FML assignment

Shivani Pitla

2022-10-03

```
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)
universalbank=read.csv("C:/Users/shiva/Downloads/UniversalBank (1).csv")

#deleting unnecessary columns, such as ID and Zip code
universalbank$ID<-NULL
universalbank$ZIP.Code<-NULL
View(universalbank)

#converting to a variable factor
universalbank$Personal.Loan=as.factor(universalbank$Personal.Loan)

#running is.na command to check if there are any NA values
head(is.na(universalbank))
```

```
##        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
```

```
#converting education into character
universalbank$Education=as.character(universalbank$Education)

#Creating dummy  variables
education_1 <- ifelse(universalbank$Education==1 ,1,0)

education_2 <- ifelse(universalbank$Education==2 ,1,0)

education_3 <- ifelse(universalbank$Education==3 ,1,0)

ub_2<-data.frame(Age=universalbank$Age,Experience=universalbank$Experience,Income=universalbank$Income,

#setting up testdata
UBtest_1<-data.frame(Age=40,Experience=10,Income=84,Family=2,CCAvg=2,education_1=0,education_2=1,educat

#separating training and test sets of data
set.seed(130)
ub_dummy<- createDataPartition(ub_2$Personal.Loan,p=.6,list=FALSE,times=1)
train1_ub <- ub_2[ub_dummy, ]
valid1_ub<- ub_2[-ub_dummy, ]

#Normalization
ub_norm=preProcess(train1_ub[,-(6:9)],method=c("center","scale"))
trainNorm_ub =predict(ub_norm,train1_ub)
validNorm_ub =predict(ub_norm,valid1_ub)
testNorm_ub =predict(ub_norm,UBtest_1)

View(trainNorm_ub)

#printing knn algorithm

predicttrain_ub<-trainNorm_ub[,-9]
trainsample_ub<-trainNorm_ub[,9]
predictvalid_ub<-validNorm_ub[,-9]
validsample_ub<-validNorm_ub[,9]

predict_ub<-knn(predicttrain_ub, testNorm_ub, cl=trainsample_ub,k=1)
predict_ub
```

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

```r
predict_uvb <- knn(predicttrain_ub, predictvalid_ub, cl=trainsample_ub, k=1)

#The customer has rejected the loan offer. When the k value is 0, it is decided.

#printing ou the best value of k
set.seed(130)
grid_ub<-expand.grid(k=seq(1:30))
model_ub<-train(Personal.Loan~.,data=trainNorm_ub,method="knn",tuneGrid=grid_ub)
model_ub
```

```
## 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  0.9498389  0.6848818
##    2  0.9456633  0.6536088
##    3  0.9457715  0.6453799
##    4  0.9456939  0.6379842
##    5  0.9464967  0.6369189
##    6  0.9468210  0.6342505
##    7  0.9476230  0.6362095
##    8  0.9475486  0.6304329
##    9  0.9474853  0.6264414
##   10  0.9454230  0.6086942
##   11  0.9455233  0.6063682
##   12  0.9445282  0.5965274
##   13  0.9439058  0.5896361
##   14  0.9425072  0.5751621
##   15  0.9412785  0.5625136
##   16  0.9410684  0.5580477
##   17  0.9403809  0.5494274
##   18  0.9392614  0.5384893
##   19  0.9381366  0.5268213
##   20  0.9379190  0.5236724
##   21  0.9371251  0.5153713
##   22  0.9373413  0.5176735
##   23  0.9369361  0.5122613
##   24  0.9363567  0.5059488
##   25  0.9357750  0.5000855
##   26  0.9350157  0.4931644
##   27  0.9346204  0.4881624
##   28  0.9340405  0.4818989
##   29  0.9334942  0.4759017
##   30  0.9328745  0.4683129
##
```

```
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

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

#confusion matrix - validation dataset
confusionMatrix(predict_uvb,validsample_ub)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1784   64
##          1   24  128
##
##                Accuracy : 0.956
##                  95% CI : (0.9461, 0.9646)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7205
##
##  Mcnemar's Test P-Value : 3.219e-05
##
##             Sensitivity : 0.9867
##             Specificity : 0.6667
##          Pos Pred Value : 0.9654
##          Neg Pred Value : 0.8421
##              Prevalence : 0.9040
##          Detection Rate : 0.8920
##    Detection Prevalence : 0.9240
##       Balanced Accuracy : 0.8267
##
##        'Positive' Class : 0
##
```

```
#50:30:20 Repartition
data_part_new <- createDataPartition(ub_2$Personal.Loan,p=0.5, list = F)
Train_new <- ub_2[data_part_new,]
Train_db_new <- ub_2[-data_part_new,]

data_part_new_1 <- createDataPartition(Train_db_new$Personal.Loan, p=0.6, list = F)
validate_new <- Train_db_new[data_part_new_1,]
test_new <- Train_db_new[-data_part_new_1,]

#Normalization
norm_new <- preProcess(Train_new[,-(6:9)], method=c("center","scale"))
Train_new_p <- predict(norm_new, Train_new)
Validate_new_p <- predict(norm_new, validate_new)
Test_new_p <- predict(norm_new, test_new)

#predictors and labels
train_pre <- Train_new_p[,-9]
```

```r
validate_pre <- Validate_new_p[,-9]
test_pre <- Test_new_p[,-9]

train_l <- Train_new_p[,9]
validate_l <- Validate_new_p[,9]
test_l <- Test_new_p[,9]

#knn
knn_t <- knn(train_pre,train_pre,cl= train_l, k=value_k)

knn_v <- knn(train_pre,validate_pre,cl=train_l, k=value_k)

knn_tes <- knn(train_pre,test_pre,cl=train_l, k=value_k)

confusionMatrix(knn_t,train_l)
```

```
## 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
##
```

```r
confusionMatrix(knn_v,validate_l)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1337   49
##          1   19   95
##
```

```
##                Accuracy : 0.9547
##                  95% CI : (0.9429, 0.9646)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 1.551e-13
##
##                   Kappa : 0.712
##
##  Mcnemar's Test P-Value : 0.0004368
##
##             Sensitivity : 0.9860
##             Specificity : 0.6597
##          Pos Pred Value : 0.9646
##          Neg Pred Value : 0.8333
##              Prevalence : 0.9040
##          Detection Rate : 0.8913
##    Detection Prevalence : 0.9240
##       Balanced Accuracy : 0.8229
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(knn_tes,test_l)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 899  34
##          1   5  62
##
##                Accuracy : 0.961
##                  95% CI : (0.9471, 0.9721)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 5.695e-12
##
##                   Kappa : 0.7402
##
##  Mcnemar's Test P-Value : 7.340e-06
##
##             Sensitivity : 0.9945
##             Specificity : 0.6458
##          Pos Pred Value : 0.9636
##          Neg Pred Value : 0.9254
##              Prevalence : 0.9040
##          Detection Rate : 0.8990
##    Detection Prevalence : 0.9330
##       Balanced Accuracy : 0.8202
##
##        'Positive' Class : 0
##
```