```r
###Data Import and Cleaning

 #firstly install and load  the pacakges "class","caret","e1071"
library(class)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

UPLOADING DATASET AND KNOWING THE DIMENSIONS

```r
UniversalBank <- read.csv("/Users/chaithanayayennam/Downloads/UniversalBank.csv")
dim(UniversalBank)
```

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

```r
head(UniversalBank)
```

```
##   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
```

```r
tail(UniversalBank)
```

```
##        ID Age Experience Income ZIP.Code Family CCAvg Education Mortgage
## 4995 4995  64         40     75    94588      3   2.0         3        0
## 4996 4996  29          3     40    92697      1   1.9         3        0
## 4997 4997  30          4     15    92037      4   0.4         1       85
## 4998 4998  63         39     24    93023      2   0.3         3        0
## 4999 4999  65         40     49    90034      3   0.5         2        0
## 5000 5000  28          4     83    92612      3   0.8         1        0
##      Personal.Loan Securities.Account CD.Account Online CreditCard
## 4995             0                  0          0      1          0
## 4996             0                  0          0      1          0
## 4997             0                  0          0      1          0
## 4998             0                  0          0      0          0
## 4999             0                  0          0      1          0
## 5000             0                  0          0      1          1
```

THE DATA FRAME IS TRANSPOSED USING THE t FUNCTION TO MAKE THE ANALYSIS EASIER

```
t(t(names(UniversalBank)))
```

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

RETRIEVING THE CURRENT WORKING DIRECTORY

```
getwd()
```

```
## [1] "/Users/chaithanayayennam"
```

REMOVING ID AND ZIP CODE COLUMNS

```
Data_original <- UniversalBank[,-c(1,5)]
dim(Data_original)
```

```
## [1] 5000   12
```

SPLITTING DATA INTO 60% TRAINING AND 40% VALIDATION

```
Data_original$Education <- as.factor(Data_original$Education)
```

CREATING DUMMY VARIABLES AND COMBINING THEM IN THE DATASET

```
dummys<- dummyVars(~.,data=Data_original)
Data_original<- as.data.frame(predict(dummys,Data_original))
```

PARTITIONING THE DATA INTO TRAINING AND TESTING DATASETS

```
set.seed(1)
data_train.set <- sample(row.names(Data_original), 0.6*dim(Data_original)[1])
data_valid.set <- setdiff(row.names(Data_original),data_train.set)
train_data <- Data_original[data_train.set,]
valid_data <- Data_original[data_valid.set,]
t(t(names(train_data)))
```

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

```r
summary(train_data)
```

```
##       Age          Experience        Income           Family     
##  Min.   :23.00   Min.   :-3.00   Min.   :  8.00   Min.   :1.000  
##  1st Qu.:36.00   1st Qu.:10.00   1st Qu.: 39.00   1st Qu.:1.000  
##  Median :45.00   Median :20.00   Median : 63.00   Median :2.000  
##  Mean   :45.43   Mean   :20.19   Mean   : 73.08   Mean   :2.388  
##  3rd Qu.:55.00   3rd Qu.:30.00   3rd Qu.: 98.00   3rd Qu.:3.000  
##  Max.   :67.00   Max.   :43.00   Max.   :224.00   Max.   :4.000  
##      CCAvg          Education.1      Education.2     Education.3    
##  Min.   : 0.000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000  
##  1st Qu.: 0.700   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000  
##  Median : 1.500   Median :0.0000   Median :0.000   Median :0.0000  
##  Mean   : 1.915   Mean   :0.4173   Mean   :0.285   Mean   :0.2977  
##  3rd Qu.: 2.500   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000  
##  Max.   :10.000   Max.   :1.0000   Max.   :1.000   Max.   :1.0000  
##     Mortgage       Personal.Loan     Securities.Account   CD.Account     
##  Min.   :  0.00   Min.   :0.00000   Min.   :0.0000     Min.   :0.00000  
##  1st Qu.:  0.00   1st Qu.:0.00000   1st Qu.:0.0000     1st Qu.:0.00000  
##  Median :  0.00   Median :0.00000   Median :0.0000     Median :0.00000  
##  Mean   : 57.34   Mean   :0.09167   Mean   :0.1003     Mean   :0.05367  
##  3rd Qu.:102.00   3rd Qu.:0.00000   3rd Qu.:0.0000     3rd Qu.:0.00000  
##  Max.   :635.00   Max.   :1.00000   Max.   :1.0000     Max.   :1.00000  
##      Online         CreditCard    
##  Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :1.0000   Median :0.0000  
##  Mean   :0.5847   Mean   :0.2927  
##  3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :1.0000
```

