Assignment_2

Tarun

2024-02-25

Description: "The object of the assignment is to apply k-NN for classification." #import the needed packages.

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(ISLR)
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
```

```
library(class)
```

#Import data

```
getwd()
```

```
## [1] "C:/Users/tarun/OneDrive/Documents"
```

```
setwd("C:\\Users\\tarun\\Downloads")
customer_data <- read.csv("UniversalBank.csv")
str(customer_data)
```

```
## 'data.frame':    5000 obs. of  14 variables:
##  $ ID                : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP.Code          : int  91107 90089 94720 94112 91330 92121 91711 93943 90089 93023 ...
##  $ Family            : int  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         : int  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal.Loan     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ Securities.Account: int  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : int  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : int  0 0 0 0 1 0 0 1 0 0 ...
```

Initial Research of Customer Data

```r
head(customer_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
```

```r
summary(customer_data)
```

```
##        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   :93153
##  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
```

Looking on values that are missing in Customer Data

```
test_missing <- is.na.data.frame('customer_data')
test_missing
```

```
##       [,1]
## [1,] FALSE
```

Selecting Key Features and Initial Data Inspection

```
library(dplyr)
cleaned_customer_data <- customer_data %>%
  select(Age, Experience, Income, Family, CCAvg, Education, Mortgage, Personal.Loan, Securities.Account

head(cleaned_customer_data)
```

```
##   Age Experience Income Family CCAvg Education Mortgage Personal.Loan
## 1  25          1     49      4   1.6         1        0             0
## 2  45         19     34      3   1.5         1        0             0
## 3  39         15     11      1   1.0         1        0             0
## 4  35          9    100      1   2.7         2        0             0
## 5  35          8     45      4   1.0         2        0             0
## 6  37         13     29      4   0.4         2      155             0
##   Securities.Account CD.Account Online CreditCard
## 1                  1          0      0          0
## 2                  1          0      0          0
## 3                  0          0      0          0
## 4                  0          0      0          0
## 5                  0          0      0          1
## 6                  0          0      1          0
```

Data Type Conversion and Testing

```
cleaned_customer_data$Education <- as.character(cleaned_customer_data$Education)
is_char <- is.character(cleaned_customer_data$Education)

cleaned_customer_data$Personal.Loan <- as.factor(cleaned_customer_data$Personal.Loan)
is_fact <- is.factor(cleaned_customer_data$Personal.Loan)
```

Dummy Encoding of Education Data

```
dummy_encoding <- dummyVars(~Education, data = cleaned_customer_data)
head(predict(dummy_encoding, cleaned_customer_data))
```

```
##   Education1 Education2 Education3
## 1          1          0          0
## 2          1          0          0
## 3          1          0          0
## 4          0          1          0
## 5          0          1          0
## 6          0          1          0
```

```
encoded_customer_data <- predict(dummy_encoding, cleaned_customer_data)
```

Final Customer Data Collection

```
final_customer_data <- cleaned_customer_data[, -6]
final_customer_data <- cbind(final_customer_data, encoded_customer_data)
head(final_customer_data)
```

```
##   Age Experience Income Family CCAvg Mortgage Personal.Loan Securities.Account
## 1  25          1     49      4   1.6        0             0                  1
## 2  45         19     34      3   1.5        0             0                  1
## 3  39         15     11      1   1.0        0             0                  0
## 4  35          9    100      1   2.7        0             0                  0
## 5  35          8     45      4   1.0        0             0                  0
## 6  37         13     29      4   0.4      155             0                  0
##   CD.Account Online CreditCard Education1 Education2 Education3
## 1          0      0          0          1          0          0
## 2          0      0          0          1          0          0
## 3          0      0          0          1          0          0
## 4          0      0          0          0          1          0
## 5          0      0          1          0          1          0
## 6          0      1          0          0          1          0
```

Splitting data for training and validation.

```
set.seed(15)
train_index <- createDataPartition(final_customer_data$Personal.Loan, p = 0.60, list = FALSE)
train_data <- final_customer_data[train_index,]
validation_data <- final_customer_data[-train_index,]
```

Testing Data Collection

```
test_data <- data.frame(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Mortgage = 0, Se
test_data
```

```
##   Age Experience Income Family CCAvg Mortgage Securities.Account CD.Account
## 1  40         10     84      2     2        0                  0          0
##   Online CreditCard Education_1 Education_2 Education_3
## 1      1          1           0           1           0
```

Data Preprocessing and Model Training Summary

```
set.seed(15)
training_preprocessed <- preProcess(train_data[, -c(7, 12:14)], method = c("center", "scale"))
model_train <- predict(training_preprocessed, train_data)
model_validate <- predict(training_preprocessed, validation_data)
model_test <- predict(training_preprocessed, test_data)
summary(model_train)
```

