Intrusion Detection with Machine Learning

Laine Beatty

5/7/2022

Contents

DA 6813: Data Analytics Applications

Libraries

```
library(car) #vif
library(dplyr)
library(ROCR)
library(caret)
library(tree)
library(h2o)
library(lime)
```

Data Processing

```
flows <- read.csv('BotNeTIoT-L01_label_NoDuplicates.csv')
```

```
str(flows)
```

```
## 'data.frame':    2426574 obs. of  25 variables:
##  $ X                   : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ MI_dir_L0.1_weight  : num  1 1.93 2.9 3.9 4.9 ...
##  $ MI_dir_L0.1_mean    : num  98 98 87 83.7 81.7 ...
##  $ MI_dir_L0.1_variance: num  0.00 1.82e-12 2.31e+02 2.04e+02 1.78e+02 ...
##  $ H_L0.1_weight       : num  1 1.93 2.9 3.9 4.9 ...
##  $ H_L0.1_mean         : num  98 98 87 83.7 81.7 ...
##  $ H_L0.1_variance     : num  0.00 1.82e-12 2.31e+02 2.04e+02 1.78e+02 ...
```

```
##  $ HH_L0.1_weight      : num  1 1.93 1 1 2 ...
##  $ HH_L0.1_mean        : num  98 98 66 74 74 ...
##  $ HH_L0.1_std         : num  0.00 1.35e-06 0.00 0.00 9.54e-07 ...
##  $ HH_L0.1_magnitude   : num  98 139 115 74 74 ...
##  $ HH_L0.1_radius      : num  0.00 1.82e-12 0.00 0.00 9.09e-13 ...
##  $ HH_L0.1_covariance  : num  0 0 0 0 0 ...
##  $ HH_L0.1_pcc         : num  0 0 0 0 0 ...
##  $ HH_jit_L0.1_weight  : num  1 1.93 1 1 2 ...
##  $ HH_jit_L0.1_mean    : num  1.51e+09 7.26e+08 1.51e+09 1.51e+09 7.53e+08 ...
##  $ HH_jit_L0.1_variance: num  0.00 5.66e+17 0.00 0.00 5.67e+17 ...
##  $ HpHp_L0.1_weight    : num  1 1.93 1 1 1 ...
##  $ HpHp_L0.1_mean      : num  98 98 66 74 74 74 74 74 74 74 ...
##  $ HpHp_L0.1_std       : num  0.00 1.35e-06 0.00 0.00 0.00 ...
##  $ HpHp_L0.1_magnitude : num  98 139 115 74 74 ...
##  $ HpHp_L0.1_radius    : num  0.00 1.82e-12 0.00 0.00 0.00 ...
##  $ HpHp_L0.1_covariance: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ HpHp_L0.1_pcc       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ label               : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
colSums(is.na(flows))
```

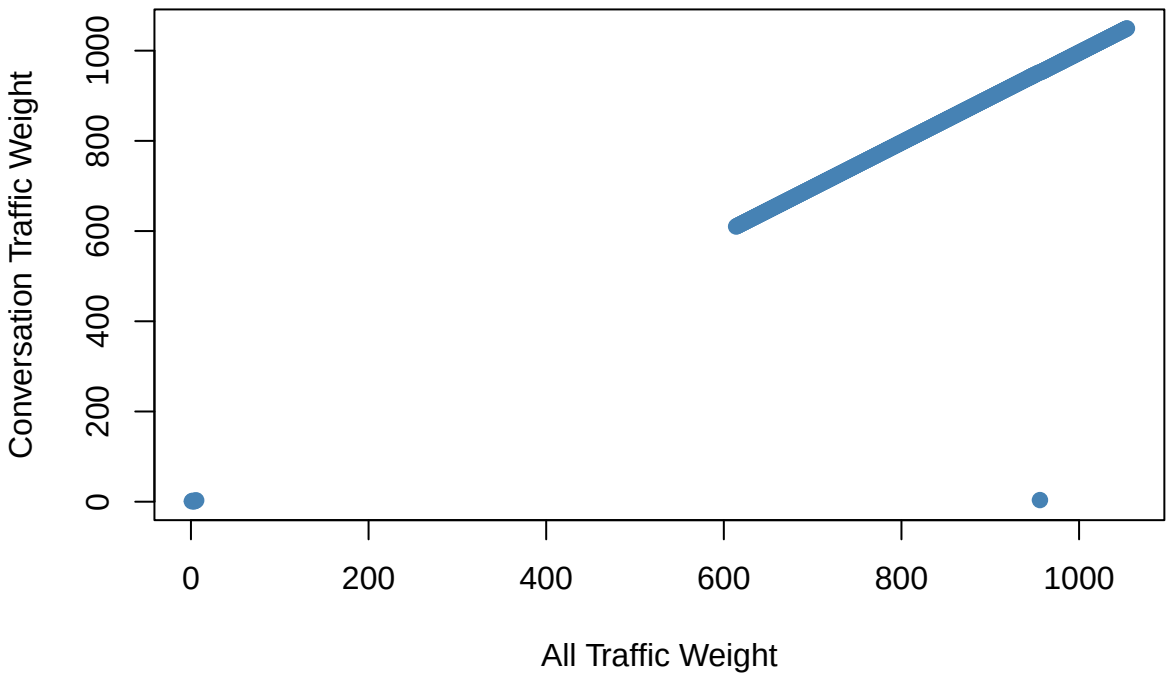
```
##                     X   MI_dir_L0.1_weight     MI_dir_L0.1_mean
##                     0                    0                    0
## MI_dir_L0.1_variance        H_L0.1_weight          H_L0.1_mean
##                     0                    0                    0
##      H_L0.1_variance       HH_L0.1_weight         HH_L0.1_mean
##                     0                    0                    0
##          HH_L0.1_std    HH_L0.1_magnitude       HH_L0.1_radius
##                     0                    0                    0
##   HH_L0.1_covariance          HH_L0.1_pcc   HH_jit_L0.1_weight
##                     0                    0                    0
##     HH_jit_L0.1_mean HH_jit_L0.1_variance     HpHp_L0.1_weight
##                     0                    0                    0
##       HpHp_L0.1_mean        HpHp_L0.1_std  HpHp_L0.1_magnitude
##                     0                    0                    0
##     HpHp_L0.1_radius HpHp_L0.1_covariance        HpHp_L0.1_pcc
##                     0                    0                    0
##                label
##                    0
```

```
table(flows$label)
```

```
##
##       0       1
## 1913077  513497
```

```
flows$label <- as.factor(flows$label)
```

```
plot(flows[1:500,"H_L0.1_weight"],flows[1:500,"HH_L0.1_weight"],
     col='steelblue', pch=19,
     xlab="All Traffic Weight", ylab="Conversation Traffic Weight")
```



Training and Testing Split

```
set.seed(1)
trainID <- sample(nrow(flows), .7*nrow(flows), replace=FALSE)
flow.train <- flows[trainID,c(2:25)]
flow.test <- flows[-trainID,c(2:25)]
```

```
flows$splitsets <- rep("test",nrow(flows))
flows[trainID,"splitsets"] <- "train"
```

```
table(flows$splitsets,flows$label)
```

```
##
##                0       1
##   test    574074  153899
##   train  1339003  359598
```

