ETM 538, Winter '19, Project Rough Draft

Introduction and K Nearest Neighbors Model

Jordan Hilton, Andey Nunes, Mengyu Li, Peter Boss

March 6, 2019

Introduction for the Rough Draft

This draft is in 4 pieces because we're still synthesizing everything. The outputs are intended to be read in numerical order. The associated .Rmd files will knit successfully into .html format on machines with current versions of both R Studio and all of the packages we used.

We are analyzing a data set on blood donations. There are 576 observations on 4 input variables, with 1 binary output variable for whether or not a specific person donated blood in a given month (March 2007). Our business question is how to predict whether or not a donation was made in that month. The answer will have clear ramifications for blood donation services, hospitals, and the healthcare system at large.

Our analysis is currently split into 4 models, with a PDF for each one: 1) K Nearest Neighbors, 2) Naive Linear Regression, 3) Decision Tree, and 4) Cross-Validation with Random Forest, including another linear regression. The naive linear regression model returned poor results, but we have included it for comparison purposes, and as an example of a the need and importance of more appropriate analysis.

Further planned work:
+ Consolidating and streamlining the analysis and presentation.
+ Direct comparison of models and final model selection based on lowest error rate.
+ More thorough discussion.
+ Dividing the KNN model into 10 different hold-outs, performing the analysis across each hold-out, and averaging the results. This should give a more robust result than using a single test set.
+ Including graphical output in the KNN model.
+ Deeper exploration of the Random Forest model (expanding the number of trees).

K Nearest Neighbors model

We're going to use the nearest neighbors method to predict blood donations. Let's first load our data, then drop the index column and the result column for our working data:

```
data<-read.csv("projectdata.csv")
head(data)
```

```
##     X Months.since.Last.Donation Number.of.Donations
## 1 619                           2                  50
## 2 664                           0                  13
## 3 441                           1                  16
## 4 160                           2                  20
## 5 358                           1                  24
## 6 335                           4                   4
##   Total.Volume.Donated..c.c.. Months.since.First.Donation
## 1                       12500                          98
## 2                        3250                          28
## 3                        4000                          35
## 4                        5000                          45
## 5                        6000                          77
## 6                        1000                           4
```

```
##   Made.Donation.in.March.2007
## 1                           1
## 2                           1
## 3                           1
## 4                           1
## 5                           0
## 6                           0
```

```
instances<-data[-c(1,6)] ##drop the index column and the result column for working purposes
head(instances)
```

```
##   Months.since.Last.Donation Number.of.Donations
## 1                          2                  50
## 2                          0                  13
## 3                          1                  16
## 4                          2                  20
## 5                          1                  24
## 6                          4                   4
##   Total.Volume.Donated..c.c.. Months.since.First.Donation
## 1                       12500                         98
## 2                        3250                         28
## 3                        4000                         35
## 4                        5000                         45
## 5                        6000                         77
## 6                        1000                          4
```

Now we'll assign weights and scale the data columns by the assigned weights. To start, we just scaled each column by the maximum value.

```
weights<-c(1/74,1/50,1/12500,1/98) ## make a vector of independent variable weights, scaling each by th
names(weights)<-c("monthssincelast", "numberdonations", "totalvolume", "monthssincefirst")  ##name them
scaledinstances<-cbind(instances[,1]*weights[1],instances[,2]*weights[2],instances[,3]*weights[3],instan
## make a scaled version of our instances so that the distances are equivalent
head(scaledinstances)
```

```
##            [,1] [,2] [,3]       [,4]
## [1,] 0.02702703 1.00 1.00 1.00000000
## [2,] 0.00000000 0.26 0.26 0.28571429
## [3,] 0.01351351 0.32 0.32 0.35714286
## [4,] 0.02702703 0.40 0.40 0.45918367
## [5,] 0.01351351 0.48 0.48 0.78571429
## [6,] 0.05405405 0.08 0.08 0.04081633
```

```
scaledinstances[1:10,]
```

```
##             [,1] [,2] [,3]       [,4]
##  [1,] 0.02702703 1.00 1.00 1.00000000
##  [2,] 0.00000000 0.26 0.26 0.28571429
##  [3,] 0.01351351 0.32 0.32 0.35714286
##  [4,] 0.02702703 0.40 0.40 0.45918367
##  [5,] 0.01351351 0.48 0.48 0.78571429
##  [6,] 0.05405405 0.08 0.08 0.04081633
##  [7,] 0.02702703 0.14 0.14 0.14285714
##  [8,] 0.01351351 0.24 0.24 0.35714286
##  [9,] 0.06756757 0.92 0.92 1.00000000
## [10,] 0.00000000 0.06 0.06 0.04081633
```

We'll now divide our working data into two buckets, one for training which we'll label "data", and one for calculating nearest neighbors to assign weights.

```
practicebucket<-scaledinstances[501:576,]
databucket<-scaledinstances[1:500,]
```

Here we built a little function to help us calculate Manhattan distances between two different points.

```
distance<-function(x,y){
  result<-dist(rbind(x,y), method="manhattan") ## find the manhattan distance between two instances
  return(as.numeric(result)) #the result is weirdly vectorized so we just grab the value out of it
}
```

Here's an example of the calculation of the distance between one of our practice points and one of our data points:

```
databucket[1,]
```

```
## [1] 0.02702703 1.00000000 1.00000000 1.00000000
```

```
practicebucket[1,]
```

```
## [1] 0.1891892 0.0800000 0.0800000 0.2653061
```

```
distance(databucket[1,], practicebucket[1,])
```

```
## [1] 2.736856
```

