Lab 6 – MATH 243

Theodore Dounias

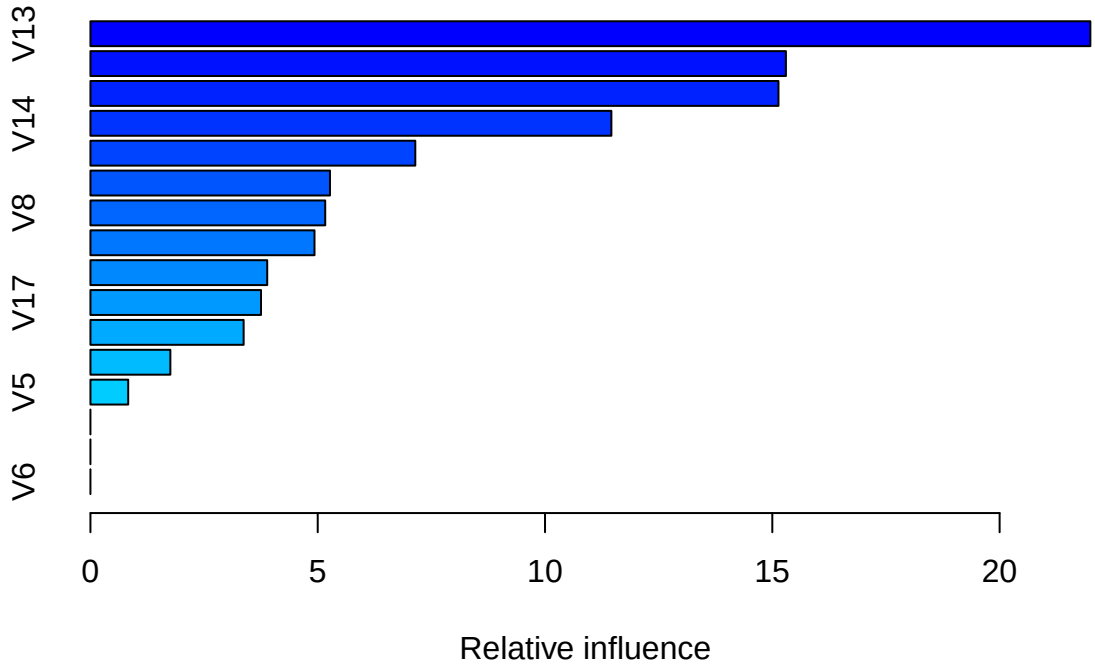
November 6, 2017

Ransom Notes Keep Falling

Build a Boosted Tree

```
boost.letters <- gbm(V1~., data = lettersdf[train,], distribution = "multinomial", n.trees = 50,
                     interaction.depth = 1, shrinkage = 0.1)

summary(boost.letters)
```




```
##     var    rel.inf
## V13 V13 21.9965247
## V11 V11 15.2993528
## V12 V12 15.1356499
## V14 V14 11.4595433
## V9   V9  7.1464605
## V15 V15  5.2694983
## V8   V8  5.1645842
## V10 V10  4.9290784
## V4   V4  3.8886848
## V17 V17  3.7521023
```

```
## V16 V16  3.3698812
## V7   V7  1.7579011
## V5   V5  0.8307385
## V2   V2  0.0000000
## V3   V3  0.0000000
## V6   V6  0.0000000
```

Variable V13 seems to be the most important variable.

Assessing Predictions

```
yhat.boost <- predict(boost.letters, newdata = lettersdf[-train, ], n.trees = 50)

predicted <- LETTERS[apply(yhat.boost, 1, which.max)]

#1
conf_tb <- table(predicted, lettersdf$V1[-train])
conf_tb
```

```
##
## 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
```

```
#2
mcr <- 1 - sum(diag(conf_tb))/5000
mcr
```

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

```
#3
df_conf_pred <- data.frame(pred = predicted)
df_conf_real <- data.frame(real = lettersdf$V1[-train])

df_conf_pred <- df_conf_pred %>%
  group_by(pred) %>%
  summarize(N = n())

df_conf_real <- df_conf_real %>%
  group_by(real) %>%
  summarize(N = n())

df_full <- inner_join(df_conf_pred, df_conf_real, by = c("pred" = "real"))

df_full <- df_full %>%
  mutate(rate = abs(N.x - N.y)/N.y)
```

```
## Warning: package 'bindrcpp' was built under R version 3.3.3
```

```
df_full[df_full$rate == max(df_full$rate), ]
```

```
## # A tibble: 1 x 4
##     pred   N.x   N.y     rate
##   <fctr> <int> <int>    <dbl>
## 1      B   285   184 0.548913
```

A note on problem 3. It is unclear here what being most difficult to predict means. I assumed that, out of the total number of each letter the method was presented with, the worse predicted would be that for which the frequency at which mistakes are made is higher. That is, the letter around which most error happens, which is not necessarily the same as being difficult to predict. B is also the letter for which the absolute number of errors made is highest, if we go with that interpretation.

4. In terms of letter pairs, BD, XE, EC seem to be particularly hard to discern. Also, several other letters in combination with B are hard as well, validating our previous claim about the letter B.