```r
cat("The size of the training dataset is:",nrow(train))
```

```
## The size of the training dataset is:
```

```r
summary(valid_data)
```

```
##       Age          Experience        Income           Family     
##  Min.   :23.0   Min.   :-3.00   Min.   :  8.00   Min.   :1.000  
##  1st Qu.:35.0   1st Qu.:10.00   1st Qu.: 39.00   1st Qu.:1.000  
##  Median :45.0   Median :20.00   Median : 64.00   Median :2.000  
##  Mean   :45.2   Mean   :19.97   Mean   : 74.81   Mean   :2.409  
##  3rd Qu.:55.0   3rd Qu.:30.00   3rd Qu.: 99.00   3rd Qu.:3.000  
##  Max.   :67.0   Max.   :43.00   Max.   :218.00   Max.   :4.000  
##      CCAvg         Education.1     Education.2     Education.3   
##  Min.   : 0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
##  1st Qu.: 0.700   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000  
##  Median : 1.600   Median :0.000   Median :0.000   Median :0.000  
##  Mean   : 1.973   Mean   :0.422   Mean   :0.274   Mean   :0.304  
##  3rd Qu.: 2.600   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.000  
##  Max.   :10.000   Max.   :1.000   Max.   :1.000   Max.   :1.000  
##     Mortgage       Personal.Loan    Securities.Account   CD.Account    
##  Min.   :  0.00   Min.   :0.0000   Min.   :0.0000     Min.   :0.0000  
##  1st Qu.:  0.00   1st Qu.:0.0000   1st Qu.:0.0000     1st Qu.:0.0000  
##  Median :  0.00   Median :0.0000   Median :0.0000     Median :0.0000  
##  Mean   : 55.24   Mean   :0.1025   Mean   :0.1105     Mean   :0.0705  
##  3rd Qu.: 97.25   3rd Qu.:0.0000   3rd Qu.:0.0000     3rd Qu.:0.0000  
##  Max.   :617.00   Max.   :1.0000   Max.   :1.0000     Max.   :1.0000  
##      Online        CreditCard   
##  Min.   :0.000   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:0.000  
##  Median :1.000   Median :0.000  
##  Mean   :0.615   Mean   :0.296  
##  3rd Qu.:1.000   3rd Qu.:1.000  
##  Max.   :1.000   Max.   :1.000
```

```r
cat("The size of the validation dataset is:",nrow(valid_data))
```

```
## The size of the validation dataset is: 2000
```

Now, let us normalize the data

```r
train_norm.set <- train_data[,-10]
valid_norm.set <- valid_data[,-10]
normtn <- preProcess(train_data[,-10],method=c("center","scale"))
5
```

```
## [1] 5
```

```r
train_norm.set <- predict(normtn,train_data[,-10])
valid_norm.set <- predict(normtn,valid_data[,-10])
```

CREATING NEW CUSTOMER DATA

```r
newdata<- data.frame(
Age = 40,
Experience = 10,
Income = 84,
Family = 2,
CCAvg = 2,
Education.1 = 0,
Education.2 = 1,
Education.3 = 0,
Mortgage = 0,
Securities.Account = 0,
CD.Account = 0,
Online = 1,
CreditCard = 1
)

# Normalize the new customer dataset
customer_norm.set <- predict(normtn, newdata)
```

PERFORMING kNN CLASSIFICATION

```r
prediction_data <- class::knn(train = train_norm.set,
                         test = customer_norm.set,
                         cl = train_data$Personal.Loan,
                         k = 1)
prediction_data
```

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

CHOICE OF k THAT BALANCES BETWEEN OVERFITTING AND IGNORING THE PREDICTOR INFORMATION