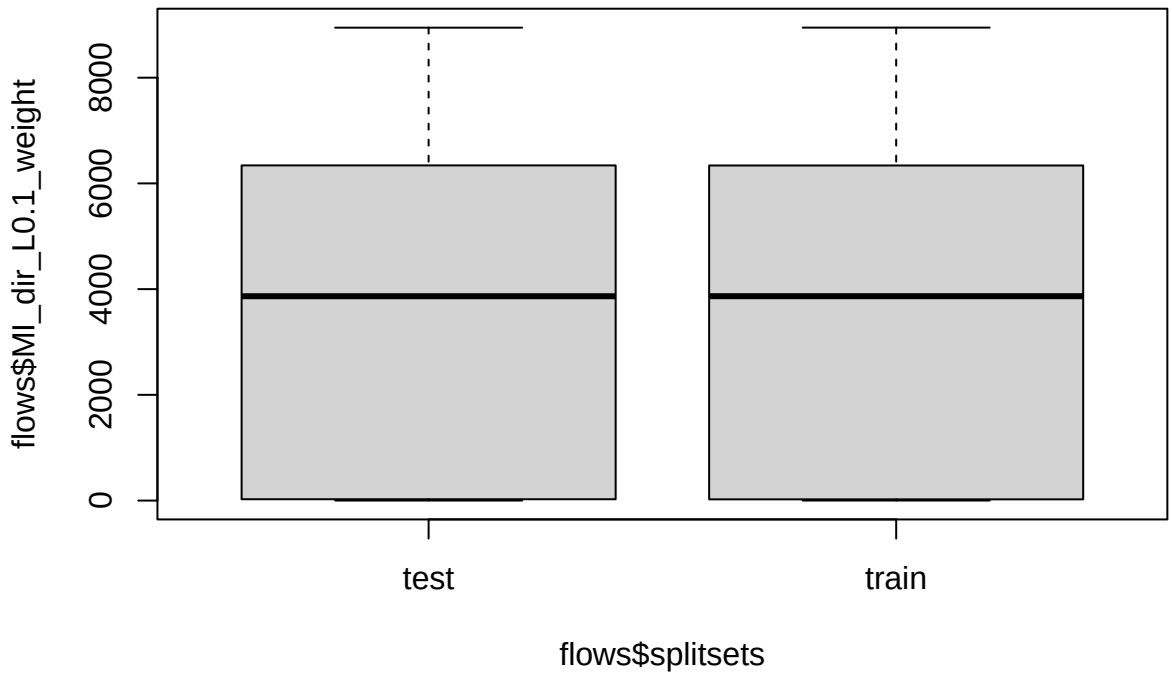
```
153899 / nrow(flow.test)
```

```
## [1] 0.2114076
```

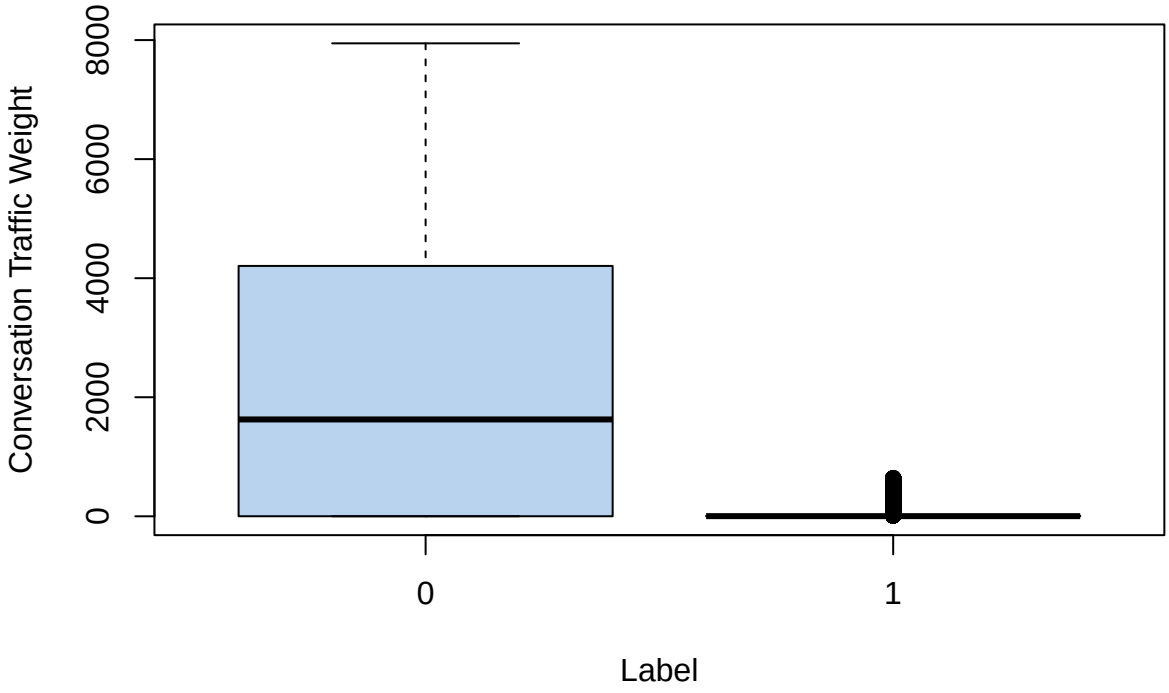
```r
359598 / nrow(flow.train)
```

```
## [1] 0.2117025
```

```r
boxplot(flows$MI_dir_L0.1_weight ~ flows$splitsets)
```




```r
boxplot(flows$HH_L0.1_weight ~ flows$label, col='slategray2',
        xlab='Label', ylab='Conversation Traffic Weight')
```



