FML_Assignment2_811213452

Jay oza

2022-10-06

#Importing required packages

```
#install.packages("FNN")
#install.packages("psych")
library(psych)
library(FNN)
library(ISLR)
library(class)
```

```
##
## Attaching package: 'class'

## The following objects are masked from 'package:FNN':
##
##     knn, knn.cv
```

```
library(caret)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha

## Loading required package: lattice
```

#Importing dataset

```
universalbank<- read.csv("UniversalBank.csv")
```

#Eliminating ZIP code and ID from the dataset

```
ds=subset(universalbank, select=-c(ID, ZIP.Code ))
```

#Using is.na() to check for missing values

```r
ds_na <- is.na.data.frame("ds")
```

#Converting Categorical variables with numeric class to factors

```r
ds$Personal.Loan =  as.factor(ds$Personal.Loan)
ds$Education= as.factor(ds$Education)
summary(ds)
```

```
##       Age          Experience       Income          Family     
##  Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   :1.000  
##  1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:1.000  
##  Median :45.00   Median :20.0   Median : 64.00   Median :2.000  
##  Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :2.396  
##  3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:3.000  
##  Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :4.000  
##      CCAvg        Education    Mortgage     Personal.Loan Securities.Account
##  Min.   : 0.000   1:2096    Min.   :  0.0   0:4520        Min.   :0.0000    
##  1st Qu.: 0.700   2:1403    1st Qu.:  0.0   1: 480        1st Qu.:0.0000    
##  Median : 1.500   3:1501    Median :  0.0                 Median :0.0000    
##  Mean   : 1.938             Mean   : 56.5                 Mean   :0.1044    
##  3rd Qu.: 2.500             3rd Qu.:101.0                 3rd Qu.:0.0000    
##  Max.   :10.000             Max.   :635.0                 Max.   :1.0000    
##    CD.Account         Online         CreditCard   
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000  
##  Median :0.0000   Median :1.0000   Median :0.000  
##  Mean   :0.0604   Mean   :0.5968   Mean   :0.294  
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000  
```

#Creating dummy variables for education (categorical variables with more than 2 categories) using library (psych) and eliminating education

```r
dummy_education <- as.data.frame(dummy.code(ds$Education))
names(dummy_education) <- c("Education_1", "Education_2","Education_3")
ds_noeducation <- subset(ds, select=-c(Education))
ub <- cbind(ds_noeducation, dummy_education)
summary(ub)
```

```
##       Age          Experience       Income          Family     
##  Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   :1.000  
##  1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:1.000  
##  Median :45.00   Median :20.0   Median : 64.00   Median :2.000  
##  Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :2.396  
##  3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:3.000  
##  Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :4.000  
##      CCAvg           Mortgage     Personal.Loan Securities.Account
##  Min.   : 0.000   Min.   :  0.0   0:4520        Min.   :0.0000    
##  1st Qu.: 0.700   1st Qu.:  0.0   1: 480        1st Qu.:0.0000    
##  Median : 1.500   Median :  0.0                 Median :0.0000    
##  Mean   : 1.938   Mean   : 56.5                 Mean   :0.1044    
##  3rd Qu.: 2.500   3rd Qu.:101.0                 3rd Qu.:0.0000    
```

```
##  Max.   :10.000   Max.   :635.0                   Max.   :1.0000  
##    CD.Account         Online         CreditCard     Education_1    
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000  
##  Median :0.0000   Median :1.0000   Median :0.000   Median :0.0000  
##  Mean   :0.0604   Mean   :0.5968   Mean   :0.294   Mean   :0.4192  
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000   Max.   :1.0000  
##   Education_2      Education_3    
##  Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.0000  
##  Mean   :0.3002   Mean   :0.2806  
##  3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :1.0000
```

#Dividing the dataset into Training and Validation set and using preProcess() to normalize the dataset

```
set.seed(123)
Train_Index <-createDataPartition(ub$Personal.Loan, p=0.6, list=FALSE)
Train_ub <-ub[Train_Index,]
Validation_ub <-ub[-Train_Index,]
```

```
Model_norm <- preProcess(Train_ub[,-c(7,12:14)],method = c("center", "scale"))
Train_norm_ub <- predict(Model_norm,Train_ub)
Validation_norm_ub<- predict(Model_norm,Validation_ub)
```

#Creating a test dataset

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

#Normalizing the test dataset using z-score

```
Test_norm_ub <- predict(Model_norm, Test_data)
```

#Q1= Implementing kNN classification using k=1

```
Train_Predictors <- Train_norm_ub[,-7]
Validation_Predictors <- Validation_norm_ub[,-7]
Train_Labels <- Train_norm_ub[,7]
Validate_Lables <- Validation_norm_ub[,7]
Knn <- knn(Train_Predictors, Test_norm_ub, cl=Train_Labels, k=1)
head(Knn)
```

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

Since success class is specified as 1, here when k=1 customer is classified as 0 which means loan is not accepted.