```r
# Calculate the accuracy for each value of k
# Set the range of k values to consider
accuracy_data <- data.frame(k = seq(1, 15, 1), overallaccuracy = rep(0, 15))
for(i in 1:15) {
kn <- class::knn(train = train_norm.set,
test = valid_norm.set,
cl = train_data$Personal.Loan, k = i)
accuracy_data[i, 2] <- confusionMatrix(kn,
as.factor(valid_data$Personal.Loan),positive = "1")$overall[1]
}
which(accuracy_data[,2] == max(accuracy_data[,2]))
```

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

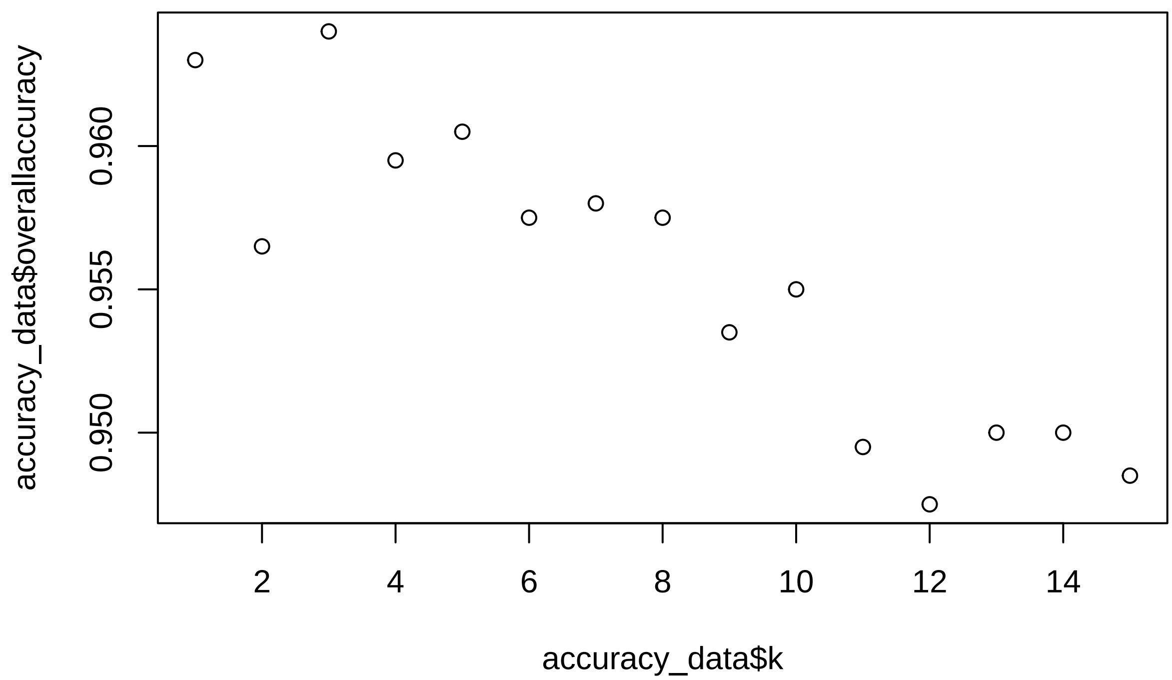
```r
accuracy_data
```

```
##     k overallaccuracy
## 1   1          0.9630
```

```
## 2   2          0.9565
## 3   3          0.9640
## 4   4          0.9595
## 5   5          0.9605
## 6   6          0.9575
## 7   7          0.9580
## 8   8          0.9575
## 9   9          0.9535
## 10 10          0.9550
## 11 11          0.9495
## 12 12          0.9475
## 13 13          0.9500
## 14 14          0.9500
## 15 15          0.9485
```

PLOTTING ACCURACY

```r
plot(accuracy_data$k,accuracy_data$overallaccuracy)
```



CONFUSION MATRIX

```r
prediction_data <- class::knn(train = train_norm.set,
                              test = valid_norm.set,
                              cl = train_data$Personal.Loan,
                              k = 3)
confusionMatrix(prediction_data, as.factor(valid_data$Personal.Loan))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1786   63
##          1    9  142
##
##                Accuracy : 0.964
##                  95% CI : (0.9549, 0.9717)
##     No Information Rate : 0.8975
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7785
##
##  Mcnemar's Test P-Value : 4.208e-10
##
##             Sensitivity : 0.9950
##             Specificity : 0.6927
##          Pos Pred Value : 0.9659
##          Neg Pred Value : 0.9404
##              Prevalence : 0.8975
##          Detection Rate : 0.8930
##    Detection Prevalence : 0.9245
##       Balanced Accuracy : 0.8438
##
##        'Positive' Class : 0
##
```

CONSIDERING 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 CreditCard = 1. **CLASSIFYING THE CUSTOMER USING THE BEST k**

```r
customer_set2 <- data.frame(
Age = 40,
Experience = 10,
Income = 84,
Family = 2,
CCAvg = 2,
Education.1 = 0,
Education.2 = 1,
Education.3 = 0,
Mortgage = 0,
Securities.Account = 0,
CD.Account = 0,
Online = 1,
CreditCard = 1)

