Assignment_2

Olayinka Sikiru

2024-02-18

The primary goal of this task is to leverage the k-Nearest Neighbors (k-NN) algorithm to predict the likelihood of liability customers (depositors) at Universal Bank accepting personal loan offers. This predictive analysis aims to support the bank's retail marketing department in crafting more effective and targeted marketing campaigns to enhance the conversion rate of depositors into loan customers, building on the success of a previous campaign that achieved a 9% conversion rate.

```
# To import Dataset
UniversalBank <- read.csv("C:/Users/DELL/Dataset/UniversalBank.csv")

#Summary of the table
summary(UniversalBank)
```

```
##        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
```

```
#Structure of the dataset 'UniversalBank'
str(UniversalBank)
```

```
## '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 ...
```

Solution 1

```
# Convert Education to a factor
UniversalBank$Education = as.factor(UniversalBank$Education)

# Remove ID and ZIP Code from the dataset and transform Education into dummy variables
UniversalBank_1 <- select(UniversalBank, -c(ZIP.Code, ID))

# Use fastDummies to create dummy variables for Education
UniversalBank_dummy <- fastDummies::dummy_cols(UniversalBank_1, select_columns = "Education")

# Display the first few rows of UniversalBank_dummy
head(UniversalBank_dummy)
```

```
##   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 Education_1 Education_2
## 1                  1          0      0          0           1           0
## 2                  1          0      0          0           1           0
## 3                  0          0      0          0           1           0
## 4                  0          0      0          0           0           1
## 5                  0          0      0          1           0           1
## 6                  0          0      1          0           0           1
##   Education_3
## 1           0
## 2           0
## 3           0
## 4           0
## 5           0
## 6           0
```

```
#Converting Personal.Loan to a factor present in the dataset 'UniversalBank_dummy'
UniversalBank_dummy$Personal.Loan = as.factor(UniversalBank_dummy$Personal.Loan)

# Set the seed for reproducibility
set.seed(123)

# Divide the dataset into training and validation sets
train.index <- sample(row.names(UniversalBank_dummy), 0.6 * nrow(UniversalBank_dummy))
train_data <- UniversalBank_dummy[train.index, ]
valid_data <- UniversalBank_dummy[-as.numeric(train.index), ]  # Convert to numeric to ensure pr
oper indexing

# Define details of the given customer
CST_Details <- data.frame(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Educati
on_1 = 0, Education_2 = 1, Education_3 = 0, Mortgage = 0, Securities.Account = 0, CD.Account =
0, Online = 1, CreditCard = 1)

CST_Details
```

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

```
# Normalize numerical features (excluding target and dummy variables)
normalize_data <- setdiff(names(train_data), c("Personal.Loan", "Education_1", "Education_2", "E
ducation_3"))
norm_model <- preProcess(train_data[normalize_data], method = c("center", "scale"))
train_data_normalized <- cbind(predict(norm_model, train_data[normalize_data]), train_data[c("Pe
rsonal.Loan", "Education_1", "Education_2", "Education_3")])
valid_data_normalized <- cbind(predict(norm_model, valid_data[normalize_data]), valid_data[c("Pe
rsonal.Loan", "Education_1", "Education_2", "Education_3")])

# Define the columns to normalize based on the numerical features in CST_Details
normalize_data <- setdiff(names(CST_Details), c("Education_1", "Education_2", "Education_3"))

# Now normalize using the defined columns
new.df_normalized <- cbind(predict(norm_model, CST_Details[normalize_data]), CST_Details[c("Educ
ation_1", "Education_2", "Education_3")])

# Identify missing columns
missing_columns <- setdiff(names(train_data_normalized), names(new.df_normalized))

# Insert missing columns into new.df_normalized with default values
for(col in missing_columns) {
  new.df_normalized[[col]] <- 0  # Or use another default value as appropriate
}

# Ensure column order in new.df_normalized matches train_data_normalized
new.df_normalized <- new.df_normalized[names(train_data_normalized)]

# Remove the target variable column from new.df_normalized
new.df_normalized <- new.df_normalized[, -which(names(new.df_normalized) == "Personal.Loan")]

# Run k-NN classification
knn_result <- knn(train = train_data_normalized[, -which(names(train_data_normalized) == "Person
al.Loan")], test = new.df_normalized, cl = train_data_normalized$Personal.Loan, k = 1, prob = TR
UE)

# Display k-NN classification result
knn_result
```

```
## [1] 0
## attr(,"prob")
## [1] 1
## Levels: 0 1
```