Logistic Regression

```
log.all <- glm(label ~ ., data=flow.train, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(log.all)
```

```
##
## Call:
## glm(formula = label ~ ., family = binomial, data = flow.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.898   0.000   0.000   0.000   7.042
##
## Coefficients: (1 not defined because of singularities)
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)            7.569e+00  1.640e-01  46.162  < 2e-16 ***
## MI_dir_L0.1_weight    -1.547e+02  2.020e+04  -0.008  0.99389
## MI_dir_L0.1_mean       5.462e-01  1.199e+03   0.000  0.99964
## MI_dir_L0.1_variance -1.771e-02  5.395e+00  -0.003  0.99738
## H_L0.1_weight          1.547e+02  2.020e+04   0.008  0.99389
```

```
## H_L0.1_mean          -5.222e-01  1.199e+03   0.000  0.99965
## H_L0.1_variance       1.768e-02  5.395e+00   0.003  0.99739
## HH_L0.1_weight       -5.374e-02  4.053e-03 -13.260  < 2e-16 ***
## HH_L0.1_mean         -1.256e-02  4.095e-03  -3.067  0.00216 **
## HH_L0.1_std           1.846e-02  2.812e-03   6.562 5.30e-11 ***
## HH_L0.1_magnitude     6.316e-03  1.971e-03   3.205  0.00135 **
## HH_L0.1_radius        2.609e-06  3.804e-06   0.686  0.49277
## HH_L0.1_covariance    2.630e-05  1.713e-05   1.535  0.12471
## HH_L0.1_pcc          -4.996e+00  2.706e-01 -18.465  < 2e-16 ***
## HH_jit_L0.1_weight           NA         NA      NA       NA
## HH_jit_L0.1_mean     -9.144e-09  9.572e-11 -95.534  < 2e-16 ***
## HH_jit_L0.1_variance -2.515e-18  3.117e-19  -8.070 7.04e-16 ***
## HpHp_L0.1_weight      6.882e-02  4.150e-03  16.586  < 2e-16 ***
## HpHp_L0.1_mean        6.270e-03  3.886e-03   1.613  0.10664
## HpHp_L0.1_std         2.774e-03  3.703e-03   0.749  0.45381
## HpHp_L0.1_magnitude  -1.251e-02  2.127e-03  -5.882 4.06e-09 ***
## HpHp_L0.1_radius      1.829e-05  7.019e-06   2.606  0.00915 **
## HpHp_L0.1_covariance  2.859e-05  3.885e-05   0.736  0.46181
## HpHp_L0.1_pcc         5.640e+00  6.221e-01   9.067  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1753647.8  on 1698600  degrees of freedom
## Residual deviance:    8410.2  on 1698578  degrees of freedom
## AIC: 8456.2
##
## Number of Fisher Scoring iterations: 20
```

```
flow.train <- flow.train[,c(7:13,15:24)] # remove MI, H, HH_jit_weight
```

```
m1.log <- glm(label ~ ., data=flow.train, family=binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(m1.log)
```

```
##
## Call:
## glm(formula = label ~ ., family = binomial, data = flow.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.0382   0.0000   0.0000   5.2894
##
## Coefficients:
##                       Estimate Std. Error  z value Pr(>|z|)
## (Intercept)           3.905e+00  3.103e-02  125.847  < 2e-16 ***
## HH_L0.1_weight       -4.197e-02  2.185e-03  -19.211  < 2e-16 ***
```

```
## HH_L0.1_mean         -1.619e-03  1.278e-03   -1.267 0.205079
## HH_L0.1_std           2.110e-02  1.039e-03   20.306  < 2e-16 ***
## HH_L0.1_magnitude     1.596e-02  7.074e-04   22.562  < 2e-16 ***
## HH_L0.1_radius       -2.414e-05  1.124e-06  -21.481  < 2e-16 ***
## HH_L0.1_covariance    6.152e-05  5.154e-06   11.936  < 2e-16 ***
## HH_L0.1_pcc          -2.046e+00  7.926e-02  -25.811  < 2e-16 ***
## HH_jit_L0.1_mean     -7.323e-09  3.604e-11 -203.183  < 2e-16 ***
## HH_jit_L0.1_variance -1.194e-17  1.654e-19  -72.219  < 2e-16 ***
## HpHp_L0.1_weight      3.197e-02  2.188e-03   14.613  < 2e-16 ***
## HpHp_L0.1_mean       -1.867e-02  1.196e-03  -15.618  < 2e-16 ***
## HpHp_L0.1_std         5.336e-02  2.132e-03   25.026  < 2e-16 ***
## HpHp_L0.1_magnitude   3.032e-03  7.045e-04    4.304 1.68e-05 ***
## HpHp_L0.1_radius      8.849e-06  2.339e-06    3.783 0.000155 ***
## HpHp_L0.1_covariance -3.249e-04  1.682e-05  -19.319  < 2e-16 ***
## HpHp_L0.1_pcc         4.036e+00  1.822e-01   22.145  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1753648  on 1698600  degrees of freedom
## Residual deviance:   49218  on 1698584  degrees of freedom
## AIC: 49252
##
## Number of Fisher Scoring iterations: 25
```

```
# library(car)
vif(m1.log)
```

```
##       HH_L0.1_weight         HH_L0.1_mean          HH_L0.1_std
##           146.059658            94.419594             3.543807
##    HH_L0.1_magnitude       HH_L0.1_radius   HH_L0.1_covariance
##            35.240122             7.786647             4.007728
##          HH_L0.1_pcc     HH_jit_L0.1_mean HH_jit_L0.1_variance
##             1.613501             1.198723             1.086807
##     HpHp_L0.1_weight       HpHp_L0.1_mean        HpHp_L0.1_std
##           144.529364            84.432359             2.255799
##  HpHp_L0.1_magnitude     HpHp_L0.1_radius HpHp_L0.1_covariance
##            39.731111            19.158866            15.283090
##        HpHp_L0.1_pcc
##             2.255684
```

```
flow.train <- flow.train[,c(2:17)] # remove HH_L0.1_weight
```

```
m2.log <- glm(label ~ ., data=flow.train, family=binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(m2.log)
```

```
##
## Call:
## glm(formula = label ~ ., family = binomial, data = flow.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
##  -8.49     0.00     0.00     0.00     8.49
##
## Coefficients:
##                          Estimate Std. Error    z value Pr(>|z|)
## (Intercept)          -2.554e+14  1.253e+05 -2.039e+09   <2e-16 ***
## HH_L0.1_mean         -9.418e+12  5.704e+03 -1.651e+09   <2e-16 ***
## HH_L0.1_std           2.160e+13  4.055e+03  5.328e+09   <2e-16 ***
## HH_L0.1_magnitude     1.445e+13  3.303e+03  4.374e+09   <2e-16 ***
## HH_L0.1_radius       -2.313e+10  6.118e+00 -3.781e+09   <2e-16 ***
## HH_L0.1_covariance    5.245e+10  1.960e+01  2.676e+09   <2e-16 ***
## HH_L0.1_pcc          -3.446e+14  6.786e+05 -5.078e+08   <2e-16 ***
## HH_jit_L0.1_mean     -2.743e+06  9.311e-05 -2.946e+10   <2e-16 ***
## HH_jit_L0.1_variance  1.109e-03  1.165e-12  9.521e+08   <2e-16 ***
## HpHp_L0.1_weight     -9.593e+11  7.973e+01 -1.203e+10   <2e-16 ***
## HpHp_L0.1_mean       -1.604e+13  5.476e+03 -2.929e+09   <2e-16 ***
## HpHp_L0.1_std         1.887e+13  4.506e+03  4.187e+09   <2e-16 ***
## HpHp_L0.1_magnitude   1.011e+13  3.397e+03  2.977e+09   <2e-16 ***
## HpHp_L0.1_radius     -1.393e+10  6.692e+00 -2.082e+09   <2e-16 ***
## HpHp_L0.1_covariance -6.457e+10  2.856e+01 -2.260e+09   <2e-16 ***
## HpHp_L0.1_pcc         4.856e+14  1.100e+06  4.413e+08   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1753648  on 1698600  degrees of freedom
## Residual deviance: 3670253  on 1698585  degrees of freedom
## AIC: 3670285
##
## Number of Fisher Scoring iterations: 25
```

```r
vif(m2.log)
```

```
##         HH_L0.1_mean          HH_L0.1_std    HH_L0.1_magnitude
##           603.964481             7.690900           203.387236
##       HH_L0.1_radius   HH_L0.1_covariance          HH_L0.1_pcc
##            10.907584             1.685328             1.563489
##     HH_jit_L0.1_mean HH_jit_L0.1_variance     HpHp_L0.1_weight
##             1.715320             1.008404             1.146243
##       HpHp_L0.1_mean        HpHp_L0.1_std  HpHp_L0.1_magnitude
##           559.291107             7.907502           218.082253
##     HpHp_L0.1_radius HpHp_L0.1_covariance        HpHp_L0.1_pcc
##             8.799998             2.969130             2.794900
```

```
flow.train <- flow.train[,c(2:16)] # remove HH_L0.1_mean

m3.log <- glm(label ~ ., data=flow.train, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
vif(m3.log)
```

```
##          HH_L0.1_std    HH_L0.1_magnitude       HH_L0.1_radius
##             2.180039            29.490695             8.024036
##   HH_L0.1_covariance          HH_L0.1_pcc     HH_jit_L0.1_mean
##             2.351669             1.898378             1.057401
## HH_jit_L0.1_variance     HpHp_L0.1_weight       HpHp_L0.1_mean
##             1.054300             1.054632            30.011334
##        HpHp_L0.1_std  HpHp_L0.1_magnitude     HpHp_L0.1_radius
##             2.237328            39.194488            15.805621
## HpHp_L0.1_covariance        HpHp_L0.1_pcc
##            14.162071             1.842132
```

```
flow.train <- flow.train[,c(1:10,12:15)] # remove HpHp_L0.1_magnitude

m4.log <- glm(label ~ ., data=flow.train, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
vif(m4.log)
```

```
##          HH_L0.1_std    HH_L0.1_magnitude       HH_L0.1_radius
##             1.693004            22.029105             3.393087
##   HH_L0.1_covariance          HH_L0.1_pcc     HH_jit_L0.1_mean
##             2.680138             1.926141             1.055693
## HH_jit_L0.1_variance     HpHp_L0.1_weight       HpHp_L0.1_mean
##             1.048206             1.044956            23.335232
##        HpHp_L0.1_std     HpHp_L0.1_radius HpHp_L0.1_covariance
##             2.106243            60.205477            58.259061
##        HpHp_L0.1_pcc
##             1.840668
```

```
flow.train <- flow.train[,c(1:10,12:14)] # remove HpHp_L0.1_radius

m5.log <- glm(label ~ ., data=flow.train, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
vif(m5.log)
```

```
##          HH_L0.1_std    HH_L0.1_magnitude       HH_L0.1_radius
##             1.684650            21.978924            10.877353
##   HH_L0.1_covariance          HH_L0.1_pcc     HH_jit_L0.1_mean
```

```
##             2.509630             1.933227             1.055577
## HH_jit_L0.1_variance     HpHp_L0.1_weight       HpHp_L0.1_mean
##             1.047875             1.044284            23.146947
##        HpHp_L0.1_std HpHp_L0.1_covariance        HpHp_L0.1_pcc
##             2.233950            12.658504             1.931129
```

```
flow.train <- flow.train[,c(1:8,10:13)] # remove HpHp_L0.1_mean

m6.log <- glm(label ~ ., data=flow.train, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
vif(m6.log)
```

```
##          HH_L0.1_std    HH_L0.1_magnitude       HH_L0.1_radius
##             1.356916             1.292001            22.551546
##   HH_L0.1_covariance          HH_L0.1_pcc     HH_jit_L0.1_mean
##            20.359832             1.344314             1.050237
## HH_jit_L0.1_variance     HpHp_L0.1_weight        HpHp_L0.1_std
##             1.055158             1.027623             2.800549
## HpHp_L0.1_covariance        HpHp_L0.1_pcc
##            40.005432             1.444299
```

```
flow.train <- flow.train[,c(1:9,11:12)] # remove HpHp_L0.1_covariance

m7.log <- glm(label ~ ., data=flow.train, family=binomial)
```