#Normalizing the 2nd client dataset
customer_set2_norm <- predict(normtn, customer_set2)
```

REEATING THE PROCESS BY DIVIDING THE DATA INTO THREE PARTS(50%, 30%, and 20%)

```
set.seed(500)
Train_index.set <- sample(row.names(Data_original), .5*dim(Data_original)[1])#create train index
9
```

```
## [1] 9
```

```
#create validation index
Valid_Index <- sample(setdiff(row.names(Data_original),Train_index.set),.3*dim(Data_original)[1])
Test_Index =setdiff(row.names(Data_original),union(Train_index.set,Valid_Index))#create test index
train_dataframe <- Data_original[Train_index.set,]
cat("The size of the new training dataset is:", nrow(train_dataframe))
```

```
## The size of the new training dataset is: 2500
```

```
valid_dataframe <- Data_original[Valid_Index, ]
cat("The size of the new validation dataset is:", nrow(valid_dataframe))
```

```
## The size of the new validation dataset is: 1500
```

```
test_dataframe <- Data_original[Test_Index, ]
cat("The size of the new test dataset is:", nrow(test_dataframe))
```

```
## The size of the new test dataset is: 1000
```

NORMALIZING THE DATA

```
normvalues <- preProcess(train_dataframe[, -10], method=c("center", "scale"))
train.df.norm <- predict(normtn, train_dataframe[, -10])
valid.df.norm <- predict(normtn, valid_dataframe[, -10])
test.df.norm <- predict(normtn ,test_dataframe[,-10])
```

PERFORMING kNN AND CREATING CONFUSION MATRIX ON TRAINING, TESTING, VALIDATION DATA

```
length_train <- nrow(train.df.norm)
length_class <- length(train_data$Personal.Loan)
if (length_train != length_class) {
  stop
} else {
  prediction_3 <- class::knn(train = train.df.norm,
                             test = test.df.norm,
                             cl = train_data$Personal.Loan,
                             k = 3)
  confusionMatrix(prediction_3, as.factor(test_dataframe$Personal.Loan))
}
```

```
## function (..., call. = TRUE, domain = NULL)
## {
##     if (...length() == 1L && inherits(..1, "condition")) {
##         cond <- ..1
```

```
##         if (nargs() > 1L)
##             warning("additional arguments ignored in stop()")
##         message <- conditionMessage(cond)
##         call <- conditionCall(cond)
##         .Internal(.signalCondition(cond, message, call))
##         .Internal(.dfltStop(message, call))
##     }
##     else .Internal(stop(call., .makeMessage(..., domain = domain)))
## }
## <bytecode: 0x7ff4e8bfdbb0>
## <environment: namespace:base>
```

```
prediction_4 <- class::knn(train = train.df.norm,
                           test = valid.df.norm,
                           cl = train_dataframe$Personal.Loan,
                           k = 3)
confusionMatrix(prediction_4, as.factor(valid_dataframe$Personal.Loan))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1332   65
##          1    8   95
##
##                Accuracy : 0.9513
##                  95% CI : (0.9392, 0.9617)
##     No Information Rate : 0.8933
##     P-Value [Acc > NIR] : 6.496e-16
##
##                   Kappa : 0.6971
##
##  Mcnemar's Test P-Value : 5.590e-11
##
##             Sensitivity : 0.9940
##             Specificity : 0.5938
##          Pos Pred Value : 0.9535
##          Neg Pred Value : 0.9223
##              Prevalence : 0.8933
##          Detection Rate : 0.8880
##    Detection Prevalence : 0.9313
##       Balanced Accuracy : 0.7939
##
##        'Positive' Class : 0
##
```

```
prediction_5 <- class::knn(train = train.df.norm,
test = train.df.norm,
cl = train_dataframe$Personal.Loan, k=3)
confusionMatrix(prediction_5,as.factor(train_dataframe$Personal.Loan))
```

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

```
##           Reference
## Prediction    0    1
##          0 2273   53
##          1    3  171
##
##                Accuracy : 0.9776
##                  95% CI : (0.971, 0.983)
##     No Information Rate : 0.9104
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8473
##
##  Mcnemar's Test P-Value : 5.835e-11
##
##             Sensitivity : 0.9987
##             Specificity : 0.7634
##          Pos Pred Value : 0.9772
##          Neg Pred Value : 0.9828
##              Prevalence : 0.9104
##          Detection Rate : 0.9092
##    Detection Prevalence : 0.9304
##       Balanced Accuracy : 0.8810
##
##        'Positive' Class : 0
##
```