Assignment 2

Varsha Karunya Mekala

2023-09-30

Goal : Use the k Nearest Neighbors model to determine whether a loan offer will be accepted by a new client.

Packages Installed: caret, psych, FNN, gmodels, class, dplyr

```
# Called the packages above using library() command
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(FNN)
library(class)
```

```
##
## Attaching package: 'class'

## The following objects are masked from 'package:FNN':
##
##     knn, knn.cv
```

```
library(gmodels)
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
```

Data Preprocessing/Cleaning

```
setwd("/Users/varshamekala/Desktop/fml")

# Importing Universal Bank data
universal_bank_data =read.csv('UniversalBank.csv')

# Top few rows of the dataset
head(universal_bank_data)
```

```
##   ID Age Experience Income ZIP.Code Family CCAvg Education Mortgage
## 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   1.0         1        0
## 4  4  35          9    100    94112      1   2.7         2        0
## 5  5  35          8     45    91330      4   1.0         2        0
## 6  6  37         13     29    92121      4   0.4         2      155
##   Personal.Loan Securities.Account CD.Account Online CreditCard
## 1             0                  1          0      0          0
## 2             0                  1          0      0          0
## 3             0                  0          0      0          0
## 4             0                  0          0      0          0
## 5             0                  0          0      0          1
## 6             0                  0          0      1          0
```

There are 5000 rows and 14 columns in this dataset

```
# Calculating number of rows and columns in this dataset
dim(universal_bank_data)
```

```
## [1] 5000   14
```

The names of the columns are:

```
# Looking at the columns of the dataset
names(universal_bank_data)
```

```
##  [1] "ID"                "Age"               "Experience"
##  [4] "Income"            "ZIP.Code"          "Family"
##  [7] "CCAvg"             "Education"         "Mortgage"
## [10] "Personal.Loan"     "Securities.Account" "CD.Account"
## [13] "Online"            "CreditCard"
```

Dropping ID and ZIP.Code columns from universal_bank_data dataset

```r
universal_bank_data=subset(universal_bank_data, select=-c(ID, ZIP.Code))
```

Final number of columns after dropping ID and Zip code are 12 and columns names are as follows:

```r
t(t(names(universal_bank_data)))
```

```
##       [,1]
##  [1,] "Age"
##  [2,] "Experience"
##  [3,] "Income"
##  [4,] "Family"
##  [5,] "CCAvg"
##  [6,] "Education"
##  [7,] "Mortgage"
##  [8,] "Personal.Loan"
##  [9,] "Securities.Account"
## [10,] "CD.Account"
## [11,] "Online"
## [12,] "CreditCard"
```

Looking at datatypes of all the columns in the dataset:

```r
sapply(universal_bank_data , class)
```

```
##                 Age          Experience              Income              Family
##           "integer"           "integer"           "integer"           "integer"
##               CCAvg           Education            Mortgage       Personal.Loan
##           "numeric"           "integer"           "integer"           "integer"
## Securities.Account         CD.Account              Online          CreditCard
##           "integer"           "integer"           "integer"           "integer"
```

Education column seems to be an integer and contains 3 unique values 1,2,3

```r
# Get the unique values in Education column
unique(universal_bank_data$Education)
```

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

Creating dummy variables for Education using ifelse commands

```r
universal_bank_data$Education_1 = ifelse(universal_bank_data$Education==1,1,0)

universal_bank_data$Education_2 = ifelse(universal_bank_data$Education==2,1,0)

universal_bank_data$Education_3 = ifelse(universal_bank_data$Education==3,1,0)
```

Dropping Education column

```r
# remove Education column
universal_bank_data <- subset(universal_bank_data, select=-c(Education))
```

After dropping Education variable, we now have 11 variables in the dataset along with 3 dummy variables which in total are 14 columns

```r
t(t(names(universal_bank_data)))
```

```
##       [,1]
##  [1,] "Age"
##  [2,] "Experience"
##  [3,] "Income"
##  [4,] "Family"
##  [5,] "CCAvg"
##  [6,] "Mortgage"
##  [7,] "Personal.Loan"
##  [8,] "Securities.Account"
##  [9,] "CD.Account"
## [10,] "Online"
## [11,] "CreditCard"
## [12,] "Education_1"
## [13,] "Education_2"
## [14,] "Education_3"
```

