mbruner3_2

Mark Bruner

9/22/2020

Libraries needed for this assignment.

```
library(ggplot2)
library(caret)
```

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

```
library(ISLR)
library(readr)
```

Imported dataset.

```
unibank <- read_csv("UniversalBank.csv", col_types = "iiiiiiidiiffff")
head(unibank)
```

```
## # A tibble: 6 x 14
##      ID   Age Experience Income `ZIP Code` Family CCAvg Education Mortgage
##   <int> <int>      <int>  <int>      <int>  <int> <dbl>     <int>    <int>
## 1     1    25          1     49      91107      4   1.6         1        0
## 2     2    45         19     34      90089      3   1.5         1        0
## 3     3    39         15     11      94720      1   1           1        0
## 4     4    35          9    100      94112      1   2.7         2        0
## 5     5    35          8     45      91330      4   1           2        0
## 6     6    37         13     29      92121      4   0.4         2      155
## # ... with 5 more variables: `Personal Loan` <fct>, `Securities Account` <fct>,
## #   `CD Account` <fct>, Online <fct>, CreditCard <fct>
```

Cleaning dataset.

Removed Zip Code and ID columns. Also, made dummy categories for Education and removed the Education column. Lastly, converted dataset into a data frame.

```
library(fastDummies)
unibank <- dummy_cols(unibank, select_columns = 'Education')
unibank <- unibank[, c(-1, -5, -8)]
unibank_df <- data.frame(unibank)
head(unibank_df)
```

```
##   Age Experience Income Family CCAvg Mortgage Personal.Loan Securities.Account
## 1  25          1     49      4   1.6        0             0                   1
## 2  45         19     34      3   1.5        0             0                   1
## 3  39         15     11      1   1.0        0             0                   0
## 4  35          9    100      1   2.7        0             0                   0
## 5  35          8     45      4   1.0        0             0                   0
## 6  37         13     29      4   0.4      155             0                   0
##   CD.Account Online CreditCard Education_1 Education_2 Education_3
## 1          0      0          0           1           0           0
## 2          0      0          0           1           0           0
## 3          0      0          0           1           0           0
## 4          0      0          0           0           1           0
## 5          0      0          1           0           1           0
## 6          0      1          0           0           1           0
```

Partitions

Created partitions for the dataset into Train & Validation set.

```
set.seed(87)
train_index <- createDataPartition(unibank_df$Personal.Loan, p = .6, list = FALSE)
train_data <- unibank_df[train_index, ]
valid_data <- unibank_df[-train_index, ]
```

Customer Data Frame.

Created a data frame for the customer.

```
customer_df <- data.frame(
  "Age" = as.integer(40),
  "Experience" = as.integer(10),
  "Income" = as.integer(84),
  "Family" = as.integer(2),
  "CCAvg" = as.double(2),
  "Mortgage" = as.integer(0),
  "Personal.Loan" = c(0, 1),
  "Securities.Account" = as.factor(0),
  "CD.Account" = as.factor(0),
  "Online" = as.factor(1),
  "CreditCard" = as.factor(1),
  "Education_1" = as.factor(0),
  "Education_2" = as.factor(1),
  "Education_3" = as.factor(0))
head(customer_df)
```

```
##   Age Experience Income Family CCAvg Mortgage Personal.Loan Securities.Account
## 1  40         10     84      2     2        0             0                   0
## 2  40         10     84      2     2        0             1                   0
##   CD.Account Online CreditCard Education_1 Education_2 Education_3
## 1          0      1          1           0           1           0
## 2          0      1          1           0           1           0
```

Copy original data

```
train_norm <- train_data
valid_norm <- valid_data
unibank_norm <- unibank_df
```

Normalization

Used the training data except for the personal loan column to normalize the rest of the train, test, and validation set.

```
norm_values <- preProcess(train_data[, c(1:6)], method = "center", "scale")

train_norm[, c(1:6)] <- predict(norm_values, train_data[,c(1:6)])
valid_norm[, c(1:6)] <- predict(norm_values, valid_data[, c(1:6)])
```

kNN Modeling

```
library(FNN)
knn_valid <- knn(train = train_norm[, c(1:6)], test = valid_norm[, c(1:6)], cl = train_norm[, 7], k = 1
```

Prediction

Combined valid and training.

```
norm_values <- preProcess(train_data[, c(1:6)], method = c("center", "scale"))


unibank_norm[, c(1:6)] <- predict(norm_values, unibank_df[, c(1:6)])
customer_df[, c(1:6)] <- predict(norm_values, customer_df[, c(1:6)])

customer_norm_predictors <- customer_df[, c(1:6)]
unibank_predictors <- unibank_norm[, c(1:6)]

unibank_norm_labels <- unibank_norm[, 7]
customer_test_labels <- customer_df[, 7]
```

