FML_Assignment 2

VaishnaviD

2024-02-16

Problem Statement -

Universal bank is a young bank growing rapidly in terms of overall customer acquisition. The majority of these customers are liability customers (depositors) with varying sizes of relationship with the bank. The customer base of asset customers (borrowers) is quite small, and the bank is interested in expanding this base rapidly in more loan business. In particular, it wants to explore ways of convering its liability customers to personal loan customers.
A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. This has encouraged the retail marketing department to devise smarter campaigns with beΣer target marketing. The goal is to use k-NN to predict whether a new customer will accept a loan offer. This will serve as the basis for the design of a new campaign.
The file UniversalBank.csv contains data on 5000 customers. The data include customer demographic information (age, income, etc.), the customer's relationship with the bank (mortgage, securities account, etc.), and the customer response to the last personal loan campaign (Personal Loan). Among these 5000 customers, only 480 (= 9.6%) accepted the personal loan that was offered to them in the earlier campaign.
Partition the data into training (60%) and validation (40%) sets.

To load required libraries

```
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(ggplot2)
library(lattice)
library(class)
library(caret)
library(e1071)
library(knitr)
```

To specify the file location and showing dimensions

```
universal.bank <- read.csv("C:\\Users\\Vaishnavi\\OneDrive - Kent State University\\FML\\Assignment 2\\UniversalB
ank.csv")
dim(universal.bank)
```

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

To transponse the data frame

The t function helps to transpose the data frame

```
t(t(names(universal.bank)))
```

```
##       [,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"
```

To do Data Preprocessing - to drop ID and ZIP.Code

```
universal.bank <- universal.bank[,-c(1,5)]
```

To transform categorical variables into dummy variables

Only Education is to be converted to factor

```
universal.bank$Education <- as.factor(universal.bank$Education)
```

To convert Education to Dummy Variables

```
groups <- dummyVars(~., data = universal.bank)
universal_m <- as.data.frame(predict(groups,universal.bank))
```

To split the data into training (60%) and validation (40%) sets

To make sure that we get the same sample if we rerun the code

```
set.seed(1)
train.index <- sample(row.names(universal_m), 0.6*dim(universal_m)[1])
valid.index <- setdiff(row.names(universal_m), train.index)
train.df <- universal_m[train.index,]
valid.df <- universal_m[valid.index,]
t(t(names(train.df)))
```

```
##       [,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"
```

To normalize the data

```
train.norm.df <- train.df[,-10]
valid.norm.df <- valid.df[,-10]
norm.values <- preProcess(train.df[, -10], method=c("center", "scale"))
train.norm.df <- predict(norm.values, train.df[, -10])
valid.norm.df <- predict(norm.values, valid.df[, -10])
```

Question 1 - 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 Credit Card = 1. Perform a k-NN classification with all predictors except ID and ZIP code using k = 1. Remember to transform categorical predictors with more than two categories into dummy variables first. Specify the success class as 1 (loan acceptance), and use the default cutoff value of 0.5. How would this customer be classified?

We have converted all categorical variables to dummy variables
Let's create a new sample

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

Normalizing the new customer

```
new.cust.norm <- new_customer
new.cust.norm <- predict(norm.values, new.cust.norm)
```

To predict using K-NN(k- nearest neighbors)

```
knn_pred <- class::knn(train = train.norm.df, test = new.cust.norm, cl = train.df$Personal.Loan, k = 1)
knn_pred
```

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

Question 2 - What is a choice of k that balances between overfitting and ignoring the predictor information?

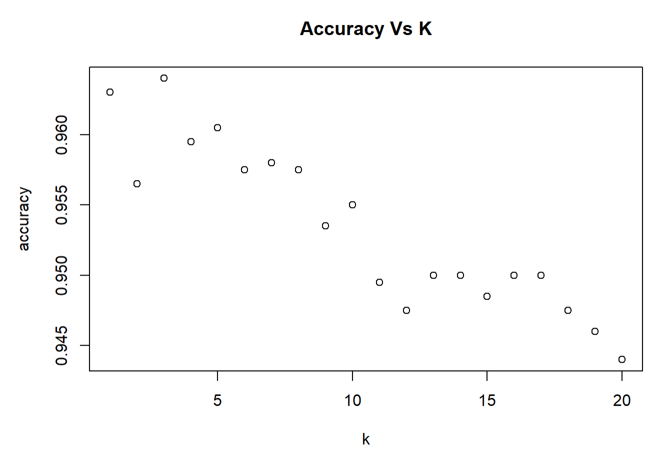
To calculate the accuracy for each value of k
To set the range of k values to consider

```
accuracy.df <- data.frame(k = seq(1, 20, 1), overallaccuracy = rep(0, 20))
for(i in 1:20)
{knn.pred <- class::knn(train = train.norm.df,
test = valid.norm.df,
cl = train.df$Personal.Loan, k = i)
accuracy.df[i, 2] <- confusionMatrix(knn.pred,as.factor(valid.df$Personal.Loan),positive = "1")$overall[1]
}
which(accuracy.df[,2] == max(accuracy.df[,2]))
```

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

To plot chart Accuracy vs K

```
plot(accuracy.df$k,accuracy.df$overallaccuracy, main = "Accuracy Vs K", xlab = "k", ylab = "accuracy")
```



Question 3 - Show the confusion matrix for the validation data that results from using the best k.