Creating a parition into two datasets(Train and Validation) with similar percentage of Personal Loan Acceptances (Personal.Loan =1) in each of the partition using Stratified sampling. Split: 60%/40% train/valid datasets

```r
#setting seed
set.seed(4546)
partition_index<- createDataPartition(universal_bank_data$Personal.Loan, p = .6, list = FALSE, times =

train_data  <- universal_bank_data[ partition_index,]
val_data  <- universal_bank_data[-partition_index,]
```

There are 3000 and 2000 rows in training data and validation data after partitioning

The percentages of Personal loan acceptances in train and validation datasets are similar(9.9% and 9.1%) which is expected from stratified sampling

```r
prop.table(table(train_data$Personal.Loan))*100
```

```
##
##         0         1
## 90.066667  9.933333
```

```r
prop.table(table(val_data$Personal.Loan))*100
```

```
##
##    0    1
## 90.9  9.1
```

Creating a new customer data from the given question into a dataframe.

```
new_cust = data.frame(Age =40, Experience=10, Income=84, Family=2, CCAvg=2, Mortgage=0, Securities.Acco
```

Initializing normalized Training, Validation data, universal_bank_data to originals

```
train_norm_df = train_data
val_norm_df =val_data
universal_bank_norm_df = universal_bank_data
```

Normalizing the training and validation data without the Personal.Loan columns(Outcome/Target variable) Used Preprocess on the train_data to use the estimated parameters to normalize the train_data, val_data and new_cust datasets

```
train_normalized = preProcess(train_data[,-7], method = c("center", "scale"))

train_norm_df[,-7] = predict(train_normalized, train_data[,-7])

val_norm_df[,-7] = predict(train_normalized, val_data[,-7])

universal_bank_norm_df[,-7]= predict(train_normalized,universal_bank_data[,-7])

new_norm_cust = predict(train_normalized, new_cust)
```

Performing K-NN classification on training data and testing against validation data using k=1

```
set.seed(4546)
prediction_vals <- knn(train = train_norm_df[,-7], test = val_norm_df[,-7],
          cl = train_norm_df[,7], k = 1, prob=TRUE)
actual_vals= val_norm_df$Personal.Loan
prediction_probabilities = attr(prediction_vals,"prob")
# confusion matrix:
table(prediction_vals,actual_vals)
```

```
##                actual_vals
## prediction_vals    0    1
##               0 1789   47
##               1   29  135
```

Question-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?

Answer:

KNN model classified the new customer data to Class 1 meaning that the customer will Accept the personal loan that is offered to them.

```
set.seed(4546)
new_norm_cust_pred <- knn(train = train_norm_df[,-7], test = new_norm_cust,
          cl = train_norm_df[,7], k = 1, prob=TRUE)
class_prob = attr(new_norm_cust_pred, 'prob')
class_prob
```

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

Question-2:

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

Finding the accuracy table to find the best K based on the accuracy on the validation data will give us k which is not prone to over-fitting while also not ignoring the information in the predictor variables.

Answer:

From the table computed below, k=3 seems to be the best performer on validation data with respect to accuracy(0.97)

```
# Initializing a data frame with two columns: k_vals, and accuracy_vals
# Iterating over 70 different values of k and computing accuracy on validation data to get the best k
accuracy_table <- data.frame(k_vals = seq(1, 70, 1), accuracy_vals = rep(0, 70))

# Computing knn for different k on validation.
for(i in 1:70) {
  knn_predictions <- knn(train_norm_df[, -7], val_norm_df[, -7],
  cl = train_norm_df[, 7], k = i)
  accuracy_table[i, 2] <- confusionMatrix(knn_predictions, as.factor(val_norm_df[, 7]))$overall[1]
}

accuracy_table
```

```
##    k_vals accuracy_vals
## 1       1        0.9620
## 2       2        0.9575
## 3       3        0.9700
## 4       4        0.9675
## 5       5        0.9660
## 6       6        0.9640
## 7       7        0.9630
## 8       8        0.9615
## 9       9        0.9615
## 10     10        0.9585
## 11     11        0.9590
```

```
## 12     12        0.9580
## 13     13        0.9560
## 14     14        0.9545
## 15     15        0.9535
## 16     16        0.9525
## 17     17        0.9525
## 18     18        0.9505
## 19     19        0.9505
## 20     20        0.9490
## 21     21        0.9470
## 22     22        0.9485
## 23     23        0.9455
## 24     24        0.9450
## 25     25        0.9440
## 26     26        0.9440
## 27     27        0.9445
## 28     28        0.9465
## 29     29        0.9440
## 30     30        0.9430
## 31     31        0.9415
## 32     32        0.9410
## 33     33        0.9400
## 34     34        0.9400
## 35     35        0.9385
## 36     36        0.9395
## 37     37        0.9395
## 38     38        0.9390
## 39     39        0.9390
## 40     40        0.9390
## 41     41        0.9395
## 42     42        0.9375
## 43     43        0.9375
## 44     44        0.9385
## 45     45        0.9380
## 46     46        0.9380
## 47     47        0.9365
## 48     48        0.9360
## 49     49        0.9375
## 50     50        0.9375
## 51     51        0.9365
## 52     52        0.9365
## 53     53        0.9355
## 54     54        0.9355
## 55     55        0.9360
## 56     56        0.9365
## 57     57        0.9360
## 58     58        0.9350
## 59     59        0.9335
## 60     60        0.9320
## 61     61        0.9320
## 62     62        0.9325
## 63     63        0.9325
## 64     64        0.9320
## 65     65        0.9325
```

```
## 66     66        0.9325
## 67     67        0.9315
## 68     68        0.9310
## 69     69        0.9310
## 70     70        0.9305
```