#Prediction of Customer

```
predicted_customer_labels <- knn(
                        unibank_predictors,
                        customer_norm_predictors,
                        cl = unibank_norm_labels,
                    k = 1)
head(predicted_customer_labels)
```

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

CrossTable

```
library(gmodels)
CrossTable(x = customer_test_labels, y = predicted_customer_labels, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  2
##
##
##                      | predicted_customer_labels
## customer_test_labels |         0 | Row Total |
## ---------------------|-----------|-----------|
##                    0 |         1 |         1 |
##                      |     0.500 |           |
## ---------------------|-----------|-----------|
##                    1 |         1 |         1 |
##                      |     0.500 |           |
## ---------------------|-----------|-----------|
##         Column Total |         2 |         2 |
## ---------------------|-----------|-----------|
##
##
```

The prediction of "0" above means that the customer would not accept the personal loan.

Created Accuracy Data Frame to hold "K" Values.

```
accuracyTest_df <- data.frame(k = seq(1, 14, 1), accuracy = rep(0, 14))
```

Hypertuning using Validation

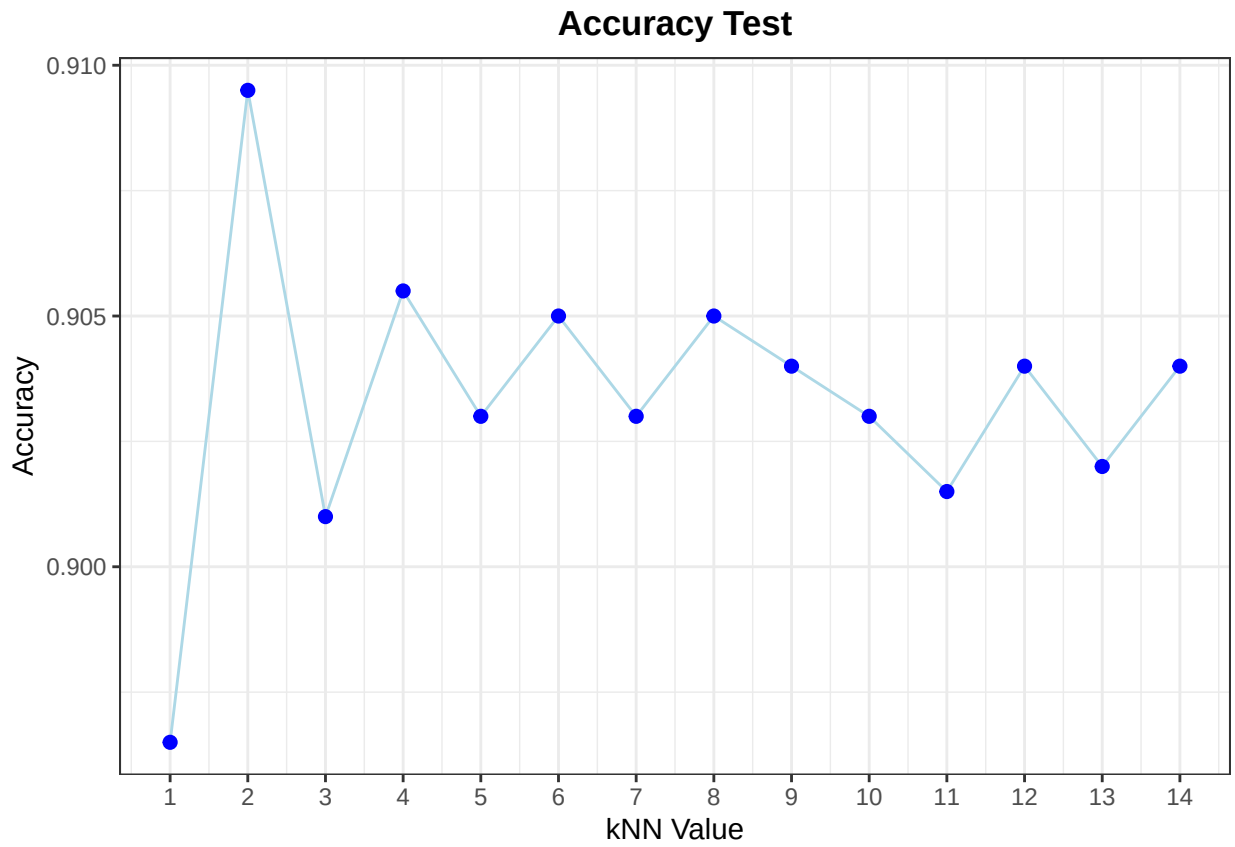
Used an accuracy test for k{i} = 1:14 and a confusion matrix.

```
library(caret)
for(i in 1:14) {
  knn_pred <- knn(
                  train_norm[, c(1:6)],
                  valid_norm[, c(1:6)],
                  cl = train_norm[, 7],
            k = i)
  accuracyTest_df[i, 2] <- confusionMatrix(knn_pred,
                                           valid_norm[, 7])$overall[1]
}

accuracyTest_df
```

```
##     k accuracy
## 1   1   0.8965
## 2   2   0.9095
## 3   3   0.9010
## 4   4   0.9055
## 5   5   0.9030
## 6   6   0.9050
## 7   7   0.9030
## 8   8   0.9050
## 9   9   0.9040
## 10 10   0.9030
## 11 11   0.9015
## 12 12   0.9040
## 13 13   0.9020
## 14 14   0.9040
```

```
library(ggplot2)
ggplot(accuracyTest_df,aes(k,accuracy)) +
      geom_line(colour = "light blue") +
      geom_point(colour = "blue", size = 2) +
      scale_x_continuous(breaks=1:14)+
      theme_bw() +
      ggtitle("Accuracy Test") +
        theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
      xlab("kNN Value") +
      ylab('Accuracy')
```



