# MIS 64060: Assignment_2: k-NN for classification

Eyob Tadele

10/02/2021

### Project Objective

The objective of this assignment is to use k-NN to predict whether a new bank customer will accept a loan offer. This in turn will be used as the basis for designing a new marketing campaign that targets customers.

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. ### Importing a UniversalBank.csv dataset into r, load relevant libraries, and printout stats about the data.

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(class)
library(gmodels)
custData <- read.csv("UniversalBank.csv")
summary(custData)
```

```
##        ID            Age          Experience       Income          ZIP.Code
##  Min.   :   1   Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   : 9307
##  1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:91911
```

```
##  Median :2500   Median :45.00   Median :20.0   Median : 64.00   Median :93437
##  Mean   :2500   Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :93152
##  3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:94608
##  Max.   :5000   Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :96651
##      Family          CCAvg          Education        Mortgage
##  Min.   :1.000   Min.   : 0.000   Min.   :1.000   Min.   :  0.0
##  1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0
##  Median :2.000   Median : 1.500   Median :2.000   Median :  0.0
##  Mean   :2.396   Mean   : 1.938   Mean   :1.881   Mean   : 56.5
##  3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0
##  Max.   :4.000   Max.   :10.000   Max.   :3.000   Max.   :635.0
##  Personal.Loan   Securities.Account   CD.Account         Online
##  Min.   :0.000   Min.   :0.0000     Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.000   Median :0.0000     Median :0.0000   Median :1.0000
##  Mean   :0.096   Mean   :0.1044     Mean   :0.0604   Mean   :0.5968
##  3rd Qu.:0.000   3rd Qu.:0.0000     3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :1.000   Max.   :1.0000     Max.   :1.0000   Max.   :1.0000
##    CreditCard
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.294
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
str(custData)
```

```
## 'data.frame':    5000 obs. of  14 variables:
##  $ ID                : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP.Code          : int  91107 90089 94720 94112 91330 92121 91711 93943 90089 93023 ...
##  $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : int  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal.Loan     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ Securities.Account: int  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : int  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : int  0 0 0 0 1 0 0 1 0 0 ...
```

**Transforming categorical predictors with more than two categories into dummy variables.**

```
# applied as.character function on Education column to facilitate the transformation
custData$Education <- as.character(custData$Education)
# use the dummyVars function to create a transformation model
dummy.custModel <- dummyVars("~ .", data = custData)
# apply the model to the custData
cust_Data <- data.frame(predict(dummy.custModel, custData))
```

**Changing Personal.Loan variable to factor as it is our target varaible**

```
cust_Data$Personal.Loan <- as.factor(cust_Data$Personal.Loan)
#levels(cust_Data$Personal.Loan) = make.names(levels(factor(cust_Data$Personal.Loan)))
str(cust_Data)
```

```
## 'data.frame':    5000 obs. of  16 variables:
##  $ ID                : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age               : num  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : num  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : num  49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP.Code          : num  91107 90089 94720 94112 91330 ...
##  $ Family            : num  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education1        : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ Education2        : num  0 0 0 1 1 1 1 0 1 0 ...
##  $ Education3        : num  0 0 0 0 0 0 0 1 0 1 ...
##  $ Mortgage          : num  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal.Loan     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ Securities.Account: num  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : num  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : num  0 0 0 0 1 0 0 1 0 0 ...
```

**Partition the data into training (60%) and validation (40%) sets. Then create a normalized model using the training set and apply that to both training and validation datasets.**

```
set.seed(420)
# partitioning cust_Data into training(60%) and validation(40%) by first creating the model
Train_idx <- createDataPartition(cust_Data$Personal.Loan, p=0.6, list = FALSE)

# creating the training and validation datasets by applying the model
Train_custData <- cust_Data[Train_idx,]
Valid_custData <- cust_Data[-Train_idx,]

