kNN Bank Subscription Prediction

Sean OMalley

EDA

```
summary(bank)
```

```
##       age                 job          marital         education
##  Min.   :19.00   management :969   divorced: 528   primary  : 678
##  1st Qu.:33.00   blue-collar:946   married :2797   secondary:2306
##  Median :39.00   technician :768   single  :1196   tertiary :1350
##  Mean   :41.17   admin.     :478                   unknown  : 187
##  3rd Qu.:49.00   services   :417
##  Max.   :87.00   retired    :230
##                  (Other)    :713
##  default       balance      housing     loan           contact
##  no :4445   Min.   :-3313   no :1962   no :3830   cellular :2896
##  yes:  76   1st Qu.:   69   yes:2559   yes: 691   telephone: 301
##             Median :  444                         unknown  :1324
##             Mean   : 1423
##             3rd Qu.: 1480
##             Max.   :71188
##
##       day            month         duration       campaign
##  Min.   : 1.00   may    :1398   Min.   :   4   Min.   : 1.000
##  1st Qu.: 9.00   jul    : 706   1st Qu.: 104   1st Qu.: 1.000
##  Median :16.00   aug    : 633   Median : 185   Median : 2.000
##  Mean   :15.92   jun    : 531   Mean   : 264   Mean   : 2.794
##  3rd Qu.:21.00   nov    : 389   3rd Qu.: 329   3rd Qu.: 3.000
##  Max.   :31.00   apr    : 293   Max.   :3025   Max.   :50.000
##                  (Other): 571
##      pdays           previous          poutcome      y
##  Min.   : -1.00   Min.   : 0.0000   failure: 490   no :4000
##  1st Qu.: -1.00   1st Qu.: 0.0000   other  : 197   yes: 521
##  Median : -1.00   Median : 0.0000   success: 129
##  Mean   : 39.77   Mean   : 0.5426   unknown:3705
##  3rd Qu.: -1.00   3rd Qu.: 0.0000
##  Max.   :871.00   Max.   :25.0000
##
```

```
glimpse(bank)
```

```
## Observations: 4,521
## Variables: 17
## $ age       <int> 30, 33, 35, 30, 59, 35, 36, 39, 41, 43, 39, 43, 36, ...
## $ job       <fctr> unemployed, services, management, management, blue-...
## $ marital   <fctr> married, married, single, married, married, single,...
## $ education <fctr> primary, secondary, tertiary, tertiary, secondary, ...
## $ default   <fctr> no, no, no, no, no, no, no, no, no, no, no, no, no,...
## $ balance   <int> 1787, 4789, 1350, 1476, 0, 747, 307, 147, 221, -88, ...
## $ housing   <fctr> no, yes, yes, yes, yes, no, yes, yes, yes, yes, yes...
## $ loan      <fctr> no, yes, no, yes, no, no, no, no, no, yes, no, no, ...
## $ contact   <fctr> cellular, cellular, cellular, unknown, unknown, cel...
## $ day       <int> 19, 11, 16, 3, 5, 23, 14, 6, 14, 17, 20, 17, 13, 30,...
## $ month     <fctr> oct, may, apr, jun, may, feb, may, may, may, apr, m...
## $ duration  <int> 79, 220, 185, 199, 226, 141, 341, 151, 57, 313, 273,...
## $ campaign  <int> 1, 1, 1, 4, 1, 2, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 5, 1...
## $ pdays     <int> -1, 339, 330, -1, -1, 176, 330, -1, -1, 147, -1, -1,...
## $ previous  <int> 0, 4, 1, 0, 0, 3, 2, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 2...
## $ poutcome  <fctr> unknown, failure, failure, unknown, unknown, failur...
## $ y         <fctr> no, no, no, no, no, no, no, no, no, no, no, no, no,...
```

Organize Data Types

There are a ton of character factors in this dataset, and knowing how kNN works, we will need to turn these character factors into numeric factor variables.

```
bank$job <- unclass(bank$job) %>% as.numeric %>% as.factor
bank$marital <- unclass(bank$marital) %>% as.numeric %>% as.factor
bank$education <- unclass(bank$education) %>% as.numeric %>% as.factor
bank$default <- unclass(bank$default) %>% as.numeric %>% as.factor
bank$housing <- unclass(bank$housing) %>% as.numeric %>% as.factor
bank$loan <- unclass(bank$loan) %>% as.numeric %>% as.factor
bank$contact <- unclass(bank$contact) %>% as.numeric %>% as.factor
bank$month <- unclass(bank$month) %>% as.numeric %>% as.factor
bank$poutcome <- unclass(bank$poutcome) %>% as.numeric %>% as.factor
bank$y <- unclass(bank$y) %>% as.numeric %>% as.factor
```

Normalize Numeric Data

The distance calculation for k-NN is heavily dependent upon the measurement scale of the input features. Looking at the summary above, we see that age, balance, duration and pdays each have much larger variance and scale than the other features so we must normalize this data to not give too much weight to these variables.

