URL Click Analysis

This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages where you can embed R codes.

The source data. Special thanks to Nandita for coming up with an idea for processing URL and Refer URL variables.

Data inspection and cleanup

load raw data

```
raw <- read.csv("coursera.sanitized.csv")
sapply(raw[1, ], class)
```

```
##  datehour      v_id      b_id      t_id    seller   country
state
##  "factor" "integer"  "factor" "integer" "integer"  "factor"
"factor"
##       url refer_url     click
##  "factor"  "factor" "integer"
```

click:

An integer representing whether or not a click occurred - this is the outcome we want to predict

```
table(raw$click)
```

```
##
##      0      1
## 782725    217
```

```
# just those actually got any clicks
clicked <- raw[raw$click == 1, ]
```

datehour:

A string describing the day and hour when the ad was served

```
head(raw$datehour)
```

```
## [1] 2013-05-29 09:00:00 2013-05-29 08:00:00 2013-05-29 08:00:00
## [4] 2013-05-29 09:00:00 2013-05-29 08:00:00 2013-05-29 08:00:00
## 100 Levels: 2013-05-28 00:00:00 2013-05-28 01:00:00 ... 2013-06-01
03:00:00
```

```
sum(is.na(raw$datehour))
```

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

```
# convert string to datetime object
pdata <- raw
datehour <- strptime(raw$datehour, "%Y-%m-%d %H:%M:%S")
pdata$wday <- datehour$wday
pdata$hour <- datehour$hour
pdata$datehour <- datehour
```

v_id:

An integer representing a channel in which a viewer has been served through.

```
table(pdata$v_id)
```

```
##
##      0      1      3
##  87067 601746  94129
```

```
# just those actually got any clicks
table(clicked$v_id)
```

```
##
##   0   1   3
##  25 165  27
```

```
# check missing values
sum(is.na(pdata$v_id))
```

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

b_id:

An integer representing a class of user based on their interests

```
length(unique(pdata$b_id))
```

```
## [1] 131
```

```
# just those actually got any clicks
length(unique(clicked$b_id))
```

```
## [1] 48
```

```
# check missing values
sum(is.na(pdata$b_id))
```

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

```
# inspect data
head(pdata$b_id)
```

```
## [1] \\N \\N \\N \\N \\N \\N
## 131 Levels: \\N 100 101 103 104 105 106 107 109 11 110 113 115 116
... 96
```

```
# check the number of NULL values
sum(pdata$b_id == "\\N")
```

```
## [1] 546712
```

```
# rename '\\N' to 0 and convert the factor into integer
levels(pdata$b_id)[levels(pdata$b_id) == "\\N"] <- "0"
pdata$b_id <- as.numeric(as.character(pdata$b_id))
```

t_id:

An integer identifying a specific ad

```
length(unique(pdata$t_id))
```

```
## [1] 18
```

```
# just those actually got any clicks
length(unique(clicked$t_id))
```

```
## [1] 17
```

```
# check missing values
sum(is.na(pdata$t_id))
```

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

seller:

An integer representing the seller providing the wholesaler with the impression

```
length(unique(pdata$seller))
```

```
## [1] 437
```

```
# just those actually got any clicks
length(unique(clicked$seller))
```

```
## [1] 41
```

```
# check missing values
sum(is.na(pdata$seller))
```

```
## [1] 94129
```

```
# check the value range
min(pdata$seller[!is.na(pdata$seller)])
```

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

```
max(pdata$seller[!is.na(pdata$seller)])
```

```
## [1] 2796
```

```
pdata$seller[is.na(pdata$seller)] <- 3000
sum(is.na(pdata$seller))
```

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

country:

A string representing the clicker's country of origin

```
table(pdata$country)
```

```
##
##              BR     CA     EG     ES     US
##   5158      1 777549      1      3    230
```

```
# just those actually got any clicks
table(clicked$country)
```

```
##
##       BR  CA  EG  ES  US
##   1    0 216   0   0   0
```

```
# check NULL values
levels(pdata$country)
```

```
## [1] ""   "BR" "CA" "EG" "ES" "US"
```

```
levels(pdata$country)[levels(pdata$country) == ""] <- "00"
levels(pdata$country)
```

```
## [1] "00" "BR" "CA" "EG" "ES" "US"
```

state:

A string representing the clicker's state of origin

```
length(unique(pdata$state))
```

```
## [1] 33
```

```
# just those actually got any clicks
length(unique(clicked$state))
```

```
## [1] 5
```

```
# check NULL values
levels(pdata$state)
```

```
##  [1] ""   "00" "AB" "AK" "AZ" "BC" "CA" "CO" "FL" "GA" "IL" "IN"
"LA" "MB"
## [15] "MI" "MO" "MS" "NB" "NJ" "NS" "NT" "NU" "NY" "OH" "ON" "OR"
"PE" "QC"
## [29] "SK" "TX" "VA" "WA" "YT"
```

```
levels(pdata$state)[levels(pdata$state) == ""] <- "00"
levels(pdata$state)
```

```
##  [1] "00" "AB" "AK" "AZ" "BC" "CA" "CO" "FL" "GA" "IL" "IN" "LA"
"MB" "MI"
## [15] "MO" "MS" "NB" "NJ" "NS" "NT" "NU" "NY" "OH" "ON" "OR" "PE"
"QC" "SK"
## [29] "TX" "VA" "WA" "YT"
```

url:

An encrypted string representing the URL the ad was displayed on

```
length(unique(pdata$url))
```

```
## [1] 12312
```

```
# just those actually got any clicks
length(unique(clicked$url))
```

```
## [1] 152
```

```
# check NULL values
levels(pdata$url)[levels(pdata$url) == ""]
```

```
## character(0)
```

Check the frequency of urls = views. Special thanks to Nandita for coming up with this metric.

```
quantile(table(pdata$url))
```

```
##    0%   25%   50%   75%  100%
##     1     1     3    11 35898
```

```
# barplot(sort(table(pdata$url),decreasing=TRUE), col='blue')
# barplot(sort(table(clicked$url),decreasing=TRUE), col='blue')
views <- as.data.frame(table(pdata$url))
names(views) <- c("url", "views")
```

Check the number of traffic sources - this represents the popularity of a given URL similar to PageRank algorithm. This is also based on Nandita's idea.

```
sourceCount <- function(page_url) {
    ref <- pdata$refer_url[pdata$url == page_url]
    return(length(unique(ref)))
}

exetime <- system.time(views$srcs <- sapply(views[, 1], sourceCount))
exetime[3]/60
```

```
## elapsed
##   276.3
```

Add the views and source counts to the data

```
pdata <- merge(pdata, views, all = TRUE)
```

refer_url:

An encrypted string representing the referrer URL

```
# length(unique(pdata$refer_url)) just those actually got any clicks
length(unique(clicked$refer_url))
```

```
## [1] 118
```

```
# check NULL values
levels(pdata$refer_url)[levels(pdata$refer_url) == ""]
```

```
## character(0)
```

Check the frequency of refer_urls

```
quantile(table(pdata$refer_url))
```

```
##    0%   25%   50%   75%  100%
##     1     1     4    17 61337
```

```
# barplot(sort(table(pdata$refer_url),decreasing=TRUE), col='blue')
# barplot(sort(table(clicked$refer_url),decreasing=TRUE), col='blue')
```

Add the frequency to the data - this represents how frequently a given referrer sends traffic - the larger frequency, the more active.

```
referrals <- as.data.frame(table(pdata$refer_url))
names(referrals) <- c("refer_url", "referrals")
pdata <- merge(pdata, referrals, all = TRUE)
```

