Assignment_2

Krishna Kumar Tavva - 811283461

2023-02-19

Loading required packages & calling libraries: readr, fastdummies, class, caret, ggplot2, lattice, ISLR, gmodels

```
#install.packages("readr")
#install.packages("fastDummies")
#install.packages("lattice")
#install.packages("ggplot2")
#install.packages("gmodels")
#install.packages("ISLR")
#install.packages("class")
#install.packages(caret")
library(readr)
library(fastDummies)
library(ISLR)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(class)
library(ggplot2)
library(gmodels)
library(lattice)
```

Load Data into R

```
ub <- data.frame(read.csv("C:/Users/krish/Downloads/UniversalBank.csv"))
str(ub)
```

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

Data cleaning (removing ID and ZIP code)

```
ub <- ub[,c(-1,-5)]
head(ub, n=5)
```

```
##   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
##   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
```

Verify if there are any null values in the datasets

```
any(is.na.data.frame(ub))
```

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

Converting data types of attributes

```
#Convert the Education variable to character
ub$Education <- as.character(ub$Education)

#Convert the Personal Loan variable to factor
ub$Personal.Loan <- as.factor(ub$Personal.Loan)
```

Dummying Variables using fastdummies package

```
ub <- dummy_cols(ub,select_columns = "Education")

#Remove Education variable after Dummy variables are created for Education
ubn <- ub[,-6]
colnames(ubn)
```

```
##  [1] "Age"                "Experience"         "Income"
##  [4] "Family"             "CCAvg"              "Mortgage"
##  [7] "Personal.Loan"      "Securities.Account" "CD.Account"
## [10] "Online"             "CreditCard"         "Education_1"
## [13] "Education_2"        "Education_3"
```

Train & Test Datasets

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

# Train Data
train.ubn <- ubn[train.index, ]
#summary(train.ubn)

# Test data
valid.ubn <- ubn[valid.index, ]
#summary(valid.ubn)
```

Normalizing the training dataset

```
Model_Z_Normalized <- preProcess(train.ubn[,-c(7,12:14)], method=c("center","scale"))

Normalized_Data_Train <- predict(Model_Z_Normalized, train.ubn)

Normalized_Data_Validation <- predict(Model_Z_Normalized, valid.ubn)

#summary(Normalized_Data_Train)
#summary(Normalized_Data_Validation)
```

Inserting a test set and normalizing it

```
test_data <- 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")
```

```
Test_Normalized <- predict(Model_Z_Normalized, test_data)
```

1.Running the knn model on the test dataset with k=1

```
Train_Predictors <- Normalized_Data_Train[,-7]
Validation_Predictors <- Normalized_Data_Validation[,-7]

Train_Labels <- Normalized_Data_Train[,7]
Validate_Lables <- Normalized_Data_Validation[,7]

Predicted_K <- knn(Train_Predictors, Test_Normalized, cl=Train_Labels, k=1)

head(Predicted_K)
```

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

When k=1 the customer is classified as 0 which indicates that the loan is not accepted. Since factor 1 is classified as loan acceptance and 0 is not accepted.

2.Choice of k that balances between overfitting and ignoring the predictor information

```
set.seed(1)
search_grid <- expand.grid(k=c(1:20))
#trtcontrol <- trainControl(method="repeatedcv")
model <- train(Personal.Loan~Age+Experience+Income+Family+CCAvg+Mortgage+
Securities.Account+CD.Account+Online+CreditCard+Education_1+Education_2+
  Education_3, data=Normalized_Data_Train, method="knn", tuneGrid = search_grid)
model
```

```
## 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
##    1  0.9507074  0.6628335
##    2  0.9440712  0.6154217
##    3  0.9447396  0.6126841
##    4  0.9432377  0.5923593
##    5  0.9425405  0.5749852
##    6  0.9422877  0.5674451
##    7  0.9405739  0.5449833
##    8  0.9401021  0.5375812
##    9  0.9394748  0.5275081
##   10  0.9386277  0.5165862
##   11  0.9382006  0.5101478
##   12  0.9380046  0.5040087
##   13  0.9377627  0.4991576
##   14  0.9369172  0.4895352
##   15  0.9363310  0.4832620
##   16  0.9362314  0.4814905
##   17  0.9355710  0.4730751
##   18  0.9351037  0.4670165
##   19  0.9349618  0.4651843
##   20  0.9347778  0.4621021
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