```
##       Age            Experience            Income            Family
##  Min.   :-1.9325   Min.   :-1.997167   Min.   :-1.4435   Min.   :-1.2237
##  1st Qu.:-0.8857   1st Qu.:-0.864443   1st Qu.:-0.7619   1st Qu.:-1.2237
##  Median :-0.0134   Median : 0.006883   Median :-0.2341   Median :-0.3482
##  Mean   : 0.0000   Mean   : 0.000000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.8589   3rd Qu.: 0.878210   3rd Qu.: 0.5355   3rd Qu.: 0.5273
##  Max.   : 1.9057   Max.   : 2.010934   Max.   : 3.3061   Max.   : 1.4028
##      CCAvg            Mortgage       Personal.Loan Securities.Account
##  Min.   :-1.1014   Min.   :-0.5591   0:2712        Min.   :-0.3388
##  1st Qu.:-0.7024   1st Qu.:-0.5591   1: 288        1st Qu.:-0.3388
##  Median :-0.2465   Median :-0.5591                 Median :-0.3388
##  Mean   : 0.0000   Mean   : 0.0000                 Mean   : 0.0000
##  3rd Qu.: 0.3234   3rd Qu.: 0.4322                 3rd Qu.:-0.3388
##  Max.   : 4.5978   Max.   : 5.6581                 Max.   : 2.9506
##    CD.Account          Online           CreditCard        Education1
##  Min.   :-0.2404   Min.   :-1.1928   Min.   :-0.640   Min.   :0.0000
##  1st Qu.:-0.2404   1st Qu.:-1.1928   1st Qu.:-0.640   1st Qu.:0.0000
##  Median :-0.2404   Median : 0.8381   Median :-0.640   Median :0.0000
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000   Mean   :0.4163
##  3rd Qu.:-0.2404   3rd Qu.: 0.8381   3rd Qu.: 1.562   3rd Qu.:1.0000
##  Max.   : 4.1578   Max.   : 0.8381   Max.   : 1.562   Max.   :1.0000
##    Education2       Education3
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000
##  Mean   :0.2873   Mean   :0.2963
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
```

K-Nearest Neighbors (KNN) Model Prediction

```
set.seed(15)
train_predictors <- model_train[, -7]
validate_predictors <- model_validate[, -7]

train_label <- model_train[, 7]
validate_label <- model_validate[, 7]

knn_model <- knn(train_predictors, model_test, cl = train_label, k = 1)
knn_model
```

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

KNN Model Tuning and Selection of Best K

```
set.seed(15)
search_grid <- expand.grid(k = c(1:40))
tr_control <-
model <- train(Personal.Loan ~ ., data = model_train, tuneGrid = search_grid, method = "knn", trControl
model
```

```
## 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, 2699, 2700, 2701, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9560044  0.7216689
##    2  0.9503321  0.6788967
##    3  0.9536710  0.6849720
##    4  0.9493355  0.6451288
##    5  0.9516699  0.6570416
##    6  0.9503332  0.6430257
##    7  0.9480010  0.6201422
##    8  0.9456688  0.6019128
##    9  0.9440021  0.5802892
##   10  0.9426688  0.5679149
##   11  0.9423332  0.5632448
##   12  0.9409999  0.5491248
##   13  0.9393354  0.5337963
##   14  0.9373343  0.5149496
##   15  0.9390010  0.5280560
##   16  0.9380010  0.5135715
##   17  0.9383343  0.5135055
##   18  0.9366677  0.4965140
##   19  0.9359999  0.4844312
##   20  0.9353343  0.4785326
##   21  0.9340021  0.4619035
##   22  0.9350032  0.4757680
##   23  0.9340032  0.4641535
##   24  0.9333354  0.4625352
##   25  0.9330010  0.4556589
##   26  0.9320032  0.4397723
##   27  0.9316699  0.4408149
##   28  0.9323354  0.4462986
##   29  0.9313354  0.4342050
##   30  0.9296676  0.4184426
##   31  0.9303332  0.4254060
##   32  0.9316687  0.4357434
##   33  0.9310043  0.4278118
##   34  0.9303365  0.4188675
##   35  0.9313376  0.4304408
##   36  0.9293343  0.4113532
##   37  0.9289998  0.4108341
##   38  0.9283332  0.4024932
##   39  0.9276665  0.3940111
##   40  0.9276687  0.3929311
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```r
best_k <- model$bestTune[[1]]
```

Model Validation and Confusion Matrix

