Assignment_2

Garsego

2025-10-03

I start by loading the file and packages needed:

```
Universal_Bank <- read.csv("C:\\Users\\arseg\\Downloads\\UniversalBank.csv")

View(Universal_Bank)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.4.3

## Loading required package: lattice
```

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.4.3
```

```
library(gmodels)
```

```
## Warning: package 'gmodels' was built under R version 4.4.3
```

Transforming "Education" into a factor (categorical)

```
#To do so, I used as.factor() and then created the dummy model with dummyVars()
Universal_Bank$Education <- as.factor(Universal_Bank$Education)
dummy_model <- dummyVars(~Education, data = Universal_Bank)
education_dummy <- as.data.frame(predict(dummy_model, Universal_Bank))
```

Replacing the "Education" column by the "education_dummy"

```
#Using cbind I replaced the Education column by the 3 columns created for the dummy model
Universal_Bank <- cbind(Universal_Bank[ , !(names(Universal_Bank) %in% "Education")],
                        education_dummy)
View(Universal_Bank)
```

Separating Personal.Loan as the target variable

```r
#So that I don't normalize Personal.Loan which is what we are trying to predict, and also ID and ZIP.Co
Target <- Universal_Bank$Personal.Loan
Predictors <- Universal_Bank[, !(names(Universal_Bank) %in% c("ID", "ZIP.Code", "Personal.Loan"))]

View(Predictors)
```

Here I normalize the data so that large variables don't overshadow smaller ones in the knn model

```r
#I use preProcess() to prepare the data before modeling
norm_model <- preProcess(Predictors, method = "range")
#And then with predict() I apply the transformation so that each variable stays in the range from 0 to
Predictors_normalized <- predict(norm_model, Predictors)
#Once again using cbind() here to create a single data frame with the normalized variables
Universal_Bank_normalized <- cbind(Predictors_normalized, Personal.Loan = Target)
#Since we want to classify a customer as "loan acceptance" or not using 1 and 0 respectively, I needed
Universal_Bank_normalized$Personal.Loan <- as.factor(Universal_Bank_normalized$Personal.Loan)
View(Universal_Bank_normalized)
```

Partitioning data into training (60%) and validation (40%):

```r
set.seed(123)
#After seting seed, I partition the data using createDataPartition()
Train_Index <- createDataPartition(Universal_Bank_normalized$Personal.Loan, p = 0.6, list = FALSE)

Training_data = Universal_Bank_normalized[Train_Index, ]
Validation_data = Universal_Bank_normalized[-Train_Index, ]
```

Question 1

```r
#For question 1 I needed to create a knn model using k=1, I used the train() function for that
knn_model1 <- train( Personal.Loan ~ ., data = Training_data, method = "knn", tuneGrid = data.frame(k =
#Here I created a data frame for the Customer, making sure that I input the columns in the same order a
Customer_1 <- data.frame( Age = 40,
                          Experience = 10,
                          Income = 84,
                          Family = 2,
                          CCAvg = 2,
                          Mortgage = 0,
                          Securities.Account = 0,
                          CD.Account = 0,
                          Online = 1,
                          CreditCard = 1,
                          Education.1 = 0,
                          Education.2 = 1,
                          Education.3 = 0)
#To normalize the Customer data I used predict() with norm_model
Customer_1_normalized <- predict(norm_model, Customer_1)
View(Customer_1_normalized)
#And then used the knn_model1 to create the prediction
Customer_1_Prediction <- predict(knn_model1, Customer_1_normalized)
Customer_1_Prediction
```

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

```
#This customer woud be classified as "not accepted", meaning that he wouldn't accept the personal loan
```

Question 2

```
#Here I used a different method from my previous submission. By creating k_choices with expand.grid() I
k_choices <- expand.grid(k=seq(1, 55, 2))
knn_model2 <- train( Personal.Loan ~ ., data = Training_data, method = "knn", tuneGrid = k_choices, trC
knn_model2
```

