k-NN Classification Model

Melissa Paniagua

9/26/2020

The purpose of this assignment is to use k-NN for classification utilizing open data on 5000 customers from a financial institution to predict whether a liability customer would accept a personal loan offer.

Variables used in the Universal Bank dataset are the following:

- ID: Customer's identifier.
- Age: Customer's age.
- Experience: Number of customer's years experience.
- Income: Annual income.
- Zip Code: Customer's location area.
- Family: Number of family members.
- CCAvg: Average spending on credit cards.
- Education: Highest customer's education level.

  1 = High School 2 = Undergraduate. 3 = Graduate.

- Mortgage: Value of debt if the customer has a mortgage.
- Personal Loan: Indicates if the customer accepted or rejected the loan offered in last campaing.

  1 = Accepted 0 = Rejected

- Securities Account: Indicates if the customer has security account.

  1 = Yes. 0 = No.

- CD Account: Indicates if the customer has a Certificate of Deposit.

  1 = Yes. 0 = No.

- Online: Indicates if the customer has internet banking facilites.

  1 = Yes. 0 = No.

- CredictCard: Indicates if the customer currently has credit cards.

  1 = Yes. 0 = No.

```r
# Load the libraries needed for the project
library(readr) #read files
library(dplyr) #To select a subset of variables
library(dummies) #To create dummies variables
library(fastDummies) #To create dummies variables
library(caret) #To split the dataset in training, validation, and testing.
library(FNN) #To use k-NN algorithm
library(ggplot2) #To create plots
library("gmodels") # To create the confusion matrix

#Import the dataset
df <- read_csv("UniversalBank.csv")
```

```r
#Preview the data
head(df)
```

```
# A tibble: 6 x 14
     ID   Age Experience Income `ZIP Code` Family CCAvg Education Mortgage
  <dbl> <dbl>      <dbl>  <dbl>      <dbl>  <dbl> <dbl>     <dbl>    <dbl>
1     1    25          1     49      91107      4   1.6         1        0
2     2    45         19     34      90089      3   1.5         1        0
3     3    39         15     11      94720      1   1           1        0
4     4    35          9    100      94112      1   2.7         2        0
5     5    35          8     45      91330      4   1           2        0
6     6    37         13     29      92121      4   0.4         2      155
# ... with 5 more variables: `Personal Loan` <dbl>, `Securities Account` <dbl>,
#   `CD Account` <dbl>, Online <dbl>, CreditCard <dbl>
```

Exploration Data

```r
# See the dataframe's structure
str(df)
```

```
Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame':    5000 obs. of  14 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 ...
 $ Education         : num  1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage          : num  0 0 0 0 0 155 0 0 104 0 ...
 $ Personal Loan     : num  0 0 0 0 0 0 0 0 0 1 ...
 $ 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 ...
 - attr(*, "spec")=
  .. cols(
  ..   ID = col_double(),
```

```
..   Age = col_double(),
..   Experience = col_double(),
..   Income = col_double(),
..   `ZIP Code` = col_double(),
..   Family = col_double(),
..   CCAvg = col_double(),
..   Education = col_double(),
..   Mortgage = col_double(),
..   `Personal Loan` = col_double(),
..   `Securities Account` = col_double(),
..   `CD Account` = col_double(),
..   Online = col_double(),
..   CreditCard = col_double()
.. )
```

```r
#Some descriptive statistcs
summary(df)
```

```
       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  
```

```r
# Identify unique values on specific variables, which could be change to dummies variables.
unique(df$Education)
```

```
[1] 1 2 3
```

Dummies variables

It is important to transform categorical variables and transform them into dummies variables. The column we will trasnform is education.

```
# Create dummies variables for education.
dumdf <- dummy_cols(df, select_columns = c("Education"))

# See the dataframe with dummies varariables
head(dumdf)
```

```
# A tibble: 6 x 17
     ID   Age Experience Income `ZIP Code` Family CCAvg Education Mortgage
  <dbl> <dbl>      <dbl>  <dbl>      <dbl>  <dbl> <dbl>     <dbl>    <dbl>
1     1    25          1     49      91107      4   1.6         1        0
2     2    45         19     34      90089      3   1.5         1        0
3     3    39         15     11      94720      1   1           1        0
4     4    35          9    100      94112      1   2.7         2        0
5     5    35          8     45      91330      4   1           2        0
6     6    37         13     29      92121      4   0.4         2      155
# ... with 8 more variables: `Personal Loan` <dbl>, `Securities Account` <dbl>,
#   `CD Account` <dbl>, Online <dbl>, CreditCard <dbl>, Education_1 <int>,
#   Education_2 <int>, Education_3 <int>
```

Select a subset of variables

For our project purposes, we will create a new dataset ignoring variables that are not important for our project such as ID number, zip code information, and also the original education columns.

```
#Select subset.
mydf <- select(dumdf, 2:4, 6:7, 9:17)

# Move the predicted variable (Personal Loan) at the end of the dataset.
mydf <- mydf%>%select(-`Personal Loan`, `Personal Loan`)

#See new dataframe
head(mydf)
```

```
# A tibble: 6 x 14
    Age Experience Income Family CCAvg Mortgage `Securities Acc~ `CD Account`
  <dbl>      <dbl>  <dbl>  <dbl> <dbl>    <dbl>            <dbl>        <dbl>
1    25          1     49      4   1.6        0                1            0
2    45         19     34      3   1.5        0                1            0
3    39         15     11      1   1          0                0            0
4    35          9    100      1   2.7        0                0            0
5    35          8     45      4   1          0                0            0
6    37         13     29      4   0.4      155                0            0
# ... with 6 more variables: Online <dbl>, CreditCard <dbl>, Education_1 <int>,
#   Education_2 <int>, Education_3 <int>, `Personal Loan` <dbl>
```