Check the cleanup result

```
pdata <- pdata[, c("datehour", "wday", "hour", "v_id", "b_id", "t_id",
"seller",
    "country", "state", "views", "srcs", "referrals", "url",
"refer_url", "click")]
summary(pdata)
```

```
##     datehour                          wday          hour
##  Min.   :2013-05-28 00:00:00   Min.   :2.00   Min.   : 0.0
##  1st Qu.:2013-05-28 19:00:00   1st Qu.:2.00   1st Qu.: 4.0
##  Median :2013-05-29 15:00:00   Median :3.00   Median :13.0
##  Mean   :2013-05-29 16:14:48   Mean   :3.19   Mean   :11.6
##  3rd Qu.:2013-05-30 13:00:00   3rd Qu.:4.00   3rd Qu.:17.0
##  Max.   :2013-06-01 03:00:00   Max.   :6.00   Max.   :23.0
##
##      v_id          b_id          t_id          seller
country
##  Min.   :0.00   Min.   :   0   Min.   :9595   Min.   :   0   00:
5158
##  1st Qu.:1.00   1st Qu.:   0   1st Qu.:9600   1st Qu.: 459   BR:
1
##  Median :1.00   Median :   0   Median :9600   Median :1263
CA:777549
##  Mean   :1.13   Mean   : 134   Mean   :9602   Mean   :1176   EG:
1
##  3rd Qu.:1.00   3rd Qu.:  20   3rd Qu.:9606   3rd Qu.:1362   ES:
3
##  Max.   :3.00   Max.   :1539   Max.   :9612   Max.   :3000   US:
230
##
##      state           views            srcs         referrals
##  AB     :469398   Min.   :    1   Min.   :  1.0   Min.   :    1
##  MB     :274160   1st Qu.:  331   1st Qu.:  2.0   1st Qu.:  937
##  SK     : 29628   Median : 2416   Median :  5.0   Median : 7688
##  00     :  5163   Mean   : 6995   Mean   : 27.4   Mean   :19395
##  BC     :  2114   3rd Qu.: 8783   3rd Qu.:  9.0   3rd Qu.:42085
##  ON     :  1951   Max.   :35898   Max.   :539.0   Max.   :61337
##  (Other):   528
##                                url
##  9de10ca8a3ca97b297d52f85d9405802: 35898
##  d41d8cd98f00b204e9800998ecf8427e: 28094
##  7fda1f7b7b41b508f042a2f1707139a5: 23717
##  2f187a80b5c7a8191b833d9f7a9cab3a: 21161
##  827ff00577f632e875ed4d5ad5110a05: 16547
##  3f4c618b986082f65d169b75eacbdf1d: 13729
##  (Other)                         :643796
##                             refer_url          click
##  61704255014d0941063906fc597813a2: 61337   Min.   :0e+00
##  50fbb858c32b045b323f64625c7499a3: 53225   1st Qu.:0e+00
##  c948dd7a61887edf9074a9f8c6461e34: 49634   Median :0e+00
##  1599908839386189af5ca7eec8b1de31: 42085   Mean   :3e-04
##  2d5027eae798c17c9f0b91d43d3432ba: 39452   3rd Qu.:0e+00
##  ecab7930ae6daaae85e0e49cbaaca87d: 29402   Max.   :1e+00
```

```
##  (Other)                         :507807
```

```
head(pdata)
```

```
##              datehour wday hour v_id b_id t_id seller country state
views
## 1 2013-05-29 17:00:00    3   17    3    0 9606   3000      00
00     1
## 2 2013-05-29 01:00:00    3    1    1  130 9611    589      CA
SK     2
## 3 2013-05-30 02:00:00    4    2    1 1392 9601    310      CA
AB     2
## 4 2013-05-29 00:00:00    3    0    0  853 9598      1      CA    AB
28094
## 5 2013-05-29 05:00:00    3    5    0    0 9597      1      CA    AB
28094
## 6 2013-05-28 21:00:00    2   21    0  853 9598      1      CA    AB
28094
##   srcs referrals                              url
## 1    1         1 000231e00491e024ac295c78f63585eb
## 2    1         2 0021ac67ce4542475a085c5d71f89a61
## 3    1         2 0021ac67ce4542475a085c5d71f89a61
## 4  539         4 d41d8cd98f00b204e9800998ecf8427e
## 5  539         4 d41d8cd98f00b204e9800998ecf8427e
## 6  539         4 d41d8cd98f00b204e9800998ecf8427e
##                          refer_url click
## 1 000231e00491e024ac295c78f63585eb     0
## 2 0021ac67ce4542475a085c5d71f89a61     0
## 3 0021ac67ce4542475a085c5d71f89a61     0
## 4 0022fc9341d50e0828b6cf556ecb47e0     0
## 5 0022fc9341d50e0828b6cf556ecb47e0     0
## 6 0022fc9341d50e0828b6cf556ecb47e0     0
```

```
save(pdata, file = "processed.rda")
# load(file='processed.rda')
```