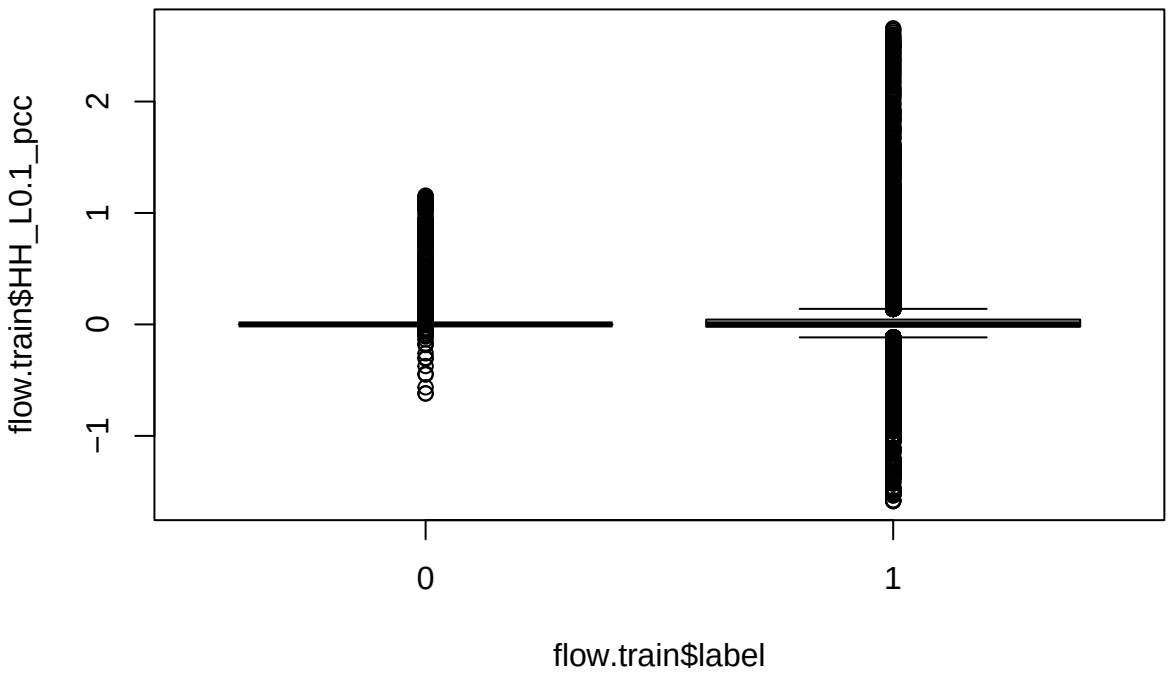
```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
vif(m7.log)
```

```
##          HH_L0.1_std    HH_L0.1_magnitude       HH_L0.1_radius
##             4.687445             1.733413             3.251294
##   HH_L0.1_covariance          HH_L0.1_pcc     HH_jit_L0.1_mean
##             1.541734             1.436985             1.624189
## HH_jit_L0.1_variance     HpHp_L0.1_weight        HpHp_L0.1_std
##             1.007636             1.139495             2.397537
##        HpHp_L0.1_pcc
##             1.347674
```

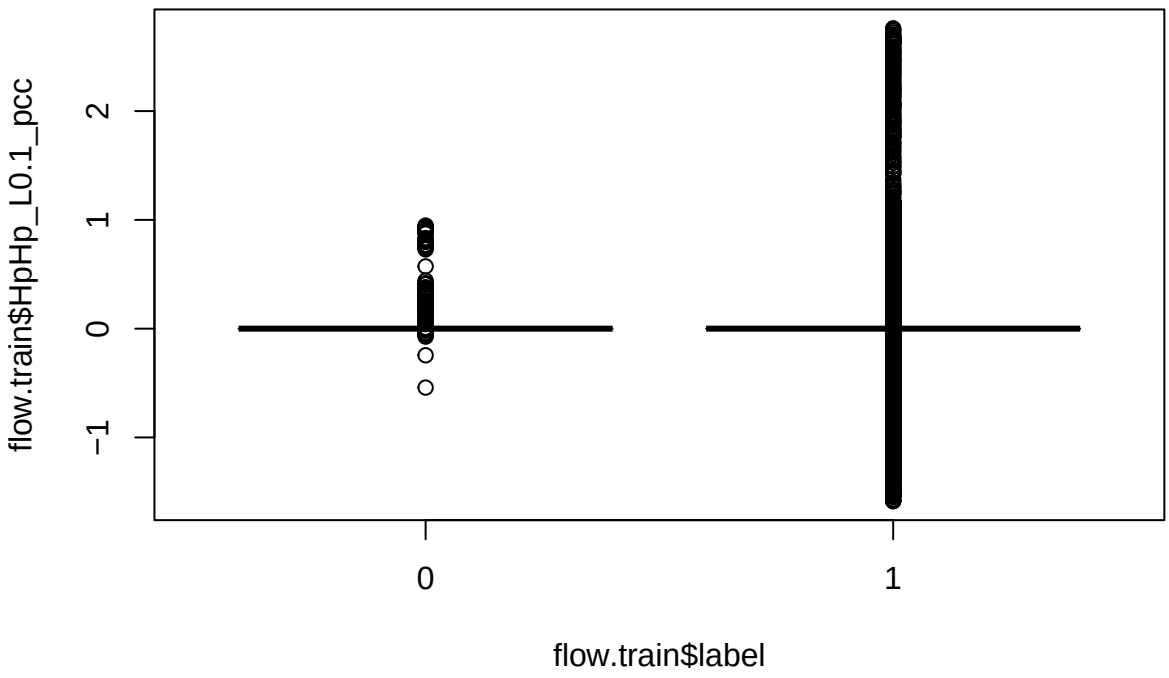
```
boxplot(flow.train$HH_L0.1_pcc ~ flow.train$label)
```




```
cor(flow.train$HH_L0.1_pcc, flow.train$HpHp_L0.1_pcc)
```

```
## [1] 0.2863835
```

```
boxplot(flow.train$HpHp_L0.1_pcc ~ flow.train$label)
```




```
flow.train$abs_HH_pcc <- abs(flow.train$HH_L0.1_pcc)
flow.test$abs_HH_pcc <- abs(flow.test$HH_L0.1_pcc)

flow.train$abs_HpHp_pcc <- abs(flow.train$HpHp_L0.1_pcc)
flow.test$abs_HpHp_pcc <- abs(flow.test$HpHp_L0.1_pcc)
```

```
flow.train <- flow.train[,c(1:4,6:9,11:13)] # remove pcc vars
```

```
mfinal.log <- glm(label ~ ., data=flow.train,
                  family=binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(mfinal.log)
```