Part I

Data Splitting

```
#It generates the same random variables (same partition training set, same partition valid set)
set.seed(1234)

# To create a partition of 60% of our data set, we will use a caret package tool
resample = createDataPartition(mydf$Income, p=0.60, list=FALSE)

# Now, let's to create a dataframe with 60% for training sets and 40% validation sets.
train_set = mydf[resample, ]
valid_set = mydf[-resample, ]

#Now, let's do summary function to get some descritive statistics for only income
# on our training and validation set.
summary(train_set$Income)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8.00   39.00   64.00   73.94   98.00  205.00
```

```
summary(valid_set$Income)
```

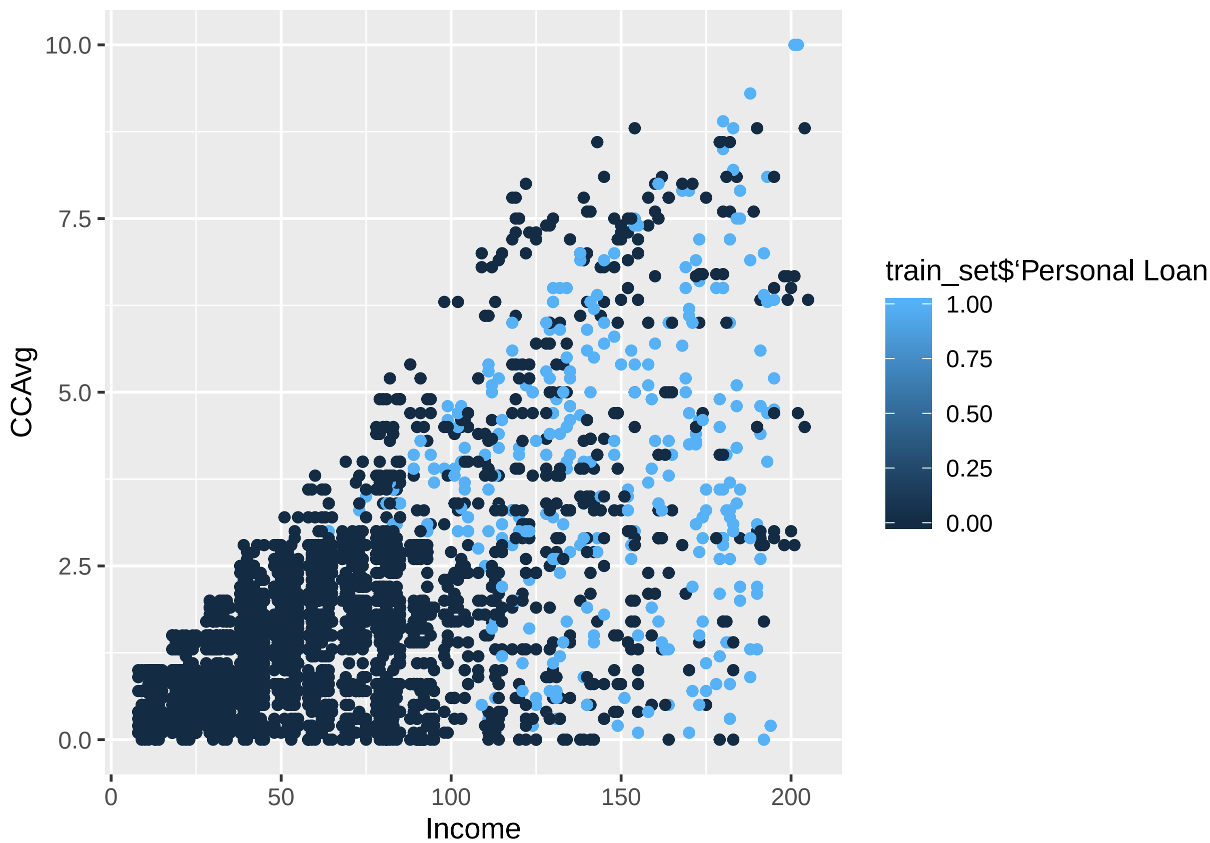
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8.00   39.00   64.00   73.53   98.00  224.00
```

As we can prove, our training and validation sets are stable and well balanced.

Plotting

```
# Plotting the data would help us get a better understanding of our data subsets.

ggplot(train_set,
       aes(x=Income,y=CCAvg, color=train_set$`Personal Loan`)) + geom_point()
```



This plot show us that customers with lower income also have an average spending on credict cards lower. Or it is possitble that they are likely to have less access to credit.

Normalization

Normalizing the dataset is essential the reduce the bias on the dataset.

For this purpose, we are going to use the preProcess function from our Caret package. This function uses the range method (min-max), which will transform those values using "center" (mean) and "scale" (standard deviation) as input method parameters.

To normalize the dataset, we have to exclude categorical variables such as Education, Securities Account, etc. The main variable (Personal Loan) is also excluded in this section.

Note: Normalizing in this context means transforming all those variables in smaller numbers to reduce the bias and it's spreadness.

