RAJESH FML ASSIGNMENT @2

2023-10-01

Assignment 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 converting 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 better 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.

##Load the libraries which are required

```
library(class)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(e1071)
library(knitr)
```

Read the data from UniversalBank.csv dataset

```
universalbank.df <- read.csv("C:\\Users\\yadla sreebhavya\\Downloads\\UniversalBank (2).csv")
dim(universalbank.df)
```

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

```
t(t(names(universalbank.df)))
```

```
##       [,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"
```

```
# The 't' function used to transpose of the  given data frame
```

ZIP and ID

```
universalbank.df <- universalbank.df[,-c(1,5)]
```

##Convert categorical variables into dummy variables

```
#  Education needs to be converted to factor

universalbank.df$Education <- as.factor(universalbank.df$Education)
```

```
# Now, convert Education into Dummy Variables

groups <- dummyVars(~., data = universalbank.df) # This creates the dummy groups

universal_m.df <- as.data.frame(predict(groups,universalbank.df))
```

The data will be seperated into 60% training and 40 % Validation

```
set.seed(1)  # Important to ensure that we get the same sample if we rerun the code
train.index <- sample(row.names(universal_m.df), 0.6*dim(universal_m.df)[1])
valid.index <- setdiff(row.names(universal_m.df), train.index)
train.df <- universal_m.df[train.index,]
valid.df <- universal_m.df[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"
```

##Normalize the data

```
train.norm.df <- train.df[,-10] # Personal Income is the 10th variable
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

```
new_customer <- data.frame(
  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,
  CreditCard = 1)
```

##Normalize the new customer

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

##Now predict using K-NN

```
knn.pred1 <- class::knn(train = train.norm.df,
                       test = new.cust.norm,
                       cl = train.df$Personal.Loan, k = 1)
knn.pred1
```

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

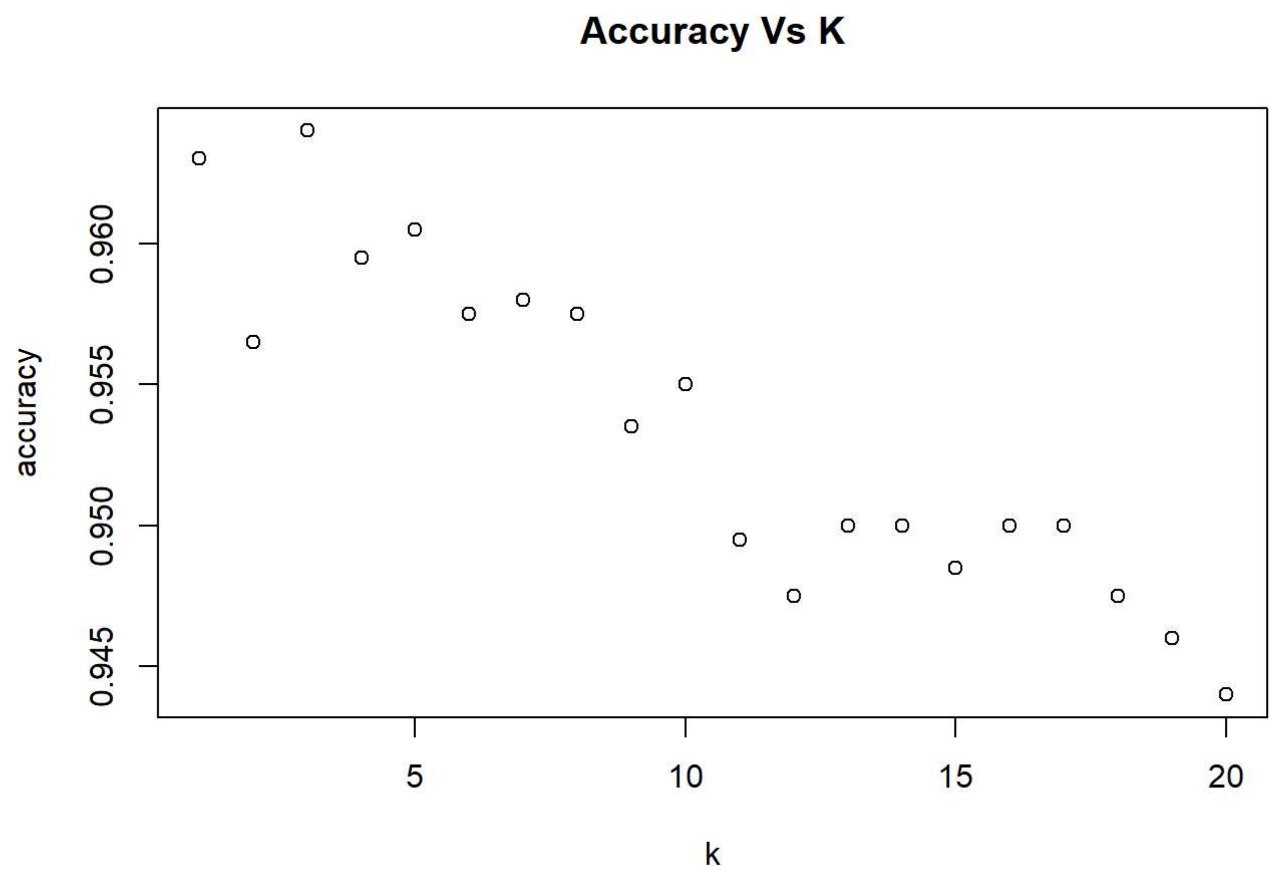
Question 2

What is a choice of k that balances between over fitting and ignoring the predictor information? Calculate the accuracy for k