#Q2= Finding the best k

```r
set.seed(123)
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+(
model
```

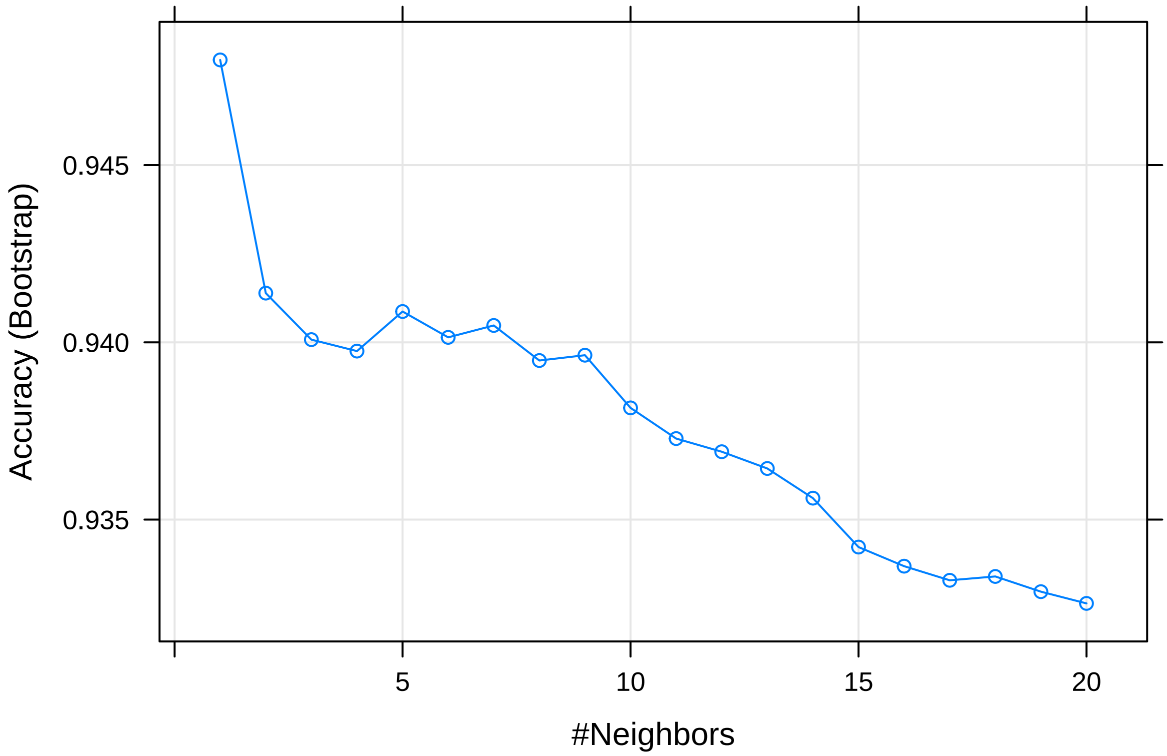
```
## k-Nearest Neighbors
##
## 3000 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3000, 3000, 3000, 3000, 3000, 3000, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9479683  0.6791568
##    2  0.9413890  0.6307845
##    3  0.9400766  0.6113089
##    4  0.9397528  0.6014080
##    5  0.9408706  0.5987998
##    6  0.9401406  0.5876125
##    7  0.9404763  0.5823387
##    8  0.9394876  0.5696284
##    9  0.9396370  0.5648137
##   10  0.9381509  0.5499292
##   11  0.9372856  0.5397043
##   12  0.9369143  0.5343188
##   13  0.9364416  0.5266224
##   14  0.9356041  0.5172636
##   15  0.9342242  0.5039270
##   16  0.9336850  0.4985215
##   17  0.9332867  0.4948477
##   18  0.9333953  0.4956182
##   19  0.9329659  0.4901981
##   20  0.9326351  0.4864292
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

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

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

#The value of best k is 1 as it provides the best result [i.e the choice of k that balances between overfitting and ignoring the predictor information]

```r
plot(model)
```



#3 Confusion matrix for the validation data that results from using the best k.

```
library(gmodels)
```

```
ConfusionMatrix<- predict(model,Validation_norm_ub[,-7])
confusionMatrix(ConfusionMatrix,Validate_Lables)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1789   54
##          1   19  138
##
##                Accuracy : 0.9635
##                  95% CI : (0.9543, 0.9713)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7711
##
##  Mcnemar's Test P-Value : 6.909e-05
##
##             Sensitivity : 0.9895
##             Specificity : 0.7188
##          Pos Pred Value : 0.9707
```

```
##          Neg Pred Value : 0.8790
##              Prevalence : 0.9040
##          Detection Rate : 0.8945
##    Detection Prevalence : 0.9215
##       Balanced Accuracy : 0.8541
##
##        'Positive' Class : 0
##
```

Miscalculation= False positive+ False negative= 73, Accuracy= 0.9635, Sensitivity= 0.9895

#4 Running best k on test data

```
test_bestk <- knn(Train_Predictors, Test_norm_ub, cl=Train_Labels, k=bestk)
head(test_bestk)
```

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

The customer is classified as 0 by choosing the best k, which means the loan is not accepted

#5 Reparting the data, this time into training, validation, and test sets and applying the k-NN method with the k chosen above.

