

# Assignment2

Niharika D

2022-10-02

*#question-1*

```
UniversalBank<-read.csv("./UniversalBank.csv")  
head(UniversalBank) #Viewing imported dataset UniversalBank
```

```
##   ID Age Experience Income ZIPCode Family CCAvg Education Mortgage PersonalLoan  
## 1  1  25         1     49   91107      4   1.6          1         0           0  
## 2  2  45        19     34   90089      3   1.5          1         0           0  
## 3  3  39        15     11   94720      1   1.0          1         0           0  
## 4  4  35         9    100   94112      1   2.7          2         0           0  
## 5  5  35         8     45   91330      4   1.0          2         0           0  
## 6  6  37        13     29   92121      4   0.4          2        155          0  
##   SecuritiesAccount CDAccount 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  
## 6                 0          0      1          0
```

*#Converting PersonalLoan to factor datatype*

```
UniversalBank$PersonalLoan = as.factor(UniversalBank$PersonalLoan)
```

*#Removing variables which are not used*

```
remove_data<-subset(UniversalBank,select=-c(ID,ZIPCode))  
head(remove_data)
```

```
##   Age Experience Income Family CCAvg Education Mortgage PersonalLoan  
## 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  
## 6  37        13     29      4   0.4          2        155          0  
##   SecuritiesAccount CDAccount 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  
## 6                 0          0      1          0
```

```
UniversalBank<-remove_data
```

```
#checking for null values
```

```
null_values <- is.na(UniversalBank)
```

```
#Creating dummy variables for categorical variable Education
```

```
Education_1<-ifelse(UniversalBank$Education == '1',1,0)
```

```
Education_2<-ifelse(UniversalBank$Education == '2',1,0)
```

```
Education_3<-ifelse(UniversalBank$Education == '3',1,0)
```

```
UniBank<-data.frame(Age=UniversalBank$Age,Experience=UniversalBank$Experience,Income=UniversalBank$Income)
```

```
#Splitting data into 60% and 40%
```

```
set.seed(123)
```

```
library(caret)
```

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

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

```
Split_data <- createDataPartition(UniBank$PersonalLoan,p=.6,list=FALSE,times=1)
```

```
Training <- UniBank[Split_data,]
```

```
Validation <- UniBank[-Split_data,]
```

```
#Normalization
```

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

```
Training_norm = predict(Normalization,Training)
```

```
Validation_norm = predict(Normalization,Validation)
```

```
#Creating TEST Dataset with given values
```

```
library(class)
```

```
Test_predictor<-data.frame(Age=40,Experience=10,Income=84,Family=2,CCAvg=2,Education_1=0,Education_2=1,Education_3=0)
```

```
Normalization_test = predict(Normalization,Test_predictor)
```

```
Train_predictor = Training_norm[, -9]
```

```
Validate_predictor = Validation_norm[, -9]
```

```
Train_label<-Training_norm[,9]
```

```
Validate_label<-Validation_norm[,9]
```

```
Prediction <-knn(Train_predictor,  
                Normalization_test,  
                cl=Train_label,  
                k=1)
```

```
Prediction
```

```
## [1] 0
```

```
## Levels: 0 1
```

```

#question-2
#BestValueK
set.seed(321)
SearchGrid <- expand.grid(k=seq(1:30))
model <- train(PersonalLoan~.,data=Training_norm,method="knn",tuneGrid=SearchGrid)
model

