Assignment 2

Sean Bradford

Question 1: 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, and Credit Card = 1. Perform a k-NN classification with all predictors except ID and ZIP code using k = 1. Remember to transform categorical predictors with more than two categories into dummy variables first. Specify the success class as 1 (loan acceptance), and use the default cutoff value of 0.5. How would this customer be classified?

```
rm(list=ls())
library(FNN)
```

```
## Warning: package 'FNN' was built under R version 4.2.1
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.1
```

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

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

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

```
## Warning: package 'psych' was built under R version 4.2.1
```

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

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

```
bank_data=(read.csv("C:\\Users\\Sean\\OneDrive\\Desktop\\Grad School\\Machine Learning\\UniversalBank.cs

# Transformed Education to a factor for dummy coding
bank_data$Education=as.factor(bank_data$Education)

# Created subset and removed ID & Zip Code
bank_subset=subset(bank_data, select=-c(ID, ZIP.Code ))
```

```
# dummyVars caused issues with new.df.norm (line 52)
# Researched alternatives and found dummy.code in psych package
# Created dummy variables for Education
bank_dumm = as.data.frame(dummy.code(bank_subset$Education))
names(bank_dumm) = c("Education_1", "Education_2","Education_3")

# Created subset without Education then combined data frames under bank_dumm1
no_ed = subset(bank_subset, select=-c(Education))
bank_dumm1=cbind(no_ed,bank_dumm)

# Transformed Personal.Loan & CCAvg for future knn implementation
bank_dumm1$Personal.Loan = as.factor(bank_dumm1$Personal.Loan)
bank_dumm1$CCAvg = as.integer(bank_dumm1$CCAvg)

# Partitioned data - 60% training and 40% validation
set.seed(111)
train.index = sample(row.names(bank_dumm1), 0.6*dim(bank_dumm1)[1])
test.index = setdiff(row.names(bank_dumm1), train.index)
train.df = bank_dumm1[train.index, ]
valid.df = bank_dumm1[test.index, ]

# Created data frame for customer test
new.df = data.frame(Age = as.integer(40), Experience = as.integer(10), Income = as.integer(84), Family =

# Normalized data (z-score)
norm.values = preProcess(train.df[, -7], method=c("center", "scale"))
train.df[, -7] = predict(norm.values, train.df[, -7])
valid.df[, -7] = predict(norm.values, valid.df[, -7])
new.df.norm = predict(norm.values, new.df)

# Performed k-NN classification
nn = knn(train = train.df[,-7],test = new.df.norm, cl = train.df[,7], k=1)
nn
```

```
## [1] 0
## attr(,"nn.index")
##      [,1]
## [1,]  955
## attr(,"nn.dist")
##           [,1]
## [1,] 0.9871344
## Levels: 0
```

Algorithm predicted customer classification of 0

Question 2: What is a choice of k that balances between overfitting and ignoring the predictor information?

```
# Initialized a data frame with columns k and accuracy
accuracy.df = data.frame(k = seq(1, 14, 1), accuracy = rep(0, 14))

# Computed k-NN for different k's applied to validation data
for(i in 1:14) {
  nn2 = knn(train = train.df[,-7],test = valid.df[,-7], cl = train.df[,7], k=i)
```

```
  accuracy.df[i, 2] = confusionMatrix(nn2, valid.df[,7])$overall[1]
}
accuracy.df
```

```
##     k accuracy
## 1   1   0.9555
## 2   2   0.9505
## 3   3   0.9625
## 4   4   0.9530
## 5   5   0.9555
## 6   6   0.9525
## 7   7   0.9535
## 8   8   0.9505
## 9   9   0.9510
## 10 10   0.9475
## 11 11   0.9480
## 12 12   0.9445
## 13 13   0.9455
## 14 14   0.9450
```

K=3 is shown to be the most accurate K value

Question 3: Show the confusion matrix for the validation data that results from using the best k.

```
nn3 = knn(train = train.df[,-7],test = valid.df[,-7], cl = train.df[,7], k=3)
confusionMatrix(nn3,valid.df[,7])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1806   69
##          1    6  119
##
##                Accuracy : 0.9625
##                  95% CI : (0.9532, 0.9704)
##     No Information Rate : 0.906
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7409
##
##  Mcnemar's Test P-Value : 8.118e-13
##
##             Sensitivity : 0.9967
##             Specificity : 0.6330
##          Pos Pred Value : 0.9632
##          Neg Pred Value : 0.9520
##              Prevalence : 0.9060
##          Detection Rate : 0.9030
##    Detection Prevalence : 0.9375
##       Balanced Accuracy : 0.8148
##
##        'Positive' Class : 0
##
```

