Jared Stadden

Assignment 2

1.

With k=1 the model predicts that the customer will not accept a loan (predicted value is 0)

```
+                          "CCAvg"=2,"Mortgage"=0,"Securities.Account"=0,"CD.Account"=0,
+                          "Online"=1,"CreditCard"=1,"Education.1"=0,"Education.2"=1,
+                          "Education.3"=0)
> #normalizing prediction values
> predict1_normalized <- predict(norm_model,predict1)
> #prediction customer with given attributes
> Predicted_Validation_labels <- knn(Train_Predictors,predict1_normalized,cl=Train_labels,k=1)
> head(Predicted_Validation_labels)
[1] 0
Levels: 0 1
> 
```

2.

The optimal k chosen by the train function is k=2

```
> #finding optimal k
> set.seed(123)
> Search_grid <-expand.grid(k=c(2,7,9,15))
> model <- train(Personal.Loan~Age+Experience+Income+Family+CCAvg+Mortgage+
+                  Securities.Account+CD.Account+Online+CreditCard+Education.1+
+                  Education.2+Education.3,data=bank_normalized,method="knn"
+                ,tuneGrid=Search_grid,preProcess='range')
> model
k-Nearest Neighbors

5000 samples
  13 predictor
   2 classes: '0', '1'

Pre-processing: re-scaling to [0, 1] (13)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 5000, 5000, 5000, 5000, 5000, 5000, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   2  0.9556847  0.7161952
   7  0.9521205  0.6673878
   9  0.9499033  0.6444999
  15  0.9427703  0.5743225

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 2.
> 
```

3.

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  2000


                  | Predicted_Validation_labels
Validation_labels |         0 |         1 | Row Total |
------------------|-----------|-----------|-----------|
                0 |      1779 |        29 |      1808 |
                  |     0.984 |     0.016 |     0.904 |
                  |     0.968 |     0.179 |           |
                  |     0.889 |     0.014 |           |
------------------|-----------|-----------|-----------|
                1 |        59 |       133 |       192 |
                  |     0.307 |     0.693 |     0.096 |
                  |     0.032 |     0.821 |           |
                  |     0.029 |     0.066 |           |
------------------|-----------|-----------|-----------|
     Column Total |      1838 |       162 |      2000 |
                  |     0.919 |     0.081 |           |
------------------|-----------|-----------|-----------|
```

Accuracy = (133+1779)/2000 = 0.96

Recall = 133/(133+59) = 0.69

Precision = 133/(133+29) = 0.82

Specificity = 1779/(1779+29) = 0.98

4.

Using the optimal k=2, knn still predicts that the customer will not accept the loan (predicted value = 0)

```
> #Set k from above
> Predicted_Validation_labels <- knn(Train_Predictors,predict1_normalized,cl=Train_labels,k=2)
> head(Predicted_Validation_labels)
[1] 0
Levels: 0 1
> 
```

5.

Confusion matrix for Test data

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1000


             | Predicted_Test_labels2
Test_labels2 |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |       883 |        21 |       904 |
             |     0.977 |     0.023 |     0.904 |
             |     0.968 |     0.239 |           |
             |     0.883 |     0.021 |           |
-------------|-----------|-----------|-----------|
           1 |        29 |        67 |        96 |
             |     0.302 |     0.698 |     0.096 |
             |     0.032 |     0.761 |           |
             |     0.029 |     0.067 |           |
-------------|-----------|-----------|-----------|
Column Total |       912 |        88 |      1000 |
             |     0.912 |     0.088 |           |
-------------|-----------|-----------|-----------|
```

Accuracy = (883+67)/1000 = 0.95

Recall = 67/(67+29) = 0.70

Precision = 67/(67+21) = 0.76

Specificity = 883/(883+21) = 0.98

Confusion matrix for Validation data

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1500


                   | Predicted_Test_labels3
Validation_labels2 |         0 |         1 | Row Total |
-------------------|-----------|-----------|-----------|
                 0 |      1332 |        24 |      1356 |
                   |     0.982 |     0.018 |     0.904 |
                   |     0.965 |     0.202 |           |
                   |     0.888 |     0.016 |           |
-------------------|-----------|-----------|-----------|
                 1 |        49 |        95 |       144 |
                   |     0.340 |     0.660 |     0.096 |
                   |     0.035 |     0.798 |           |
                   |     0.033 |     0.063 |           |
-------------------|-----------|-----------|-----------|
      Column Total |      1381 |       119 |      1500 |
                   |     0.921 |     0.079 |           |
-------------------|-----------|-----------|-----------|
```

Accuracy = (1332+95)/1500 = 0.95

Recall = 95/(95+49) = 0.66

Precision = 95/(95+24) = 0.80

Specificity = 1332/(1332+24) = 0.98

The confusion matrix for the Test data has a smaller sample size than the confusion matrix for the Validation data, but this simply because that is the way the data was portioned: Validation = 30%, Test = 20%. The Accuracy and Specificity of the two confusion matrices are essentially the same. The Recall and Precision are slightly different between the two sets. This is because the Validation set had a slightly lower proportion of False Positives and a slightly higher proportion of False Negatives. The differences are likely just due to noise in the data from the slight differences in the Test and Validation sets. Stratified sampling using the Personal.Loan variable was used to keep the characteristics of the partitioned data sets similar. This helps to ensure a fair assessment of model performance.