Qu.:0.000   3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :1.000      Max.   :1.000   Max.   :1.000   Max.   :1.000
```

Normalize the data using z-score scaling

```r
train.norm.df <- train_data
valid.norm.df <- validation_data
test.norm.df <- test_data

# z-score scaling
# normalize columns Age, Experience, Income, Family, CCAvg, Education and Mortgage
norm.model <- preProcess(train_data[, 1:7], method=c("center", "scale"))

# Apply the model
train.norm.df[, 1:7] <- predict(norm.model, train_data[, 1:7])
valid.norm.df[, 1:7] <- predict(norm.model, validation_data[, 1:7])
test.norm.df[, 1:7] <- predict(norm.model, test_data[, 1:7])

summary(train.norm.df)
```

```
##       Age              Experience           Income            Family
##  Min.   :-1.95774   Min.   :-2.02223   Min.   :-1.4293   Min.   :-1.1980
##  1st Qu.:-0.91199   1st Qu.:-0.89139   1st Qu.:-0.7582   1st Qu.:-1.1980
##  Median :-0.04052   Median :-0.02151   Median :-0.2169   Median :-0.3303
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.83094   3rd Qu.: 0.84836   3rd Qu.: 0.5408   3rd Qu.: 0.5374
##  Max.   : 1.87670   Max.   : 1.97920   Max.   : 3.1172   Max.   : 1.4051
##      CCAvg            Education          Mortgage        Personal Loan
##  Min.   :-1.1138   Min.   :-1.0538   Min.   :-0.5550   Min.   :0.00000
##  1st Qu.:-0.7171   1st Qu.:-1.0538   1st Qu.:-0.5550   1st Qu.:0.00000
##  Median :-0.2070   Median : 0.1343   Median :-0.5550   Median :0.00000
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   :0.09533
##  3rd Qu.: 0.3597   3rd Qu.: 1.3223   3rd Qu.: 0.4431   3rd Qu.:0.00000
##  Max.   : 4.5536   Max.   : 1.3223   Max.   : 5.5426   Max.   :1.00000
##  Securities Account   CD Account          Online          CreditCard
##  Min.   :0.0000     Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000     1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000     Median :0.00000   Median :1.0000   Median :0.0000
##  Mean   :0.1053     Mean   :0.05767   Mean   :0.5917   Mean   :0.2987
##  3rd Qu.:0.0000     3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:1.0000
```

```
##  Max.   :1.0000     Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
```

```r
summary(valid.norm.df)
```

```
##       Age               Experience             Income              Family
##  Min.   :-1.957741   Min.   :-2.022235   Min.   :-1.42935   Min.   :-1.198000
##  1st Qu.:-0.824840   1st Qu.:-0.804405   1st Qu.:-0.73653   1st Qu.:-1.198000
##  Median : 0.046623   Median : 0.065473   Median :-0.19527   Median :-0.330303
##  Mean   :-0.004096   Mean   :-0.002204   Mean   : 0.02372   Mean   : 0.001157
##  3rd Qu.: 0.830940   3rd Qu.: 0.848363   3rd Qu.: 0.60579   3rd Qu.: 0.537393
##  Max.   : 1.876695   Max.   : 1.892217   Max.   : 2.83579   Max.   : 1.405090
##      CCAvg             Education            Mortgage           Personal Loan
##  Min.   :-1.11378   Min.   :-1.05381   Min.   :-0.555031   Min.   :0.000
##  1st Qu.:-0.77374   1st Qu.:-1.05381   1st Qu.:-0.555031   1st Qu.:0.000
##  Median :-0.26368   Median : 0.13425   Median :-0.555031   Median :0.000
##  Mean   :-0.01415   Mean   :-0.08079   Mean   : 0.001683   Mean   :0.111
##  3rd Qu.: 0.35973   3rd Qu.: 1.32232   3rd Qu.: 0.423360   3rd Qu.:0.000
##  Max.   : 4.55355   Max.   : 1.32232   Max.   : 5.275788   Max.   :1.000
##  Securities Account   CD Account        Online          CreditCard
##  Min.   :0.000       Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:0.000       1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000
##  Median :0.000       Median :0.000   Median :1.000   Median :0.000
##  Mean   :0.104       Mean   :0.073   Mean   :0.583   Mean   :0.291
##  3rd Qu.:0.000       3rd Qu.:0.000   3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :1.000       Max.   :1.000   Max.   :1.000   Max.   :1.000
```