Confusion Matrix using best K=3

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

Question 4 - Consider 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 Credit Card = 1. Classify the customer using the best k.

To load new customer profile

```
new_customer2<-data.frame( Age = 40, Experience = 10, Income = 84, family =2, CCAvg = 2, Education_1 = 0, Educati
on_2 = 1, Education_3 = 0, Mortgage = 0, Securities.Account = 0, CDAccount = 0, Online = 1, CreditCard = 1)
```

To do k-NN Prediction

```
knn_pred <- class::knn(train = train.norm.df, test = new.cust.norm, cl = train.df$Personal.Loan, k = 3)
knn_pred
```

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

To print the predicted class (1 for loan acceptance, 0 for loan rejection)

```
print("This customer is classified as Loan Rejected")
```

```
## [1] "This customer is classified as Loan Rejected"
```

Question 5 - Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%). Apply the k-NN method with the k chosen above. Compare the confusion matrix of the test set with that of the training and validation sets. Comment on the differences and their reason.

Split the data to 50% training and 30% Validation and 20% Testing

```
set.seed(1)
Train_Index1 <- sample(row.names(universal_m), 0.5*dim(universal_m)[1])
Val_Index1 <- sample(setdiff(row.names(universal_m),Train_Index1),0.3*dim(universal_m)[1])
Test_Index1 <-setdiff(row.names(universal_m),union(Train_Index1,Val_Index1))
Train_Data <- universal_m[Train_Index1,]
Validation_Data <- universal_m[Val_Index1,]
Test_Data <- universal_m[Test_Index1,]
```

To normalize the data

```
train.norm.df1 <- Train_Data[,-10]
valid.norm.df1 <- Validation_Data[,-10]
Test.norm.df1 <-Test_Data[,-10]
norm.values1 <- preProcess(Train_Data[, -10], method=c("center", "scale"))
train.norm.df1 <- predict(norm.values1, Train_Data[,-10])
valid.norm.df1 <- predict(norm.values1, Validation_Data[,-10])
Test.norm.df1 <-predict(norm.values1,Test_Data[,-10])
```

To predict using K-NN(k- Nearest neighbors)

```
validation_knn = class::knn(train = train.norm.df1,
test = valid.norm.df1,
cl = Train_Data$Personal.Loan,
k = 3)
test_knn = class::knn(train = train.norm.df1,
test = Test.norm.df1,
cl = Train_Data$Personal.Loan,
k = 3)
Train_knn = class::knn(train = train.norm.df1,
test = train.norm.df1,
cl = Train_Data$Personal.Loan,
k = 3)
```

Now performing Validation confusion Matrix

```
validation_confusion_matrix = confusionMatrix(validation_knn,
as.factor(Validation_Data$Personal.Loan),
positive = "1")
validation_confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1358   42
##          1    6   94
##
##                Accuracy : 0.968
##                  95% CI : (0.9578, 0.9763)
##     No Information Rate : 0.9093
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7797
##
##  Mcnemar's Test P-Value : 4.376e-07
##
##             Sensitivity : 0.69118
##             Specificity : 0.99560
##          Pos Pred Value : 0.94000
##          Neg Pred Value : 0.97000
##              Prevalence : 0.09067
##          Detection Rate : 0.06267
##    Detection Prevalence : 0.06667
##       Balanced Accuracy : 0.84339
##
##        'Positive' Class : 1
##
```

Now performing Test confusion Matrix

```
test_confusion_matrix = confusionMatrix(test_knn, as.factor(Test_Data$Personal.Loan), positive = "1")
test_confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 884  35
##          1   4  77
##
##                Accuracy : 0.961
##                  95% CI : (0.9471, 0.9721)
##     No Information Rate : 0.888
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.777
##
##  Mcnemar's Test P-Value : 1.556e-06
##
##             Sensitivity : 0.6875
##             Specificity : 0.9955
##          Pos Pred Value : 0.9506
##          Neg Pred Value : 0.9619
##              Prevalence : 0.1120
##          Detection Rate : 0.0770
##    Detection Prevalence : 0.0810
##       Balanced Accuracy : 0.8415
##
##        'Positive' Class : 1
##
```

Now performing training confusion Matrix

```
Training_confusion_matrix = confusionMatrix(Train_knn, as.factor(Train_Data$Personal.Loan), positive = "1")
Training_confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2263   54
##          1    5  178
##
##                Accuracy : 0.9764
##                  95% CI : (0.9697, 0.982)
##     No Information Rate : 0.9072
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8452
##
##  Mcnemar's Test P-Value : 4.129e-10
##
##             Sensitivity : 0.7672
##             Specificity : 0.9978
##          Pos Pred Value : 0.9727
##          Neg Pred Value : 0.9767
##              Prevalence : 0.0928
##          Detection Rate : 0.0712
##    Detection Prevalence : 0.0732
##       Balanced Accuracy : 0.8825
##
##        'Positive' Class : 1
##
```

Test vs.Train:

Accuracy: Train has a higher accuracy compared to Test.
Reason : This is due to differences in the datasets used for evaluation.

Train vs.Validation:

Accuracy: Train has a higher accuracy compared to Validation.
Reason: this is due to train can have a more balanced or easy to predict dataset.

Reasons for Differences: This can be due to randomness, sample & model variability, Data set difference, etc