# make specific selection of the training set indices based on the question(i.e. dropping columns 1,5,a
TrainData <- Train_custData[,-c(1,5,12)]
ValidData <- Valid_custData[,-c(1,5,12)]

# creating a normalized model using range on the training data
Model_Train_Norm <- preProcess(TrainData, method = c("range"))

# apply normalization model on training and validation set
TrainPredictors <- predict(Model_Train_Norm, TrainData)
ValidPredictors <- predict(Model_Train_Norm, ValidData)
summary(TrainPredictors)
```

```
##       Age           Experience          Income            Family
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2727   1st Qu.:0.2826   1st Qu.:0.1523   1st Qu.:0.0000
```

```
##  Median :0.5000   Median :0.5000   Median :0.2741   Median :0.3333
##  Mean   :0.5085   Mean   :0.5033   Mean   :0.3285   Mean   :0.4623
##  3rd Qu.:0.7273   3rd Qu.:0.7174   3rd Qu.:0.4416   3rd Qu.:0.6667
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##      CCAvg          Education1      Education2       Education3
##  Min.   :0.00000   Min.   :0.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.07527   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.16129   Median :0.000   Median :0.0000   Median :0.0000
##  Mean   :0.20367   Mean   :0.412   Mean   :0.2817   Mean   :0.3063
##  3rd Qu.:0.26882   3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.00000   Max.   :1.000   Max.   :1.0000   Max.   :1.0000
##     Mortgage       Securities.Account   CD.Account        Online
##  Min.   :0.00000   Min.   :0.0000     Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.0000     1st Qu.:0.000   1st Qu.:0.0000
##  Median :0.00000   Median :0.0000     Median :0.000   Median :1.0000
##  Mean   :0.08947   Mean   :0.1067     Mean   :0.063   Mean   :0.5943
##  3rd Qu.:0.15906   3rd Qu.:0.0000     3rd Qu.:0.000   3rd Qu.:1.0000
##  Max.   :1.00000   Max.   :1.0000     Max.   :1.000   Max.   :1.0000
##    CreditCard
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.2967
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

```
summary(ValidPredictors)
```

```
##       Age          Experience        Income          Family
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2727   1st Qu.:0.2826   1st Qu.:0.1574   1st Qu.:0.0000
##  Median :0.5000   Median :0.5000   Median :0.2893   Median :0.3333
##  Mean   :0.5065   Mean   :0.5007   Mean   :0.3420   Mean   :0.4702
##  3rd Qu.:0.7273   3rd Qu.:0.7174   3rd Qu.:0.4721   3rd Qu.:0.6667
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0964   Max.   :1.0000
##      CCAvg          Education1     Education2      Education3
##  Min.   :0.00000   Min.   :0.00   Min.   :0.000   Min.   :0.000
##  1st Qu.:0.07527   1st Qu.:0.00   1st Qu.:0.000   1st Qu.:0.000
##  Median :0.17204   Median :0.00   Median :0.000   Median :0.000
##  Mean   :0.21544   Mean   :0.43   Mean   :0.279   Mean   :0.291
##  3rd Qu.:0.27957   3rd Qu.:1.00   3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :1.07527   Max.   :1.00   Max.   :1.000   Max.   :1.000
##     Mortgage       Securities.Account   CD.Account         Online
##  Min.   :0.00000   Min.   :0.000      Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.000      1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.00000   Median :0.000      Median :0.0000   Median :1.0000
##  Mean   :0.08824   Mean   :0.101      Mean   :0.0565   Mean   :0.6005
##  3rd Qu.:0.16063   3rd Qu.:0.000      3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :0.92441   Max.   :1.000      Max.   :1.0000   Max.   :1.0000
##    CreditCard
##  Min.   :0.00
##  1st Qu.:0.00
##  Median :0.00
##  Mean   :0.29
```

```
##  3rd Qu.:1.00
##  Max.   :1.00
```

```r
# the training and validation labels are on index 12 i.e. Personal.Loan
TrainLabels <- Train_custData[,12]
ValidLabels <- Valid_custData[,12]
summary(TrainLabels)
```

```
##    0    1
## 2712  288
```

```r
summary(ValidLabels)
```

```
##    0    1
## 1808  192
```