Deal with the class imbalance

Reduce the majority class by downsumpling

Here we are making an assumption that we can ignore most of the cases where a click didn't occur without impeding our ability to detect the cases where it did occur.

```
# first check the ratio between the two classes
table(pdata$click)
```

```
##
##      0      1
## 782725    217
```

```
# drop url and refer_url variables
pdata <- pdata[, !(names(pdata) %in% c("datehour", "url", "refer_url"))]
# convert click to a factor
pdata$click <- as.factor(pdata$click)

# down sample the majority class to reduce the ratio of two classes to
10:1
maj <- pdata[pdata$click == 0, ]
subSampling <- sample(1:dim(maj)[1], size = 217 * 10, replace = FALSE)
subSampled <- rbind(pdata[pdata$click == 1, ], maj[subSampling, ])
table(subSampled$click)
```

```
##
##    0    1
## 2170  217
```

```
summary(subSampled)
```

```
##      wday         hour          v_id          b_id
t_id
##  Min.   :2.0   Min.   : 0.0   Min.   :0.00   Min.   :   0   Min.
:9595
##  1st Qu.:2.0   1st Qu.: 4.0   1st Qu.:1.00   1st Qu.:   0   1st
Qu.:9600
##  Median :3.0   Median :14.0   Median :1.00   Median :   0   Median
:9600
##  Mean   :3.2   Mean   :11.7   Mean   :1.14   Mean   : 134   Mean
:9602
##  3rd Qu.:4.0   3rd Qu.:18.0   3rd Qu.:1.00   3rd Qu.:  22   3rd
Qu.:9606
##  Max.   :6.0   Max.   :23.0   Max.   :3.00   Max.   :1539   Max.
:9612
##
##     seller     country      state         views
srcs
##  Min.   :   0   00:  12   AB     :1462   Min.   :    1   Min.   :
1.0
##  1st Qu.: 459   BR:   0   MB     : 814   1st Qu.:  278   1st Qu.:
2.0
##  Median :1263   CA:2372   SK     :  87   Median : 2297   Median :
5.0
##  Mean   :1184   EG:   0   00     :  12   Mean   : 6953   Mean   :
26.9
##  3rd Qu.:1362   ES:   0   ON     :   5   3rd Qu.: 8783   3rd Qu.:
9.0
##  Max.   :3000   US:   3   BC     :   4   Max.   :35898   Max.
:539.0
##                            (Other):
3
##    referrals     click
##  Min.   :    1   0:2170
##  1st Qu.:  801   1: 217
##  Median : 7327
##  Mean   :18973
##  3rd Qu.:42085
##  Max.   :61337
##
```

Setup Cross Validation

In order to test the predictive model, we will split the downsampled data into two subsets. We will also take a sampling from the original data without downsampling the majority class.

```
# split data into two subsets - 2/3 training set, 1/3 test set
set.seed(333)
trainSamples <- sample(1:dim(subSampled)[1], size =
(dim(subSampled)[1]/3 *
    2), replace = FALSE)
train <- subSampled[trainSamples, ]
test <- subSampled[-trainSamples, ]
# we will also have a validation set from the original data
valSamples <- sample(1:dim(pdata)[1], size = dim(pdata)[1]/50, replace
= FALSE)
validation <- pdata[valSamples, ]
# check the number of minority class - should be around 145:72
sum(train$click == 1)
```

```
## [1] 140
```

```
sum(test$click == 1)
```

```
## [1] 77
```

```
sum(validation$click == 1)
```

```
## [1] 4
```

Random Forest

We try to build a predictive model using a random forest, with parameters set to take 10 samples from each class per iteration to make sure we get a balanced result.

```
suppressPackageStartupMessages(library(randomForest))
table(train$click)
```

```
##
##    0    1
## 1451  140
```

```
set.seed(1234)
rf.model1 <- randomForest(click ~ ., data = train, importance = TRUE,
prox = TRUE,
    strata = train$click, sampsize = c(10, 10))
rf.model1
```

```
##
## Call:
##  randomForest(formula = click ~ ., data = train, importance =
TRUE,      prox = TRUE, strata = train$click, sampsize = c(10, 10))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 41.92%
## Confusion matrix:
##     0   1 class.error
## 0 833 618      0.4259
## 1  49  91      0.3500
```

The class error rate for click = 1 is 35.0% - which is not so great, but at least we are getting more than half of them right.