k = 12 is the value that provide the best performance.

Confusion Matrix and optimal k for Validation

```
knn_valid_optimal <- knn(
                  train_norm[, c(1:6)],
                  test = valid_norm[, c(1:6)],
                  cl = train_norm[, 7],
          k = 12)
confusionMatrix(valid_norm[, 7], knn_valid, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1717   91
##          1  116   76
##
##                Accuracy : 0.8965
##                  95% CI : (0.8823, 0.9095)
##     No Information Rate : 0.9165
##     P-Value [Acc > NIR] : 0.99925
##
##                   Kappa : 0.3668
```

```
##
##  Mcnemar's Test P-Value : 0.09529
##
##             Sensitivity : 0.4551
##             Specificity : 0.9367
##          Pos Pred Value : 0.3958
##          Neg Pred Value : 0.9497
##              Prevalence : 0.0835
##          Detection Rate : 0.0380
##    Detection Prevalence : 0.0960
##       Balanced Accuracy : 0.6959
##
##        'Positive' Class : 1
##
```

```
confusionMatrix(valid_norm[, 7], knn_valid_optimal, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1771   37
##          1  155   37
##
##                Accuracy : 0.904
##                  95% CI : (0.8902, 0.9166)
##     No Information Rate : 0.963
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2375
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5000
##             Specificity : 0.9195
##          Pos Pred Value : 0.1927
##          Neg Pred Value : 0.9795
##              Prevalence : 0.0370
##          Detection Rate : 0.0185
##    Detection Prevalence : 0.0960
##       Balanced Accuracy : 0.7098
##
##        'Positive' Class : 1
##
```

There are two types of errors in the confusion matrix which are: Type 1 and Type 2. The first type of error is a false positive. What this means for this problem is that the model predicted a person as accepting a personal loan but in reality (according to validation data) they rejected it. According to the confusion matrix above, the model predicted 58 (second confusion matrix) people as accepting a personal loan but in reality they rejected it.

The second type of error is a false negative. What this means is that the model predicted a person as not accepting the personal loan but in reality they did accepted it. According to the confusion matrix above,

the model said that 138 people would reject the personal loan but in reality (according to the validation set) they accepted it.

The precision of this model is 96.8% (second confusion matrix), meaning that it correctly predicts a person of accepting a loan 96.8% of the time, which I believe is good. Compared to the kNN model where k = 1 (top confusion matrix) is used, this model predicts correctly predicted 33 more people.

However, the specificity (predicted rejection of loan/actual rejection of loan) is higher in the k = 1 model compared to the k = 12 model. The k =1 model predicted 22 more people who would reject the loan correctly compared to the k = 12 model.

In conclusion, in this specific situation I would think the bank would prefer maximizing the precision of the model over the other metrics. I make this assumption due to believing that the bank would prefer maximizing getting more personal loans from customers (i.e. the models ability to correctly identify those would accept an offer of a personal loan).

