Assignment_2

Garsego

2025-09-30

```r
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
```

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

View(Universal_Bank)
```

Transforming "Education" into a factor (categorical)

```r
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"

```r
Universal_Bank <- cbind(Universal_Bank[ , !(names(Universal_Bank) %in% "Education")],
                        education_dummy)
View(Universal_Bank)
```

Separating Personal.Loan as the target variable

```r
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

```r
norm_model <- preProcess(Predictors, method = "range")
Predictors_normalized <- predict(norm_model, Predictors)
Universal_Bank_normalized <- cbind(Predictors_normalized, Personal.Loan = Target)

Universal_Bank_normalized$Personal.Loan <- as.factor(Universal_Bank_normalized$Personal.Loan)
View(Universal_Bank_normalized)
```

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

```
set.seed(123)

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

```
knn_model1 <- train( Personal.Loan ~ ., data = Training_data, method = "knn", tuneGrid = data.frame(k =

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)

Customer_1_normalized <- predict(norm_model, Customer_1)
View(Customer_1_normalized)

Customer_1_Prediction <- predict(knn_model1, Customer_1_normalized)
Customer_1_Prediction
```

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

Question 2

```
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  0.9414429  0.5701188
##   7  0.9406703  0.5494197
##   9  0.9393626  0.5281562
## 
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

Question 3

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

Validation_Prediction <- predict(Best_k, Validation_data)
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               
## 
```

Question 4

```r
Customer_Prediction_2 <- predict(Best_k, Customer_1_normalized)
Customer_Prediction_2
```

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

Question 5

```r
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, ]

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
##   0.938123  0.5617227
##
## Tuning parameter 'k' was held constant at a value of 5
```

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

```
##    [1] 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 0 0 0 0 0 0 0 0 0 0 0 0
##   [38] 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
##   [75] 0 1 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 1 0
##  [112] 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 1 0 0 0 1
##  [149] 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 0 0 0 0 0
##  [186] 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
##  [223] 0 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 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
##  [297] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [334] 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 1 0 0 0 0 0 0 0 0
##  [371] 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 1 0 0 0 0 0 0 0 0 0 0 1
##  [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 0 0 0 0 0 0 0 0
##  [445] 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 1 1 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 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 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
##  [593] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [667] 0 0 0 0 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 0 0 0 0 0 0
##  [704] 0 0 0 0 0 0 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
##  [741] 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
##  [778] 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 0
```

```
##  [815] 0 0 0 0 0 0 0 0 0 1 0 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 0
##  [852] 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 1 0 0 0 0 0 0
##  [889] 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 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 1 0 0 0 0 0
##  [963] 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 1 1 0 0 0 0 0
## [1000] 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 0 0 0 0 0 0 0
## [1037] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0
## [1111] 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 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 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 0 0 0 0 0 0
## [1222] 0 0 0 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
## [1259] 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 0
## [1296] 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 0 0 0 0 0 0 0 0
## [1333] 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
## [1370] 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 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 1 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
## [1481] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 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
##   [38] 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 1 0 0 0 0 0 0 0
##   [75] 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 1
##  [112] 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 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 1 1 0 0
##  [186] 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 0 0 0 0 0 0 0 0 0 0
##  [223] 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
##  [260] 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 0 0 0 0 0 0 0 0
##  [297] 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 1 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 1 0 0 0 0 0 0
##  [371] 0 0 0 1 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
##  [408] 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 1 0 0 0 0
##  [445] 0 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
##  [482] 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
##  [519] 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 1 0
##  [556] 0 0 0 0 0 0 0 0 0 0 0 1 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
##  [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 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [704] 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 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 1 0 0 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 1 0 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 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
##  [852] 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 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 1 0 0
##  [926] 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 0 0 0 0 0 0 0
##  [963] 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 1 0 0 0 0 0 0 0 0
## [1000] 0
## Levels: 0 1
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1354   74
##          1    2   70
##
##                Accuracy : 0.9493
##                  95% CI : (0.937, 0.9599)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 6.336e-11
##
##                   Kappa : 0.6241
##
##  Mcnemar's Test P-Value : 3.816e-16
##
##             Sensitivity : 0.48611
##             Specificity : 0.99853
##          Pos Pred Value : 0.97222
##          Neg Pred Value : 0.94818
##              Prevalence : 0.09600
##          Detection Rate : 0.04667
##    Detection Prevalence : 0.04800
##       Balanced Accuracy : 0.74232
##
##        'Positive' Class : 1
##
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 904  35
##          1   0  61
##
##                Accuracy : 0.965
##                  95% CI : (0.9517, 0.9755)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 9.645e-14
##
##                   Kappa : 0.7591
##
##  Mcnemar's Test P-Value : 9.081e-09
##
##             Sensitivity : 0.6354
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9627
```

```
##              Prevalence : 0.0960
##          Detection Rate : 0.0610
##    Detection Prevalence : 0.0610
##       Balanced Accuracy : 0.8177
##
##        'Positive' Class : 1
##
```