```
# Copy the original data and create new normalized dataframes
train_norm_set <- train_set
valid_norm_set <- valid_set
#traval_norm_set <- traval_set

# use preProcess() from the caret package to normalize the all the variables in our dataset.
norm_values <- preProcess(train_set[, 1:6], method=c("center", "scale"))

# Replace the columns with normalized values
train_norm_set[, 1:6] <- predict(norm_values, train_set[, 1:6])
```

```r
valid_norm_set[, 1:6] <- predict(norm_values, valid_set[, 1:6])

# Now, let's see the differences between our original dataframe and our normalized dataframe.
#Calculate the descriptive stats for out normalized train set
summary(train_norm_set)
```

```
      Age            Experience           Income            Family       
 Min.   :-1.93925   Min.   :-2.004577   Min.   :-1.4330   Min.   :-1.2064  
 1st Qu.:-0.89223   1st Qu.:-0.870011   1st Qu.:-0.7593   1st Qu.:-1.2064  
 Median :-0.01971   Median : 0.002733   Median :-0.2160   Median :-0.3286  
 Mean   : 0.00000   Mean   : 0.000000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 0.85282   3rd Qu.: 0.875476   3rd Qu.: 0.5229   3rd Qu.: 0.5491  
 Max.   : 1.89984   Max.   : 1.922768   Max.   : 2.8483   Max.   : 1.4269  
     CCAvg            Mortgage       Securities Account   CD Account     
 Min.   :-1.1091   Min.   :-0.5516   Min.   :0.0000     Min.   :0.00000  
 1st Qu.:-0.7054   1st Qu.:-0.5516   1st Qu.:0.0000     1st Qu.:0.00000  
 Median :-0.2440   Median :-0.5516   Median :0.0000     Median :0.00000  
 Mean   : 0.0000   Mean   : 0.0000   Mean   :0.1099     Mean   :0.06596  
 3rd Qu.: 0.3904   3rd Qu.: 0.4279   3rd Qu.:0.0000     3rd Qu.:0.00000  
 Max.   : 4.6584   Max.   : 5.6066   Max.   :1.0000     Max.   :1.00000  
     Online         CreditCard      Education_1      Education_2    
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
 Median :1.0000   Median :0.0000   Median :0.0000   Median :0.0000  
 Mean   :0.5883   Mean   :0.2965   Mean   :0.4091   Mean   :0.2835  
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
  Education_3     Personal Loan   
 Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.0000  
 Median :0.0000   Median :0.0000  
 Mean   :0.3075   Mean   :0.1023  
 3rd Qu.:1.0000   3rd Qu.:0.0000  
 Max.   :1.0000   Max.   :1.0000  
```

```r
#Calculate the variance for our normalized train set
var(train_norm_set[, 1:6])
```

```
                   Age  Experience      Income      Family       CCAvg
Age         1.00000000  0.99418741 -0.04617853 -0.03281504 -0.05140346
Experience  0.99418741  1.00000000 -0.03670307 -0.03741685 -0.04765026
Income     -0.04617853 -0.03670307  1.00000000 -0.15164511  0.63670078
Family     -0.03281504 -0.03741685 -0.15164511  1.00000000 -0.09700959
CCAvg      -0.05140346 -0.04765026  0.63670078 -0.09700959  1.00000000
Mortgage    0.01965612  0.02254167  0.22577922 -0.03085638  0.10707322
              Mortgage
Age         0.01965612
Experience  0.02254167
Income      0.22577922
Family     -0.03085638
CCAvg       0.10707322
Mortgage    1.00000000
```

```r
#Calculate the descriptive stats for out normalized valid set
summary(valid_norm_set)
```

```
      Age             Experience             Income              Family
 Min.   :-1.93925   Min.   :-2.004577   Min.   :-1.43297   Min.   :-1.20639
 1st Qu.:-0.80498   1st Qu.:-0.870011   1st Qu.:-0.75928   1st Qu.:-1.20639
 Median : 0.06755   Median : 0.002733   Median :-0.21597   Median :-0.32864
 Mean   : 0.02458   Mean   : 0.029684   Mean   :-0.00891   Mean   : 0.04829
 3rd Qu.: 0.94007   3rd Qu.: 0.875476   3rd Qu.: 0.52292   3rd Qu.: 1.42685
 Max.   : 1.89984   Max.   : 2.010042   Max.   : 3.26116   Max.   : 1.42685
     CCAvg             Mortgage          Securities Account  CD Account
 Min.   :-1.10913   Min.   :-0.551650   Min.   :0.0000     Min.   :0.00000
 1st Qu.:-0.70540   1st Qu.:-0.551650   1st Qu.:0.0000     1st Qu.:0.00000
 Median :-0.18632   Median :-0.551650   Median :0.0000     Median :0.00000
 Mean   : 0.02148   Mean   :-0.009318   Mean   :0.0961     Mean   :0.05205
 3rd Qu.: 0.33275   3rd Qu.: 0.427850   3rd Qu.:0.0000     3rd Qu.:0.00000
 Max.   : 4.65842   Max.   : 5.432028   Max.   :1.0000     Max.   :1.00000
     Online          CreditCard       Education_1      Education_2
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :1.0000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :0.6096   Mean   :0.2903   Mean   :0.4344   Mean   :0.2763
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
  Education_3     Personal Loan
 Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.0000   1st Qu.:0.00000
 Median :0.0000   Median :0.00000
 Mean   :0.2893   Mean   :0.08659
 3rd Qu.:1.0000   3rd Qu.:0.00000
 Max.   :1.0000   Max.   :1.00000
```

```r
#Calculate the variance for out normalized valid set
var(valid_norm_set[, 1:6])
```

```
                   Age  Experience      Income      Family       CCAvg
Age         1.00107163  0.99691546 -0.06890733 -0.06847225 -0.05433237
Experience  0.99691546  1.00426859 -0.06142022 -0.07729387 -0.05523825
Income     -0.06890733 -0.06142022  1.00250134 -0.16918536  0.67392892
Family     -0.06847225 -0.07729387 -0.16918536  1.03608725 -0.13259335
CCAvg      -0.05433237 -0.05523825  0.67392892 -0.13259335  1.04029149
Mortgage   -0.06036765 -0.05985953  0.17152809 -0.00421576  0.11276287
              Mortgage
Age        -0.06036765
Experience -0.05985953
Income      0.17152809
Family     -0.00421576
CCAvg       0.11276287
Mortgage    0.93293385
```

As we can see, after normalizing the training set, the mean and variance of the columns are between 0 and 1. It proves that the data is in equal scale to be analized. Regarding the validation set, it is not true. We will use the mean and standard deviation from the training set to normalize the validation and testing set.

Training k-NN Model

Now, let apply the k-NN model where k = 1. First, create the x variables (all the ones we want to train) and create another one to save the Y variable, which is Personal Loan (last column). This column has values of 1=Yes, or 0=No.

We will use "drop = TRUE" argument to transform the dataframe into a vector because k-NN works only with vectors.