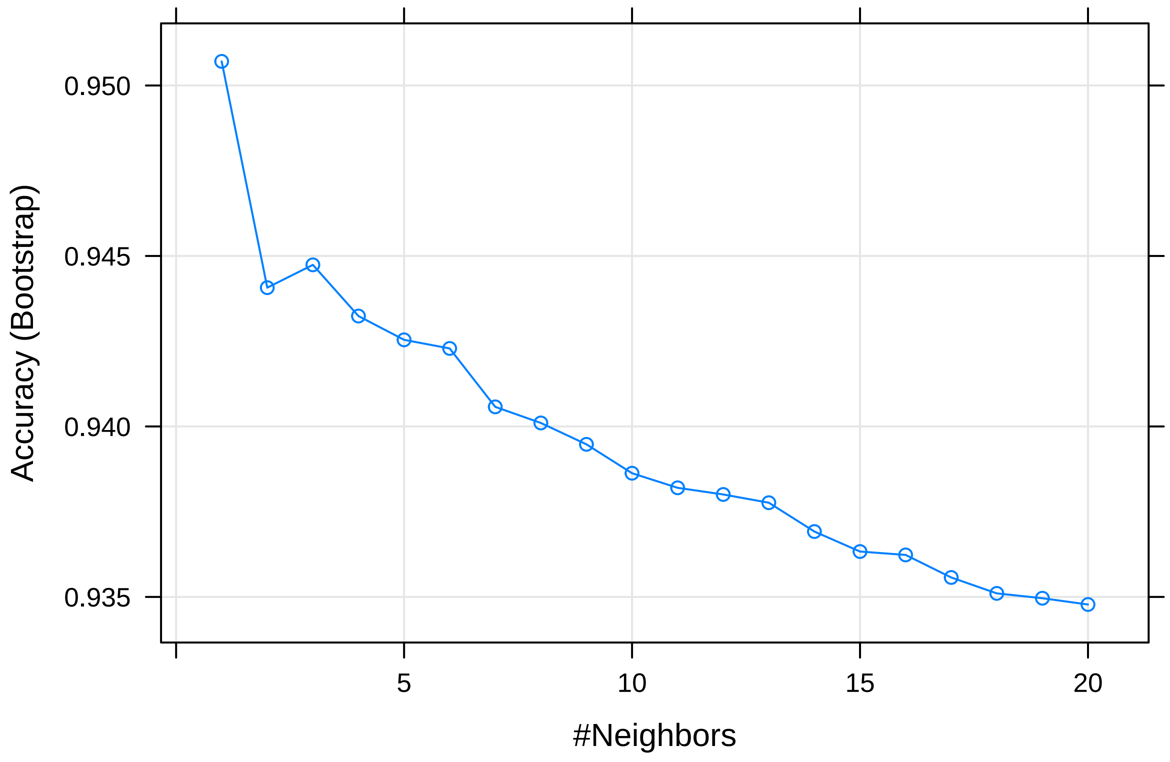
```
best_k <- model$bestTune[[1]]
best_k
```

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

The k value which balances between over fitting and ignoring the predictor information is k = 1.

Plotting the model

```
plot(model)
```



3.Confusion matrix being deployed over the validation data

```
pred_training <- predict(model,Normalized_Data_Validation[,-7])
confusionMatrix(pred_training, Validate_Lables)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1774   66
##          1   21  139
##
##                Accuracy : 0.9565
##                  95% CI : (0.9466, 0.965)
##     No Information Rate : 0.8975
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7381
##
##  Mcnemar's Test P-Value : 2.39e-06
##
##             Sensitivity : 0.9883
##             Specificity : 0.6780
##          Pos Pred Value : 0.9641
##          Neg Pred Value : 0.8688
##              Prevalence : 0.8975
##          Detection Rate : 0.8870
##    Detection Prevalence : 0.9200
##       Balanced Accuracy : 0.8332
##
##        'Positive' Class : 0
##
```

Miscalculations = 87, Accuracy = 0.9565, Sensitivity = 0.9883

4.Running the test data with best k choosen above

```
test_best_k <- knn(Train_Predictors, Test_Normalized, cl=Train_Labels, k=best_k)
head(test_best_k)
```

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

With the best k being choosen, the customer is classified as 0 which indicates that the loan is not accepted.