```r
summary(test.norm.df)
```

```
##       Age              Experience            Income              Family
##  Min.   :-1.95774   Min.   :-1.93525   Min.   :-1.42935   Min.   :-1.1980
##  1st Qu.:-0.91199   1st Qu.:-0.89139   1st Qu.:-0.77984   1st Qu.:-1.1980
##  Median :-0.04052   Median :-0.02151   Median :-0.28187   Median :-0.3303
##  Mean   :-0.05107   Mean   :-0.05988   Mean   :-0.05026   Mean   : 0.0671
##  3rd Qu.: 0.83094   3rd Qu.: 0.76138   3rd Qu.: 0.43259   3rd Qu.: 1.4051
##  Max.   : 1.87670   Max.   : 1.89222   Max.   : 3.24715   Max.   : 1.4051
##      CCAvg             Education            Mortgage          Personal Loan
##  Min.   :-1.11378   Min.   :-1.05381   Min.   :-0.55503   Min.   :0.000
##  1st Qu.:-0.71706   1st Qu.:-1.05381   1st Qu.:-0.55503   1st Qu.:0.000
##  Median :-0.26368   Median : 0.13425   Median :-0.55503   Median :0.000
##  Mean   :-0.06327   Mean   : 0.04515   Mean   : 0.01498   Mean   :0.083
##  3rd Qu.: 0.24638   3rd Qu.: 1.32232   3rd Qu.: 0.45548   3rd Qu.:0.000
##  Max.   : 3.98682   Max.   : 1.32232   Max.   : 5.72051   Max.   :1.000
##  Securities Account   CD Account        Online          CreditCard
##  Min.   :0.000       Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:0.000       1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000
##  Median :0.000       Median :0.000   Median :1.000   Median :0.000
##  Mean   :0.102       Mean   :0.056   Mean   :0.626   Mean   :0.283
##  3rd Qu.:0.000       3rd Qu.:0.000   3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :1.000       Max.   :1.000   Max.   :1.000   Max.   :1.000
```

k-NN modeling

```r
library(FNN)
# Personal Loan is the dependent variable (class output) so exclude that
train_predictors <- subset(train.norm.df, select=-c(`Personal Loan`))
valid_predictors <- subset(valid.norm.df, select=-c(`Personal Loan`))
```

```r
test_predictors <- subset(test.norm.df, select=-c(`Personal Loan`))

# Mark labels, for some reason, knn expects labes to be a vector and not a set which is what you get fr
# that is why we use dplyr::pull() to extract `Personal Loan` as a vector.
train_labels <- dplyr::pull(train.norm.df, `Personal Loan`)
valid_labels <- dplyr::pull(valid.norm.df, `Personal Loan`)
test_labels <- dplyr::pull(test.norm.df, `Personal Loan`)

# build a k-NN model
nn <- knn(train = train_predictors, test = test_predictors,
          cl = train_labels, k = 1, prob=TRUE)

head(nn)
```

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

```r
#print(nn)
```

Prediction

Problem 1.

Problem statement:

Given 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. Success class is 1 (loan acceptance), and default cutoff value of 0.5. How would this customer be classified?