Slow Learning

```
boost.letters.slow <- gbm(V1~., data = lettersdf[train,], distribution = "multinomial", n.trees = 100,
                     interaction.depth = 1, shrinkage = 0.01)

yhat.boost.slow <- predict(boost.letters.slow, newdata = lettersdf[-train, ], n.trees = 100)

predicted.slow <- LETTERS[apply(yhat.boost.slow, 1, which.max)]

conf_tb.slow <- table(predicted.slow, lettersdf$V1[-train])
conf_tb.slow
```

```
##
## predicted.slow   A   B   C   D   E   F   G   H   I   J   K   L   M   N   O
##              A 163   0   0   1   0   0   1   0   3  11   5  67   3   3   0
##              B   0  87   2  16   7  13   2   8   6  19   1   8   0   0   0
##              C  10   0 111   0  38   0  18   0   1  19  12   4   1   3   0
##              D   1  40   0 114   0  45   6   9  10  10   2   0   2  27  19
##              E   0   0   6   0  16   0   0   1   0   0   2   0   0   0   0
##              F   0   0   1   0   0  60   0   0   0   1   0   0   0   0   0
##              G   1   0   9   0  44   4  82  10   0   0  17   7   0   0   2
##              H   0   0   0   3   0   0   1  62   0   0   5   0   0  16   0
##              I   0   0   0   0   0   0   0   0 129   1   0   4   0   0   0
##              J   2   0   0   0   0   6   0   0  18  67   0   0   0   1   0
##              K   2   0  32   1  11   0  11   5   1   0  83  10   3   7   7
##              L   0   0   0   0   0   0   0   0   0   0   0  71   0   0   0
##              M   4   5   0   0   0   1   0   3   0   0   5   1 158   6   0
##              N   4   6   1  10   1   5   6  21   0   2  11   0  23  95   7
##              O   5   3   8   1   0   0  12  25   0   3   0   2   0  10 129
##              P   0   0   0  21   0  25   0   0   4   4   0   0   0  13   0
##              Q   3   2   0   1   5   2  11  21   1   5   1   1   0   0   9
##              R   5  12   0   8  13   0  22   7   4  41  14   6   0   0   3
##              S   3  23   3   9   9   4   8   6   8   8   2   4   1   0   0
##              T   0   0   0   0   0  27   0   3   0   0   0   0   0   3   0
##              U   0   0   0   0   2   0   0   6   0   0   0   0   0   0   1
##              V   0   0   2   0   0   1   0   1   0   0   2   0   0   5   0
##              W   1   5   2   4   0   0   8   5   0   0   6   0  16  19  10
##              X   0   1   3   0  31   2   0   0   8   0   7   0   0   0   0
##              Y   0   0   2   0   2   2   0   1   0   0   4   6   0   0   0
##              Z   0   0   4  14  27   0   6   0   0   0   0   3   0   0   1
##
## predicted.slow   P   Q   R   S   T   U   V   W   X   Y   Z
##              A   0   1   4   8   0   0   0   0   7   0   1
##              B   7   5  19  18   3   0   0   0   7   0   5
##              C   0   6   0   0   0   1   0   0   0   0   0
##              D  21   0  21  13   7   3   0   0   4   9   3
##              E   0   0   0   3   1   4   0   0   0   0   0
##              F   2   0   0   0   5   0   0   5   0   1   0
##              G   2  27   1   1   0   2   0   0   0   0   0
##              H   0   0   0   0   2   4   0   0  38   0   0
##              I   0   0   0   1   6   0   0   0   1   4   0
##              J   7   1   0   7   1   0   0   0   0   0   1
```

```
##              K   0  14   3   0   2  13   0   0   3   0   0
##              L   0   0   0   0   0   0   0   0   0   0   0
##              M   0   3  12   0   0  12   1  11   1   1   0
##              N   0   2  13   2   6  22  11  14   0   3   0
##              O  15  41  16  10   6  16   0   1   4   2   0
##              P 124   0   0   0  39   0   1   0   0   3   0
##              Q   3  46   0  12   2   4   0   0   4   7   3
##              R   0  11 105   6   0   0   0   0   2   0   2
##              S   0   7   1  77   1   0   1   3   6   5  21
##              T   5   0   2   0  82   0  10  20   0   7   3
##              U   0   0   0   4   0  65   5   0   0   0   0
##              V   0   0   0   1   6  42  85   5  10  22   0
##              W   9   2   8   0   0   2   8 101   0   1   0
##              X   0   0   0  24   7  15   0   0  55   5  10
##              Y   5   0   0   1  30   0  52  10   4 107   0
##              Z   0   0   0  12   4   1   0   0  40   1 128
```

```
mcr.slow <- 1 - sum(diag(conf_tb.slow))/5000
mcr.slow
```

```
## [1] 0.5196
```

Several letter pairs became much harder to predict, including RB, BS, ND, NV, OH and many others. This does not seem to have solved many issues relating to the previous pairs, and as the mcr shows is a much worse model. This is probably because we do not have a high enough B to compensate for a lamda that is ten times smaller.

