Assignment_2

Hannah Cronin

2022-09-27

```
UniversalBankOrig <- read.csv("~/Downloads/UniversalBank.csv")
library(caret)
```

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

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

```
library(ISLR)
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
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(FNN)
```

```
##
## Attaching package: 'FNN'
```

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

Create Dummy Variables

```
UniversalBank = subset(UniversalBankOrig, select = -c(ID, ZIP.Code)) #strip ID, ZIP.Code
education_dummy = as.data.frame(dummy.code(UniversalBank$Education)) #creates dummy vari
able
names(education_dummy) = c('Education_1', 'Education_2', 'Education_3') #renames
bank = subset(UniversalBank, select = -c(Education)) #strip education from dataset
UBank = cbind(bank, education_dummy)
UBank$Personal.Loan = as.factor(UBank$Personal.Loan)
```

Partitioning the data

```
Train_Index = createDataPartition(UBank$Personal.Loan, p=.6, list = FALSE)
train.df = UBank[Train_Index,]
valid.df = UBank[-Train_Index,]
new.df = data.frame(Age=40, Experience=10, Income=84, Family=2, CCAvg=2, Education_1=0,
 Education_2=1, Education_3=0, Mortgage=0, Securities.Account=0, CD.Account=0, Online=1,
CreditCard=1) #new customer info
```

Compare partitions

```
summary(train.df$Personal.Loan)
```

```
##    0    1
## 2712  288
```

```
summary(valid.df$Personal.Loan)
```

```
##    0    1
## 1808  192
```

Data Normalization

```
train.norm.df = train.df
valid.norm.df = valid.df
new.norm.df = new.df
Ubank.norm.df = UBank

norm.values = preProcess(train.df[,-7], method=c('scale','center'))
train.norm.df[, -7] = predict(norm.values, train.df[,-7])
valid.norm.df[, -7] = predict(norm.values, valid.df[,-7])
new.norm.df <- predict(norm.values, new.df)
Ubank.norm.df[,-7] <- predict(norm.values,UBank[,-7])
```

```
nn = knn(train = train.norm.df[,-7], test = valid.norm.df[,-7],
          cl = train.norm.df[,7], k = 1, prob=TRUE)
head(nn)
```

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

```
# This customer would be classified as a loan denial (0)
```

```
accuracy.df <- data.frame(k = seq(1, 20, 1), accuracy = rep(0,20))

for(i in 1:20) {
  knn.pred <- knn(train.norm.df[, -7], valid.norm.df[, -7],
                  cl = train.norm.df[, 7], k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn.pred, valid.norm.df[, 7])$overall[1]
}
accuracy.df
```

```
##     k accuracy
## 1   1   0.9620
## 2   2   0.9565
## 3   3   0.9640
## 4   4   0.9590
## 5   5   0.9630
## 6   6   0.9585
## 7   7   0.9625
## 8   8   0.9550
## 9   9   0.9570
## 10 10   0.9530
## 11 11   0.9550
## 12 12   0.9505
## 13 13   0.9545
## 14 14   0.9505
## 15 15   0.9520
## 16 16   0.9455
## 17 17   0.9480
## 18 18   0.9455
## 19 19   0.9475
## 20 20   0.9440
```

```
# K = 3 is the best option
```

```
cm <- knn(train = train.df[,-7],test = valid.df[,-7], cl = train.df[,7], k=3, prob=TRUE)
confusionMatrix(cm, valid.df[,7])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1739  115
##          1   69   77
##
##                Accuracy : 0.908
##                  95% CI : (0.8945, 0.9203)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.2869433
##
##                   Kappa : 0.4064
##
##  Mcnemar's Test P-Value : 0.0009085
##
##             Sensitivity : 0.9618
##             Specificity : 0.4010
##          Pos Pred Value : 0.9380
##          Neg Pred Value : 0.5274
##              Prevalence : 0.9040
##          Detection Rate : 0.8695
##    Detection Prevalence : 0.9270
##       Balanced Accuracy : 0.6814
##
##        'Positive' Class : 0
##
```

```
#Confusion Matrix
```

```
knn.2 <- knn(train = train.df[,-7],test = new.df, cl = train.df[,7], k=3, prob=TRUE)
knn.2
```

```
## [1] 0
## attr(,"prob")
## [1] 1
## attr(,"nn.index")
##      [,1] [,2] [,3]
## [1,]  423 1405 2776
## attr(,"nn.dist")
##          [,1]     [,2]     [,3]
## [1,] 4.004997 4.548626 4.657252
## Levels: 0
```

```
# Customer is classified as 0
```