Solution 2

```
# To Set seed for reproducibility of random operations
set.seed(123)

# To define the sequence of odd k values from 1 to 20 for the k-NN algorithm
k_values <- seq(1, 20, 2)

# To calculate accuracy for each k value using the validation set
accuracy <- sapply(k_values, function(k){
  predicted <- knn(train = train_data[, -which(names(train_data) == "Personal.Loan")],
                   test = valid_data[, -which(names(valid_data) == "Personal.Loan")],
                   cl = train_data$Personal.Loan, k=k)

# To calculate and return the mean accuracy for this k
  mean(predicted == valid_data$Personal.Loan)
})

best_k <- k_values[which.max(accuracy)]
print(best_k)
```

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

Solution 3

```
# Perform k-NN classification using the best k value on the validation data
predicted <- knn(train = train_data[, -which(names(train_data) == "Personal.Loan")],
                 test = valid_data[, -which(names(valid_data) == "Personal.Loan")],
                 cl = train_data$Personal.Loan, k = best_k)

# Create a confusion matrix
conf_matrix <- confusionMatrix(table(predicted, valid_data$Personal.Loan))

# Print the confusion matrix
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##
## predicted    0    1
##         0 1746  122
##         1   52   80
##
##                Accuracy : 0.913
##                  95% CI : (0.8998, 0.925)
##     No Information Rate : 0.899
##     P-Value [Acc > NIR] : 0.01903
##
##                   Kappa : 0.4338
##
##  Mcnemar's Test P-Value : 1.687e-07
##
##             Sensitivity : 0.9711
##             Specificity : 0.3960
##          Pos Pred Value : 0.9347
##          Neg Pred Value : 0.6061
##              Prevalence : 0.8990
##          Detection Rate : 0.8730
##    Detection Prevalence : 0.9340
##       Balanced Accuracy : 0.6836
##
##        'Positive' Class : 0
##
```

Solution 4

```
CST_Details2 <- data.frame(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Educat
ion_1 = 0, Education_2 = 1, Education_3 = 0, Mortgage = 0, Securities.Account = 0, CD.Account =
0, Online = 1, CreditCard = 1)

CST_Details2_normalized <- predict(norm_model, CST_Details2)

# Identify missing columns
missing_columns <- setdiff(names(train_data_normalized), names(CST_Details2_normalized))

# Add missing columns to CST_Details2_normalized with default values
for (col in missing_columns) {

# To Assume numerical features; adjust as necessary
CST_Details2_normalized[[col]] <- 0}

# To ensure the order of columns in CST_Details2_normalized matches train_data_normalized
CST_Details2_normalized <-CST_Details2_normalized[names(train_data_normalized)[names(train_data_
normalized) != "Personal.Loan"]]

predicted_class <- knn(train = train_data_normalized[, -which(names(train_data_normalized) == "P
ersonal.Loan")], test = CST_Details2_normalized, cl = train_data_normalized$Personal.Loan, k = b
est_k)

# Display the predicted class for CST_Details2
predicted_class
```

