Lab 8: Ransom notes keep falling

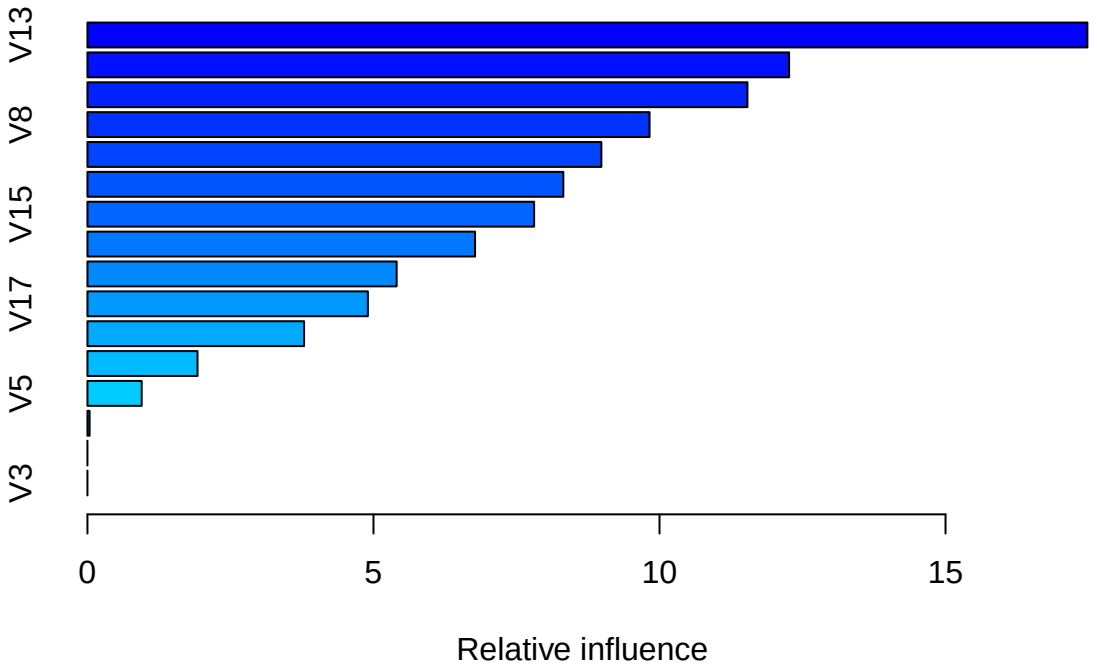
Paul Nguyen

11/14/2019

```
library(ggplot2)
library(dplyr)
library(gbm)
```

```
lettersdf <- read.csv("https://raw.githubusercontent.com/stat-learning/course-materials/master/data/let
                      header = FALSE)
set.seed(1)
train <- sample(1:nrow(lettersdf), nrow(lettersdf) * .75)
trainingdata <- lettersdf[train,]
testdata <- lettersdf[-train,]

boost.tree <- gbm(V1 ~ .-V1, data = trainingdata,
                  distribution = "multinomial",
                  n.trees = 50,
                  shrinkage = .1,
                  interaction.depth = 1)
summary(boost.tree)
```




```
##     var     rel.inf
## V13 V13 17.47975897
## V12 V12 12.26575586
## V14 V14 11.53730240
## V8   V8  9.82580683
## V10 V10  8.98367860
## V11 V11  8.31923619
## V15 V15  7.80833554
```

```
## V9   V9  6.77590463
## V16 V16  5.40537511
## V17 V17  4.90210556
## V7   V7  3.78671390
## V4   V4  1.92421542
## V5   V5  0.94803360
## V6   V6  0.03777739
## V2   V2  0.00000000
## V3   V3  0.00000000
```

```
yhat <- predict(boost.tree, newdata = testdata, n.trees = 50, type = "response")
predicted <- LETTERS[apply(yhat, 1, which.max)]
```