```
#First, create X dataframe and Y vector
train_predictors<-train_norm_set[,1:13, drop = TRUE]
valid_predictors<-valid_norm_set[,1:13, drop = TRUE]
train_labels <-train_norm_set[,14, drop = TRUE]
valid_labels  <-valid_norm_set[,14, drop = TRUE]

#Run the model using k = 1
set.seed(1234)
my_knn <-knn(train_predictors,
                            valid_predictors,
                            cl=train_labels,
                            k=1 )

# See the 6 first values of predicted class in the validation set
head(my_knn)
```

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

```
# To summarized the model
summary(my_knn)
```

```
   0    1
1869  129
```

Confusion Matrix

```
# Create a confusion matrix
conf_matrix <- CrossTable(x=valid_labels,y=my_knn, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1998
```

```
             | my_knn
valid_labels |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |      1807 |        18 |      1825 |
             |     0.990 |     0.010 |     0.913 |
             |     0.967 |     0.140 |           |
             |     0.904 |     0.009 |           |
-------------|-----------|-----------|-----------|
           1 |        62 |       111 |       173 |
             |     0.358 |     0.642 |     0.087 |
             |     0.033 |     0.860 |           |
             |     0.031 |     0.056 |           |
-------------|-----------|-----------|-----------|
Column Total |      1869 |       129 |      1998 |
             |     0.935 |     0.065 |           |
-------------|-----------|-----------|-----------|
```

The confusion matrix show us that the model wrongly predicted 80 customers. The model is classified 95.99% correctly.

Probability Output

```r
# Create a new variable for our probability
set.seed(1234)
my_knnprob <-knn(train_predictors,
        valid_predictors,
        cl=train_labels, k=1, prob=TRUE )

class_prob<-attr(my_knnprob, 'prob')

# See the first rows
head(class_prob)
```

```
[1] 1 1 1 1 1 1
```

Calcutale the accuracy, recall, precision, specificity

```r
#Calcutale the accuracy
k1_accuracy <- (conf_matrix$t[2,2] + conf_matrix$t[1,1])/ sum(conf_matrix$t)
print(k1_accuracy)
```

```
[1] 0.95996
```

```r
#Calcutale the recall
k1_recall <- conf_matrix$t[2,2]/ (conf_matrix$t[2,2] + conf_matrix$t[2,1])
print(k1_recall)
```

```
[1] 0.6416185
```

```r
#Calcutale the precision
k1_precision <- conf_matrix$t[2,2]/ (conf_matrix$t[2,2] + conf_matrix$t[1,2])
print(k1_precision)
```

```
[1] 0.8604651
```

```r
#Calcutale the specificity
k1_specificity <- conf_matrix$t[1,1]/ (conf_matrix$t[1,1] + conf_matrix$t[1,2])
print(k1_specificity)
```

```
[1] 0.990137
```

As we can determine, the model is learning well.

Create a new observation

Now, let's add a new customer and run the k-NN model the validation set.

```r
# Create a new observation
new_obs <- c(40, 10, 84, 2, 2, 0, 0, 0, 1, 1, 0, 1, 0)
```

Normalize the new observation

```r
# To select only the first six columns that we want to normalize
x <- as.data.frame(t(new_obs[1:6]))
colnames (x) <- c('Age', 'Expereince', 'Income', 'Family', 'CCAvg', 'Mortgage')

#To normalize my new obervation
n=(x-norm_values$mean)/norm_values$std

# To get the same dimensions
m = c(n$Age,n$Expereince,n$Income,n$Family,n$CCAvg,n$Mortgage,new_obs[7], new_obs[8],
      new_obs[9], new_obs[10], new_obs[11], new_obs[12], new_obs[13])

# To show how my the variable looks like
m
```

```
 [1] -0.45596688 -0.87001050  0.21866737 -0.32864440  0.04437672 -0.55164997
 [7]  0.00000000  0.00000000  1.00000000  1.00000000  0.00000000  1.00000000
[13]  0.00000000
```

Run K-NN Model with the new observation

```r
# Calculate the probability for the new observation
my_knnprob2 <-knn(train_predictors,
                            m,
                            cl=train_labels,
                            k=1, prob = TRUE)

# To get the probability output
class_prob2<-attr(my_knnprob2, 'prob')
# Show the first lines
head(class_prob2)
```

```
[1] 1
```

Explain how would this customer be classified?

The probability output the new customer "1." It means that the new customer will accept the personal loan offer from the Universal Bank.

Determing the optimal using Hyperparameter Tuning

Let's find the optimal k using tuning parameters

```r
set.seed(1234)
Search_grid <- expand.grid(k=c(1:20))
train_predict_labels <- train_predictors
train_predict_labels$Personal_Loan = train_labels

modeltest<-train(factor(Personal_Loan)~Age+Experience+Income+Family+
                   CCAvg+Mortgage+`Securities Account`+`CD Account`+Online+
                   CreditCard+Education_1+Education_2+Education_3,
                 data = train_predict_labels, method="knn",
                 tuneGrid=Search_grid,
                 preProcess='range')