```
Model.norm<- preProcess(ub[,-c(7,12:14)],method=c("center","scale"))
universalbank_norm <- predict(Model.norm,ub)
```

```
set.seed(422)
univbank <-createDataPartition(ub$Personal.Loan, p=0.5, list=FALSE)
Train_univbank <-ub[univbank,]
Testdata_univbank <-ub[-univbank,]

univbank_v <-createDataPartition(Testdata_univbank$Personal.Loan,p=0.6,list = FALSE)
Validate_univbank <- Testdata_univbank[univbank_v,]
Test_univbank <- Testdata_univbank[-univbank_v,]
```

```
Model.norm<- preProcess(ub[,-c(7,12:14)],method=c("center","scale"))

Train_norm <- predict(Model.norm,Train_univbank)
Validate_norm <- predict(Model.norm,Validate_univbank)
Test_norm<- predict(Model.norm,Test_univbank)
```

#Performing Knn classification with the k chosen above

```
Trainub_predictor <- Train_norm[,-7]
Validateub_predictor <- Validate_norm[,-7]
Testub_predictor <- Test_norm[,-7]

Trainub_labels <- Train_norm[,7]
Validateub_labels <- Validate_norm[,7]
Testub_labels <- Test_norm[,7]
```

#KNN classification over train dataset using the best k

```r
T_KNN_model <- knn(Trainub_predictor,Trainub_predictor,cl= Trainub_labels,k=bestk)
head(T_KNN_model)
```

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

#KNN classification over validation dataset using the best k

```r
V_KNN_model <- knn(Trainub_predictor,Validateub_predictor,cl=Trainub_labels,k=bestk)
head(V_KNN_model)
```

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

#KNN classification over test dataset using the best k

```r
TE_KNN_model<- knn(Trainub_predictor,Testub_predictor,cl=Trainub_labels,k=bestk)
head(TE_KNN_model)
```

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

#Confusion matrix to compare test set with that of the training and validation sets.

```r
confusionMatrix(T_KNN_model,Trainub_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
##
```

#The reason for 0 miscalculations, Accuracy=1 and Sensitivity= 1 is that train and test dataset are same. Therefore, it cannot predict any miscalculations and has an Accuracy of 100%

```
confusionMatrix(V_KNN_model,Validateub_labels)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1332   56
##          1   24   88
##
##                Accuracy : 0.9467
##                  95% CI : (0.9341, 0.9575)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 9.186e-10
##
##                   Kappa : 0.6588
##
##  Mcnemar's Test P-Value : 0.0005284
##
##             Sensitivity : 0.9823
##             Specificity : 0.6111
##          Pos Pred Value : 0.9597
##          Neg Pred Value : 0.7857
##              Prevalence : 0.9040
##          Detection Rate : 0.8880
##    Detection Prevalence : 0.9253
##       Balanced Accuracy : 0.7967
##
##        'Positive' Class : 0
##
```

#Miscalucations= False positive+ False Negative= 56+24= 80, Accuracy= 0.9467, Sensitivity = 0.9823

```
confusionMatrix(TE_KNN_model,Testub_labels)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 891  26
##          1  13  70
##
##                Accuracy : 0.961
##                  95% CI : (0.9471, 0.9721)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 5.695e-12
##
##                   Kappa : 0.7608
##
##  Mcnemar's Test P-Value : 0.05466
##
```

```
##            Sensitivity : 0.9856
##            Specificity : 0.7292
##         Pos Pred Value : 0.9716
##         Neg Pred Value : 0.8434
##             Prevalence : 0.9040
##         Detection Rate : 0.8910
##   Detection Prevalence : 0.9170
##      Balanced Accuracy : 0.8574
##
##       'Positive' Class : 0
##
```

Miscalculations= False positive+ False negative= 26+13= 39, Accuracy= 0.961, Sensitivity= 0.9856

#Interpretation: The training data shall be excluded from the consideration because it has already seen the data. Therefore, it will give a 100% accuracy when compared with other two models.

#Miscalculations: Validation - 80, Test - 39 #Accuracy: Validation - 0.9467, Test - 0.961 #Sensitivty: Validation - 0.9823, Test - 0.9856

#When we compare test model with that of validation model we see that test model has fewer miscalculations as compared to validation. It also has higher accuracy and sensitivity, making it work well.