Fundamentals Of Machine Learning- Assignment 2

Elmy Luka

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)
data_1 <- read.csv("~/Desktop/MS BA/Fundamentals Of Machine Learning/Assignment-2/UniversalBank.csv")

#Elimination the ID AND ZIP CODE Columns
data_1$ID<-NULL
data_1$ZIP.Code<-NULL
View(data_1)

#converting to factor variable
data_1$Personal.Loan=as.factor(data_1$Personal.Loan)

#Checking if there is any null variables
head(is.na(data_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
```

```
#Transforming Education to character
data_1$Education=as.character(data_1$Education)

#Creating dummy  variables
Education_1 <- ifelse(data_1$Education==1 ,1,0)

Education_2 <- ifelse(data_1$Education==2 ,1,0)

Education_3 <- ifelse(data_1$Education==3 ,1,0)




data_2<-data.frame(Age=data_1$Age,Experience=data_1$Experience,Income=data_1$Income,Family=data_1$Famil


#defining testdata
test_1<-data.frame(Age=40,Experience=10,Income=84,Family=2,CCAvg=2,Education_1=0,Education_2=1,Educatio

#splitting data to 60:40
set.seed(250)
temp<- createDataPartition(data_2$Personal.Loan,p=.6,list=FALSE,times=1)
train_1 <- data_2[temp, ]
valid_1<- data_2[-temp, ]

#Normalization
Norm_Model=preProcess(test_1[,-(6:9)],method=c("center","scale"))
```

```
## Warning in preProcess.default(test_1[, -(6:9)], method = c("center", "scale")):
## Std. deviations could not be computed for: Age, Experience, Income, Family,
## CCAvg, Securities.Account, CD.Account, Online, CreditCard
```

```
train_1_Norm =predict(Norm_Model,train_1)
valid_1_Norm =predict(Norm_Model,valid_1)
test_1_Norm =predict(Norm_Model,test_1)

View(train_1_Norm)

#running knn algorithm

predict_train<-train_1_Norm[,-9]
train_sample<-train_1_Norm[,9]
predict_valid<-valid_1_Norm[,-9]
```

```
valid_sample<-valid_1_Norm[,9]

predict<-knn(predict_train, test_1_Norm, cl=train_sample,k=1)
predict
```

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

```
#The loan offer has been denied by the customer. It is determined when the k value=0

#Finding the best value of k

set.seed(350)
grid_1<-expand.grid(k=seq(1:30))
model_1<-train(Personal.Loan~.,data=train_1_Norm,method="knn",tuneGrid=grid_1)

model_1
```

```
## 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<-model_1$bestTune[[1]]



#confusion matrix

predicted<-predict(model_1,valid_1_Norm[-9])
confusionMatrix(predicted,valid_sample)
```

```
## 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
##
```

```
#5 data is splitted to 50:30:20 ratio again

 set.seed(346)

label_1<-createDataPartition(data_2$Personal.Loan,p=0.5,list=FALSE)
```

```
label_2<-createDataPartition(data_2$Personal.Loan,p=0.3,list=FALSE)
label_3<-createDataPartition(data_2$Personal.Loan,p=0.2,list=FALSE)

train_2<-data_2[label_1,]
valid_2<-data_2[label_2,]
test_2<-data_2[label_3,]


#normalizing new dataset

normal_1<-preProcess(train_1[,-(6:9)],method=c("center","scale"))

normalized_train_1 <- predict(normal_1,train_1)

normalized_valid_1<-predict(normal_1,valid_1)

normalized_test_1<-predict(normal_1,test_1)

#running knn for train,validation and test data

predict_new_train= normalized_train_1[,-9]

predict_new_train_1= normalized_train_1[,9]

predict_new_valid=normalized_valid_1[,-9]

predict_new_valid_1=normalized_valid_1[,9]

predict_new_test=normalized_test_1[,-9]

predict_new_test_1=normalized_test_1[,9]

View(predict_new_test_1)

Predict_train_new<-knn(predict_new_train,predict_new_train,cl=predict_new_train_1,k=value_k)

Predict_valid_new<-knn(predict_new_train,predict_new_valid,cl=predict_new_train_1,k=value_k)


#training ,validation and test data confusion matrix

confusionMatrix(Predict_train_new,predict_new_train_1)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction    0    1
##          0 2704  195
##          1    8   93
##
##                Accuracy : 0.9323
##                  95% CI : (0.9227, 0.9411)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 1.99e-08
##
##                   Kappa : 0.4508
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9971
##             Specificity : 0.3229
##          Pos Pred Value : 0.9327
##          Neg Pred Value : 0.9208
##              Prevalence : 0.9040
##          Detection Rate : 0.9013
##    Detection Prevalence : 0.9663
##       Balanced Accuracy : 0.6600
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(Predict_valid_new,Predict_valid_new)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1938    0
##          1    0   62
##
##                Accuracy : 1
##                  95% CI : (0.9982, 1)
##     No Information Rate : 0.969
##     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 : 0.969
##          Detection Rate : 0.969
##    Detection Prevalence : 0.969
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 0
##
```

```
#CONCLUSION- We can conclude that the model performs well on the unseen data because the Test data has
```