# To show the result
modeltest
```

```
k-Nearest Neighbors

3002 samples
  13 predictor
   2 classes: '0', '1'

Pre-processing: re-scaling to [0, 1] (13)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 3002, 3002, 3002, 3002, 3002, 3002, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9536046  0.7263191
   2  0.9471997  0.6821014
   3  0.9458861  0.6670179
```

```
  4  0.9452363  0.6582538
  5  0.9446240  0.6448406
  6  0.9441927  0.6366618
  7  0.9432076  0.6260626
  8  0.9411801  0.6052238
  9  0.9399465  0.5922334
 10  0.9381334  0.5738815
 11  0.9370963  0.5640625
 12  0.9356435  0.5523524
 13  0.9350647  0.5445418
 14  0.9330659  0.5249848
 15  0.9319625  0.5132661
 16  0.9313246  0.5060602
 17  0.9298700  0.4910055
 18  0.9283937  0.4769321
 19  0.9273087  0.4643694
 20  0.9261396  0.4539903

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 1.
```

As we can see, the hypertuning parameter uses a resampling of Bootstrapped (25 reps), which is very robust, and it gives the optimal k = 1

Discuss the choice of k that balances between overfitting and ignoring the predictor information?

Overfitting is when the model is capturing the personal characteristics from the training set. We can prove that comparing the accuracy from the training and validation set.

Based on our outputs showing the accuracy of the model trained on different data partitioning, we can prove out the model is not overfitting because those values have similar results.

Show the confusion matrix for the validation data that results from using the best k and explain different error types that you observe.

Confusion Matrix using the optimal k.

In the universal Bank problem, the hypertuning parameter uses a resampling of Bootstrapped (25 reps), which is very robust, and gives the optimal k = 1. So, we will continue using k=1.

Part II

Data Splitting for the second part

Here we are going to divide the dataframe in three sections: Training, Validation, and Testing

```r
#It generates the same random variables (same partition training set, same partition valid set)
set.seed(1234)

# To create a partition of 80% of our data set, we will use a caret package tool
resample2 = createDataPartition(mydf$Income, p=0.80, list=FALSE)

# Now, let's to create a dataframe with 50% for training sets, 30% validation sets,
# and 20% testing sets.
```

```r
traval_set2 = mydf[resample2, ] # It will get 80%
test_set2 = mydf[-resample2, ] # It will get 20%

# To create a partition of 50% of our data set, we will use a caret package tool
resample3 = createDataPartition(traval_set2$Income, p=0.80, list=FALSE)

# To share the data form training and validation
train_set2 = mydf[resample3, ] # It will get 50%
valid_set2 = mydf[-resample3, ] # It will get 30%

#Now, let's do summary function to get some descritive statistics for only income
# on our training and validation set.
summary(train_set2$Income)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8.00   39.00   63.00   73.57   98.00  224.00
```

```r
summary(valid_set2$Income)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8.00   39.00   64.00   74.14   99.00  218.00
```

```r
summary(test_set2$Income)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8.00   39.00   63.00   73.33   98.00  224.00
```

Normalizing step

This steps is to renormalize our dataframes. It is important to notice that we have to renormalize our traval (training+validation) set.

```r
# Copy the original data and create new normalization dataframes
train_norm_set1 <- train_set2
valid_norm_set1 <- valid_set2
traval_norm_set1 <- traval_set2
test_norm_set1 <- test_set2

# use preProcess() from the caret package to normalize the all the variables for the training set.
norm_values2 <- preProcess(train_set2[, 1:6], method=c("center", "scale"))

# Replace the columns with normalized values for the training and validation set
train_norm_set1[, 1:6] <- predict(norm_values2, train_set2[, 1:6])
valid_norm_set1[, 1:6] <- predict(norm_values2, valid_set2[, 1:6])

# use preProcess() from the caret package to normalize the all the variables for the training and valid
norm_values3 <- preProcess(traval_set2[, 1:6], method=c("center", "scale"))

# Replace the columns with normalized values
traval_norm_set1[, 1:6] <- predict(norm_values3, traval_set2[, 1:6])
```

```r
test_norm_set1[, 1:6] <- predict(norm_values3, test_set2[, 1:6])

# Now, let's see the differences between our original dataframe and our normalized dataframe.
summary(train_norm_set1)
```

```
      Age             Experience          Income            Family       
 Min.   :-1.93922   Min.   :-2.00430   Min.   :-1.4372   Min.   :-1.2185  
 1st Qu.:-0.89920   1st Qu.:-0.87802   1st Qu.:-0.7577   1st Qu.:-1.2185  
 Median :-0.03252   Median :-0.01166   Median :-0.2317   Median :-0.3545  
 Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 0.83416   3rd Qu.: 0.85471   3rd Qu.: 0.5355   3rd Qu.: 1.3736  
 Max.   : 1.87417   Max.   : 1.98098   Max.   : 3.2971   Max.   : 1.3736  
     CCAvg            Mortgage       Securities Account   CD Account     
 Min.   :-1.1112   Min.   :-0.5616   Min.   :0.0000     Min.   :0.00000  
 1st Qu.:-0.7104   1st Qu.:-0.5616   1st Qu.:0.0000     1st Qu.:0.00000  
 Median :-0.2522   Median :-0.5616   Median :0.0000     Median :0.00000  
 Mean   : 0.0000   Mean   : 0.0000   Mean   :0.1074     Mean   :0.05963  
 3rd Qu.: 0.3204   3rd Qu.: 0.4442   3rd Qu.:0.0000     3rd Qu.:0.00000  
 Max.   : 4.6152   Max.   : 5.5223   Max.   :1.0000     Max.   :1.00000  
     Online         CreditCard      Education_1      Education_2   
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000  
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000  
 Median :1.0000   Median :0.0000   Median :0.0000   Median :0.000  
 Mean   :0.5982   Mean   :0.2882   Mean   :0.4131   Mean   :0.285  
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.000  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000  
  Education_3     Personal Loan   
 Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.0000  
 Median :0.0000   Median :0.0000  
 Mean   :0.3019   Mean   :0.1005  
 3rd Qu.:1.0000   3rd Qu.:0.0000  
 Max.   :1.0000   Max.   :1.0000  
```