Prediction using optimal k.

```
predicted_customer_labels <- knn(
                        unibank_predictors,
                        customer_norm_predictors,
                        cl = unibank_norm_labels,
                      k = 12)
head(predicted_customer_labels)
```

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

CrossTable

```
library(gmodels)
CrossTable(x = customer_test_labels, y = predicted_customer_labels, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  2
##
##
##                      | predicted_customer_labels
## customer_test_labels |         0 | Row Total |
## ---------------------|-----------|-----------|
##                    0 |         1 |         1 |
```

```
##                      |     0.500 |           |
## ---------------------|-----------|-----------|
##                    1 |         1 |         1 |
##                      |     0.500 |           |
## ---------------------|-----------|-----------|
##         Column Total |         2 |         2 |
## ---------------------|-----------|-----------|
##
##
```

The customer would still not accept the loan based on the higher k-value.

Partitions with training, test, and validation

Created partitions for the dataset into Train & Validation set.

```
set.seed(15)
test_index <- createDataPartition(unibank_df$Personal.Loan, p = .2, list = FALSE)
test_data <- unibank_df[test_index, ]
train_valid_data <- unibank_df[-test_index, ]

train_index <- createDataPartition(train_valid_data$Personal.Loan, p = .625, list = FALSE)
train_data <- train_valid_data[train_index, ]
valid_data <- train_valid_data[-train_index, ]
```

Copy original data

```
train_norm <- train_data
valid_norm <- valid_data
train_valid_norm <- train_valid_data
test_norm <- test_data
```

Normalization

Used the training data except for the personal loan column to normalize the rest of the train, test, and validation set.

```
norm_values <- preProcess(train_data[, c(1:6)], method = "center", "scale")

train_norm[, c(1:6)] <- predict(norm_values, train_data[, c(1:6)])
valid_norm[, c(1:6)] <- predict(norm_values, valid_data[, c(1:6)])
train_valid_norm[, c(1:6)] <- predict(norm_values, train_valid_norm[, c(1:6)])

train_norm_predictors <- train_norm[, c(1:6)]
valid_norm_predictors <- valid_norm[, c(1:6)]
```

kNN Modeling

```
library(FNN)
knn_valid <- knn(
                train_norm_predictors,
                valid_norm_predictors,
                cl = train_norm[, 7],
        k = 12)
confusionMatrix(knn_valid, valid_norm[, 7], positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1313  104
##          1   43   40
##
##                Accuracy : 0.902
##                  95% CI : (0.8858, 0.9166)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.6245
##
##                   Kappa : 0.3035
##
##  Mcnemar's Test P-Value : 7.47e-07
##
##             Sensitivity : 0.27778
##             Specificity : 0.96829
##          Pos Pred Value : 0.48193
##          Neg Pred Value : 0.92661
##              Prevalence : 0.09600
##          Detection Rate : 0.02667
##    Detection Prevalence : 0.05533
##       Balanced Accuracy : 0.62303
##
##        'Positive' Class : 1
##
```

Combining sets.

Combined valid and training.

```
norm_values <- preProcess(train_valid_norm[, c(1:6)], method = c("center", "scale"))

train_valid_norm[, c(1:6)] <- predict(norm_values, train_valid_data[, c(1:6)])
test_norm[, c(1:6)] <- predict(norm_values, test_data[, c(1:6)])


train_valid_labels <- unibank_norm[, 7]
test_labels <- customer_df[, 7]
```

Prediction of test set

```
predicted_test_labels <- knn(
                        unibank_predictors,
                        test_norm[, c(1:6)],
                        cl = train_valid_labels,
                    k = 12)
head(predicted_test_labels)
```

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

Confusion Matrix for test set.

```
confusionMatrix(test_norm[, 7], predicted_test_labels, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 609 295
##          1   2  94
##
##                Accuracy : 0.703
##                  95% CI : (0.6736, 0.7312)
##     No Information Rate : 0.611
##     P-Value [Acc > NIR] : 7.746e-10
##
##                   Kappa : 0.2762
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.2416
##             Specificity : 0.9967
##          Pos Pred Value : 0.9792
##          Neg Pred Value : 0.6737
##              Prevalence : 0.3890
##          Detection Rate : 0.0940
##    Detection Prevalence : 0.0960
##       Balanced Accuracy : 0.6192
##
##        'Positive' Class : 1
##
```

Comparing Confusion Matrices.

The accuracy for the train/validation was significantly higher than the test confusion matrix by about 22%. A reason for this may be due to the fact that we did not hypertune the model to make sure we had the

optimal k for the model. We used the optimal k from the previous situation which used different partition sizes compared to this situation.

The specificity is much higher for the test confusion matrix by almost 4%. Interestly enough the model predicted correctly the positive class almost 100% of the time. Where the model performed poorly was that it had said that 331 people would accept the loan but they did not in reality. Comparatively, the train and validation confusion matrix said the model only had 97 false-positives. A reason for this could be due to the fact that the train and validation model for this problem was more similiar to the train and validation model from the previous part.