Question 4: Consider the following customer: 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 and Credit Card = 1. Classify the customer using the best k.

```
customer_data= data.frame(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Education_1 = 
nn4 = knn(train = train.df[,-7],test = customer_data, cl = train.df[,7], k=3)
nn4
```

```
## [1] 1
## attr(,"nn.index")
##      [,1] [,2] [,3]
## [1,] 1769 1898  769
## attr(,"nn.dist")
##         [,1]     [,2]     [,3]
## [1,] 90.5376 90.54006 90.60584
## Levels: 1
```

Algorithm predicted customer classification of 1 with best k (k=3)

Question 5: Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%). Apply the k-NN method with the k chosen above. Compare the confusion matrix of the test set with that of the training and validation sets. Comment on the differences and their reason.

```
# Followed same sequence as question 1
bank_data$Education=as.factor(bank_data$Education)

bank_subset=subset(bank_data, select=-c(ID, ZIP.Code ))

bank_dumm = as.data.frame(dummy.code(bank_subset$Education))
names(bank_dumm) = c("Education_1", "Education_2","Education_3")
no_ed = subset(bank_subset, select=-c(Education))
bank_dumm1=cbind(no_ed,bank_dumm)

bank_dumm1$Personal.Loan = as.factor(bank_dumm1$Personal.Loan)
bank_dumm1$CCAvg = as.integer(bank_dumm1$CCAvg)

# Partitioned data: 50% training, 30% validation, 20% testing
set.seed(111)
train.index = sample(rownames(bank_dumm1), 0.5*dim(bank_dumm1)[1])
set.seed(111)
valid.index = sample(setdiff(rownames(bank_dumm1),train.index), 0.3*dim(bank_dumm1)[1])
test.index = setdiff(rownames(bank_dumm1), union(train.index, valid.index))

# Created data frames for partitioned data
train.df = bank_dumm1[train.index, ]
valid.df = bank_dumm1[valid.index, ]
test.df = bank_dumm1[test.index, ]

# Normalized data
norm.values = preProcess(train.df[, -7], method=c("center", "scale"))
train.df[, -7] = predict(norm.values, train.df[, -7])
valid.df[, -7] = predict(norm.values, valid.df[, -7])
test.df[,-7] = predict(norm.values, test.df[,-7])
```

```r
# Created test, valid, and train variables for each k-NN test
test.knn = knn(train = train.df[,-7],test = test.df[,-7], cl = train.df[,7], k=3)
valid.knn = knn(train = train.df[,-7],test = valid.df[,-7], cl = train.df[,7], k=3)
train.knn = knn(train = train.df[,-7],test = train.df[,-7], cl = train.df[,7], k=3)

confusionMatrix(test.knn, test.df[,7])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 906  34
##          1   4  56
##
##                Accuracy : 0.962
##                  95% CI : (0.9482, 0.973)
##     No Information Rate : 0.91
##     P-Value [Acc > NIR] : 1.116e-10
##
##                   Kappa : 0.727
##
##  Mcnemar's Test P-Value : 2.546e-06
##
##             Sensitivity : 0.9956
##             Specificity : 0.6222
##          Pos Pred Value : 0.9638
##          Neg Pred Value : 0.9333
##              Prevalence : 0.9100
##          Detection Rate : 0.9060
##    Detection Prevalence : 0.9400
##       Balanced Accuracy : 0.8089
##
##        'Positive' Class : 0
##
```

```r
confusionMatrix(valid.knn, valid.df[,7])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1355   49
##          1    3   93
##
##                Accuracy : 0.9653
##                  95% CI : (0.9548, 0.974)
##     No Information Rate : 0.9053
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7634
##
##  Mcnemar's Test P-Value : 4.365e-10
##
```

```
##             Sensitivity : 0.9978
##             Specificity : 0.6549
##          Pos Pred Value : 0.9651
##          Neg Pred Value : 0.9688
##              Prevalence : 0.9053
##          Detection Rate : 0.9033
##    Detection Prevalence : 0.9360
##       Balanced Accuracy : 0.8264
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(train.knn, train.df[,7])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2249   64
##          1    3  184
##
##                Accuracy : 0.9732
##                  95% CI : (0.9661, 0.9792)
##     No Information Rate : 0.9008
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8316
##
##  Mcnemar's Test P-Value : 2.299e-13
##
##             Sensitivity : 0.9987
##             Specificity : 0.7419
##          Pos Pred Value : 0.9723
##          Neg Pred Value : 0.9840
##              Prevalence : 0.9008
##          Detection Rate : 0.8996
##    Detection Prevalence : 0.9252
##       Balanced Accuracy : 0.8703
##
##        'Positive' Class : 0
##
```

The test set was the least accurate at .962, the validation set had a better accuracy at .9653, and the training set had the best accuracy at .9732. These differences exist because all three models are trained with the training data. Since the training model was tested the training data, it was the best fit. It also contained the most data (50%). The validation model tested more data (30%) than the testing model (20%) which explains the accuracy difference between them.