Question-3:

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

Finding confusion matrix with the best K(3) using 2 different methodologies 1) Using CrossTable from gmodels 2) Using confusionmatrix() as well

```
set.seed(4546)
knn_predictions <- knn(train_norm_df[, -7], val_norm_df[, -7],
  cl = train_norm_df[, 7], k = 3)

CrossTable(x=as.factor(val_norm_df[, 7]), y=knn_predictions, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  2000
##
##
##                              | knn_predictions
## as.factor(val_norm_df[, 7]) |         0 |         1 | Row Total |
## ----------------------------|-----------|-----------|-----------|
##                           0 |      1811 |         7 |      1818 |
##                             |     0.996 |     0.004 |     0.909 |
##                             |     0.972 |     0.051 |           |
##                             |     0.905 |     0.004 |           |
## ----------------------------|-----------|-----------|-----------|
##                           1 |        53 |       129 |       182 |
##                             |     0.291 |     0.709 |     0.091 |
##                             |     0.028 |     0.949 |           |
##                             |     0.026 |     0.065 |           |
## ----------------------------|-----------|-----------|-----------|
##               Column Total |      1864 |       136 |      2000 |
##                             |     0.932 |     0.068 |           |
## ----------------------------|-----------|-----------|-----------|
##
##
```

```
confusionMatrix(knn_predictions, as.factor(val_norm_df[, 7]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1811   53
##          1    7  129
##
##                Accuracy : 0.97
##                  95% CI : (0.9616, 0.977)
##     No Information Rate : 0.909
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7954
##
##  Mcnemar's Test P-Value : 6.267e-09
##
##             Sensitivity : 0.9961
##             Specificity : 0.7088
##          Pos Pred Value : 0.9716
##          Neg Pred Value : 0.9485
##              Prevalence : 0.9090
##          Detection Rate : 0.9055
##    Detection Prevalence : 0.9320
##       Balanced Accuracy : 0.8525
##
##        'Positive' Class : 0
##
```

Precision = TP/(TP+FP) = 0.948 Recall = TP/(TP+FN) = 0.71

Question-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.

Answer:

Using the best k from the accuracy table: 3 to classify the new data point. The new customer is classified as the customer that will Accept the personal loan that is offered to them(class =1)

```
knn_new_predictions_w_best_k <- knn(universal_bank_norm_df[, -7], new_norm_cust,
cl = universal_bank_norm_df[, 7], k = 3, prob=TRUE)
class_prob = attr(knn_new_predictions_w_best_k, 'prob')
class_prob
```

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

Question-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.