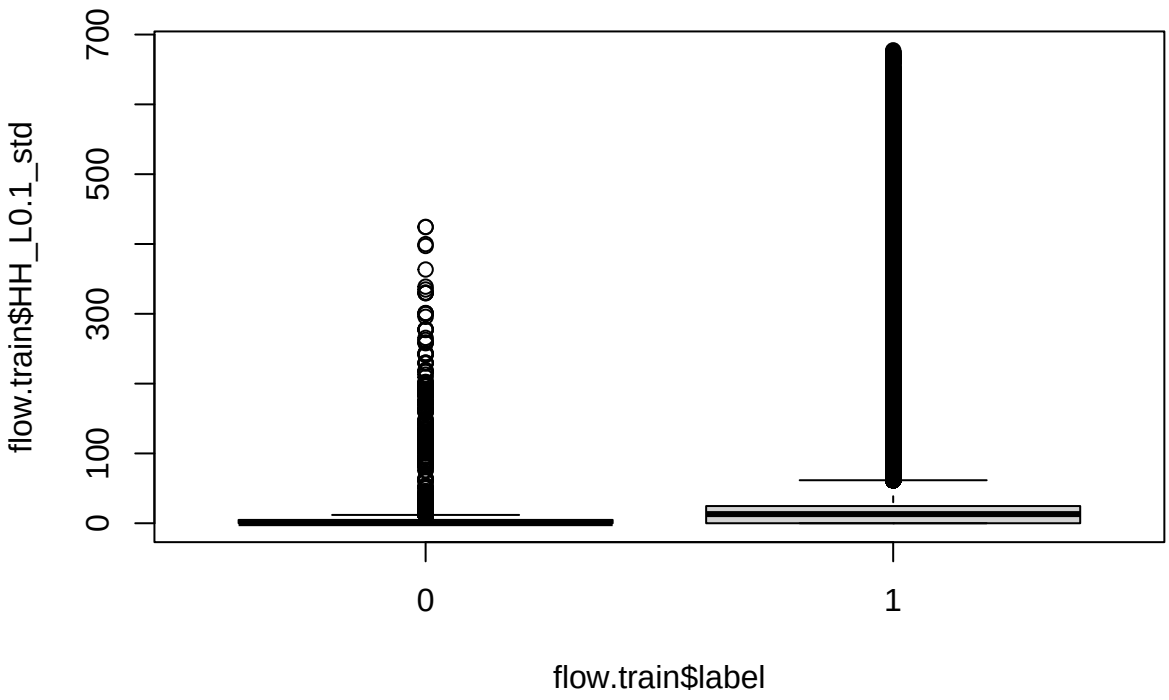
```
##
## Call:
## glm(formula = label ~ ., family = binomial, data = flow.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
##  -8.49    0.00    0.00    0.00    8.49
##
## Coefficients:
##                          Estimate Std. Error    z value Pr(>|z|)
## (Intercept)          -1.626e+15  1.237e+05 -1.315e+10   <2e-16 ***
## HH_L0.1_std           3.394e+13  3.164e+03  1.073e+10   <2e-16 ***
## HH_L0.1_magnitude     2.174e+12  3.080e+02  7.058e+09   <2e-16 ***
## HH_L0.1_radius       -1.696e+10  3.333e+00 -5.088e+09   <2e-16 ***
## HH_L0.1_covariance    7.932e+10  1.762e+01  4.502e+09   <2e-16 ***
## HH_jit_L0.1_mean     -1.928e+06  9.205e-05 -2.095e+10   <2e-16 ***
## HH_jit_L0.1_variance  1.810e-04  1.165e-12  1.553e+08   <2e-16 ***
## HpHp_L0.1_weight     -1.314e+12  7.950e+01 -1.653e+10   <2e-16 ***
## HpHp_L0.1_std         6.113e+12  2.606e+03  2.345e+09   <2e-16 ***
## abs_HH_pcc            1.076e+16  6.862e+05  1.568e+10   <2e-16 ***
## abs_HpHp_pcc          2.531e+15  9.287e+05  2.725e+09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance:  1753648  on 1698600  degrees of freedom
## Residual deviance: 15418970  on 1698590  degrees of freedom
## AIC: 15418992
##
## Number of Fisher Scoring iterations: 25
```

```
vif(mfinal.log)
```

```
##          HH_L0.1_std    HH_L0.1_magnitude       HH_L0.1_radius
##             4.682132             1.769194             3.238149
##   HH_L0.1_covariance     HH_jit_L0.1_mean HH_jit_L0.1_variance
##             1.362052             1.676289             1.007990
##     HpHp_L0.1_weight        HpHp_L0.1_std           abs_HH_pcc
##             1.139581             2.646058             1.509288
##         abs_HpHp_pcc
##             1.951865
```

```
boxplot(flow.train$HH_L0.1_std ~ flow.train$label)
```



Logistic Regression Predictions

```
lr.probs <- predict(mfinal.log, flow.test, type='response')
```

```
summary(lr.probs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1196  0.0000  1.0000
```

Logistic Regression Performance

```
lr.preds <- ifelse(lr.probs > .5, 1, 0)
lr.cm <- confusionMatrix(as.factor(lr.preds), flow.test$label)
lr.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 561849  79031
##          1  12225  74868
```

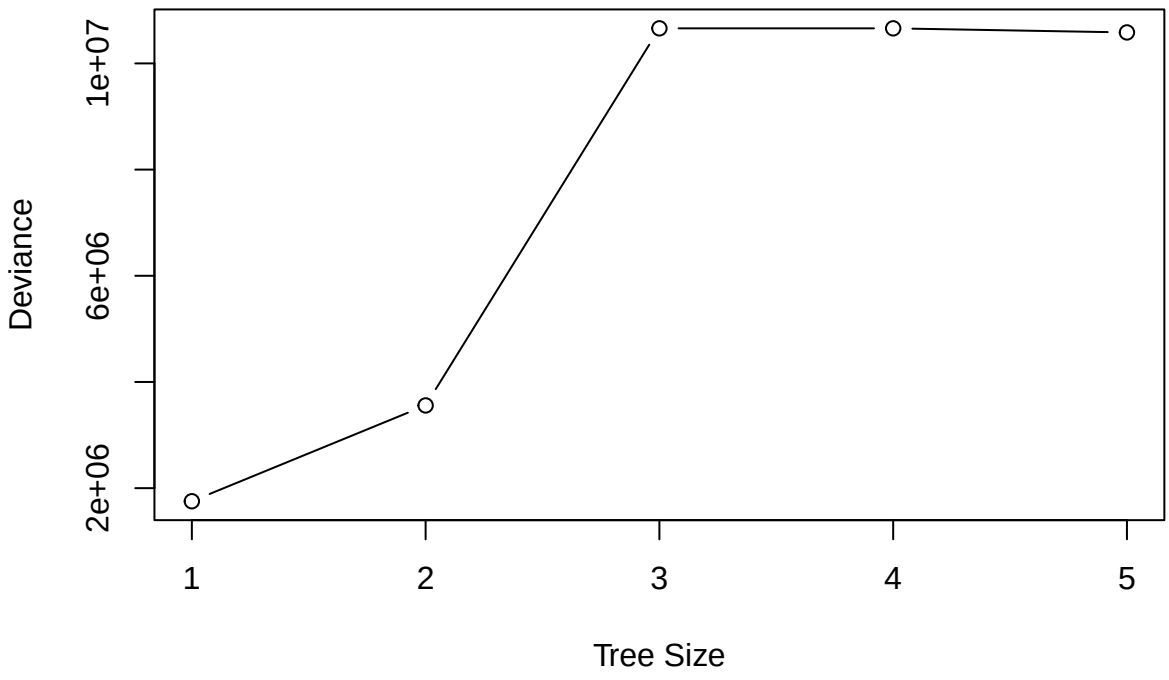
```
##
##                Accuracy : 0.8746
##                  95% CI : (0.8739, 0.8754)
##     No Information Rate : 0.7886
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.553
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9787
##             Specificity : 0.4865
##          Pos Pred Value : 0.8767
##          Neg Pred Value : 0.8596
##              Prevalence : 0.7886
##          Detection Rate : 0.7718
##    Detection Prevalence : 0.8804
##       Balanced Accuracy : 0.7326
##
##        'Positive' Class : 0
##
```

```
lr.acc <- round(lr.cm$overall[["Accuracy"]],4)*100
lr.ppv <- round(lr.cm$byClass[["Neg Pred Value"]],4)*100
```