Now we'll create a table where we calculate the distances between each of our practice points and each of the data points. Each column, iterated by j, is a practice point and each row, iterated by j, is a data point.

```
distancesbyrow<-data.frame() #length(data) rows iterated by i for data, length(practice) columns iterat

for(i in 1:length(databucket[,1])){
  for(j in 1:length(practicebucket[,1])){
    distancesbyrow[i,j]<-distance(databucket[i,],practicebucket[j,])  # calculate the distance between
  }
}

head(distancesbyrow[,1:6])
```

```
##          V1        V2        V3        V4        V5        V6
## 1 2.7368560 2.6409046 2.8278764 1.7634749 2.6307005 2.3319581
## 2 0.5695974 0.5144622 0.6606178 0.8362162 0.5246663 0.6136790
## 3 0.7475124 0.6515609 0.8385328 0.6312741 0.6413569 0.6487369
## 4 0.9960397 0.9000883 1.0870601 0.6757198 0.8898842 0.6931826
## 5 1.4960838 1.4001324 1.5871042 0.5227027 1.3899283 1.0911859
## 6 0.3596249 0.3453061 0.3486045 1.3870601 0.3555102 0.7645229
```

Now we create a table where we tabulate the results: which point in the data is nearest the ith practice point, and what is the value of that point for our class variable (whether or not a donation was made). There is some additional fussing about the case where there are multiple nearest points, but as you can see there are no "ambiguous" values, where multiple nearest points have different values for the class variable.

```
distanceresults<-data.frame() #length(practice) rows, one for each test case. first column is minimum d

for(i in 1:length(practicebucket[,1])){ #traversing across our practice data
  distanceresults[i,1]<-min(distancesbyrow[,i]) #the minimum distance for each column of distancesbyrow
  distanceresults[i,2]<-sum(distanceresults[i,1]==distancesbyrow[,i]) #the number of occurences of this
  distanceresults[i,3]<-data[which.min(distancesbyrow[,i]),6] ## pulls the "donation" value for the firs
```

```
  for(j in 1:length(practicebucket[,1])){ ## traversing across the distances for one possible case
    if(distancesbyrow[j,i]==distanceresults[i,1] & data[j,6]!=distanceresults[i,3]){
     distanceresults[i,3]<-"amb"
    } ## if the distance for this case is minimum AND the playvalue is not equal to the first playvalue
  }
}

colnames(distanceresults)<-c("minimumdistance","occurrences", "donation")
head(distanceresults)
```

```
##   minimumdistance occurrences donation
## 1      0.00000000           1        0
## 2      0.01020408           1        0
## 3      0.02040816           1        1
## 4      0.09845560           1        0
## 5      0.02040816           1        0
## 6      0.00000000           1        1
```

```
sum(distanceresults$occurrences==1) # the number of possible classes that have a unique closest neighbo
```

```
## [1] 54
```

```
sum(distanceresults$donation=="amb") # the number of unambiguous results
```

```
## [1] 0
```

Let's compare the results of our training against the real donation values:

```
predictedresults<-distanceresults[,3]
originaldata<-data[501:576,6]
correctanswers<-sum(predictedresults==originaldata)
errorrate<-1-correctanswers/length(originaldata)
errorrate
```

```
## [1] 0.2105263
```

```
a <- c(1:6, NA)
mean(a)
```

```
## [1] NA
```

We have an error rate of 21.1%. Ordinarily, the next step would be to run several adjustments to the weight vector, trying to minimize the error rate of the calculated nearest neighbors. Let's instead graduate to using a real k-nearest neighbors package, in the "class" library.

```
train<-databucket
test<-practicebucket
cl<-data[1:500,6]
libraryresults<-knn(train, test, cl, k=3, prob=TRUE)
libraryresults
```

```
##  [1] 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
## [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [71] 0 0 0 0 0 0
## attr(,"prob")
##  [1] 0.7500000 1.0000000 1.0000000 0.6666667 1.0000000 0.6666667 1.0000000
##  [8] 1.0000000 0.6666667 1.0000000 1.0000000 0.6666667 0.8333333 0.8333333
## [15] 0.8333333 1.0000000 1.0000000 0.6666667 0.6666667 0.6666667 1.0000000
## [22] 0.6666667 0.6666667 1.0000000 1.0000000 0.6666667 0.6666667 1.0000000
```

```
## [29] 0.6666667 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## [36] 1.0000000 1.0000000 1.0000000 1.0000000 0.6666667 1.0000000 1.0000000
## [43] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.6666667 1.0000000
## [50] 0.6666667 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## [57] 1.0000000 1.0000000 0.9090909 0.9090909 0.6666667 0.9090909 0.9090909
## [64] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## [71] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.6666667
## Levels: 0 1
```

Note that we've told R to look at the 3 nearest neighbors, and the library returns (in the first result) the winning classification for each test row, and also the probability of that classification based on the nearest neighbors. Let's calculate the error rate here:

```
predictedresults<-as.numeric(libraryresults)-1
originaldata<-data[501:576,6]
correctanswers<-sum(predictedresults==originaldata)
errorrate<-1-correctanswers/length(originaldata)
errorrate
```

```
## [1] 0.1184211
```

Excellent, the professionals have reduced our error rate by almost half. Let's quickly calculate the predictions using the prebuilt library for the test data set for the contest.

```
testdata<-read.csv("project test data.csv")
testdata<-testdata[-c(1,6)]
testdata<-cbind(testdata[,1]*weights[1],testdata[,2]*weights[2],testdata[,3]*weights[3],testdata[,4]*we
train<-scaledinstances
test<-testdata
cl<-data[,6]
libraryfullresults<-knn(train, test, cl, k=3, prob=TRUE)
head(libraryfullresults)
```

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