Now let's see how much accuracy we get on the test set.

```
# make prediction from the test set
test.pred1 <- predict(rf.model1, test[, -12])
# compare it to the actual outcome
confusionMatrix <- table(observed = test$click, predicted = test.pred1)
confusionMatrix
```

```
##         predicted
## observed   0   1
##        0 394 325
##        1  23  54
```

```
# class error rate of click=1
confusionMatrix[2, 1]/sum(confusionMatrix[2, ])
```

```
## [1] 0.2987
```

The class error rate is 29.9% - a bit better.

Now let's make prediction with validation set

```
# make prediction from the validation set
validation.pred1 <- predict(rf.model1, validation[, -12])
# compare it to the actual outcome
confusionMatrix <- table(observed = validation$click, predicted =
validation.pred1)
confusionMatrix
```

```
##         predicted
## observed    0    1
##        0 8856 6798
##        1    1    3
```

```
# class error rate of click=1
confusionMatrix[2, 1]/sum(confusionMatrix[2, ])
```

```
## [1] 0.25
```

The class error rate is 25.0% - we called 3 out of 4 correctly.

Now that the model seems to be working OK, we can now turn to the variable importance metric Random Forest produces.

```
result <- importance(rf.model1, )[, "MeanDecreaseAccuracy"]
importance(rf.model1)[order(result, decreasing = TRUE), ]
```

```
##                  0        1 MeanDecreaseAccuracy MeanDecreaseGini
## seller     1.55867  2.69475              2.08787           1.3146
## b_id       1.86772  0.30772              2.00006           0.6068
## referrals  0.81327  4.45428              1.77695           1.3559
## views      0.42939  6.03709              1.34419           1.6460
## v_id       0.82073  1.37522              1.05345           0.3338
## srcs      -0.08476  5.87896              0.63277           1.3031
## country    0.00000  0.00000              0.00000           0.0000
## hour       0.36502 -1.58071             -0.03936           1.3653
## state     -0.24397 -0.84135             -0.36731           0.2404
## t_id      -0.40296  0.02626             -0.42430           0.9689
## wday      -0.62656 -1.78701             -1.09131           0.7621
```

The result rank the variable's importance by how much each contribute to reduce prediction error. However, the differences are pretty small, so I am not confident that this is a robust result.

Improving the model

Wenjia points out that two variables that show high importance, seller and b_id, happen to contain a lot of

NULL values. These NULL values may be artificially raising the apparent importance of those variables.

Seller variable

Let's examine “seller” variable.

```
# 94129 null values in 'seller' variable were re-coded with 3000
table(pdata$seller[pdata$seller == 3000], pdata$v_id[pdata$seller ==
3000])
```

```
##
##             3
##   3000 94129
```

```
length(pdata$v_id[pdata$v_id == 3])
```

```
## [1] 94129
```

It looks none of the v_id==3 conains valid seller id. So what happens if we remove v_id==3?

```
# New random forest without v_id==3
rf.model2 <- randomForest(click ~ ., data = train[train$v_id != 3, ],
importance = TRUE,
    prox = TRUE, strata = train$click[train$v_id != 3], sampsize =
c(10, 10))
rf.model2
```

```
##
## Call:
##  randomForest(formula = click ~ ., data = train[train$v_id !=
3, ], importance = TRUE, prox = TRUE, strata = train$click[train$v_id
!=      3], sampsize = c(10, 10))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 39.77%
## Confusion matrix:
##     0   1 class.error
## 0 763 500      0.3959
## 1  52  73      0.4160
```

```
result <- importance(rf.model2, )[, "MeanDecreaseAccuracy"]
importance(rf.model2)[order(result, decreasing = TRUE), ]
```

```
##                   0       1 MeanDecreaseAccuracy MeanDecreaseGini
## views      4.330538  4.8310               5.3026         1.646611
## srcs       4.578119  3.5439               5.2679         1.411746
## b_id       3.266409  0.1011               2.9831         0.549895
## referrals  1.683398  4.4927               2.6569         1.457910
## seller     0.663660  2.0719               1.0503         1.312233
## hour       1.064918 -1.0578               0.7595         1.417638
## v_id       0.437230  0.9651               0.5857         0.178795
## t_id       0.447652 -1.5972               0.1733         0.883117
## wday       0.546101 -1.2312               0.1632         0.744873
## state      0.009176 -0.9674              -0.1418         0.264481
## country   -1.417050 -1.4170              -1.6375         0.006166
```