```r
# let's combine training and validation datasets before we predict
train_valid_data <- rbind(train_data, validation_data)

# use new variables for Problem 1
train_valid.norm.df <- train_valid_data
test1.norm.df <- test_data

norm.train_valid.model <- preProcess(train_valid_data[, 1:7], method=c("center", "scale"))

# Apply the model
train_valid.norm.df[, 1:7] <- predict(norm.train_valid.model, train_valid_data[, 1:7])
test1.norm.df[, 1:7] <- predict(norm.train_valid.model, test_data[, 1:7])

#summary(train_valid.norm.df)
#summary(test1.norm.df)

# Build Model
# Personal Loan is the dependent variable (class output) so exclude that
train_valid_predictors <- subset(train_valid.norm.df, select=-c(`Personal Loan`))
test1_predictors <- subset(test1.norm.df, select=-c(`Personal Loan`))
problem1.test.data <- c(40, 10, 84, 2, 2, 2, 0, 0, 0, 1, 1)

# Mark labels, for some reason, knn expects labes to be a vector and not a set which is what you get fr
# that is why we use dplyr::pull() to extract `Personal Loan` as a vector.
train_valid_labels <- dplyr::pull(train_valid.norm.df, `Personal Loan`)
```

```r
test1_labels <- dplyr::pull(test1.norm.df, `Personal Loan`)

# build a k-NN model
predicted_test_labels <- knn(train = train_valid_predictors, test = problem1.test.data,
          cl = train_valid_labels, k = 1, prob=TRUE)

#print(predicted_test_labels)

sprintf("Nearest neighbor is: %s",row.names(train_data)[attr(predicted_test_labels, "nn.index")])
```

```
## [1] "Nearest neighbor is: 540"
```

```r
sprintf("Classification probability is: %f",attr(predicted_test_labels, "prob"))
```

```
## [1] "Classification probability is: 1.000000"
```

Analysis:

Data prep: The data was divided in to 60% training and 40% validation and test (requirements). Which meant 20% of the data was used for validation and 20% for test. Looking at the output it appears that the customer would be a good target for the personal loan offer. Since, our K value is too small (k=1) we are seeing overfitting.

Problem 2

Problem statement:

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

```r
# Hypertuning with multiple K values
library(caret)
set.seed(13)
# variable for number of attempts
attempts_var = 20
# define 20 k values with initial accuracy set to 0
accuracy.val.df <- data.frame(k = seq(1, attempts_var, 1), accuracy = rep(0, attempts_var))

# Need to convert DF to factors to work with confusion matrix
test_labels.factor <- as.factor(test_labels)

for(i in 1:attempts_var) {
  knn.pred <- knn(train = train_predictors, test = test_predictors,
          cl = train_labels, k = i, prob=TRUE)
  # Populate the accuracy value
  accuracy.val.df[i, 2] <- confusionMatrix(knn.pred, test_labels.factor)$overall[1]
}
accuracy.val.df
```

```
##     k accuracy
## 1   1    0.964
## 2   2    0.961
## 3   3    0.965
## 4   4    0.964
## 5   5    0.966
## 6   6    0.963
## 7   7    0.965
```

```
## 8   8    0.964
## 9   9    0.964
## 10 10    0.961
## 11 11    0.964
## 12 12    0.962
## 13 13    0.963
## 14 14    0.958
## 15 15    0.959
## 16 16    0.957
## 17 17    0.959
## 18 18    0.958
## 19 19    0.958
## 20 20    0.956
```

Analysis:

Looking at the output above the optimal value of K is 3. At k = 3 we see the accuracy is 0.968, better than others. As K is increased accuracy does not increase but it goes down. For values of k between 1 and 20 the best accuracy was observed at k = 3. This is the best value of K that balances between overfitting and ignoring the predictor information.

Problem 3

Problem Statement

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

Analysis

Following is the confusion matrix for our best K (k=3). We can see that the model has Accuracy = 0.968, Sensitivity : 0.9989 and Specificity : 0.6771.