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

Solution 5

```
set.seed(123)  # Ensure reproducibility

# Calculate indices for the splits
train_indices <- sample(1:nrow(UniversalBank_dummy), 0.5 * nrow(UniversalBank_dummy))
remaining_indices <- setdiff(1:nrow(UniversalBank_dummy), train_indices)
validation_indices <- sample(remaining_indices, 0.6 * length(remaining_indices))
test_indices <- setdiff(remaining_indices, validation_indices)

# Create the new data partitions
train_set <- UniversalBank_dummy[train_indices, ]
validation_set <- UniversalBank_dummy[validation_indices, ]
test_set <- UniversalBank_dummy[test_indices, ]

norm_model_new <- preProcess(train_set[, -which(names(train_set) == "Personal.Loan")], method =
c("center", "scale"))
train_set_normalized <- predict(norm_model_new, train_set)
validation_set_normalized <- predict(norm_model_new, validation_set)
test_set_normalized <- predict(norm_model_new, test_set)

# Apply k-NN to the validation set
predicted_validation <- knn(train = train_set_normalized[, -which(names(train_set_normalized) ==
"Personal.Loan")], test = validation_set_normalized[, -which(names(validation_set_normalized) ==
"Personal.Loan")], cl = train_set$Personal.Loan, k = best_k)

# Apply k-NN to the train set
predicted_train <- knn(train = train_set_normalized[, -which(names(train_set_normalized) == "Per
sonal.Loan")], test = train_set_normalized[, -which(names(train_set_normalized) == "Personal.Loa
n")], cl = train_set$Personal.Loan, k = best_k)

# Apply k-NN to the test set
predicted_test <- knn(train = train_set_normalized[, -which(names(train_set_normalized) == "Pers
onal.Loan")],
test = test_set_normalized[, -which(names(test_set_normalized) == "Personal.Loan")], cl = train_
set$Personal.Loan, k = best_k)

# For the validation set
conf_matrix_validation <- confusionMatrix(table(Predicted = predicted_validation, Actual = valid
ation_set$Personal.Loan))
print(conf_matrix_validation)
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted    0    1
##         0 1351   61
##         1    6   82
##
##                Accuracy : 0.9553
##                  95% CI : (0.9436, 0.9652)
##     No Information Rate : 0.9047
##     P-Value [Acc > NIR] : 1.242e-13
##
##                   Kappa : 0.6872
##
##  Mcnemar's Test P-Value : 4.191e-11
##
##             Sensitivity : 0.9956
##             Specificity : 0.5734
##          Pos Pred Value : 0.9568
##          Neg Pred Value : 0.9318
##              Prevalence : 0.9047
##          Detection Rate : 0.9007
##    Detection Prevalence : 0.9413
##       Balanced Accuracy : 0.7845
##
##        'Positive' Class : 0
##
```

```
# For the train set
conf_matrix_train <- confusionMatrix(table(Predicted = predicted_train, Actual = train_set$Perso
nal.Loan))
print(conf_matrix_train)
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted    0    1
##         0 2266   77
##         1    5  152
##
##                Accuracy : 0.9672
##                  95% CI : (0.9594, 0.9738)
##     No Information Rate : 0.9084
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7705
##
##  Mcnemar's Test P-Value : 4.483e-15
##
##             Sensitivity : 0.9978
##             Specificity : 0.6638
##          Pos Pred Value : 0.9671
##          Neg Pred Value : 0.9682
##              Prevalence : 0.9084
##          Detection Rate : 0.9064
##    Detection Prevalence : 0.9372
##       Balanced Accuracy : 0.8308
##
##        'Positive' Class : 0
##
```

```
# For the test set
conf_matrix_test <- confusionMatrix(table(Predicted = predicted_test, Actual = test_set$Persona
l.Loan))
print(conf_matrix_test)
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted   0   1
##         0 888  43
##         1   4  65
##
##                Accuracy : 0.953
##                  95% CI : (0.938, 0.9653)
##     No Information Rate : 0.892
##     P-Value [Acc > NIR] : 3.996e-12
##
##                   Kappa : 0.71
##
##  Mcnemar's Test P-Value : 2.976e-08
##
##             Sensitivity : 0.9955
##             Specificity : 0.6019
##          Pos Pred Value : 0.9538
##          Neg Pred Value : 0.9420
##              Prevalence : 0.8920
##          Detection Rate : 0.8880
##    Detection Prevalence : 0.9310
##       Balanced Accuracy : 0.7987
##
##        'Positive' Class : 0
##
```