V12 seems to be the most important. (The mean value of the squared horizontal distance times the vertical distance for each "on" pixel. This measures the correlation of the horizontal variance with the vertical position.)

```
confusiontable <- table(predicted, testdata$V1)
confusiontable
```

```
##
## predicted   A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P
##         A 176   0   0   0   0   0   1   0   1   0   2  10   5   0   0   0
##         B   0 129   0  26   5  15   3   7  12  17   2   3   1   5   1   6
##         C   3   0 130   0  26   0  15   0   1   3   7   7   1   3   1   0
##         D   0  20   0 131   0  13   6  10   6   6   4   0   1   4  10  13
##         E   0   0  11   1  72   1   3   0   0   0   5   1   0   0   0   1
##         F   0   0   3   0   0 119   0   1   2   3   0   0   0   0   0  16
##         G   1   2   6   0  22   6 112   4   1   0   4   8   0   0   5   3
##         H   0   0   0   1   0   0   1  82   0   0   4   0   1   1   0   0
##         I   0   0   0   0   0   4   0   0 148   2   0   0   0   0   0   1
##         J   3   0   0   7   0   2   0   1   9 131   0   0   0   1   0   2
##         K   0   1  17   2  10   0   4  13   0   0 108   3   3   0   1   0
##         L   2   0   0   0   0   0   2   0   0   0   1 146   0   0   3   0
##         M   3   7   0   1   0   1   0   3   0   3   6   0 178   8   0   0
##         N   0   2   0   4   1   0   0   5   0   0   5   0   5 157   1   0
##         O   5   1   9   7   0   0   1  28   0   4   0   0   5   8 147  10
##         P   0   0   0   8   0  14   0   0   3   3   0   0   0   9   0 134
##         Q   1   1   1   0   8   0  19   6   1   5   0   2   1   0   5   1
##         R   0  12   0   7   7   3  13   9   2   7  17   2   1   2   2   0
##         S   2   5   4   6   9   5   6   2   2   7   0   3   1   0   0   0
##         T   0   0   0   2   0   6   0   2   0   0   0   0   0   0   0   0
##         U   0   0   2   0   2   1   0  12   0   0   0   0   0   1   2   0
##         V   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         W   0   1   2   0   0   0   6   6   0   0   2   0   4   4  10   9
##         X   5   3   0   0  31   2   0   3   5   0  10   1   0   0   0   0
##         Y   2   0   0   0   2   5   0   0   0   0   2   8   0   5   0   4
##         Z   1   0   1   0  11   0   2   0   0   0   0   0   0   0   0   0
##
## predicted   Q   R   S   T   U   V   W   X   Y   Z
##         A   0   0   8   0   0   0   0   0   0   0
##         B   8  16  13   2   1   0   0   8   1   4
##         C   6   0   0   0   1   0   0   0   0   0
##         D   0   8   5   0   1   0   0   4   4   1
##         E   1   5   2   4   3   0   0   2   0   7
##         F   0   0   2  15   0   4   4   0   5   0
```

```
##         G  14   1   1   0   1   0   0   0   0   1
##         H   0   0   2   0   0   0   0  12   0   0
##         I   0   0   4   5   0   0   0   0   2   0
##         J   2   0   1   0   0   0   0   0   0   2
##         K   0   2   0   2   2   0   0   8   0   1
##         L  11   0   1   0   0   0   0   0   0   0
##         M   1   9   0   0  13   2  13   1   1   0
##         N   0   2   0   5  15   7   4   0   0   0
##         O  16   3   3   3  11   1   4   4   1   0
##         P   0   0   0   1   0   2   0   0   1   0
##         Q  98   1   3   0   3   4   0   0   8   1
##         R   1 152  11   0   0   0   1   1   0   4
##         S   6   0 122   0   0   1   0   4   4  17
##         T   0   0   2 137   2   5   0   0  11   3
##         U   0   0   1   5 145   3   0   0   3   0
##         V   0   0   0  10   5 127   2   0  15   0
##         W   1   3   0   0   1  14 142   0   2   0
##         X   0   3  15   8   0   0   0 128   0   3
##         Y   1   0   1  11   2   4   0   9 120   0
##         Z   0   0   3   2   0   0   0   5   0 133
```

```
misclass.rate <- (sum(confusiontable)-sum(diag(confusiontable))) / sum(confusiontable)
misclass.rate
```

```
## [1] 0.3192
```

```
lettermisclass.rate <-
  (colSums(confusiontable) - diag(confusiontable)) / colSums(confusiontable)
lettermisclass.rate
```