### Q1: Determine how a specific customer(i.e. 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) will be classified. This means that it is out test set.

```r
set.seed(420)
# Create a dataframe for the test set
TestData <- data.frame(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2,
                       Education1 = 0, Education2 = 1, Education3 = 0, Mortgage = 0,
                       Securities.Account = 0, CD.Account = 0, Online = 1, CreditCard = 1)

# normalize the test data with the normalization model created earler
TestPredictor <- predict(Model_Train_Norm, TestData)
# train a KNN model using k=1 and the class package
PredLabel <- knn(TrainPredictors, TestPredictor, cl=TrainLabels, k=1)
# The customer will be classified as a 0
PredLabel
```

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

**The customer will be classified as a 0. Meaning the customer will decline the offer.**

### Q2: What is a choice of k that balances between overfitting and ignoring the predictor information?

I applied the expand.grid() function to search for a specific optimal value of k by supplying certain number of potential k values.

```r
# searching for a specific choice of k by customizing a grid search
set.seed(420)
# create a search_grid set containing potential k values
search_grid <- expand.grid(k=c(3,5,7,9,11,13,15,17,19,21))
# apply it to the knn model
bestK_model <- train(TrainPredictors, TrainLabels, method = "knn", tuneGrid = search_grid)
bestK_model
```

```
## k-Nearest Neighbors
##
## 3000 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3000, 3000, 3000, 3000, 3000, 3000, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    3  0.9474290  0.6413997
##    5  0.9451361  0.6077868
##    7  0.9434825  0.5790797
##    9  0.9409041  0.5460397
##   11  0.9381898  0.5149350
##   13  0.9352977  0.4838324
##   15  0.9325375  0.4507476
##   17  0.9308300  0.4264440
##   19  0.9288720  0.3984997
##   21  0.9263341  0.3690868
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

**We can observe from the output that accuracy starts to decline after k=3. This indicates that 3 is the optimum value of k.**

**Q3: Show the confusion matrix for the validation data that results from using the best k**

```
set.seed(420)
# train a KNN model using k=3 and the class package
PredLabelBestK <- knn(TrainPredictors, ValidPredictors, cl=TrainLabels, k=3)
head(PredLabelBestK)
```

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

```
# confusion matrix for predicted values against validation data for best k (i.e. k=3) and positive case
confusionMatrix(PredLabelBestK, ValidLabels, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1803   69
##          1    5  123
##
##                Accuracy : 0.963
##                  95% CI : (0.9538, 0.9708)
```

```
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7495
##
##  Mcnemar's Test P-Value : 2.414e-13
##
##             Sensitivity : 0.6406
##             Specificity : 0.9972
##          Pos Pred Value : 0.9609
##          Neg Pred Value : 0.9631
##              Prevalence : 0.0960
##          Detection Rate : 0.0615
##    Detection Prevalence : 0.0640
##       Balanced Accuracy : 0.8189
##
##        'Positive' Class : 1
##
```

**Q4: 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.**

- The best k here is 3 and using the already normalized TestPredictor from question 1

```
set.seed(420)
# train a KNN model using k=3 and the class package
PredLabelK3t <- knn(TrainPredictors, TestPredictor, cl=TrainLabels, k=3)
PredLabelK3t
```

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

**The customer will be classified as a 0, which means a declined offer.**

**Q5: 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.**

In order to crate a 50-30-20 partition, I first partitioned the dataset as 80-20. Then the 80% training dataset is further divided into training and validation set with approximately 60-40 ratio.