```
## k-Nearest Neighbors
##
## 3000 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2700, 2700, 2700, 2700, 2700, 2700, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9586543  0.73433203
##    3  0.9519888  0.66446105
##    5  0.9469977  0.61356433
##    7  0.9403321  0.54292037
##    9  0.9393321  0.52627868
##   11  0.9363354  0.49290552
##   13  0.9319976  0.44698779
##   15  0.9303332  0.42743138
##   17  0.9279976  0.39487727
##   19  0.9259976  0.37588030
##   21  0.9239999  0.34858282
##   23  0.9213365  0.31174661
##   25  0.9186687  0.28364280
##   27  0.9173343  0.25704932
##   29  0.9143332  0.21089030
##   31  0.9133331  0.19117668
##   33  0.9113331  0.15957347
##   35  0.9099998  0.13862781
##   37  0.9086654  0.11163055
##   39  0.9066665  0.07880396
##   41  0.9059998  0.06772844
##   43  0.9053331  0.05006312
##   45  0.9053331  0.03996958
##   47  0.9050009  0.03422605
##   49  0.9050009  0.02895601
##   51  0.9049998  0.02345591
##   53  0.9056654  0.02999069
##   55  0.9056654  0.02999069
##
```

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

```r
#Since k = 1 probably means overfitting, I decided to stick with what I did for my previous submission:
knn_model2 <- train( Personal.Loan ~ ., data = Training_data, method = "knn")
knn_model2
```

```
## 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
##   5  0.9436374  0.5902912
##   7  0.9412346  0.5544764
##   9  0.9392437  0.5254829
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```r
#knn_model2 returns k = 5 as the best k value.
```

Question 3

```r
#Since k = 5 is the best k, we use it to test on the Validation_Data

Best_k <- train( Personal.Loan ~ ., data = Training_data, method = "knn", tuneGrid = data.frame(k = 5))

Validation_Prediction <- predict(Best_k, Validation_data)
#For my first submission I have used confusionMatrix():
confusionMatrix(Validation_Prediction, as.factor(Validation_data$Personal.Loan), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1799   86
##          1    9  106
##
##                Accuracy : 0.9525
##                  95% CI : (0.9422, 0.9614)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 4.861e-16
##
##                   Kappa : 0.6666
##
```

```
##  Mcnemar's Test P-Value : 6.318e-15
##
##             Sensitivity : 0.5521
##             Specificity : 0.9950
##          Pos Pred Value : 0.9217
##          Neg Pred Value : 0.9544
##              Prevalence : 0.0960
##          Detection Rate : 0.0530
##    Detection Prevalence : 0.0575
##       Balanced Accuracy : 0.7736
##
##        'Positive' Class : 1
##
```

```
#For easier visualization of the confusion matrix, I have now used CrossTable():
CrossTable(x=Validation_Prediction, y=Validation_data$Personal.Loan, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  2000
##
##
##                       | Validation_data$Personal.Loan
## Validation_Prediction |         0 |         1 | Row Total |
## ----------------------|-----------|-----------|-----------|
##                     0 |      1799 |        86 |      1885 |
##                       |     0.954 |     0.046 |     0.942 |
##                       |     0.995 |     0.448 |           |
##                       |     0.899 |     0.043 |           |
## ----------------------|-----------|-----------|-----------|
##                     1 |         9 |       106 |       115 |
##                       |     0.078 |     0.922 |     0.058 |
##                       |     0.005 |     0.552 |           |
##                       |     0.004 |     0.053 |           |
## ----------------------|-----------|-----------|-----------|
##          Column Total |      1808 |       192 |      2000 |
##                       |     0.904 |     0.096 |           |
## ----------------------|-----------|-----------|-----------|
##
##
```

Question 4

```r
#Using predict() we can predict that the Customer would not accept the personal loan offer:
Customer_Prediction_2 <- predict(Best_k, Customer_1_normalized)
Customer_Prediction_2
```

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

Question 5

```r
#Here similar to our Extra credit assignment, I created to data partitionings. 20% of Test Data, and th
Train_Index_2 <- createDataPartition(Universal_Bank_normalized$Personal.Loan, p = 0.8, list = FALSE)

Temporary_data = Universal_Bank_normalized[Train_Index_2, ]
Test_data = Universal_Bank_normalized[-Train_Index_2, ]