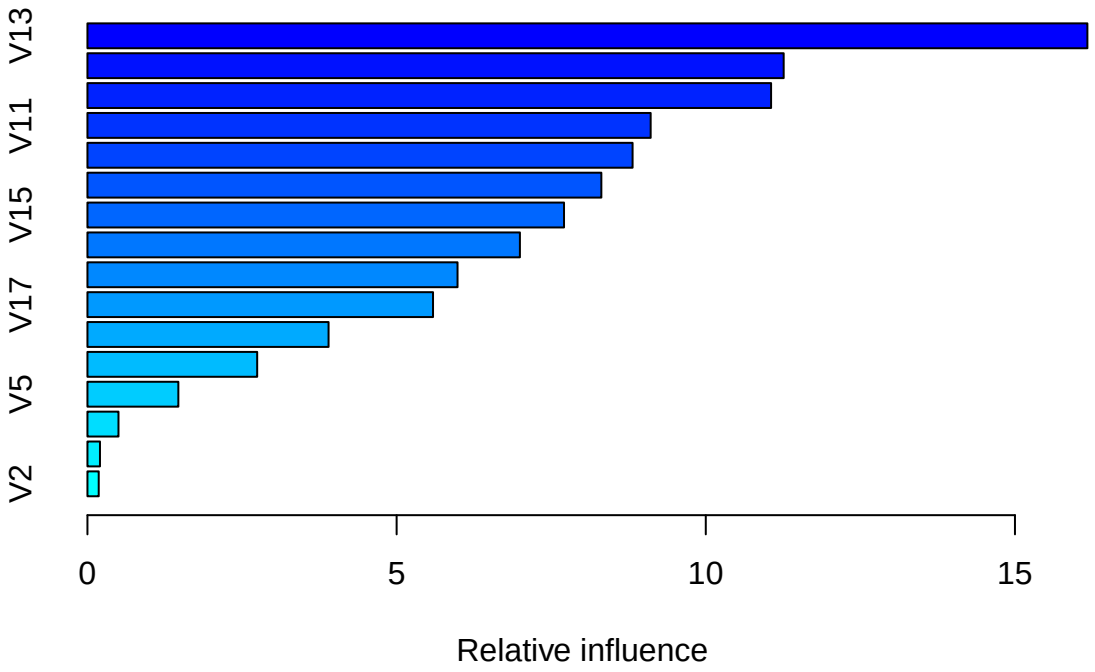
```
##         A         B         C         D         E         F         G
## 0.1372549 0.2989130 0.3010753 0.3546798 0.6504854 0.3959391 0.4226804
##         H         I         J         K         L         M         N
## 0.5773196 0.2331606 0.3141361 0.3966480 0.2474227 0.1400966 0.2451923
##         O         P         Q         R         S         T         U
## 0.2180851 0.3300000 0.4096386 0.2585366 0.3900000 0.3476190 0.2961165
##         V         W         X         Y         Z
## 0.2701149 0.1647059 0.3118280 0.3258427 0.2485876
```

```
max(lettermisclass.rate)
```

```
## [1] 0.6504854
```