```
# partitioning cust_Data into training(60%) and validation(40%) by first creating the model
set.seed(420)
Train_idx2 <- createDataPartition(cust_Data$Personal.Loan, p=0.8, list = FALSE)

# creating the training and test datasets(80-20)
Train_custData2 <- cust_Data[Train_idx2,]  # this is both TRAINING and VALIDATION for now
Test_custData2 <- cust_Data[-Train_idx2,]

# partitioning the Train_custData(i.e. TRAINING + VALIDATION) into separate
```

```r
# training and validation datasets in 60-40 ratio
Train_idx3 <- createDataPartition(Train_custData2$Personal.Loan, p=0.625, list = FALSE)
Train_custData22 <- Train_custData2[Train_idx3,] # resizing the training data
Valid_custData2 <- Train_custData2[-Train_idx3,] # creating the validation data

# make specific selection of the training set indices based on the quesion
# (i.e. dropping columns 1,5,and 12)
TrainData2 <- Train_custData22[,-c(1,5,12)]
ValidData2 <- Valid_custData2[,-c(1,5,12)]
TestData2 <- Test_custData2[,-c(1,5,12)]

# creating a normalized model using range on the training data
Model_Train_N <- preProcess(TrainData2, method = c("range"))

# apply normalization model on training and validation set
TrainPredictors2 <- predict(Model_Train_N, TrainData2)
ValidPredictors2 <- predict(Model_Train_N, ValidData2)
TestPredictor2 <-  predict(Model_Train_N, TestData2)
str(TrainPredictors2)
```

```
## 'data.frame':    2500 obs. of  13 variables:
##  $ Age               : num  0.0455 0.2727 0.2727 0.6818 0.25 ...
##  $ Experience        : num  0.0667 0.2444 0.2222 0.6444 0.2444 ...
##  $ Income            : num  0.19 0.426 0.171 0.296 0.796 ...
##  $ Family            : num  1 0 1 0.333 0 ...
##  $ CCAvg             : num  0.16 0.27 0.1 0.15 0.89 0.01 0.38 0.2 0.47 0.05 ...
##  $ Education1        : num  1 0 0 0 0 0 0 1 0 0 ...
##  $ Education2        : num  0 1 1 1 0 1 0 0 0 1 ...
##  $ Education3        : num  0 0 0 0 1 0 1 0 1 0 ...
##  $ Mortgage          : num  0 0 0 0 0 ...
##  $ Securities.Account: num  1 0 0 0 0 0 1 1 0 1 ...
##  $ CD.Account        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : num  0 0 0 1 0 1 0 0 0 0 ...
##  $ CreditCard        : num  0 0 1 0 0 0 0 0 0 1 ...
```

```r
str(ValidPredictors2)
```

```
## 'data.frame':    1500 obs. of  13 variables:
##  $ Age               : num  0.318 0.614 0.273 0.818 0.841 ...
##  $ Experience        : num  0.333 0.578 0.267 0.756 0.711 ...
##  $ Income            : num  0.0972 0.0648 0.338 0.1481 0.0648 ...
##  $ Family            : num  1 0 0.667 1 0 ...
##  $ CCAvg             : num  0.04 0.03 0.06 0.25 0.15 0.24 0.81 0.18 0.29 0.5 ...
##  $ Education1        : num  0 0 0 0 0 1 0 0 1 0 ...
##  $ Education2        : num  1 0 1 1 0 0 0 0 0 0 ...
##  $ Education3        : num  0 1 0 0 1 0 1 1 0 1 ...
##  $ Mortgage          : num  0.251 0 0.169 0 0 ...
##  $ Securities.Account: num  0 0 0 0 0 0 0 0 0 1 ...
##  $ CD.Account        : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ Online            : num  1 0 1 1 1 0 0 1 0 1 ...
##  $ CreditCard        : num  0 1 0 0 1 0 0 0 1 0 ...
```