Re-partitioning into train, valid and test datasets based on the outcome variable(Personal.Loan) using stratified sampling

```
# Create a stratified random split of the data into test dataset and use the remaining dataset for furt
set.seed(4546)

partition_index_mod<- createDataPartition(universal_bank_data$Personal.Loan, p = 0.2, list = FALSE, tim
test  <- universal_bank_data[partition_index_mod,]
data_remaining  <- universal_bank_data[-partition_index_mod,]

# Create a stratified random split of the training data into validation and testing sets
split_train_val <- createDataPartition(data_remaining$Personal.Loan, p = 0.625, list = FALSE, times=1)
train <- data_remaining[split_train_val,]
val  <- data_remaining[-split_train_val,]
```

Best k chosen was 3. I will be using k=3 in the KNN model with this re-partitioned dataset.

```
# Normalizing the data here
train_norm = train
val_norm =val
test_norm =test
data_remaining_norm = data_remaining

train_normalized_mod = preProcess(train[,-7], method = c("center", "scale"))

train_norm[,-7] = predict(train_normalized_mod, train[,-7])
val_norm[,-7] = predict(train_normalized_mod, val[,-7])
test_norm[,-7] = predict(train_normalized_mod, test[,-7])
data_remaining_norm[,-7]=predict(train_normalized_mod, data_remaining[,-7])
```

Performing KNN using k=3 here on Training data and calculating the confusion matrix for validation data:

```
set.seed(4546)
prediction_repart_1 <- knn(train = train_norm[,-7], test = val_norm[,-7],
          cl = train_norm[,7], k = 3, prob=TRUE)
actual_repart_1= val_norm$Personal.Loan
prediction_prob_repart_1 = attr(prediction_repart_1,"prob")

CrossTable(x=actual_repart_1,y=prediction_repart_1, prop.chisq = FALSE)
```

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

```
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1500
##
##
##                  | prediction_repart_1
## actual_repart_1 |         0 |         1 | Row Total |
## ----------------|-----------|-----------|-----------|
##               0 |      1349 |         6 |      1355 |
##                 |     0.996 |     0.004 |     0.903 |
##                 |     0.964 |     0.059 |           |
##                 |     0.899 |     0.004 |           |
## ----------------|-----------|-----------|-----------|
##               1 |        50 |        95 |       145 |
##                 |     0.345 |     0.655 |     0.097 |
##                 |     0.036 |     0.941 |           |
##                 |     0.033 |     0.063 |           |
## ----------------|-----------|-----------|-----------|
##    Column Total |      1399 |       101 |      1500 |
##                 |     0.933 |     0.067 |           |
## ----------------|-----------|-----------|-----------|
##
##
```

```
confusionMatrix(prediction_repart_1, as.factor(actual_repart_1))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1349   50
##          1    6   95
##
##                Accuracy : 0.9627
##                  95% CI : (0.9518, 0.9717)
##     No Information Rate : 0.9033
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7527
##
##  Mcnemar's Test P-Value : 9.132e-09
##
##             Sensitivity : 0.9956
##             Specificity : 0.6552
##          Pos Pred Value : 0.9643
##          Neg Pred Value : 0.9406
##              Prevalence : 0.9033
##          Detection Rate : 0.8993
##    Detection Prevalence : 0.9327
##       Balanced Accuracy : 0.8254
##
##        'Positive' Class : 0
```

##

Performing KNN using k=3 here on combined Training and Validation data and calculating the confusion matrix for testing data:

```
set.seed(2019)
prediction_repart_2 <- knn(train = data_remaining_norm[,-7], test = test_norm[,-7],
          cl = data_remaining_norm[,7], k = 3, prob=TRUE)
actual_repart_2= test_norm$Personal.Loan
prediction_prob = attr(prediction_repart_2,"prob")

#Confusion Matrix of Testing data
CrossTable(x=actual_repart_2,y=prediction_repart_2, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1000
##
##
##                  | prediction_repart_2
## actual_repart_2 |         0 |         1 | Row Total |
## ----------------|-----------|-----------|-----------|
##               0 |       907 |         4 |       911 |
##                 |     0.996 |     0.004 |     0.911 |
##                 |     0.957 |     0.077 |           |
##                 |     0.907 |     0.004 |           |
## ----------------|-----------|-----------|-----------|
##               1 |        41 |        48 |        89 |
##                 |     0.461 |     0.539 |     0.089 |
##                 |     0.043 |     0.923 |           |
##                 |     0.041 |     0.048 |           |
## ----------------|-----------|-----------|-----------|
##    Column Total |       948 |        52 |      1000 |
##                 |     0.948 |     0.052 |           |
## ----------------|-----------|-----------|-----------|
##
##
```

```
confusionMatrix(prediction_repart_2, as.factor(actual_repart_2))
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction   0   1
##          0 907  41
##          1   4  48
##
##                Accuracy : 0.955
##                  95% CI : (0.9402, 0.967)
##     No Information Rate : 0.911
##     P-Value [Acc > NIR] : 6.696e-08
##
##                   Kappa : 0.6584
##
##  Mcnemar's Test P-Value : 8.025e-08
##
##             Sensitivity : 0.9956
##             Specificity : 0.5393
##          Pos Pred Value : 0.9568
##          Neg Pred Value : 0.9231
##              Prevalence : 0.9110
##          Detection Rate : 0.9070
##    Detection Prevalence : 0.9480
##       Balanced Accuracy : 0.7675
##
##        'Positive' Class : 0
##
```