misclassification rate is .3142 highest letter being misclassified is E. One pair that the tree found hard to predict was D and B. Also, C and E was not predicted that well either.

```
#even slower tree
slow.boost <- gbm(V1 ~ .-V1, data = trainingdata,
                  distribution = "multinomial",
                  n.trees = 2000,
                  shrinkage = .01,
                  interaction.depth = 1)
summary(slow.boost)
```




```
##     var    rel.inf
## V13 V13 16.1714337
## V14 V14 11.2594609
## V12 V12 11.0561370
## V11 V11  9.1085031
## V8   V8  8.8160159
## V10 V10  8.3108860
## V15 V15  7.7073711
## V9   V9  6.9941397
## V16 V16  5.9850713
## V17 V17  5.5888077
## V7   V7  3.8993516
## V4   V4  2.7452101
## V5   V5  1.4699957
## V6   V6  0.5021764
## V3   V3  0.2028339
## V2   V2  0.1826058
```

```
yhatslow <- predict(slow.boost, newdata = testdata, n.trees = 2000, type = "response")
predictedslow <- LETTERS[apply(yhatslow, 1, which.max)]

slowtable <- table(predictedslow, testdata$V1)
slowtable
```

```
##
## predictedslow   A   B   C   D   E   F   G   H   I   J   K   L   M   N   O
##             A 183   0   0   1   0   0   1   1   0   0   2   1   3   0   0
##             B   0 143   0  16   2   7   3  10   7   3   3   1   2   1   0
##             C   3   0 139   0   5   0  13   0   2   0   1   3   0   0   1
##             D   0   6   0 146   0   2   4   9   4   3   5   0   1   4   7
##             E   0   1   6   0 142   6   1   0   0   0   4   2   0   0   0
##             F   0   1   0   5   0 135   0   3   1   4   0   0   0   0   0
##             G   1   2  11   1  10   6 134   1   0   0   0   5   1   0   3
```

```
##             H   0   2   0   8   0   0   2 130   1   0   7   0   2   0   1
##             I   1   0   0   0   0   7   0   0 164   3   0   0   0   0   0
##             J   1   0   0   4   0   1   0   0   2 154   0   0   0   1   0
##             K   1   1  14   5   8   0   2  11   0   0 138   2   3   1   0
##             L   1   0   0   0   6   0   1   0   2   0   0 167   0   0   3
##             M   2   5   0   0   0   0   0   2   0   0   0   0 184   4   3
##             N   1   0   0   1   0   1   0   0   0   1   3   0   3 175   0
##             O   0   1   5   2   0   0   1   8   0   3   0   0   4   6 154
##             P   0   1   0   4   0  13   0   0   1   4   0   0   0   4   1
##             Q   1   1   0   0   6   0  16   2   0   2   1   4   0   0   5
##             R   0  12   0   4   0   0   7   7   0   3   5   1   1   5   4
##             S   4   3   6   2   9   6   4   0   3   5   0   2   0   0   0
##             T   0   0   0   4   4   6   0   1   0   0   1   1   0   0   0
##             U   0   0   2   0   0   0   1   6   1   1   0   0   1   1   0
##             V   0   3   0   0   0   1   0   0   0   0   1   0   0   1   0
##             W   0   1   0   0   0   1   3   3   0   0   1   0   2   2   6
##             X   3   1   2   0   7   2   0   0   5   2   6   3   0   0   0
##             Y   2   0   0   0   0   3   0   0   0   0   0   2   0   3   0
##             Z   0   0   1   0   7   0   1   0   0   3   1   0   0   0   0
##
## predictedslow   P   Q   R   S   T   U   V   W   X   Y   Z
##             A   0   0   0   9   0   0   0   0   0   0   0
##             B   5   1  13   5   2   0   0   2   3   0   1
##             C   0   3   0   0   0   1   0   0   0   0   0
##             D   1   0   7   3   1   1   1   0   1   3   0
##             E   3   3   4   6   5   0   0   1   1   0   4
##             F  12   0   1   2   8   0   0   0   0   5   0
##             G   3   5   1   2   1   1   0   0   0   0   0
##             H   0   1   1   3   3   0   3   3   1   0   0
##             I   2   0   0   3   0   0   0   0   0   1   0
##             J   1   0   0   0   0   0   0   0   1   0   5
##             K   0   0   2   0   2   1   0   0  13   0   0
##             L   0   8   0   2   0   0   0   0   0   0   2
##             M   0   2   3   0   0  13   0  10   0   0   0
##             N   0   0   1   0   0   6   5   0   0   0   0
##             O   3  11   0   0   1   4   1   2  14   1   0
##             P 157   0   0   0   0   0   4   0   1   2   0
##             Q   0 122   1   3   0   2   1   0   0   5   1
##             R   0   0 165   5   0   0   0   1   1   0   2
##             S   0   4   0 146   3   0   1   0   2   3  12
##             T   0   0   0   4 171   2   2   0   0   2   3
##             U   0   1   0   0   5 170   1   0   1   2   0
##             V   3   0   0   1   0   2 145   0   1  14   0
##             W   3   0   0   0   0   0   6 151   0   0   0
##             X   0   0   6   0   6   1   1   0 142   0   3
##             Y   7   1   0   0   2   2   3   0   4 140   0
##             Z   0   4   0   6   0   0   0   0   0   0 144
```

```
misclass.rateslow <-
  (sum(slowtable) - sum(diag(slowtable))) / sum(slowtable)
misclass.rateslow
```

```
## [1] 0.2118
```

```
lettermisclass.rateslow <-
  (colSums(slowtable) - diag(slowtable)) / colSums(slowtable)
lettermisclass.rateslow
```

```
##         A         B         C         D         E         F         G
## 0.1029412 0.2228261 0.2526882 0.2807882 0.3106796 0.3147208 0.3092784
##         H         I         J         K         L         M         N
## 0.3298969 0.1502591 0.1937173 0.2290503 0.1391753 0.1111111 0.1586538
##         O         P         Q         R         S         T         U
## 0.1808511 0.2150000 0.2650602 0.1951220 0.2700000 0.1857143 0.1747573
##         V         W         X         Y         Z
## 0.1666667 0.1117647 0.2365591 0.2134831 0.1864407
```

```
lettermisclass.rate-lettermisclass.rateslow
```

```
##          A          B          C          D          E          F
## 0.03431373 0.07608696 0.04838710 0.07389163 0.33980583 0.08121827
##          G          H          I          J          K          L
## 0.11340206 0.24742268 0.08290155 0.12041885 0.16759777 0.10824742
##          M          N          O          P          Q          R
## 0.02898551 0.08653846 0.03723404 0.11500000 0.14457831 0.06341463
##          S          T          U          V          W          X
## 0.12000000 0.16190476 0.12135922 0.10344828 0.05294118 0.07526882
##          Y          Z
## 0.11235955 0.06214689
```