```
knn.pred <- knn(train = train_predictors, test = test_predictors,
         cl = train_labels, k = 3, prob=TRUE)
  # Populate the accuracy value
  confusionMatrix(knn.pred, test_labels.factor)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 912  30
##          1   5  53
##
##                Accuracy : 0.965
##                  95% CI : (0.9517, 0.9755)
##     No Information Rate : 0.917
##     P-Value [Acc > NIR] : 6.240e-10
##
##                   Kappa : 0.7336
##
##  Mcnemar's Test P-Value : 4.976e-05
##
##             Sensitivity : 0.9945
##             Specificity : 0.6386
##          Pos Pred Value : 0.9682
```

```
##          Neg Pred Value : 0.9138
##              Prevalence : 0.9170
##          Detection Rate : 0.9120
##    Detection Prevalence : 0.9420
##       Balanced Accuracy : 0.8166
##
##        'Positive' Class : 0
##
```

Problem 4

Problem Statement

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.

Analysis

Classifying with k=3, results displayed below. The customer would be a good target for the loan offer.

```r
problem4.test.data <- c(40, 10, 84, 2, 2, 2, 0, 0, 0, 1, 1)

# building on previous code ...
# build a k-NN model
predicted_test_labels <- knn(train = train_valid_predictors, test = problem4.test.data,
          cl = train_valid_labels, k = 3, prob=TRUE)

#predicted_test_labels

sprintf("Nearest neighbor is: %s",row.names(train_data)[attr(predicted_test_labels, "nn.index")][1])
```

```
## [1] "Nearest neighbor is: 540"
```

```r
sprintf("Closest Distance is: %s",row.names(train_data)[attr(predicted_test_labels, "nn.dist")][1])
```

```
## [1] "Closest Distance is: 90"
```

```r
sprintf("Classification probability is: %f",attr(predicted_test_labels, "prob"))
```

```
## [1] "Classification probability is: 1.000000"
```

Problem 5

Problem Statement

Repartition the data 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.

```r
set.seed(13)
train_index = createDataPartition(filtered_data$`Personal Loan`, p=0.5, list=FALSE) # 50% training data
# Train Data (60%)
train_data = filtered_data[train_index,]

val_test_data = filtered_data[-train_index,] # rest of the data for validation and testing

test_index = createDataPartition(val_test_data$`Personal Loan`, p=0.4, list=FALSE)
```

```r
# Test Data (20%)
test_data = val_test_data[test_index,]
# Validation Data (30%)
validation_data = val_test_data[-test_index,]

## Normalize
train.norm.df <- train_data
valid.norm.df <- validation_data
test.norm.df <- test_data

# z-score scaling
# normalize columns Age, Experience, Income, Family, CCAvg, Education and Mortgage
norm.model <- preProcess(train_data[, 1:7], method=c("center", "scale"))

# Apply the model
train.norm.df[, 1:7] <- predict(norm.model, train_data[, 1:7])
valid.norm.df[, 1:7] <- predict(norm.model, validation_data[, 1:7])
test.norm.df[, 1:7] <- predict(norm.model, test_data[, 1:7])

## k-nn modeling
# Personal Loan is the dependent variable (class output) so exclude that
train_predictors <- subset(train.norm.df, select=-c(`Personal Loan`))
valid_predictors <- subset(valid.norm.df, select=-c(`Personal Loan`))
test_predictors <- subset(test.norm.df, select=-c(`Personal Loan`))

# Mark labels, for some reason, knn expects labes to be a vector and not a set which is what you get fr
# that is why we use dplyr::pull() to extract `Personal Loan` as a vector.
train_labels <- dplyr::pull(train.norm.df, `Personal Loan`)
valid_labels <- dplyr::pull(valid.norm.df, `Personal Loan`)
test_labels <- dplyr::pull(test.norm.df, `Personal Loan`)

# build a k-NN model for test
nn_test <- knn(train = train_predictors, test = test_predictors,
          cl = train_labels, k = 3, prob=TRUE)

## confusion matrix
# Need to convert DF to factors to work with confusion matrix
test_labels.factor <- as.factor(test_labels)
print("Confusion matrix for test set")
```

