Chase Holland
10/1/23

Question 1

```
> #removing ID and zip
> bank$ID<-NULL
> bank$ZIP.Code<-NULL
> #creating dummy variables
> bank$Education <- as.factor(bank$Education)
> bank$Personal.Loan <- as.factor(bank$Personal.Loan)
> bank$Securities.Account <- as.factor(bank$Securities.Account)
> bank$CD.Account <- as.factor(bank$CD.Account)
> bank$Online <- as.factor(bank$Online)
> bank$CreditCard <- as.factor(bank$CreditCard)
> #splitting training and validation data
> train_index <- createDataPartition(bank$Personal.Loan,p=0.6, list = FALSE)
> train <- bank[train_index, ]
> validation <- bank[-train_index, ]
> #new customer
> new_cust <- data.frame(Age=40, Experience=10, Income=84, Family=2, CCAvg=2,
Education_1=0, Education_2=1, Education_3=0, Mortgage=0, .... [TRUNCATED]
> classification <- knn(train=train[,1:7],test=new_cust[,1:7],cl=train$Person
al.Loan,k=1)

> summary(classification)
0 1
1 0
```

Question 2

```
> #determining k
> set.seed(123)
> fitcontrol<-trainControl(method="repeatedcv",number=3,repeats=2)
> searchGrid=expand.grid(k=1:10)
> knnmodel <- train(Personal.Loan~.,
+                   data = train,
+                   method = 'knn',
+                   tuneGrid = searchGrid,
 .... [TRUNCATED]
> knnmodel
k-Nearest Neighbors
3000 samples
  11 predictor
   2 classes: '0', '1'
No pre-processing
Resampling: Cross-Validated (3 fold, repeated 2 times)
Summary of sample sizes: 2000, 2000, 2000, 2000, 2000, 2000, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.8998333  0.3834427
   2  0.8938333  0.3795006
   3  0.9103333  0.4200501
   4  0.9058333  0.3938384
   5  0.9098333  0.3839552
   6  0.9096667  0.3870453
```

```
   7  0.9105000  0.3716218
   8  0.9081667  0.3543793
   9  0.9093333  0.3428830
  10  0.9086667  0.3253807

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 7.
```

Question 3

```
> confusionMatrix(predictions,validation$Personal.Loan)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1750  129
         1   58   63

               Accuracy : 0.9065
                 95% CI : (0.8929, 0.9189)
    No Information Rate : 0.904
    P-Value [Acc > NIR] : 0.3698

                  Kappa : 0.3547

 Mcnemar's Test P-Value : 3.073e-07

            Sensitivity : 0.9679
            Specificity : 0.3281
         Pos Pred Value : 0.9313
         Neg Pred Value : 0.5207
             Prevalence : 0.9040
         Detection Rate : 0.8750
   Detection Prevalence : 0.9395
      Balanced Accuracy : 0.6480

       'Positive' Class : 0
```

Question 4

```
> #classify using k
>
> cust_prediction=data.frame(Age=40, Experience=10, Income=84, Family=2, CCAv
g=2, Education=1, Mortgage=0, Securities.Acc .... [TRUNCATED]

> ####predict(knnmodel,cust_prediction)
>
```

Question 5

```
> #changing data sizes
> test_size = 0.2
> test_index = createDataPartition(bank$Personal.Loan, p = 0.2, list = FALSE)
> test = bank[train_index,]
> test_index = createDataPartition(bank$Personal.Loan, p=0.2, list = FALSE)
> train_index = createDataPartition(bank$Personal.Loan, p = 0.5, list = FALSE
)
> validation_index = createDataPartition(bank$Personal.Loan, p = 0.3, list =
FALSE)
> trainknn = knn(train=train[,-8], test = train[,-8], cl = train[,8], k =3)
```

```
> testknn = knn(train = train[,-8], test = test[,-8], cl = train[,8], k =3)

> validationknn = knn(train = train[,-8], test = validation[,-8], cl = train[
,8], k =3)
> #confusion matrices

> confusionMatrix(trainknn, train[,8])
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2675  106
         1   37  182

               Accuracy : 0.9523
                 95% CI : (0.9441, 0.9597)
    No Information Rate : 0.904
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6924

 Mcnemar's Test P-Value : 1.297e-08

            Sensitivity : 0.9864
            Specificity : 0.6319
         Pos Pred Value : 0.9619
         Neg Pred Value : 0.8311
             Prevalence : 0.9040
         Detection Rate : 0.8917
   Detection Prevalence : 0.9270
      Balanced Accuracy : 0.8092

       'Positive' Class : 0


> confusionMatrix(testknn, test[,8])
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2675  106
         1   37  182

               Accuracy : 0.9523
                 95% CI : (0.9441, 0.9597)
    No Information Rate : 0.904
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6924

 Mcnemar's Test P-Value : 1.297e-08

            Sensitivity : 0.9864
            Specificity : 0.6319
         Pos Pred Value : 0.9619
         Neg Pred Value : 0.8311
             Prevalence : 0.9040
         Detection Rate : 0.8917
   Detection Prevalence : 0.9270
      Balanced Accuracy : 0.8092

       'Positive' Class : 0
```

```
> confusionMatrix(validationknn, validation[,8])
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1736  121
         1   72   71

               Accuracy : 0.9035
                 95% CI : (0.8897, 0.9161)
    No Information Rate : 0.904
    P-Value [Acc > NIR] : 0.5493170

                  Kappa : 0.3724

 Mcnemar's Test P-Value : 0.0005501

            Sensitivity : 0.9602
            Specificity : 0.3698
         Pos Pred Value : 0.9348
         Neg Pred Value : 0.4965
             Prevalence : 0.9040
         Detection Rate : 0.8680
   Detection Prevalence : 0.9285
      Balanced Accuracy : 0.6650

       'Positive' Class : 0
```

The biggest thing I noticed with these confusion matrices was that the accuracy for test and training were exactly the same. I had some confusion through the assignment, so I am not sure if that is due to an error or more coincidence. Accuracy for the validation set fell from the original data set. This could be because of the smaller sample size in the second example.