```r
#Calculate the variance
var(train_norm_set1[, 1:6])
```

```
                   Age  Experience      Income      Family       CCAvg
Age         1.00000000  0.99429778 -0.04732990 -0.05657548 -0.03250474
Experience  0.99429778  1.00000000 -0.03896261 -0.06246274 -0.03155934
Income     -0.04732990 -0.03896261  1.00000000 -0.14804842  0.63761632
Family     -0.05657548 -0.06246274 -0.14804842  1.00000000 -0.09710343
CCAvg      -0.03250474 -0.03155934  0.63761632 -0.09710343  1.00000000
Mortgage   -0.02014256 -0.01665128  0.21263362 -0.02366249  0.11644228
              Mortgage
Age        -0.02014256
Experience -0.01665128
Income      0.21263362
Family     -0.02366249
CCAvg       0.11644228
Mortgage    1.00000000
```

```r
summary(valid_norm_set1)
```

```
      Age              Experience            Income             Family
 Min.   :-1.939220   Min.   :-2.004297   Min.   :-1.43715   Min.   :-1.21851
 1st Qu.:-0.812536   1st Qu.:-0.878023   1st Qu.:-0.75770   1st Qu.:-1.21851
 Median :-0.032524   Median :-0.011658   Median :-0.20975   Median :-0.35447
 Mean   :-0.008892   Mean   :-0.007222   Mean   : 0.01247   Mean   :-0.03327
 3rd Qu.: 0.834156   3rd Qu.: 0.854707   3rd Qu.: 0.55738   3rd Qu.: 0.50958
 Max.   : 1.874171   Max.   : 1.980981   Max.   : 3.16561   Max.   : 1.37362
     CCAvg              Mortgage        Securities Account   CD Account
 Min.   :-1.111203   Min.   :-0.56158   Min.   :0.00000    Min.   :0.00000
 1st Qu.:-0.727536   1st Qu.:-0.56158   1st Qu.:0.00000    1st Qu.:0.00000
 Median :-0.252247   Median :-0.56158   Median :0.00000    Median :0.00000
 Mean   :-0.004081   Mean   :-0.01246   Mean   :0.09905    Mean   :0.06177
 3rd Qu.: 0.377655   3rd Qu.: 0.40474   3rd Qu.:0.00000    3rd Qu.:0.00000
 Max.   : 4.615173   Max.   : 5.69979   Max.   :1.00000    Max.   :1.00000
     Online          CreditCard       Education_1      Education_2
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :1.0000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :0.5943   Mean   :0.3044   Mean   :0.4302   Mean   :0.2727
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
  Education_3     Personal Loan
 Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.0000   1st Qu.:0.00000
 Median :0.0000   Median :0.00000
 Mean   :0.2972   Mean   :0.08792
 3rd Qu.:1.0000   3rd Qu.:0.00000
 Max.   :1.0000   Max.   :1.00000
```

```r
#Calculate the variance
var(valid_norm_set1[, 1:6])
```

```
                    Age    Experience      Income      Family       CCAvg
Age         0.964380417  0.9588103166 -0.06975031 -0.02660781 -0.08601218
Experience  0.958810317  0.9646945756 -0.06043017 -0.03293567 -0.08234398
Income     -0.069750310 -0.0604301674  1.05058126 -0.17440245  0.67881739
Family     -0.026607815 -0.0329356683 -0.17440245  0.95345056 -0.12880977
CCAvg      -0.086012183 -0.0823439846  0.67881739 -0.12880977  1.00486295
Mortgage    0.001065446  0.0002804293  0.20349936 -0.01467524  0.09941529
               Mortgage
Age         0.0010654464
Experience  0.0002804293
Income      0.2034993599
Family     -0.0146752358
CCAvg       0.0994152864
Mortgage    1.0168521539
```

Training k-NN Model for Validation Set

Traing the k-NN Model using the training and validation set

```r
#First, create the variables for The Y variable which is Personal Loan (last column).
train_predictors1 <-train_norm_set1[,1:13, drop = TRUE]
valid_predictors1 <-valid_norm_set1[,1:13, drop = TRUE]
train_labels1 <-train_norm_set1[,14, drop = TRUE]
valid_labels1  <-valid_norm_set1[,14, drop = TRUE]

#Run the model using k = 1
set.seed(1234)
my_knn2 <-knn(train_predictors1,
                            valid_predictors1,
                            cl=train_labels1,
                            k=1 )

# See the 6 first values of predicted class in the validation set
head(my_knn2)
```

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

```r
# To summarized the model
summary(my_knn2)
```

```
   0    1
1664  133
```

Determing the optimal using Hyperparameter Tuning for Validation Set

Let's find the optimal k using tuning parameters

```r
set.seed(1234)
Search_grid <- expand.grid(k=c(1:20))
train_predict_labels1 <- train_predictors1
train_predict_labels1$Personal_Loan = train_labels1
modeltest1<-train(factor(Personal_Loan)~Age+Experience+Income+Family+
                   CCAvg+Mortgage+`Securities Account`+`CD Account`+Online+
                   CreditCard+Education_1+Education_2+Education_3,
                 data = train_predict_labels1, method="knn",
                 tuneGrid=Search_grid,
                 preProcess='range')