Communities and Crime

Growing a Random Forest

```
crime_test <- crime_test %>%
  select(-(1:4), -(101:126))

crime_train <- crime_train %>%
  select(-(1:4), -(101:126))

bag.crimes <- randomForest(ViolentCrimesPerPop~., data = crime_train, mtry = 96, importance = TRUE)

yhat.bag <- predict(bag.crimes, newdata = crime_test)

mse.bag <- mean((yhat.bag - crime_test$ViolentCrimesPerPop)^2)

mse.bag
```

```
## [1] 0.017121
```

```
rforest.crimes <- randomForest(ViolentCrimesPerPop~., data = crime_train, importance = TRUE)

yhat.rforest <- predict(rforest.crimes, newdata = crime_test)

mse.rforest <- mean((yhat.rforest - crime_test$ViolentCrimesPerPop)^2)

mse.rforest
```

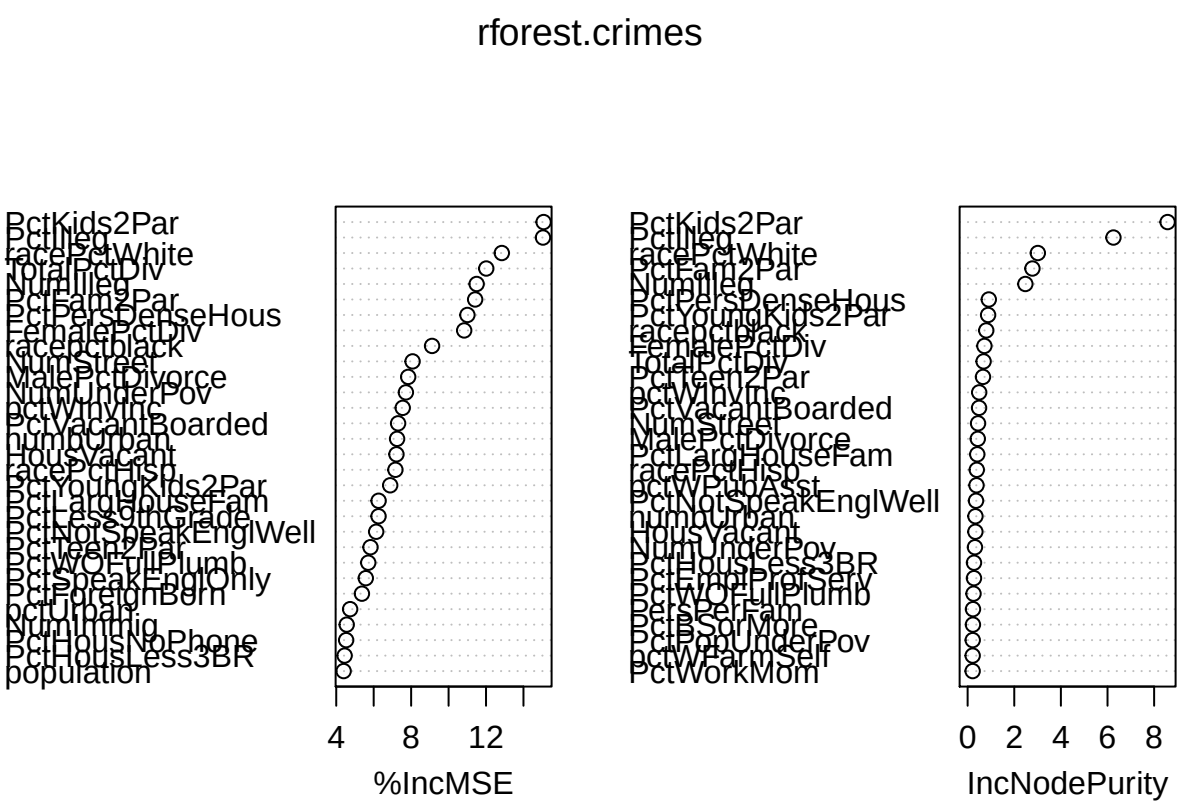
```
## [1] 0.01705606
```

The random forest test MSE is lower than that of the simple bagging method, which was to be expected. Both are lower than the test MSE that our group had.

Variance Importance

```
#I will construct this only for the more efficient model, which is the random forest

varImpPlot(rforest.crimes)
```



With this relatively more interpretable plot, it seems like the best variables to use are the percentage of undocumented individuals, the percentage of children with two parents, the percentage of families with two parents, the percentage of white individuals, and the number of undocumented individuals. These are all variables that where included in our model, maybe perhaps without some redundancies that, at least here, seem important to the model.

One Last Boost

```
boost.crime <- gbm(ViolentCrimesPerPop~., data = crime_train, distribution = "gaussian",
                   n.trees = 5000, interaction.depth = 1, shrinkage = 0.01)

yhat.boost.crime <- predict(boost.crime, newdata = crime_test, n.trees = 1000)

mse.boost <- mean((yhat.boost.crime - crime_test$ViolentCrimesPerPop)^2)

mse.boost
```

```
## [1] 0.01674509
```

The MSE here is better than in all previous models.