```
## [1] "Confusion matrix for test set"
```

```r
confusionMatrix(nn_test, test_labels.factor)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 883  37
##          1   3  77
##
##                Accuracy : 0.96
##                  95% CI : (0.9459, 0.9713)
##     No Information Rate : 0.886
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7724
##
##  Mcnemar's Test P-Value : 1.811e-07
##
##             Sensitivity : 0.9966
##             Specificity : 0.6754
##          Pos Pred Value : 0.9598
##          Neg Pred Value : 0.9625
##              Prevalence : 0.8860
##          Detection Rate : 0.8830
##    Detection Prevalence : 0.9200
##       Balanced Accuracy : 0.8360
##
##        'Positive' Class : 0
##
```

```r
# build a k-NN model for training
nn_train <- knn(train = train_predictors, test = train_predictors,
          cl = train_labels, k = 3, prob=TRUE)

## confusion matrix
# Need to convert DF to factors to work with confusion matrix
train_labels.factor <- as.factor(train_labels)
print("Confusion matrix for train set")
```

```
## [1] "Confusion matrix for train set"
```

```r
confusionMatrix(nn_train, train_labels.factor)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2263   44
##          1    2  191
##
##                Accuracy : 0.9816
##                  95% CI : (0.9755, 0.9865)
##     No Information Rate : 0.906
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8826
##
##  Mcnemar's Test P-Value : 1.493e-09
##
##             Sensitivity : 0.9991
##             Specificity : 0.8128
##          Pos Pred Value : 0.9809
##          Neg Pred Value : 0.9896
##              Prevalence : 0.9060
##          Detection Rate : 0.9052
##    Detection Prevalence : 0.9228
##       Balanced Accuracy : 0.9059
```

```
##
##        'Positive' Class : 0
##
```

```r
# build a k-NN model for validation
nn_valid <- knn(train = train_predictors, test = valid_predictors,
          cl = train_labels, k = 3, prob=TRUE)

## confusion matrix
# Need to convert DF to factors to work with confusion matrix
valid_labels.factor <- as.factor(valid_labels)
print("Confusion matrix for validation set")
```

```
## [1] "Confusion matrix for validation set"
```

```r
confusionMatrix(nn_valid, valid_labels.factor)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1362   53
##          1    7   78
##
##                Accuracy : 0.96
##                  95% CI : (0.9488, 0.9693)
##     No Information Rate : 0.9127
##     P-Value [Acc > NIR] : 4.899e-13
##
##                   Kappa : 0.7017
##
##  Mcnemar's Test P-Value : 6.267e-09
##
##             Sensitivity : 0.9949
##             Specificity : 0.5954
##          Pos Pred Value : 0.9625
##          Neg Pred Value : 0.9176
##              Prevalence : 0.9127
##          Detection Rate : 0.9080
##    Detection Prevalence : 0.9433
##       Balanced Accuracy : 0.7952
##
##        'Positive' Class : 0
##
```

Analysis

Comapring the confusion matrix for the train, validation and test set we see the following 1. Training set has the highest accuracy followed by validation set (Train Accuracy : 0.9792 > Validation Accuracy : 0.968 > Test Accuracy : 0.963) - which is as expected since, we trained the data on training set and validation set so the model has already seen the data unlike test data set. 2. Validation set has the highest sensitivity (proportion of positives correctly classified) followed by the training set (Validation Sensitivity : 1 > Train Sensitivity : 0.9996 > Test Sensitivity : 0.9989) 3. Training set has the highest specificity (proportion of negative cases correctly identified as negative) (Train Specificity : 0.7830 > Validation Specificity : 0.6690 > Test Specificity : 0.6400) - which is again what we expected. 4. I was expecting training set numbers (accuracy, sensitivity, specificity) to be better than this given the model uses this training data. This could be because of the lazy leraning nature of k-nn algorithm. 5. We can see that negative prediction value for for validation set is a bit better than training set. 6. I do not understand why this is the case (slighly lower negative prediction value for training set) as a result I need to do some more reading into this subject to better understand the numbers.