Train_Validation <- createDataPartition(Temporary_data$Personal.Loan, p = 0.625, list = FALSE)
Training_data_2 <- Temporary_data[Train_Validation, ]
Validation_data_2 <- Temporary_data[-Train_Validation, ]

#After partitioning the data, I have created a new model, using k = 5, since that is the best k found i

knn_model3 <- train( Personal.Loan ~ ., data = Training_data_2, method = "knn", tuneGrid = data.frame(k
knn_model3
```

```
## k-Nearest Neighbors
##
## 2500 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2500, 2500, 2500, 2500, 2500, 2500, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.939429  0.560938
##
## Tuning parameter 'k' was held constant at a value of 5
```

```r
#Here I added the prediction for the Training Data, which I hadn't done in my previous submission:
Training_data_2_Pred <- predict(knn_model3, Training_data_2)
Training_data_2_Pred
```

```
##    [1] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0
##   [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
##   [75] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [112] 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
##  [149] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
##  [186] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1
##  [223] 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [260] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
```

```
##  [297] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
##  [334] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0
##  [371] 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [408] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [445] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [482] 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
##  [519] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [556] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
##  [593] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [630] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0
##  [667] 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
##  [704] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
##  [741] 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
##  [778] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0
##  [815] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
##  [852] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
##  [889] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
##  [926] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
##  [963] 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1000] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
## [1037] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1074] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0
## [1111] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
## [1148] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1
## [1185] 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
## [1222] 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0
## [1259] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
## [1296] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1333] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1370] 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
## [1407] 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0
## [1444] 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0
## [1481] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
## [1518] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1555] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1592] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0
## [1629] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1666] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
## [1703] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1740] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1777] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1814] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1851] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1888] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1925] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1962] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
## [1999] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [2036] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
## [2073] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
## [2110] 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
## [2147] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
## [2184] 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [2221] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [2258] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
```

```
## [2295] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [2332] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [2369] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
## [2406] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [2443] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [2480] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## Levels: 0 1
```

```r
Validation_data_2_Pred <- predict(knn_model3, Validation_data_2)
Validation_data_2_Pred
```

```
##    [1] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [38] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
##   [75] 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
##  [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
##  [186] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
##  [223] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
##  [260] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
##  [297] 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0
##  [334] 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
##  [371] 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [408] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [445] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [482] 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [519] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [556] 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [593] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [630] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [667] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
##  [704] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [741] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
##  [778] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [815] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0
##  [852] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0
##  [889] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [926] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [963] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
## [1000] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1037] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1074] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
## [1111] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1148] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1185] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
## [1222] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
## [1259] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1296] 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1333] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1370] 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1407] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1444] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
## [1481] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
## Levels: 0 1
```

```r
Test_data_Pred <- predict(knn_model3, Test_data)
Test_data_Pred
```

```
##    [1] 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [75] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [149] 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
##  [186] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0
##  [223] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [260] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [297] 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
##  [334] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
##  [371] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
##  [408] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0
##  [445] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [482] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [519] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0
##  [556] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [593] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [630] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0
##  [667] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
##  [704] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [741] 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [778] 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
##  [815] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [852] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0
##  [889] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
##  [926] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [963] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1000] 0
## Levels: 0 1
```

```r
#As I did for question 3, I have used CrossTable() to create the confusion matrix and for better visual

confusionMatrix(Training_data_2_Pred, as.factor(Training_data_2$Personal.Loan), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2255   93
##          1    5  147
##
##                Accuracy : 0.9608
##                  95% CI : (0.9524, 0.9681)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7299
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
##             Sensitivity : 0.6125
##             Specificity : 0.9978
##          Pos Pred Value : 0.9671
##          Neg Pred Value : 0.9604
##              Prevalence : 0.0960
##          Detection Rate : 0.0588
##    Detection Prevalence : 0.0608
##       Balanced Accuracy : 0.8051
##
##        'Positive' Class : 1
##
```