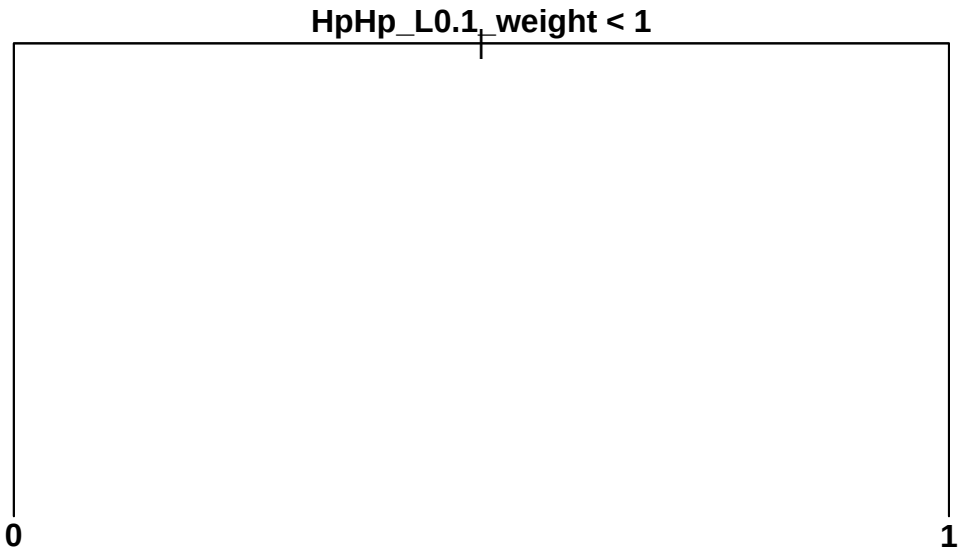
Decision Tree

```
set.seed(1)
tree.m1 <- tree(label ~ ., data=flow.train)

tree.cv <- cv.tree(tree.m1, FUN=prune.tree)
plot(tree.cv$size, tree.cv$dev, type='b',
     xlab='Tree Size', ylab='Deviance')
```

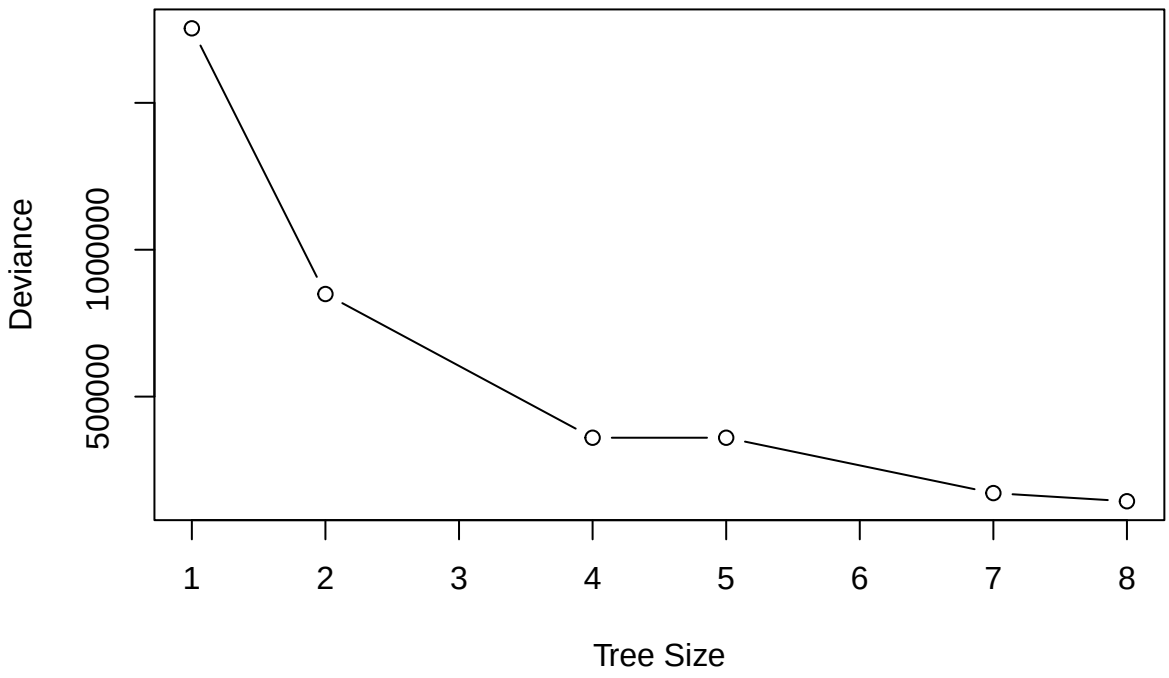



```
tree.prune <- prune.tree(tree.m1, best=2)
plot(tree.prune)
text(tree.prune, font=2)
```

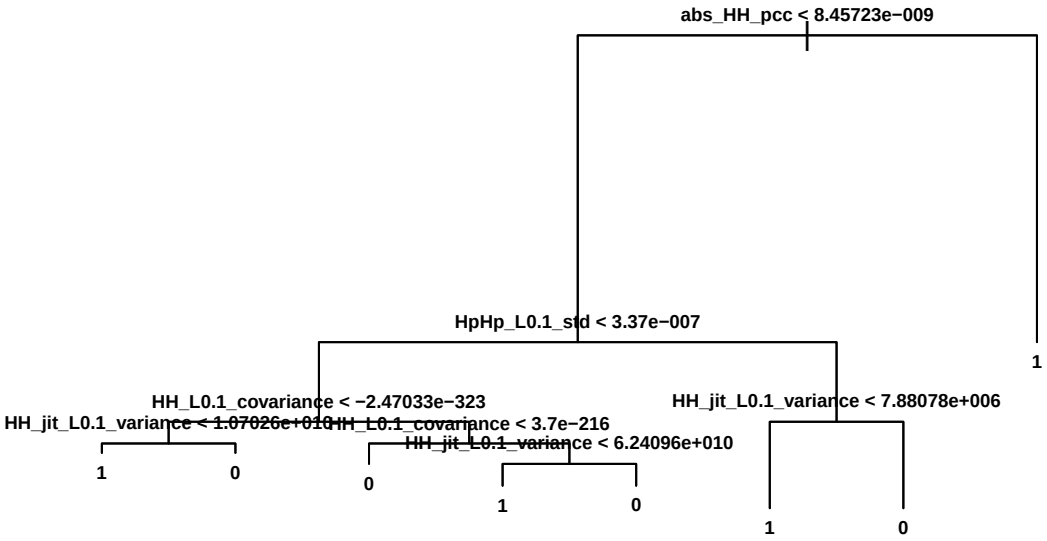



```
set.seed(1)
tree.m2 <- tree(label ~ . -HpHp_L0.1_weight, data=flow.train)

tree.cv2 <- cv.tree(tree.m2, FUN=prune.tree)
plot(tree.cv2$size, tree.cv2$dev, type='b',
     xlab='Tree Size', ylab='Deviance')
```




```
tree.prune2 <- prune.tree(tree.m2, best=8)
plot(tree.prune2)
text(tree.prune2, font=2, cex=.6)
```