```
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
```

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



Question 3

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

##Perform Confusion Matrix for 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.

#New customer profile

```
new_customer2<-data.frame(
  Age = 40,
  Experience = 10,
  Income = 84,
  family =2,
  CCAvg = 2,
  Education_1 = 0,
  Education_2 = 1,
  Education_3 = 0,
  Mortgage = 0,
  Securities.Account = 0,
  CDAccount = 0,
  Online = 1,
  CreditCard = 1)
```

```
knn.pred1 <- class::knn(train = train.norm.df,
                       test = new.cust.norm,
                       cl = train.df$Personal.Loan, k = 3)
knn.pred1
```

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

Print the predicted class (1 for Approval , 0 for Rejection)

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

```
## [1] "This customer is classified as: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.

##Seperate the data to 50% training and 30% Validation and 20% Testing

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

##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])
```

##Predict using k- Nearest neighbors

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

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

Train = class::knn(train = train.norm.df1,
                    test = train.norm.df1,
                    cl = Train_Data$Personal.Loan,
                    k = 3)
```

##Validate confusion Matrix

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

validation_conf_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
##
```

Test confusion Matrix

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


test_conf_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
##
```

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

Training_conf_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
##
```

Differences in Model Evaluation Metrics

Training vs. Testing:

Accuracy: The training dataset exhibits a higher accuracy score (0.9764) compared to the test dataset (0.961).

Explanation: This difference in accuracy can be attributed to variations in the datasets used for evaluation. The training data might be more balanced or contain patterns that are easier for the model to learn.

Sensitivity (True Positive Rate): The training dataset shows higher sensitivity (0.7672) than the test dataset (0.6875).

Explanation: This indicates that the model trained on the training data is better at correctly identifying positive cases, such as loan approvals. It likely has a lower rate of false negatives.

Specificity (True Negative Rate): The training dataset exhibits higher specificity (0.9978) compared to the test dataset (0.9955).

Explanation: This suggests that the model trained on the training data is better at correctly identifying negative cases, such as loan rejections. It may have a lower rate of false positives.

Positive Predictive Value (Precision): The training dataset shows a higher positive predictive value (0.9727) compared to the test dataset (0.9506).

Explanation: The model trained on the training data is more precise in predicting positive cases, resulting in fewer false positive predictions.

Training vs. Validation:

Accuracy:The training dataset still has a higher accuracy score (0.968) compared to the validation dataset (0.961).

Explanation: Similar to the comparison with the test dataset, the training data may have characteristics that make it easier for the model to achieve high accuracy.

Sensitivity (True Positive Rate): The training dataset exhibits higher sensitivity (0.69118) compared to the validation dataset (0.6875).

Explanation:The model trained on the training data is better at correctly identifying positive cases. This indicates that the validation dataset's model may have a higher rate of false negatives.

Specificity (True Negative Rate): The training dataset still shows higher specificity (0.99560) compared to the validation dataset (0.9955).

Explanation: The model trained on the training data is better at correctly identifying negative cases. The validation dataset's model may have a slightly higher rate of false positives.

Positive Predictive Value (Precision): The training dataset continues to have a higher positive predictive value (0.9506) compared to the validation dataset (0.9400).

Explanation:The model trained on the training data maintains its precision in predicting positive cases, resulting in fewer false positive predictions.

Reasons for Discrepancies:

Data Set Differences: Variations in data composition and distribution among different sets can significantly influence model performance. For instance, one data set may be more imbalanced, making it challenging to predict rare events.

Model Variability: Differences in model configurations or the random initialization of model parameters can lead to variations in performance.

Hyper parameter Tuning: Different hyper parameter settings, such as the choice of k in k-NN or other model-specific parameters, can affect model performance.

Data Splitting If data is randomly divided into training, validation, and test sets for each evaluation, it can introduce variations in results, especially with small datas ets.

Sample Variability: In small data sets, variations in the specific samples included in the validation and test sets can impact performance metrics.

Randomness: Some models, like neural networks, incorporate randomness in their optimization process, resulting in slight performance variations.