```
CrossTable(x=Training_data_2_Pred, y=Training_data_2$Personal.Loan, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  2500
##
##
##                      | Training_data_2$Personal.Loan
## Training_data_2_Pred |         0 |         1 | Row Total |
## ---------------------|-----------|-----------|-----------|
##                    0 |      2255 |        93 |      2348 |
##                      |     0.960 |     0.040 |     0.939 |
##                      |     0.998 |     0.388 |           |
##                      |     0.902 |     0.037 |           |
## ---------------------|-----------|-----------|-----------|
##                    1 |         5 |       147 |       152 |
##                      |     0.033 |     0.967 |     0.061 |
##                      |     0.002 |     0.613 |           |
##                      |     0.002 |     0.059 |           |
## ---------------------|-----------|-----------|-----------|
##         Column Total |      2260 |       240 |      2500 |
##                      |     0.904 |     0.096 |           |
## ---------------------|-----------|-----------|-----------|
##
##
```

```
confusionMatrix(Validation_data_2_Pred, as.factor(Validation_data_2$Personal.Loan), positive = "1")
```

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

```
## Prediction    0    1
##          0 1352   66
##          1    4   78
##
##                Accuracy : 0.9533
##                  95% CI : (0.9414, 0.9634)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 7.606e-13
##
##                   Kappa : 0.6671
##
##  Mcnemar's Test P-Value : 3.079e-13
##
##             Sensitivity : 0.54167
##             Specificity : 0.99705
##          Pos Pred Value : 0.95122
##          Neg Pred Value : 0.95346
##              Prevalence : 0.09600
##          Detection Rate : 0.05200
##    Detection Prevalence : 0.05467
##       Balanced Accuracy : 0.76936
##
##        'Positive' Class : 1
##
```

```
CrossTable(x=Validation_data_2_Pred, y=Validation_data_2$Personal.Loan, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1500
##
##
##                        | Validation_data_2$Personal.Loan
## Validation_data_2_Pred |         0 |         1 | Row Total |
## -----------------------|-----------|-----------|-----------|
##                      0 |      1352 |        66 |      1418 |
##                        |     0.953 |     0.047 |     0.945 |
##                        |     0.997 |     0.458 |           |
##                        |     0.901 |     0.044 |           |
## -----------------------|-----------|-----------|-----------|
##                      1 |         4 |        78 |        82 |
##                        |     0.049 |     0.951 |     0.055 |
##                        |     0.003 |     0.542 |           |
##                        |     0.003 |     0.052 |           |
## -----------------------|-----------|-----------|-----------|
```

```
##           Column Total |      1356 |       144 |      1500 |
##                        |     0.904 |     0.096 |           |
## -----------------------|-----------|-----------|-----------|
##
##
```

```r
confusionMatrix(Test_data_Pred, as.factor(Test_data$Personal.Loan), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 900  51
##          1   4  45
##
##                Accuracy : 0.945
##                  95% CI : (0.929, 0.9583)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 1.502e-06
##
##                   Kappa : 0.5944
##
##  Mcnemar's Test P-Value : 5.552e-10
##
##             Sensitivity : 0.4688
##             Specificity : 0.9956
##          Pos Pred Value : 0.9184
##          Neg Pred Value : 0.9464
##              Prevalence : 0.0960
##          Detection Rate : 0.0450
##    Detection Prevalence : 0.0490
##       Balanced Accuracy : 0.7322
##
##        'Positive' Class : 1
##
```

```r
CrossTable(x=Test_data_Pred, y=Test_data$Personal.Loan, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1000
##
##
##               | Test_data$Personal.Loan
```

```
## Test_data_Pred |         0 |         1 | Row Total |
## ---------------|-----------|-----------|-----------|
##              0 |       900 |        51 |       951 |
##                |     0.946 |     0.054 |     0.951 |
##                |     0.996 |     0.531 |           |
##                |     0.900 |     0.051 |           |
## ---------------|-----------|-----------|-----------|
##              1 |         4 |        45 |        49 |
##                |     0.082 |     0.918 |     0.049 |
##                |     0.004 |     0.469 |           |
##                |     0.004 |     0.045 |           |
## ---------------|-----------|-----------|-----------|
##   Column Total |       904 |        96 |      1000 |
##                |     0.904 |     0.096 |           |
## ---------------|-----------|-----------|-----------|
##
##
```

```
#When comparing the 3 confusion matrices, we notice that accuracy goes down from the Training Data whic
```