Decision Tree Performance

```
prune1.probs <- predict(tree.prune, newdata=flow.test)
prune1.preds <- ifelse(prune1.probs[,"1"] > prune1.probs[,"0"], 1, 0)
```

```
tree1.cm <- confusionMatrix(as.factor(prune1.preds), flow.test$label)
```

```
## Warning in confusionMatrix.default(as.factor(prune1.preds), flow.test$label):
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
tree1.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0      0      0
##          1 574074 153899
##
##                Accuracy : 0.2114
##                  95% CI : (0.2105, 0.2123)
##     No Information Rate : 0.7886
```

```
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.2114
##              Prevalence : 0.7886
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

```
prune.probs <- predict(tree.prune2, newdata=flow.test)
prune.preds <- ifelse(prune.probs[,"1"] > prune.probs[,"0"], 1, 0)
```

```
tree.cm <- confusionMatrix(as.factor(prune.preds), flow.test$label)
tree.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 573603   4687
##          1    471 149212
##
##                Accuracy : 0.9929
##                  95% CI : (0.9927, 0.9931)
##     No Information Rate : 0.7886
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9785
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9992
##             Specificity : 0.9695
##          Pos Pred Value : 0.9919
##          Neg Pred Value : 0.9969
##              Prevalence : 0.7886
##          Detection Rate : 0.7879
##    Detection Prevalence : 0.7944
##       Balanced Accuracy : 0.9844
##
##        'Positive' Class : 0
##
```

```
tree.acc <- round(tree.cm$overall[["Accuracy"]],4)*100
tree.ppv <- round(tree.cm$byClass[["Neg Pred Value"]],4)*100
```

Gradient Boosting Machines

```
h2o.no_progress()
h2o.init(max_mem_size="5g")
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         18 hours 52 minutes
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.36.0.4
##     H2O cluster version age:    2 years and 17 days !!!
##     H2O cluster name:           H2O_started_from_R_laine_dmd671
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   4.59 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     R Version:                  R version 4.1.1 (2021-08-10)

## Warning in h2o.clusterInfo():
## Your H2O cluster version is too old (2 years and 17 days)!
## Please download and install the latest version from http://h2o.ai/download/
```

```
y <- "label"
train.features <- flow.train[,c(1:8,10:11)]
x <- setdiff(names(train.features), y)
```

```
train.h2o <- as.h2o(flow.train)

h2o.fit1 <- h2o.gbm(x=x, y=y, training_frame=train.h2o, nfolds=5)
h2o.fit1
```

```
## Model Details:
## ==============
##
## H2OBinomialModel: gbm
## Model ID:  GBM_model_R_1713308884557_676
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1              50                       50               21570         5
##   max_depth mean_depth min_leaves max_leaves mean_leaves
```

```
## 1         5    5.00000         26         32    29.64000
##
##
## H2OBinomialMetrics: gbm
## ** Reported on training data. **
##
## MSE:  0.00100865
## RMSE:  0.03175925
## LogLoss:  0.009304987
## Mean Per-Class Error:  0.0008976558
## AUC:  0.9998699
## AUCPR:  0.9997214
## Gini:  0.9997398
## R^2:  0.993956
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##              0      1    Error           Rate
## 0      1337597   1406 0.001050  =1406/1339003
## 1          268 359330 0.000745    =268/359598
## Totals 1337865 360736 0.000986  =1674/1698601
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold          value idx
## 1                        max f1  0.109152       0.997676 339
## 2                        max f2  0.082538       0.998630 349
## 3                  max f0point5  0.655442       0.998328 209
## 4                  max accuracy  0.109152       0.999014 339
## 5                 max precision  0.996248       1.000000   0
## 6                    max recall  0.002642       1.000000 398
## 7               max specificity  0.996248       1.000000   0
## 8             max absolute_mcc  0.109152       0.997053 339
## 9   max min_per_class_accuracy  0.109152       0.998950 339
## 10 max mean_per_class_accuracy  0.082538       0.999104 349
## 11                      max tns  0.996248 1339003.000000   0
## 12                      max fns  0.996248  358516.000000   0
## 13                      max fps  0.002638 1339003.000000 399
## 14                      max tps  0.002642  359598.000000 398
## 15                      max tnr  0.996248       1.000000   0
## 16                      max fnr  0.996248       0.996991   0
## 17                      max fpr  0.002638       1.000000 399
## 18                      max tpr  0.002642       1.000000 398
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
##
## H2OBinomialMetrics: gbm
## ** Reported on cross-validation data. **
## ** 5-fold cross-validation on training data (Metrics computed for combined holdout predictions) **
##
## MSE:  0.001213997
## RMSE:  0.03484247
## LogLoss:  0.01072465
## Mean Per-Class Error:  0.002010686
## AUC:  0.9998177
## AUCPR:  0.9996546
```

```
## Gini:  0.9996354
## R^2:  0.9927255
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##              0      1    Error           Rate
## 0      1338433    570 0.000426   =570/1339003
## 1         1293 358305 0.003596   =1293/359598
## Totals 1339726 358875 0.001097  =1863/1698601
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                          metric threshold          value idx
## 1                        max f1  0.378482       0.997407 242
## 2                        max f2  0.112226       0.997378 307
## 3                  max f0point5  0.729176       0.998158 174
## 4                  max accuracy  0.378482       0.998903 242
## 5                 max precision  0.996571       1.000000   0
## 6                    max recall  0.002529       1.000000 391
## 7               max specificity  0.996571       1.000000   0
## 8             max absolute_mcc  0.378482       0.996712 242
## 9   max min_per_class_accuracy  0.039155       0.997928 334
## 10 max mean_per_class_accuracy  0.112226       0.998325 307
## 11                      max tns  0.996571 1339003.000000   0
## 12                      max fns  0.996571  359330.000000   0
## 13                      max fps  0.001735 1339003.000000 399
## 14                      max tps  0.002529  359598.000000 391
## 15                      max tnr  0.996571       1.000000   0
## 16                      max fnr  0.996571       0.999255   0
## 17                      max fpr  0.001735       1.000000 399
## 18                      max tpr  0.002529       1.000000 391
##
## Gains/Lift Table: Extract with ‘h2o.gainsLift(<model>, <data>)‘ or ‘h2o.gainsLift(<model>, valid=<T/
## Cross-Validation Metrics Summary:
##                              mean        sd cv_1_valid cv_2_valid cv_3_valid
## accuracy                 0.998991  0.000181   0.999146   0.998808   0.999076
## auc                      0.999802  0.000050   0.999816   0.999764   0.999879
## err                      0.001009  0.000181   0.000854   0.001192   0.000924
## err_count              342.600000 61.435333 290.000000 405.000000 314.000000
## f0point5                 0.997706  0.000546   0.997260   0.997649   0.998590
## f1                       0.997618  0.000429   0.997978   0.997178   0.997815
## f2                       0.997530  0.001094   0.998697   0.996707   0.997040
## lift_top_group           4.723642  0.013699   4.741951   4.732519   4.722115
## logloss                  0.010725  0.001642   0.008955   0.012497   0.010641
## max_per_class_error      0.002552  0.001547   0.000862   0.003607   0.003476
## mcc                      0.996979  0.000544   0.997438   0.996423   0.997230
## mean_per_class_accuracy  0.998436  0.000675   0.999157   0.997924   0.998143
## mean_per_class_error     0.001564  0.000675   0.000843   0.002076   0.001857
## mse                      0.001214  0.000272   0.000966   0.001484   0.001090
## pr_auc                   0.999666  0.000089   0.999753   0.999579   0.999745
## precision                0.997765  0.001000   0.996783   0.997964   0.999108
## r2                       0.992726  0.001623   0.994197   0.991092   0.993472
## recall                   0.997472  0.001580   0.999176   0.996393   0.996524
## rmse                     0.034673  0.003854   0.031076   0.038529   0.033010
## specificity              0.999400  0.000269   0.999138   0.999455   0.999761
##                         cv_4_valid cv_5_valid
```