# To show the result
modeltest1
```

```
k-Nearest Neighbors

3203 samples
  13 predictor
   2 classes: '0', '1'

Pre-processing: re-scaling to [0, 1] (13)
Resampling: Bootstrapped (25 reps)
```

```
Summary of sample sizes: 3203, 3203, 3203, 3203, 3203, 3203, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9503554  0.6930832
   2  0.9452988  0.6577146
   3  0.9438377  0.6433481
   4  0.9435997  0.6338016
   5  0.9429516  0.6196916
   6  0.9424734  0.6102541
   7  0.9416765  0.5987671
   8  0.9402743  0.5819964
   9  0.9384071  0.5633323
  10  0.9367152  0.5464649
  11  0.9360400  0.5387656
  12  0.9342771  0.5207271
  13  0.9333490  0.5121482
  14  0.9319518  0.4969679
  15  0.9309023  0.4855445
  16  0.9297518  0.4742680
  17  0.9284311  0.4594290
  18  0.9271093  0.4455629
  19  0.9261323  0.4325538
  20  0.9248099  0.4174314

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 1.
```

Training k-NN Model for Test Set

Traing the k-NN Model using the traval set (training + validation) and testing set

```r
#First, create the variables for The Y variable which is Personal Loan (last column).
traval_predictors1 <-traval_norm_set1[,1:13, drop = TRUE]
test_predictors1 <-test_norm_set1[,1:13, drop = TRUE]
traval_labels1 <-traval_norm_set1[,14, drop = TRUE]
test_labels1  <-test_norm_set1[,14, drop = TRUE]


#Run the model using k = 1
set.seed(1234)
my_knn3 <-knn(traval_predictors1,
                            test_predictors1,
                            cl=traval_labels1,
                            k=1 )

# See the 6 first values of predicted class in the validation set
head(my_knn3)
```

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

```r
# To summarized the model
summary(my_knn3)
```

```
  0   1
914  84
```

Determing the optimal using Hyperparameter Tuning for Test Set

Let's find the optimal k using tuning parameters

```r
set.seed(1234)
Search_grid <- expand.grid(k=c(1:20))
traval_predict_labels1 <- traval_predictors1
traval_predict_labels1$Personal_Loan = traval_labels1
modeltest2<-train(factor(Personal_Loan)~Age+Experience+Income+Family+
                   CCAvg+Mortgage+`Securities Account`+`CD Account`+Online+
                   CreditCard+Education_1+Education_2+Education_3,
                 data = traval_predict_labels1, method="knn",
                 tuneGrid=Search_grid,
                 preProcess='range')

# To show the result
modeltest2
```

```
k-Nearest Neighbors

4002 samples
  13 predictor
   2 classes: '0', '1'

Pre-processing: re-scaling to [0, 1] (13)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 4002, 4002, 4002, 4002, 4002, 4002, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9591370  0.7342128
   2  0.9548035  0.7010894
   3  0.9536538  0.6864198
   4  0.9530212  0.6769620
   5  0.9521833  0.6646934
   6  0.9516948  0.6577040
   7  0.9509923  0.6461417
   8  0.9498775  0.6347656
   9  0.9485197  0.6197664
  10  0.9475126  0.6085382
  11  0.9466458  0.5993160
  12  0.9446654  0.5787931
  13  0.9436382  0.5686613
  14  0.9427109  0.5591373
  15  0.9415228  0.5469459
  16  0.9410365  0.5419454
```

```
  17  0.9394884  0.5257653
  18  0.9384533  0.5147062
  19  0.9372660  0.5024011
  20  0.9358777  0.4867357

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 1.
```

K-NN for Training set

```r
#Run the model using k = 1
set.seed(1234)
my_knn0 <-knn(train_predictors1,
                            train_predictors1,
                            cl=train_labels1,
                            k=1 )

# See the 6 first values of predicted class in the validation set
head(my_knn0)
```

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

```r
# To summarized the model
summary(my_knn0)
```

```
   0    1
2881  322
```

Confusion Matrix for Training set

```r
# Create a confusion matrix
conf_matrix0 <- CrossTable(x=train_labels1,y=my_knn0, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  3203
```

```
               | my_knn0 
train_labels1 |         0 |         1 | Row Total | 
--------------|-----------|-----------|-----------|
            0 |      2881 |         0 |      2881 | 
              |     1.000 |     0.000 |     0.899 | 
              |     1.000 |     0.000 |           | 
              |     0.899 |     0.000 |           | 
--------------|-----------|-----------|-----------|
            1 |         0 |       322 |       322 | 
              |     0.000 |     1.000 |     0.101 | 
              |     0.000 |     1.000 |           | 
              |     0.000 |     0.101 |           | 
--------------|-----------|-----------|-----------|
 Column Total |      2881 |       322 |      3203 | 
              |     0.899 |     0.101 |           | 
--------------|-----------|-----------|-----------|
```

This confusion matrix is showing a perfect output because it is classifing the same data that is used to train the model.

I showed this confusion matrix just to confirm its accuracy. However, it is not fair to compare this result against the confision matrix obtained from the validation and test set.

Calcutale the accuracy, recall, precision, specificity for Training set

```r
#Calcutale the accuracy
k1_accuracy0 <- (conf_matrix0$t[2,2] + conf_matrix0$t[1,1])/ sum(conf_matrix0$t)
print(k1_accuracy0)
```

```
[1] 1
```

```r
#Calcutale the recall
k1_recall0 <- conf_matrix0$t[2,2]/ (conf_matrix0$t[2,2] + conf_matrix0$t[2,1])
print(k1_recall0)
```

