ML assignment2

Varshini

2022-10-04

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

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

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

```
library(tinytex)
UB_1 <- read.csv("~/Downloads/UniversalBank (1).csv")

#removing the unwanted Columns
UB_1$ID<-NULL
UB_1$ZIP.Code<-NULL
View(UB_1)
```

```
## Warning in system2("/usr/bin/otool", c("-L", shQuote(DSO)), stdout = TRUE):
## running command ''/usr/bin/otool' -L '/Library/Frameworks/R.framework/Resources/
## modules/R_de.so'' had status 1
```

```
#transformin into factor variable
UB_1$Personal.Loan=as.factor(UB_1$Personal.Loan)

#Checking for null variables,if any available and converting Education to character
head(is.na(UB_1))
```

```
##        Age Experience Income Family CCAvg Education Mortgage Personal.Loan
## [1,] FALSE      FALSE  FALSE  FALSE FALSE     FALSE    FALSE         FALSE
## [2,] FALSE      FALSE  FALSE  FALSE FALSE     FALSE    FALSE         FALSE
## [3,] FALSE      FALSE  FALSE  FALSE FALSE     FALSE    FALSE         FALSE
## [4,] FALSE      FALSE  FALSE  FALSE FALSE     FALSE    FALSE         FALSE
## [5,] FALSE      FALSE  FALSE  FALSE FALSE     FALSE    FALSE         FALSE
## [6,] FALSE      FALSE  FALSE  FALSE FALSE     FALSE    FALSE         FALSE
##      Securities.Account CD.Account Online CreditCard
```

```
## [1,]              FALSE      FALSE  FALSE      FALSE
## [2,]              FALSE      FALSE  FALSE      FALSE
## [3,]              FALSE      FALSE  FALSE      FALSE
## [4,]              FALSE      FALSE  FALSE      FALSE
## [5,]              FALSE      FALSE  FALSE      FALSE
## [6,]              FALSE      FALSE  FALSE      FALSE
```

```
UB_1$Education=as.character(UB_1$Education)

#Create dummy  variables
Education1 <- ifelse(UB_1$Education==1 ,1,0)
Education2 <- ifelse(UB_1$Education==2 ,1,0)
Education3 <- ifelse(UB_1$Education==3 ,1,0)

dataset<-data.frame(Age=UB_1$Age,Experience=UB_1$Experience,Income=UB_1$Income,Family=UB_1$Family,CCAvg=



#Testdata defined
testset1<-data.frame(Age=40,Experience=10,Income=84,Family=2,CCAvg=2,Education_1=0,Education_2=1,Educat

#Data splitted in the ratio of 60:40
set.seed(250)
dummy<- createDataPartition(dataset$Personal.Loan,p=.6,list=FALSE,times=1)
trainset1 <- dataset[dummy, ]
validset1<- dataset[-dummy, ]

#Normalization
NormalMod=preProcess(testset1[,-(6:9)],method=c("center","scale"))
```

```
## Warning in preProcess.default(testset1[, -(6:9)], method = c("center",
## "scale")): Std. deviations could not be computed for: Age, Experience, Income,
## Family, CCAvg, Securities.Account, CD.Account, Online, CreditCard
```

```
trainNorm1 =predict(NormalMod,trainset1)
validNorm1 =predict(NormalMod,validset1)
testNorm1 =predict(NormalMod,testset1)

View(trainNorm1)
```

```
## Warning in system2("/usr/bin/otool", c("-L", shQuote(DSO)), stdout = TRUE):
## running command ''/usr/bin/otool' -L '/Library/Frameworks/R.framework/Resources/
## modules/R_de.so'' had status 1
```

```
#running knn
predicttrain1<-trainNorm1[,-9]
trainsamp<-trainNorm1[,9]
predictval<-validNorm1[,-9]
valsamp<-validNorm1[,9]

predict<-knn(predicttrain1, testNorm1, cl=trainsamp,k=1)
predict
```

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

```r
#Since determined when the k value=0, the customer denies the loan offered by the bank.



#Finding the best value of k

set.seed(350)
gr1<-expand.grid(k=seq(1:30))
mod1<-train(Personal.Loan~.,data=trainNorm1,method="knn",tuneGrid=gr1)

mod1
```

```
## 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  0.8951060  0.3541846
##    2  0.8908686  0.3468868
##    3  0.8908353  0.3381355
##    4  0.8929916  0.3327281
##    5  0.8944879  0.3219320
##    6  0.8948579  0.3158801
##    7  0.8959002  0.3069303
##    8  0.8962740  0.3082854
##    9  0.8981687  0.3118348
##   10  0.8969284  0.2986558
##   11  0.8962672  0.2830013
##   12  0.8980720  0.2985802
##   13  0.8985606  0.2985591
##   14  0.8988758  0.2960062
##   15  0.8980779  0.2829428
##   16  0.8990203  0.2820534
##   17  0.8996846  0.2804805
##   18  0.9003788  0.2782489
##   19  0.9006987  0.2755507
##   20  0.9001869  0.2704506
##   21  0.9004577  0.2639580
##   22  0.9006881  0.2688475
##   23  0.9013691  0.2652772
##   24  0.9015180  0.2620757
##   25  0.9016180  0.2625285
##   26  0.9003549  0.2528309
##   27  0.9014785  0.2562667
##   28  0.9013470  0.2469323
##   29  0.9014758  0.2480895
##   30  0.9022686  0.2536026
```

```
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 30.
```

```
value_k<-mod1$bestTune[[1]]



#confusion matrix

predicted<-predict(mod1,validNorm1[-9])
confusionMatrix(predicted,valsamp)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1786  156
##          1   22   36
##
##                Accuracy : 0.911
##                  95% CI : (0.8977, 0.9231)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.1526
##
##                   Kappa : 0.2548
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9878
##             Specificity : 0.1875
##          Pos Pred Value : 0.9197
##          Neg Pred Value : 0.6207
##              Prevalence : 0.9040
##          Detection Rate : 0.8930
##    Detection Prevalence : 0.9710
##       Balanced Accuracy : 0.5877
##
##        'Positive' Class : 0
##
```

```
#data is split in the ratio of 50:30:20

 set.seed(346)

lab1<-createDataPartition(dataset$Personal.Loan,p=0.5,list=FALSE)
lab2<-createDataPartition(dataset$Personal.Loan,p=0.3,list=FALSE)
lab3<-createDataPartition(dataset$Personal.Loan,p=0.2,list=FALSE)

train2<-dataset[lab1,]
valid2<-dataset[lab2,]
test2<-dataset[lab3,]


#normalizing new dataset
```

```
norm1<-preProcess(trainset1[,-(6:9)],method=c("center","scale"))

normtrain1 <- predict(norm1,trainset1)

normvalid1<-predict(norm1,validset1)

normtest1<-predict(norm1,testset1)
```