For this normalization I will create a min/max normalization function of which to normalize the necessary variables.

```
normalize <- function(x) {
                          return((x-min(x)) / (max(x) - min(x)))
             }

bank$age <- normalize(bank$age)
bank$balance <- normalize(bank$balance)
bank$duration <- normalize(bank$duration)
bank$pdays <- normalize(bank$pdays)
bank$day <- normalize(bank$day)
```

EDA 2

This looks much better.

```
summary(bank)
```

```
##       age              job      marital  education default
##  Min.   :0.0000   5      :969   1: 528   1: 678    1:4445
##  1st Qu.:0.2059   2      :946   2:2797   2:2306    2:  76
##  Median :0.2941   10     :768   3:1196   3:1350
##  Mean   :0.3260   1      :478            4: 187
##  3rd Qu.:0.4412   8      :417
##  Max.   :1.0000   6      :230
##                   (Other):713
##     balance        housing  loan     contact       day
##  Min.   :0.00000   1:1962   1:3830   1:2896   Min.   :0.0000
##  1st Qu.:0.04540   2:2559   2: 691   2: 301   1st Qu.:0.2667
##  Median :0.05043                     3:1324   Median :0.5000
##  Mean   :0.06356                              Mean   :0.4972
##  3rd Qu.:0.06433                              3rd Qu.:0.6667
##  Max.   :1.00000                              Max.   :1.0000
##
##      month         duration          campaign          pdays
##  9      :1398   Min.   :0.00000   Min.   : 1.000   Min.   :0.00000
##  6      : 706   1st Qu.:0.03310   1st Qu.: 1.000   1st Qu.:0.00000
##  2      : 633   Median :0.05991   Median : 2.000   Median :0.00000
##  7      : 531   Mean   :0.08605   Mean   : 2.794   Mean   :0.04675
##  10     : 389   3rd Qu.:0.10758   3rd Qu.: 3.000   3rd Qu.:0.00000
##  1      : 293   Max.   :1.00000   Max.   :50.000   Max.   :1.00000
##  (Other): 571
##     previous       poutcome y
##  Min.   : 0.0000   1: 490   1:4000
##  1st Qu.: 0.0000   2: 197   2: 521
##  Median : 0.0000   3: 129
##  Mean   : 0.5426   4:3705
##  3rd Qu.: 0.0000
##  Max.   :25.0000
##
```

```
glimpse(bank)
```

```
## Observations: 4,521
## Variables: 17
## $ age       <dbl> 0.16176471, 0.20588235, 0.23529412, 0.16176471, 0.58...
## $ job       <fctr> 11, 8, 5, 5, 2, 5, 7, 10, 3, 8, 8, 1, 10, 9, 2, 5, ...
## $ marital   <fctr> 2, 2, 3, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, ...
## $ education <fctr> 1, 2, 3, 3, 2, 3, 3, 2, 3, 1, 2, 2, 3, 2, 2, 3, 2, ...
## $ default   <fctr> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ balance   <dbl> 0.06845546, 0.10875022, 0.06258976, 0.06428102, 0.04...
## $ housing   <fctr> 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1, 2, 1, 1, ...
## $ loan      <fctr> 1, 2, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 1, ...
## $ contact   <fctr> 1, 1, 1, 3, 3, 1, 1, 1, 3, 1, 3, 1, 1, 1, 1, 1, 1, ...
## $ day       <dbl> 0.60000000, 0.33333333, 0.50000000, 0.06666667, 0.13...
## $ month     <fctr> 11, 9, 1, 7, 9, 4, 9, 9, 9, 1, 9, 1, 2, 1, 5, 2, 2,...
## $ duration  <dbl> 0.02482622, 0.07149950, 0.05991394, 0.06454816, 0.07...
## $ campaign  <int> 1, 1, 1, 4, 1, 2, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 5, 1...
## $ pdays     <dbl> 0.0000000, 0.3899083, 0.3795872, 0.0000000, 0.000000...
## $ previous  <int> 0, 4, 1, 0, 0, 3, 2, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 2...
## $ poutcome  <fctr> 4, 1, 1, 4, 4, 1, 2, 4, 4, 1, 4, 4, 4, 4, 1, 4, 4, ...
## $ y         <fctr> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, ...
```

Test and Training Data Sets

I created a function that serves as a device to set the seed, randomize the data and split it in a user defined split, in this case 70/30.

```
split.data <- function(data, p = 0.7, s = 777) {
                      set.seed(s)
                      index = sample(1:dim(data)[1])
                      train = data[index[1:floor(dim(data)[1]*p)],]
                      test  = data[index[((ceiling(dim(data)[1]*p))+1):dim(data)[1]],]
                      return(list(train = train, test = test))
                      }

# apply this function to our df4 dataset that included email addresses
churn_splits <- split.data(bank)

# split out hte return of test and train into their own data frames
bank_train <- churn_splits$train

bank_test <- churn_splits$test
```

Apply k-NN Algorithm

The training phase actually involves no model building; the process of training a lazy learner like k-NN simply involves storing the input data in a structured format.