5.Repartitioning the data into training(50%), validation(30%) and test(20%) and running the entire model with best k

```
set.seed(1)
data_part <- createDataPartition(ubn$Personal.Loan, p=0.5, list = F)
n_train_data <- ubn[data_part,]
nd_test_data <- ubn[-data_part,]

data_part_v <- createDataPartition(nd_test_data$Personal.Loan,p=0.6, list =F)
n_validate_data <- nd_test_data[data_part_v,]
n_test_data <- nd_test_data[-data_part_v,]

#Normalization

norm_m <- preProcess(n_train_data[,-c(7,12:14)],method=c("center","scale"))

train_z <- predict(norm_m, n_train_data)
validate_z <- predict(norm_m, n_validate_data)
test_z <- predict(norm_m, n_test_data)

#Defining the predictors and labels

n_train_predictor <- train_z[,-7]
n_validate_predictor <- validate_z[,-7]
n_test_predictor <- test_z[,-7]
```

```r
n_train_labels <- train_z[,7]
n_validate_labels <- validate_z[,7]
n_test_labels <- test_z[,7]

#running the knn model over train dataset

n_model <- knn(n_train_predictor,n_train_predictor,cl=n_train_labels,k=best_k)
head(n_model)
```

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

```r
n_model1 <- knn(n_train_predictor,n_validate_predictor,cl=n_train_labels,k=best_k)
head(n_model1)
```

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

```r
n_model2 <- knn(n_train_predictor,n_test_predictor,cl=n_train_labels,k=best_k)
head(n_model2)
```

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

Using CrossTable to compare the Test vs Training and Validation

```r
confusionMatrix(n_model,n_train_labels)
```

```
## 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
##
```

```
#Train_Data - Miscalculations = 0 Accuracy = 1 Sensitivity = 1

#(This is because both the train and test datasets are same, model has already seen the data and hence

confusionMatrix(n_model1,n_validate_labels)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1343   50
##          1   13   94
##
##                Accuracy : 0.958
##                  95% CI : (0.9466, 0.9676)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 2.227e-15
##
##                   Kappa : 0.7266
##
##  Mcnemar's Test P-Value : 5.745e-06
##
##             Sensitivity : 0.9904
##             Specificity : 0.6528
##          Pos Pred Value : 0.9641
##          Neg Pred Value : 0.8785
##              Prevalence : 0.9040
##          Detection Rate : 0.8953
##    Detection Prevalence : 0.9287
##       Balanced Accuracy : 0.8216
##
##        'Positive' Class : 0
##
```

```
#Validation Data - Miscalculations = 13+50 = 63 Accuracy = 0.958 Sensitivity = 0.9904

confusionMatrix(n_model2,n_test_labels)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 881  34
##          1  23  62
##
```

```
##                Accuracy : 0.943
##                  95% CI : (0.9268, 0.9565)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 4.95e-06
##
##                   Kappa : 0.6539
##
##  Mcnemar's Test P-Value : 0.1853
##
##             Sensitivity : 0.9746
##             Specificity : 0.6458
##          Pos Pred Value : 0.9628
##          Neg Pred Value : 0.7294
##              Prevalence : 0.9040
##          Detection Rate : 0.8810
##    Detection Prevalence : 0.9150
##       Balanced Accuracy : 0.8102
##
##        'Positive' Class : 0
##
```

```
#Test_Data - Miscalculations = 57 Accuracy = 0.943 Sensitivity = 0.9746

#Interpretation: When comparing the test with that of training and validation, we shall exclude train f
```

Miscalculations: Validation - 63, Test - 57 Accuracy: Validation - 0.958, Test - 0.943 Sensitivity: Validation - 0.9904, Test - 0.9746

We see that the Test data has fewer miscalculations, less accuracy and sensitivity when compared to that of the validation data, by this we can say that the model works OK on the unseen data.