```

```

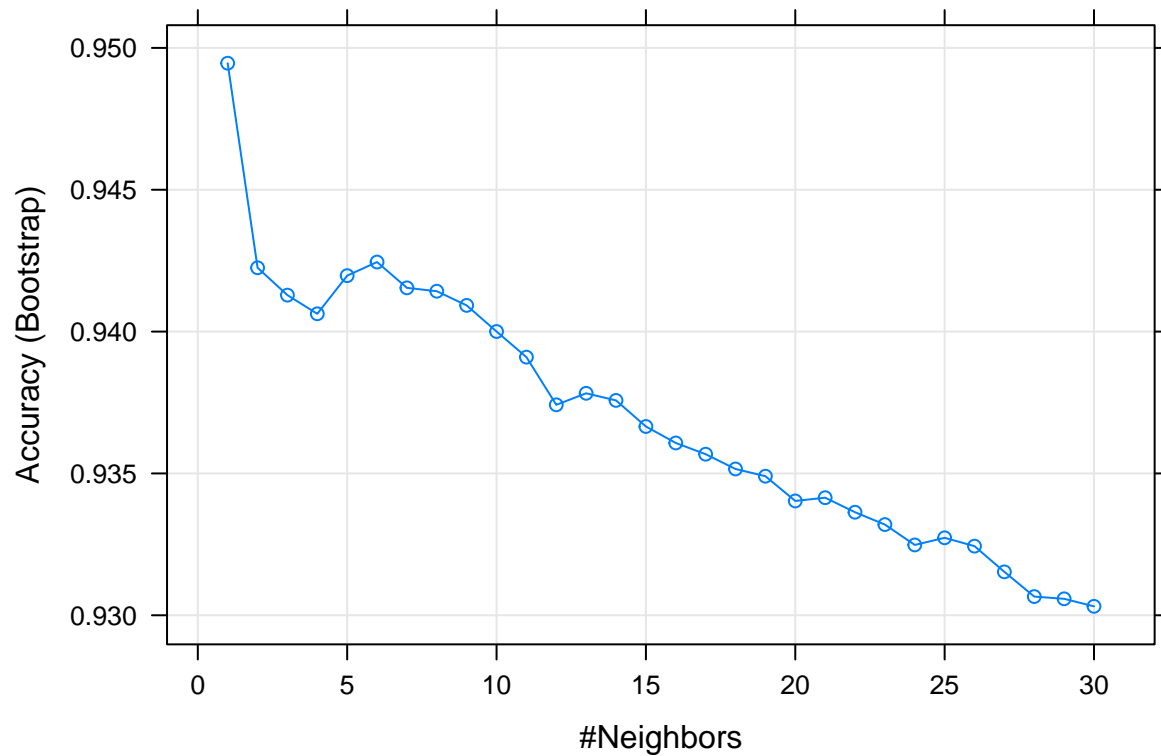
## k-Nearest Neighbors
##
## 3000 samples
## 13 predictor
## 2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3000, 3000, 3000, 3000, 3000, 3000, ...
## Resampling results across tuning parameters:
##
##  k  Accuracy  Kappa
##  1  0.9494606  0.6790540
##  2  0.9422495  0.6280947
##  3  0.9412847  0.6134538
##  4  0.9406270  0.6016192
##  5  0.9419724  0.6002352
##  6  0.9424516  0.5964779
##  7  0.9415414  0.5813768
##  8  0.9414230  0.5739398
##  9  0.9409234  0.5622319
## 10  0.9400033  0.5504214
## 11  0.9390992  0.5405417
## 12  0.9374193  0.5250147
## 13  0.9378248  0.5264158
## 14  0.9375736  0.5209409
## 15  0.9366545  0.5095868
## 16  0.9360740  0.5031104
## 17  0.9356776  0.4995591
## 18  0.9351571  0.4930217
## 19  0.9349029  0.4902960
## 20  0.9340303  0.4808043
## 21  0.9341448  0.4819498
## 22  0.9336325  0.4767749
## 23  0.9331972  0.4711490
## 24  0.9324792  0.4637587
## 25  0.9327311  0.4626125
## 26  0.9324367  0.4611510
## 27  0.9315305  0.4514573
## 28  0.9306597  0.4421025
## 29  0.9305808  0.4389133
## 30  0.9303154  0.4362343
##
## 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
```

```
plot(model)
```