For each instance in the test data, the function will identify the k-Nearest Neighbors, using Euclidean distance, where k is a user-specified number.

Training and classification using the knn() function is performed in a single function call

Build Prediction : $k = 2$

```
bank_predict <- knn(train = bank_train[1:16], test = bank_test[1:16],
                     cl = bank_train[,17], k = 2)
```

Confusion Matrix: Results

```
CrossTable(x = bank_test[,17], y = bank_predict, prop.chisq = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1356
##
##
##                  | bank_predict
## bank_test[, 17] |         1 |         2 | Row Total |
## ----------------|-----------|-----------|-----------|
##               1 |      1105 |        90 |      1195 |
##                 |     0.925 |     0.075 |     0.881 |
##                 |     0.893 |     0.763 |           |
##                 |     0.815 |     0.066 |           |
## ----------------|-----------|-----------|-----------|
##               2 |       133 |        28 |       161 |
##                 |     0.826 |     0.174 |     0.119 |
##                 |     0.107 |     0.237 |           |
##                 |     0.098 |     0.021 |           |
## ----------------|-----------|-----------|-----------|
##    Column Total |      1238 |       118 |      1356 |
##                 |     0.913 |     0.087 |           |
## ----------------|-----------|-----------|-----------|
##
##
```

We see that overall, given a k of two, we can properly predict the subscription status of 85% of the individuals given the rest of their information to predict. Confusion matricies are not entirely optimal when there is a lopsided binary choice, however it does tell us that our model is certainly headed in the right direction, now lets expliore with some diffferent sized k's.

Build Prediction : $k = 5$

```
bank_predict <- knn(train = bank_train[1:16], test = bank_test[1:16],
                    cl = bank_train[,17], k = 5)
```

Confusion Matrix: Results

```
CrossTable(x = bank_test[,17], y = bank_predict, prop.chisq = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1356
##
##
##                  | bank_predict
## bank_test[, 17] |         1 |         2 | Row Total |
## ----------------|-----------|-----------|-----------|
##               1 |      1188 |         7 |      1195 |
##                 |     0.994 |     0.006 |     0.881 |
##                 |     0.888 |     0.389 |           |
##                 |     0.876 |     0.005 |           |
## ----------------|-----------|-----------|-----------|
##               2 |       150 |        11 |       161 |
##                 |     0.932 |     0.068 |     0.119 |
##                 |     0.112 |     0.611 |           |
##                 |     0.111 |     0.008 |           |
## ----------------|-----------|-----------|-----------|
##    Column Total |      1338 |        18 |      1356 |
##                 |     0.987 |     0.013 |           |
## ----------------|-----------|-----------|-----------|
##
##
```

Super interesting, given a k of 5, we see that we have properly predicted 88% of the total, which is better than the previous model where k was 2; however we see a much higher error rate which improperly classifies 'yes' for 'no' in terms of bank subscriptions. This could be troublesome in presenting over-inflated subscription estimates, but will have to be something as determined by the business decision makers.

Build Prediction : $k = 10$

```
bank_predict <- knn(train = bank_train[1:16], test = bank_test[1:16],
                    cl = bank_train[,17], k = 10)
```

Confusion Matrix: Results

```
CrossTable(x = bank_test[,17], y = bank_predict, prop.chisq = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1356
##
##
##                  | bank_predict
## bank_test[, 17] |         1 |         2 | Row Total |
## ----------------|-----------|-----------|-----------|
##               1 |      1194 |         1 |      1195 |
##                 |     0.999 |     0.001 |     0.881 |
##                 |     0.886 |     0.125 |           |
##                 |     0.881 |     0.001 |           |
## ----------------|-----------|-----------|-----------|
##               2 |       154 |         7 |       161 |
##                 |     0.957 |     0.043 |     0.119 |
##                 |     0.114 |     0.875 |           |
##                 |     0.114 |     0.005 |           |
## ----------------|-----------|-----------|-----------|
##    Column Total |      1348 |         8 |      1356 |
##                 |     0.994 |     0.006 |           |
## ----------------|-----------|-----------|-----------|
##
##
```

Once again, as we see with a k of 5, in the k of 10 there is an increased number of false ‘yes’ subscriptions. Nevertheless, we see increased predictive power again, properly classifying 88.6% of the observations. We also only see one false ‘no’ subcription, which is a large decrease from 7 in the k=5 model.

We can see, based on accuracy, a k of 10 has the highest predictive power based purely on the confusion matrix with a predictive power of nearly 88.7% accuracy. As businesspeople we need to then assess which type of model error we are more comfortable with, and if that matters in our actionable decisions from our model insight. We want people to subscribe to our bank service, thus we want to predict if they will or will not be willing to receive a subscription. The repercussions of misclassifying the observation is not better or worse based on error, thus I think that we go with k=10. Taking increased accuracy over all else in subscription prediction.