```
## accuracy                  0.998783   0.999145
## auc                       0.999752   0.999796
## err                       0.001217   0.000855
## err_count               413.000000 291.000000
## f0point5                  0.997776   0.997254
## f1                        0.997131   0.997987
## f2                        0.996487   0.998721
## lift_top_group            4.709202   4.712426
## logloss                   0.012266   0.009267
## max_per_class_error       0.003942   0.000873
## mcc                       0.996359   0.997445
## mean_per_class_accuracy   0.997788   0.999169
## mean_per_class_error      0.002212   0.000831
## mse                       0.001529   0.001001
## pr_auc                    0.999566   0.999685
## precision                 0.998206   0.996766
## r2                        0.990856   0.994011
## recall                    0.996058   0.999210
## rmse                      0.039108   0.031642
## specificity               0.999517   0.999127
```

```
h2o.fit2 <- h2o.gbm(x=x, y=y, training_frame = train.h2o,
                    nfolds=5, ntrees=5000, stopping_rounds=5,
                    seed=1)
```

```
## Warning in .h2o.processResponseWarnings(res): early stopping is enabled but neither score_tree_interv
```

```
h2o.fit2
```

```
## Model Details:
## ==============
##
## H2OBinomialModel: gbm
## Model ID:  GBM_model_R_1713308884557_777
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1             184                      184               70219         0
##   max_depth mean_depth min_leaves max_leaves mean_leaves
## 1         5    4.97283          1         32    25.62500
##
##
## H2OBinomialMetrics: gbm
## ** Reported on training data. **
##
## MSE:  0.0001634165
## RMSE:  0.01278345
## LogLoss:  0.0008047985
## Mean Per-Class Error:  0.0001896669
## AUC:  0.9999989
## AUCPR:  0.9999953
## Gini:  0.9999977
## R^2:  0.9990208
##
```

```
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##              0      1    Error          Rate
## 0      1338752    251 0.000187  =251/1339003
## 1           69 359529 0.000192    =69/359598
## Totals 1338821 359780 0.000188  =320/1698601
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold          value idx
## 1                        max f1  0.360779       0.999555 244
## 2                        max f2  0.309881       0.999707 255
## 3                  max f0point5  0.762582       0.999602 151
## 4                  max accuracy  0.360779       0.999812 244
## 5                 max precision  0.999969       1.000000   0
## 6                    max recall  0.007144       1.000000 366
## 7                max specificity  0.999969       1.000000   0
## 8               max absolute_mcc  0.360779       0.999436 244
## 9   max min_per_class_accuracy  0.360779       0.999808 244
## 10 max mean_per_class_accuracy  0.360779       0.999810 244
## 11                      max tns  0.999969 1339003.000000   0
## 12                      max fns  0.999969  308282.000000   0
## 13                      max fps  0.000014 1339003.000000 399
## 14                      max tps  0.007144  359598.000000 366
## 15                      max tnr  0.999969       1.000000   0
## 16                      max fnr  0.999969       0.857296   0
## 17                      max fpr  0.000014       1.000000 399
## 18                      max tpr  0.007144       1.000000 366
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
##
## H2OBinomialMetrics: gbm
## ** Reported on cross-validation data. **
## ** 5-fold cross-validation on training data (Metrics computed for combined holdout predictions) **
##
## MSE:  0.0002146344
## RMSE:  0.0146504
## LogLoss:  0.001009734
## Mean Per-Class Error:  0.0004120193
## AUC:  0.9999984
## AUCPR:  0.9999934
## Gini:  0.9999968
## R^2:  0.9987139
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##              0      1    Error          Rate
## 0      1338797    206 0.000154  =206/1339003
## 1          241 359357 0.000670   =241/359598
## Totals 1339038 359563 0.000263  =447/1698601
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold          value idx
## 1                        max f1  0.628064       0.999378 183
## 2                        max f2  0.213297       0.999602 272
## 3                  max f0point5  0.695129       0.999444 167
## 4                  max accuracy  0.628064       0.999737 183
```

```
## 5                max precision  0.999966       1.000000   0
## 6                   max recall  0.002774       1.000000 370
## 7             max specificity  0.999966       1.000000   0
## 8            max absolute_mcc  0.628064       0.999212 183
## 9   max min_per_class_accuracy  0.277595       0.999711 256
## 10 max mean_per_class_accuracy  0.213297       0.999740 272
## 11                     max tns  0.999966 1339003.000000   0
## 12                     max fns  0.999966  310628.000000   0
## 13                     max fps  0.000011 1339003.000000 399
## 14                     max tps  0.002774  359598.000000 370
## 15                     max tnr  0.999966       1.000000   0
## 16                     max fnr  0.999966       0.863820   0
## 17                     max fpr  0.000011       1.000000 399
## 18                     max tpr  0.002774       1.000000 370
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
## Cross-Validation Metrics Summary:
##                              mean       sd cv_1_valid cv_2_valid cv_3_valid
## accuracy                 0.999745 0.000012   0.999752   0.999729   0.999759
## auc                      0.999998 0.000001   0.999999   0.999999   0.999999
## err                      0.000256 0.000012   0.000248   0.000271   0.000241
## err_count               86.800000 3.962322  84.000000  92.000000  82.000000
## f0point5                 0.999360 0.000132   0.999358   0.999445   0.999371
## f1                       0.999397 0.000027   0.999416   0.999362   0.999429
## f2                       0.999433 0.000140   0.999475   0.999278   0.999488
## lift_top_group           4.723618 0.008106   4.713984   4.717192   4.733939
## logloss                  0.001010 0.000063   0.000960   0.000954   0.001109
## max_per_class_error      0.000552 0.000211   0.000487   0.000777   0.000473
## mcc                      0.999235 0.000034   0.999259   0.999190   0.999276
## mean_per_class_accuracy  0.999640 0.000086   0.999665   0.999544   0.999674
## mean_per_class_error     0.000360 0.000086   0.000335   0.000456   0.000326
## mse                      0.000215 0.000013   0.000202   0.000212   0.000236
## pr_auc                   0.999994 0.000005   0.999995   0.999997   0.999995
## precision                0.999335 0.000220   0.999319   0.999500   0.999332
## r2                       0.998714 0.000077   0.998794   0.998732   0.998586
## recall                   0.999458 0.000228   0.999514   0.999223   0.999527
## rmse                     0.014645 0.000424   0.014195   0.014555   0.015349
## specificity              0.999822 0.000059   0.999817   0.999866   0.999821
##                          cv_4_valid cv_5_valid
## accuracy                   0.999738   0.999744
## auc                        0.999997   0.999999
## err                        0.000262   0.000256
## err_count                 89.000000  87.000000
## f0point5                   0.999143   0.999483
## f1                         0.999381   0.999395
## f2                         0.999619   0.999308
## lift_top_group             4.728183   4.724792
## logloss                    0.000996   0.001030
## max_per_class_error        0.000273   0.000750
## mcc                        0.999215   0.999233
## mean_per_class_accuracy    0.999752   0.999563
## mean_per_class_error       0.000248   0.000437
## mse                        0.000212   0.000212
## pr_auc                     0.999985   0.999997
```

```
## precision                0.998984   0.999541
## r2                       0.998728   0.998729
## recall                   0.999777   0.999250
## rmse                     0.014565   0.014562
## specificity              0.999727   0.999877
```

```
h2o.varimp_plot(h2o.fit2)
```