```r
set.seed(15)
model_validate <- knn(train_predictors, validate_predictors, cl = train_label, k = best_k)

confusionMatrix(model_validate, validate_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1767   69
##          1   41  123
##
##                Accuracy : 0.945
##                  95% CI : (0.9341, 0.9546)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 1.359e-11
##
##                   Kappa : 0.661
##
##  Mcnemar's Test P-Value : 0.01004
##
##             Sensitivity : 0.9773
##             Specificity : 0.6406
##          Pos Pred Value : 0.9624
##          Neg Pred Value : 0.7500
##              Prevalence : 0.9040
##          Detection Rate : 0.8835
##    Detection Prevalence : 0.9180
##       Balanced Accuracy : 0.8090
##
##        'Positive' Class : 0
##
```

Data division in training, validation, and testing

```r
set.seed(15)
train_data_partition <- createDataPartition(final_customer_data$Personal.Loan, p = 0.5, list = FALSE)
train_customer_data <- final_customer_data[train_data_partition,]
test_customer_data <- final_customer_data[-train_data_partition,]

customer_data_validate <- createDataPartition(test_customer_data$Personal.Loan, p = 0.6, list = FALSE)
validate_customer_data <- test_customer_data[customer_data_validate,]
test_customer_data <- test_customer_data[-customer_data_validate,]
```

Data Normalization

```r
set.seed(15)
normalized_customer_data <- preProcess(train_customer_data[, -c(7, 12:14)], method = c("center", "scale

train_data_normalized <- predict(normalized_customer_data, train_customer_data)
validate_data_normalized <- predict(normalized_customer_data, validate_customer_data)
test_data_normalized <- predict(normalized_customer_data, test_customer_data)
```

Separating Predictors and Labels for Training, Validation, and Testing

```r
set.seed(15)
train_predictor <- train_data_normalized[, -7]
validate_predictor <- validate_data_normalized[, -7]
test_predictor <- test_data_normalized[, -7]

train_label <- train_data_normalized[, 7]
validate_label <- validate_data_normalized[, 7]
test_label <- test_data_normalized[, 7]
```

KNN Model Training

```r
set.seed(15)
train_model <- knn(train_predictor, train_predictor, cl = train_label, k = best_k)
head(train_model)
```

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

KNN Model Validation

```r
set.seed(15)
validation_model <- knn(train_predictor, validate_predictor, cl = train_label, k = best_k)
head(validation_model)
```

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

KNN Model Testing

```r
set.seed(15)
test_model <- knn(train_predictor, test_predictor, cl = train_label, k = best_k)
head(test_model)
```

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

Confusion Matrix for Training Model

```r
set.seed(15)
confusionMatrix(train_model, train_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2260    0
##          1    0  240
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.904
##          Detection Rate : 0.904
##    Detection Prevalence : 0.904
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 0
##
```

```r
# Number of miscalculations = 0. Accuracy is 100% for training model.
```

Confusion Matrix for Validation Model

```r
set.seed(15)
confusionMatrix(validation_model, validate_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1335   51
##          1   21   93
##
##                Accuracy : 0.952
##                  95% CI : (0.9399, 0.9623)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 3.516e-12
```

```
##
##                   Kappa : 0.6951
##
##  Mcnemar's Test P-Value : 0.0006316
##
##             Sensitivity : 0.9845
##             Specificity : 0.6458
##          Pos Pred Value : 0.9632
##          Neg Pred Value : 0.8158
##              Prevalence : 0.9040
##          Detection Rate : 0.8900
##    Detection Prevalence : 0.9240
##       Balanced Accuracy : 0.8152
##
##        'Positive' Class : 0
##
```

```
# Number of miscalculations = 68. Accuracy is 95% for validation model.
```

Confusion Matrix for Test Model

```
set.seed(15)
confusionMatrix(test_model, test_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 890  25
##          1  14  71
##
##                Accuracy : 0.961
##                  95% CI : (0.9471, 0.9721)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 5.695e-12
##
##                   Kappa : 0.7632
##
##  Mcnemar's Test P-Value : 0.1093
##
##             Sensitivity : 0.9845
##             Specificity : 0.7396
##          Pos Pred Value : 0.9727
##          Neg Pred Value : 0.8353
##              Prevalence : 0.9040
##          Detection Rate : 0.8900
##    Detection Prevalence : 0.9150
##       Balanced Accuracy : 0.8620
##
##        'Positive' Class : 0
##
```

```
# Number of miscalculations = 36. Accuracy is 96% for Test Model.
```

#conclusion:the training data results are more accurate and sensitive. #The matrices provided were applied for calculating the results for the Test, Training, and Validation sets, which are 96%,100% and 95%, respectively. #The model performs well in all sets, with the training set showing the best accuracy. It means that the model has properly learned from the training data and is able to apply well to unseen data.The K-Nearest Neighbors (KNN) algorithm reliably predicts customer behavior in this dataset, making it valuable for informing business decisions.