Assignment 2

2022-10-01

```r
library(psych)
library(caret)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

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

## Loading required package: lattice
```

```r
library(FNN)
library(class)
```

```
##
## Attaching package: 'class'

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

```r
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
```

```r
library(dummy)
```

```
## dummy 0.1.3

## dummyNews()
```

```r
library(readr)
```

```r
data <- read.csv("./UniversalBank.csv")
```

```r
# Remove ID and Zip Code
df=subset(data, select=-c(ID, ZIP.Code))
```

```r
# Create dummy variables for education
dummy_edu <- as.data.frame(dummy.code(df$Education))
```

```
names(dummy_edu) <- c("Education.1", "Education.2","Education.3")
df_without_education <- subset(df, select=-c(Education))
bank_data <- cbind(df_without_education, dummy_edu)
```

```
Test_Data = data.frame(Age = as.integer(40), Experience = as.integer(10), Income = as.integer(84), Famil
```

```
# Partition data
set.seed(1)

Train_Index = createDataPartition(bank_data$Age, p=0.6, list=FALSE)
Train_Data = bank_data[Train_Index,]
Validation_Data = bank_data[-Train_Index,]
Main_Data = bank_data
```

```
# Normalize

# Copy the original data
train.norm.df    <- Train_Data
valid.norm.df    <- Validation_Data
test.norm.df     <- Test_Data
main.norm.df     <- Main_Data

norm.values <- preProcess(Train_Data[,-7], method=c("center", "scale"))
train.norm.df[,-7] <- predict(norm.values, Train_Data[,-7])
valid.norm.df [,-7]<- predict(norm.values, Validation_Data[,-7])
test.norm.df <- predict(norm.values, Test_Data)
main.norm.df[,-7] <- predict(norm.values, bank_data[,-7])
```

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

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

```
# The customer is classified as 0 (not having accepted the loan)
```

```
accuracy.df <- data.frame(k = seq(1, 14, 1), accuracy = rep(0, 14))
# compute knn for different k on validation.
for(i in 1:14) {
  prediction <- knn(train.norm.df[, -7], valid.norm.df[, -7],
                cl = train.norm.df[, 7], k = i, prob=TRUE)
   accuracy.df[i, 2] <- confusionMatrix(prediction, as.factor(valid.norm.df[,7]))$overall[1]
}

accuracy.df
```

```
##     k  accuracy
## 1   1 0.9644822
## 2   2 0.9584792
## 3   3 0.9629815
## 4   4 0.9634817
## 5   5 0.9649825
## 6   6 0.9599800
## 7   7 0.9639820
## 8   8 0.9619810
## 9   9 0.9604802
## 10 10 0.9589795
## 11 11 0.9609805
## 12 12 0.9594797
## 13 13 0.9609805
## 14 14 0.9594797
```

```
# k=3 balances between overfitting and ignoring the predictor information
```

```
prediction_test <- knn(train = train.norm.df[,-7],test = valid.norm.df[,-7], cl = train.norm.df[,7], k=
confusionMatrix(prediction_test, as.factor(valid.norm.df[,7]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1812   64
##          1   10  113
##
##                Accuracy : 0.963
##                  95% CI : (0.9537, 0.9708)
##     No Information Rate : 0.9115
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.734
##
##  Mcnemar's Test P-Value : 7.223e-10
##
##             Sensitivity : 0.9945
##             Specificity : 0.6384
##          Pos Pred Value : 0.9659
##          Neg Pred Value : 0.9187
##              Prevalence : 0.9115
##          Detection Rate : 0.9065
##    Detection Prevalence : 0.9385
##       Balanced Accuracy : 0.8165
##
##        'Positive' Class : 0
##
```

```
# Using best k
prediction <- knn(train = main.norm.df[,-7], test = Test_Data, cl = main.norm.df[,7], k = 3, prob = TRU
head(prediction)
```

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

```
# Customer is classified as Personal.Loan = 1 when k = 3, the customer would have accepted the loan
```

```
# Repartition data

Test_Index_1 = createDataPartition(bank_data$Age, p=0.2, list=FALSE)
Test_Data_1 = bank_data[Test_Index_1,]

train_val_data = bank_data[-Test_Index_1,]

Train_Index_1 = createDataPartition(train_val_data$Age, p=0.6245, list=FALSE)
Train_Data_1 = train_val_data[Train_Index_1,]
```

```
Validation_Data_1 = train_val_data[-Train_Index_1,]
```

```
# Copy the original data
train.norm.df_1 <- Train_Data_1
valid.norm.df_1 <- Validation_Data_1
test.norm.df_1 <- Test_Data_1
train_val_data.norm.df <- train_val_data

norm.values_1 <- preProcess(Train_Data_1[,-7], method=c("center", "scale"))
train.norm.df_1[,-7] <- predict(norm.values_1, Train_Data_1[,-7])
valid.norm.df_1[,-7] <- predict(norm.values_1, Validation_Data_1[,-7])
test.norm.df_1[,-7] <- predict(norm.values_1, test.norm.df_1[,-7])
```

```
test_knn <- knn(train = train.norm.df_1[,-7], test = test.norm.df_1[,-7], cl = train.norm.df_1[,7], k =
valid_knn <- knn(train = train.norm.df_1[,-7], test = valid.norm.df_1[,-7], cl = train.norm.df_1[,7], k
train_knn <- knn(train = train.norm.df_1[,-7], test = train.norm.df_1[,-7], cl = train.norm.df_1[,7], k

confusionMatrix(test_knn, as.factor(test.norm.df_1[,7]))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##          0 893  42
##          1   7  59
##
##                Accuracy : 0.951
##                  95% CI : (0.9358, 0.9636)
##     No Information Rate : 0.8991
##     P-Value [Acc > NIR] : 1.565e-09
##
##                   Kappa : 0.6812
##
##  Mcnemar's Test P-Value : 1.191e-06
##
##             Sensitivity : 0.9922
##             Specificity : 0.5842
##          Pos Pred Value : 0.9551
##          Neg Pred Value : 0.8939
##              Prevalence : 0.8991
##          Detection Rate : 0.8921
##    Detection Prevalence : 0.9341
##       Balanced Accuracy : 0.7882
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(valid_knn, as.factor(valid.norm.df_1[,7]))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0    1
##          0 1363   47
##          1   10   79
```

```
##
##                Accuracy : 0.962
##                  95% CI : (0.951, 0.9711)
##     No Information Rate : 0.9159
##     P-Value [Acc > NIR] : 7.947e-13
##
##                   Kappa : 0.7151
##
##  Mcnemar's Test P-Value : 1.858e-06
##
##             Sensitivity : 0.9927
##             Specificity : 0.6270
##          Pos Pred Value : 0.9667
##          Neg Pred Value : 0.8876
##              Prevalence : 0.9159
##          Detection Rate : 0.9093
##    Detection Prevalence : 0.9406
##       Balanced Accuracy : 0.8099
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(train_knn, as.factor(train.norm.df_1[,7]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2242   49
##          1    5  204
##
##                Accuracy : 0.9784
##                  95% CI : (0.9719, 0.9837)
##     No Information Rate : 0.8988
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8713
##
##  Mcnemar's Test P-Value : 4.87e-09
##
##             Sensitivity : 0.9978
##             Specificity : 0.8063
##          Pos Pred Value : 0.9786
##          Neg Pred Value : 0.9761
##              Prevalence : 0.8988
##          Detection Rate : 0.8968
##    Detection Prevalence : 0.9164
##       Balanced Accuracy : 0.9020
##
##        'Positive' Class : 0
##
```