Create Partions

```
Train_Index2 = createDataPartition(UBank$Personal.Loan, p=.5, list = FALSE)
training_index = UBank[Train_Index2,]
Bank_Data_Index = UBank[-Train_Index2,]
```

```
testvalid_index = createDataPartition(Bank_Data_Index$Personal.Loan, p = .6, list = FALS
E)
validation_index = Bank_Data_Index[testvalid_index,]
test_index = Bank_Data_Index[-testvalid_index,]
```

Normalize Data

```
train.norm_index = training_index
valid.norm_index = validation_index
test.norm_index = test_index

norm.values = preProcess(training_index[,-7], method=c('scale','center'))
train.norm_index[, -7] = predict(norm.values, training_index[,-7])
valid.norm_index[, -7] = predict(norm.values, validation_index[,-7])
test.norm_index[, -7] = predict(norm.values, test_index[,-7])
```

knn

```
testingknn <- knn(train = training_index[,-7],test = test_index[,-7], cl = training_inde
x[,7], k=3, prob=TRUE)
validationknn <- knn(train = training_index[,-7],test = validation_index[,-7], cl = trai
ning_index[,7], k=3, prob=TRUE)
trainknn <- knn(train = training_index[,-7],test = training_index[,-7], cl = training_in
dex[,7], k=3, prob=TRUE)
```

Confusion matrix - test

```
confusionMatrix(testingknn, test_index[,7])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 863  66
##          1  41  30
##
##                Accuracy : 0.893
##                  95% CI : (0.8722, 0.9115)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.89021
##
##                   Kappa : 0.3023
##
##  Mcnemar's Test P-Value : 0.02033
##
##             Sensitivity : 0.9546
##             Specificity : 0.3125
##          Pos Pred Value : 0.9290
##          Neg Pred Value : 0.4225
##              Prevalence : 0.9040
##          Detection Rate : 0.8630
##    Detection Prevalence : 0.9290
##       Balanced Accuracy : 0.6336
##
##        'Positive' Class : 0
##
```

Confusion matrix - validation

```
confusionMatrix(validationknn, validation_index[,7])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1308   91
##          1   48   53
##
##                Accuracy : 0.9073
##                  95% CI : (0.8915, 0.9215)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.3503489
##
##                   Kappa : 0.3839
##
##  Mcnemar's Test P-Value : 0.0003675
##
##             Sensitivity : 0.9646
##             Specificity : 0.3681
##          Pos Pred Value : 0.9350
##          Neg Pred Value : 0.5248
##              Prevalence : 0.9040
##          Detection Rate : 0.8720
##    Detection Prevalence : 0.9327
##       Balanced Accuracy : 0.6663
##
##        'Positive' Class : 0
##
```

Confusion matrix - training

```
confusionMatrix(trainknn, training_index[,7])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2235   89
##          1   25  151
##
##                Accuracy : 0.9544
##                  95% CI : (0.9455, 0.9622)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7017
##
##  Mcnemar's Test P-Value : 3.624e-09
##
##             Sensitivity : 0.9889
##             Specificity : 0.6292
##          Pos Pred Value : 0.9617
##          Neg Pred Value : 0.8580
##              Prevalence : 0.9040
##          Detection Rate : 0.8940
##    Detection Prevalence : 0.9296
##       Balanced Accuracy : 0.8091
##
##        'Positive' Class : 0
##
```

The testknn has the highest accuracy, then the trainknn, and lastly the validationknn. The fact that validationknn isn't as high as the trainknn means the model is not overfitting. Lastly, the trainingknn has an accuracy of .9464, which looks good as it suggests the model isn't over/underfitting and is producing accurate predicted results.