my misclassification rate became better when I slowed down the growth of my tree. It seems like my slow model predicts each letter better than my normal boosted tree. Some letters that are particularly good are E and H.

```
#back to communities and crime
crime <- read.csv("http://andrewpbray.github.io/data/crime-train.csv")

dw<-select (crime,-c(state,county,community,communityname,LemasSwornFT,
 LemasSwFTPerPop,LemasSwFTFieldOps,LemasSwFTFieldPerPop,LemasTotalReq,
 LemasTotReqPerPop,PolicReqPerOffic,PolicPerPop,RacialMatchCommPol,
 PctPolicWhite,PctPolicBlack,PctPolicHisp,PctPolicAsian,PctPolicMinor
 ,OfficAssgnDrugUnits,NumKindsDrugsSeiz,PolicAveOTWorked,LandArea,PopDens,
 PctUsePubTrans,PolicCars,PolicOperBudg,LemasPctPolicOnPatr,
 LemasGangUnitDeploy,LemasPctOfficDrugUn,PolicBudgPerPop))

boosted.crime.tree <- gbm(ViolentCrimesPerPop ~ . - ViolentCrimesPerPop,
                          data = dw,
                          distribution = "gaussian",
                          n.trees = 1000,
                          shrinkage = .01,
                          interaction.depth = 1)
test_datacrime <- read.csv("https://bit.ly/2PYS8Ap")
yhatcrime <- predict(boosted.crime.tree, newdata = test_datacrime,
                     n.trees = 1000, type = "response")
boostedmse <- mean((test_datacrime$ViolentCrimesPerPop - yhatcrime)^2)
boostedmse
```

```
## [1] 0.01466922
```

mse for my pruned tree: 0.0216835 bagged tree: 0.003132427 random forest: 0.00309084 my boosted tree's MSE was .01483, which is higher than the rest of my bagged trees. It does beat my normal pruned tree

though.

Problem Set Questions

Ch 8

5) $P(ClassisRed|X) = .1, .15, .2, .2, .55, .6, .6, .65, .7, .75$ Under the majority vote approach, the final classification is: Class is Red, since the majority of bootstrapped trees are greater than .5 When we classify based on average probability, we get P = (.1 + .15 + .2 + .2 + .55 + .6 + .6 + .65 + .7 + .75) / 10 = .45, so we would predict Class is not Red
6) To fit a regression tree, first we look at our predictors and make a split down one of them, that results in the highest nodal purity by using either the GINI index, entropy, deviance, minimizing RSS, or some other metric. We then assign a value to the regions that were split that results in the lowest MSE (if doing a regression),or the majority (if doing a classification), or we can create another split based on nodal purity again. Repeat these steps until stopping criterion, such as each region having 5 observations. This results in a top down greedy tree, since we made the splits only considering what would be the best next step and not looking ahead. Then, we can prune the tree by adding an $\alpha|T|$ term that punishes trees with many nodes. This gives us a pruned tree that is not too complex.