Performing KNN using k=3 here on combined Training data and calculating the confusion matrix for training data

```
set.seed(4546)
prediction_repart_3 <- knn(train = train_norm[,-7], test = train_norm[,-7],
          cl = train_norm[,7], k = 3, prob=TRUE)
actual_repart_3= train_norm$Personal.Loan
prediction_prob_repart_3 = attr(prediction_repart_3,"prob")

#Confusion Matrix of Testing data
CrossTable(x=actual_repart_3,y=prediction_repart_3, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  2500
##
##
##                 | prediction_repart_3
## actual_repart_3 |         0 |         1 | Row Total |
```

```
## ----------------|-----------|-----------|-----------|
##               0 |      2249 |         5 |      2254 |
##                 |     0.998 |     0.002 |     0.902 |
##                 |     0.976 |     0.026 |           |
##                 |     0.900 |     0.002 |           |
## ----------------|-----------|-----------|-----------|
##               1 |        56 |       190 |       246 |
##                 |     0.228 |     0.772 |     0.098 |
##                 |     0.024 |     0.974 |           |
##                 |     0.022 |     0.076 |           |
## ----------------|-----------|-----------|-----------|
##    Column Total |      2305 |       195 |      2500 |
##                 |     0.922 |     0.078 |           |
## ----------------|-----------|-----------|-----------|
##
##
```

```r
confusionMatrix(prediction_repart_3, as.factor(actual_repart_3))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2249   56
##          1    5  190
##
##                Accuracy : 0.9756
##                  95% CI : (0.9688, 0.9813)
##     No Information Rate : 0.9016
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8485
##
##  Mcnemar's Test P-Value : 1.535e-10
##
##             Sensitivity : 0.9978
##             Specificity : 0.7724
##          Pos Pred Value : 0.9757
##          Neg Pred Value : 0.9744
##              Prevalence : 0.9016
##          Detection Rate : 0.8996
##    Detection Prevalence : 0.9220
##       Balanced Accuracy : 0.8851
##
##        'Positive' Class : 0
##
```

Comparing the differences between the Confusion matrices:

Training Dataset:

Accuracy = 0.9756 TN =2249 FN=56 FP=5 TP=190 Precision = TP/ (TP+FP) = 190/(190+5) = 0.9744 Recall = TP / (TP+FN)= 190/(190+56) = 0.7724

Validation Dataset

Accuracy = 0.9627 TN =1349 FN=50 FP=6 TP=95 Precision = 95/(95+6) = 0.9406 Recall = 95/(95+50) = 0.6551

Testing Dataset:

Accuracy = 0.955 TN =907 FN=41 FP=4 TP=48 Precision = 48/(48+4) = 0.923 Recall = 48/(48+41) = 0.539

Conclusions:

We can observe that the training set has the highest accuracy, which is followed by the validation set and the test set. This is to be expected as the model has only been tested on the validation and test sets, and has only been trained on the training set. The model may not generalize well to new data if it is overfit to the training set, which would result in decreased accuracy on the validation and test sets. Since the accuracy on the validation and test sets is still relatively good in this instance, it is likely that the model is not overfitting and will generalize well to new data.

Additionally, as each set represents a different sample of the population, we may anticipate to notice some variations in the distribution of the predictor variables between the training, validation, and test sets' confusion matrices.

Since accuracy and precision appear to be near to each other across all datasets, the model may be successfully adapted to new data and resistant to overfitting.

Recall, however, appears to be the metric with the biggest disparity between the three data sets mentioned above. This disparity could be caused by numerous factors, including overfitting, unbalanced classes, and poor data quality.

Since we use stratified sampling, unbalanced classes might not be a problem in this situation.

To address these issues: 1. We can regularize the model to prevent overfitting 2. We can improve the quality of training data 3. We can use recall and Precision as a metric to find the best K instead of accuracy which is not always a good metric.