```
[1] 1
```

```r
#Calcutale the precision
k1_precision0 <- conf_matrix0$t[2,2]/ (conf_matrix0$t[2,2] + conf_matrix0$t[1,2])
print(k1_precision0)
```

```
[1] 1
```

```r
#Calcutale the specificity
k1_specificity0 <- conf_matrix0$t[1,1]/ (conf_matrix0$t[1,1] + conf_matrix0$t[1,2])
print(k1_specificity0)
```

```
[1] 1
```

Confusion Matrix for Validation set

```r
# Create a confusion matrix
conf_matrix1 <- CrossTable(x=valid_labels1,y=my_knn2, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1797


              | my_knn2
valid_labels1 |         0 |         1 | Row Total |
--------------|-----------|-----------|-----------|
            0 |      1622 |        17 |      1639 |
              |     0.990 |     0.010 |     0.912 |
              |     0.975 |     0.128 |           |
              |     0.903 |     0.009 |           |
--------------|-----------|-----------|-----------|
            1 |        42 |       116 |       158 |
              |     0.266 |     0.734 |     0.088 |
              |     0.025 |     0.872 |           |
              |     0.023 |     0.065 |           |
--------------|-----------|-----------|-----------|
 Column Total |      1664 |       133 |      1797 |
              |     0.926 |     0.074 |           |
--------------|-----------|-----------|-----------|
```

This confution matrix show us that utilizing the validation set the model is wrongly classifying 59 customers, which means 21 extra people were correctly classified.

Probability Output for Validation set

```r
# Create a new variable for our probability
set.seed(1234)
my_knnprob2 <-knn(train_predictors1,
        valid_predictors1,
        cl=train_labels1, k=1, prob=TRUE )

class_prob2 <-attr(my_knnprob2, 'prob')
```

```r
# See the first rows
head(class_prob2)
```

```
[1] 1 1 1 1 1 1
```

Calcutale the accuracy, recall, precision, specificity for Validation set

```r
#Calcutale the accuracy
k1_accuracy2 <- (conf_matrix1$t[2,2] + conf_matrix1$t[1,1])/ sum(conf_matrix1$t)
print(k1_accuracy2)
```

```
[1] 0.9671675
```

```r
#Calcutale the recall
k1_recall2 <- conf_matrix1$t[2,2]/ (conf_matrix1$t[2,2] + conf_matrix1$t[2,1])
print(k1_recall2)
```

```
[1] 0.7341772
```

```r
#Calcutale the precision
k1_precision2 <- conf_matrix1$t[2,2]/ (conf_matrix1$t[2,2] + conf_matrix1$t[1,2])
print(k1_precision2)
```

```
[1] 0.8721805
```

```r
#Calcutale the specificity
k1_specificity2 <- conf_matrix1$t[1,1]/ (conf_matrix1$t[1,1] + conf_matrix1$t[1,2])
print(k1_specificity2)
```

```
[1] 0.9896278
```

Confusion Matrix for Testing set

```r
# Create a confusion matrix
conf_matrix2 <- CrossTable(x=test_labels1,y=my_knn3, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|
```

```
Total Observations in Table:  998


             | my_knn3
test_labels1 |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |       884 |         9 |       893 |
             |     0.990 |     0.010 |     0.895 |
             |     0.967 |     0.107 |           |
             |     0.886 |     0.009 |           |
-------------|-----------|-----------|-----------|
           1 |        30 |        75 |       105 |
             |     0.286 |     0.714 |     0.105 |
             |     0.033 |     0.893 |           |
             |     0.030 |     0.075 |           |
-------------|-----------|-----------|-----------|
Column Total |       914 |        84 |       998 |
             |     0.916 |     0.084 |           |
-------------|-----------|-----------|-----------|
```

When applying the testing, the model is wrongly classifying 39 customers. It proves that the model is perfoming better than the previous one.

Probability Output for Testing set

```
# Create a new variable for our probability
set.seed(1234)
my_knnprob3 <-knn(traval_predictors1,
        test_predictors1,
        cl=traval_labels1, k=1, prob=TRUE )

class_prob3<-attr(my_knnprob3, 'prob')

# See the first rows
head(class_prob3)
```

```
[1] 1 1 1 1 1 1
```

Calcutale the accuracy, recall, precision, specificity for Testing set

```
#Calcutale the accuracy
k1_accuracy3 <- (conf_matrix2$t[2,2] + conf_matrix2$t[1,1])/ sum(conf_matrix2$t)
print(k1_accuracy3)
```

```
## [1] 0.9609218
```

```r
#Calcutale the recall
k1_recall3 <- conf_matrix2$t[2,2]/ (conf_matrix2$t[2,2] + conf_matrix2$t[2,1])
print(k1_recall3)
```

```
## [1] 0.7142857
```

```r
#Calcutale the precision
k1_precision3 <- conf_matrix2$t[2,2]/ (conf_matrix2$t[2,2] + conf_matrix2$t[1,2])
print(k1_precision3)
```

```
## [1] 0.8928571
```

```r
#Calcutale the specificity
k1_specificity3 <- conf_matrix2$t[1,1]/ (conf_matrix2$t[1,1] + conf_matrix2$t[1,2])
print(k1_specificity3)
```

```
## [1] 0.9899216
```

Run K-NN Model to try to predict again the same customer with the new sets

```r
# Calculate the probability for the new observation
my_finalpred <-knn(test_predictors1,
                              m,
                              cl=test_labels1,
                              k=1, prob = TRUE)

# To get the probability output
class_finalprob<-attr(my_finalpred , 'prob')
# Show the first lines
head(class_finalprob)
```

```
[1] 1
```

In conclusion, the new customer is going to be classify as accepting the personal loan form the Universal Bank from the new marketing campaing.