*#Conclusion: K=1 is the choice that balances between overfitting and ignoring the predictor information*

```
#question-3
```

```
#Confusion Matrix
```

```
Prediction_new <- predict(model,Validate_predictor)
```

```
confusionMatrix(Prediction_new,Validate_label)
```

```
## 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
##
```

*#question-4*

```
testing_best_k <- knn(Train_predictor, Normalization_test , cl=Train_label, k=best_k)
head(testing_best_k)
```

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

*#Conclusion: Based on the K value- we can conclude that the customer won't accept a personal loan.*

*#question-5*

*#Splitting data into 50%,30%,20%*

```
set.seed(456)
```

```
Split_data_train <- createDataPartition(UniBank$PersonalLoan,p=.5,list=FALSE,times=1)
Train.df <- UniBank[Split_data_train,]
```

```
Split_data_validate <- createDataPartition(UniBank$PersonalLoan,p=.3,list=FALSE,times=1)
Validate.df <- UniBank[Split_data_validate,]
```

```
Split_data_test <- createDataPartition(UniBank$PersonalLoan,p=.2,list=FALSE,times=1)
Test.df <- UniBank[Split_data_test,]
```

*#Normalizing the data*

```
Normalize <- preProcess(Train.df[,-(6:9)],method = c("center","scale"))
Train_norm.df = predict(Normalize,Train.df)
Validate_norm.df = predict(Normalize,Validate.df)
Test_norm.df = predict(Normalize,Test.df)
```

*#Finding knn value*

```
Train_predict = Train_norm.df[, -9]
Valid_predict = Validate_norm.df[, -9]
Test_predict = Test_norm.df[, -9]
```

```

Train_label_new = Train_norm.df[,9]
Valid_label_new = Validate_norm.df[,9]
Test_label_new = Test_norm.df[,9]

Prediction_train <- knn(Train_predict,Train_predict,c1=Train_label_new,k=best_k)

Prediction_valid <- knn(Train_predict,Valid_predict,c1=Train_label_new,k=best_k)

Prediction_test <- knn(Train_predict,Test_predict,c1=Train_label_new,k=best_k)

#Training_data:
confusionMatrix(Prediction_train,Train_label_new)

```

```

## 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
##

```

*#Accuracy: 1; Sensitivity:1*

```

#Validation_data
confusionMatrix(Prediction_valid,Valid_label_new)

```

```

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0      1
##              0 1344    21
##              1    12  123
##

```

```
##
##           Accuracy : 0.978
##           95% CI : (0.9692, 0.9848)
##      No Information Rate : 0.904
##      P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.8696
##
##  McNemar's Test P-Value : 0.1637
##
##           Sensitivity : 0.9912
##           Specificity : 0.8542
##      Pos Pred Value : 0.9846
##      Neg Pred Value : 0.9111
##           Prevalence : 0.9040
##      Detection Rate : 0.8960
##      Detection Prevalence : 0.9100
##      Balanced Accuracy : 0.9227
##
##      'Positive' Class : 0
##
```

*#Accuracy:0.978;Sensitivity:0.991*

*#Test\_data*

```
confusionMatrix(Prediction_test,Test_label_new)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0    1
##           0 900   8
##           1   4  88
##
##           Accuracy : 0.988
##           95% CI : (0.9791, 0.9938)
##      No Information Rate : 0.904
##      P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9296
##
##  McNemar's Test P-Value : 0.3865
##
##           Sensitivity : 0.9956
##           Specificity : 0.9167
##      Pos Pred Value : 0.9912
##      Neg Pred Value : 0.9565
##           Prevalence : 0.9040
##      Detection Rate : 0.9000
##      Detection Prevalence : 0.9080
##      Balanced Accuracy : 0.9561
##
##      'Positive' Class : 0
##
```

*#Accuracy:0.988;Sensitivity:0.9956*

*#Conclusion:Based on the values of Accuracy and Sensitivity observed through confusionMatrix, we can co*