```r
str(TestPredictor2)
```

```
## 'data.frame':    1000 obs. of  13 variables:
##  $ Age               : num  0.5 0.364 0.955 0.136 0.477 ...
##  $ Experience        : num  0.467 0.378 0.911 0.156 0.444 ...
##  $ Income            : num  0.1204 0.0139 0.4491 0.25 0.162 ...
##  $ Family            : num  0.667 0 1 0 0.333 ...
##  $ CCAvg             : num  0.15 0.1 0.24 0.12 0.07 0.33 0.12 0.07 0.8 0.17 ...
##  $ Education1        : num  1 1 0 1 1 0 0 0 1 0 ...
##  $ Education2        : num  0 0 0 0 0 1 0 0 0 1 ...
##  $ Education3        : num  0 0 1 0 0 0 1 1 0 0 ...
##  $ Mortgage          : num  0 0 0 0.421 0.264 ...
##  $ Securities.Account: num  1 0 0 0 1 0 0 0 0 1 ...
##  $ CD.Account        : num  0 0 0 0 0 1 0 0 0 0 ...
##  $ Online            : num  0 0 0 1 0 1 1 1 1 1 ...
##  $ CreditCard        : num  0 0 0 0 0 1 0 0 0 0 ...
```

```r
# the training and validation labels are on index 12 i.e. Personal.Loan
TrainLabels2 <- Train_custData22[,12]
ValidLabels2 <- Valid_custData2[,12]
TestLabels2 <- Test_custData2[,12]
```

Building a KNN model with k=3 and applying it on the training and validation sets. Then based on the outcome, apply it to the training set and finally compare results with the use of confusion matrices.

```r
set.seed(420)
# train a KNN model using k=3 for classification on the training set
PredLabel2 <- knn(TrainPredictors2, ValidPredictors2, cl=TrainLabels2, k=3)
head(PredLabel2)
```

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

```r
# confusion matrix for predicted values against validation data for best k (i.e. k=3)
confusionMatrix(PredLabel2, ValidLabels2, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1347   61
##          1    9   83
##
##                Accuracy : 0.9533
##                  95% CI : (0.9414, 0.9634)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 7.606e-13
##
##                   Kappa : 0.6794
##
##  Mcnemar's Test P-Value : 1.090e-09
```

```
##
##             Sensitivity : 0.57639
##             Specificity : 0.99336
##          Pos Pred Value : 0.90217
##          Neg Pred Value : 0.95668
##              Prevalence : 0.09600
##          Detection Rate : 0.05533
##    Detection Prevalence : 0.06133
##       Balanced Accuracy : 0.78488
##
##        'Positive' Class : 1
##
```

```
# train a KNN model using k=3 for classification on the test set
# I re-combined the training and validation sets to use on the training set here
set.seed(420)
Train_custDataTrValCombined <- predict(Model_Train_N,Train_custData2[,-c(1,5,12)])
Train_custDataTrValCombinedLabel <- Train_custData2[,12]
PredLabel2 <- knn(Train_custDataTrValCombined, TestPredictor2, cl=Train_custDataTrValCombinedLabel, k=3)
head(PredLabel2)
```

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

```
# confusion matrix for predicted values against test data for best k (i.e. k=3)
confusionMatrix(PredLabel2, TestLabels2, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 901  40
##          1   3  56
##
##                Accuracy : 0.957
##                  95% CI : (0.9425, 0.9687)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 2.214e-10
##
##                   Kappa : 0.7007
##
##  Mcnemar's Test P-Value : 4.021e-08
##
##             Sensitivity : 0.5833
##             Specificity : 0.9967
##          Pos Pred Value : 0.9492
##          Neg Pred Value : 0.9575
##              Prevalence : 0.0960
##          Detection Rate : 0.0560
##    Detection Prevalence : 0.0590
##       Balanced Accuracy : 0.7900
##
##        'Positive' Class : 1
##
```

**It can be observed from the results of the two confusion matrices that the model in the test set has performed slightly better. Accuracy has improved from 0.9533 to 0.957. The improvement is the result of combining training and validation datasets(i.e. 80%), as the new training set. Meaning, the model has learned from a much larger data pool, as opposed to only 50% in the initial training with 30% validation set.**