The importance of "seller" variable drops significantly once the null values are removed. For this reason, we can probably ignore this variable altogether.

```
# New random forest without seller variable
noseller <- train
noseller$seller <- NULL
rf.model3 <- randomForest(click ~ ., data = noseller, importance =
TRUE, prox = TRUE,
    strata = noseller$click, sampsize = c(10, 10))
rf.model3
```

```
##
## Call:
##  randomForest(formula = click ~ ., data = noseller, importance =
TRUE,      prox = TRUE, strata = noseller$click, sampsize = c(10, 10))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 39.66%
## Confusion matrix:
##     0   1 class.error
## 0 884 567      0.3908
## 1  64  76      0.4571
```

```
result <- importance(rf.model3, )[, "MeanDecreaseAccuracy"]
importance(rf.model3)[order(result, decreasing = TRUE), ]
```

```
##                 0        1 MeanDecreaseAccuracy MeanDecreaseGini
## srcs       4.0676  1.99375               4.5541         1.443125
## views      3.7457  1.72963               4.2118         1.854874
## referrals  2.9128  3.98252               3.9521         1.832449
## b_id       1.8838  0.08293               1.9398         0.672024
## v_id       1.7177 -0.14739               1.7842         0.344518
## country    1.3441  0.00000               1.3441         0.003091
## t_id       1.3985 -1.18428               1.2658         1.112469
## hour       1.0358 -2.07756               0.4789         1.463926
## state      0.6713 -2.69325               0.3920         0.262547
## wday      -1.4438 -1.62507              -1.8332         0.870442
```

b_id variable

Let's now examine “b_id” variable. 546712 values “//N” variable were re-coded with 0.

```
# how many clicks does that class contain?
table(pdata$click[pdata$b_id == 0])
```

```
##
##      0      1
## 546580    132
```

```
# how many clicks do all other classes contain?
table(pdata$click[pdata$b_id != 0])
```

```
##
##      0      1
## 236145     85
```

It turned out Null value is a majority class for this variable. So what happens if we remove it?

```
# New random forest without null values in b_id
nonulls <- noseller[noseller$b_id != 0, ]
rf.model4 <- randomForest(click ~ ., data = nonulls, importance = TRUE,
prox = TRUE,
    strata = nonulls$click, sampsize = c(10, 10))
rf.model4
```

```
##
## Call:
##  randomForest(formula = click ~ ., data = nonulls, importance =
TRUE,      prox = TRUE, strata = nonulls$click, sampsize = c(10, 10))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 42.83%
## Confusion matrix:
##     0   1 class.error
## 0 267 194      0.4208
## 1  24  24      0.5000
```

```
result <- importance(rf.model4, )[, "MeanDecreaseAccuracy"]
importance(rf.model4)[order(result, decreasing = TRUE), ]
```

```
##                  0        1 MeanDecreaseAccuracy MeanDecreaseGini
## b_id       1.04063  1.65759               1.4848           1.6480
## t_id       1.21699  0.20917               1.3126           1.0666
## country    1.00100  0.00000               1.0010           0.0032
## wday       0.43049 -0.00269               0.4125           0.8254
## srcs      -0.01367  2.02503               0.4084           1.2027
## v_id       0.41046 -0.59283               0.2935           0.4907
## state     -0.22732 -2.66343              -0.7488           0.3561
## hour      -0.79484 -0.30514              -0.8560           1.3164
## referrals -2.71990  4.77042              -1.4052           1.5247
## views     -1.62686  0.29365              -1.6943           1.5056
```

b_id still remains high, and our class error rate worsened noticeably. So it is probably not good idea to remove the records with null values. We should keep those records, but we shouldn't use this variable because of the class imbalance.

```
# New random forest without b_id
noBID <- noseller
noBID$b_id <- NULL
rf.model5 <- randomForest(click ~ ., data = noBID, importance = TRUE,
prox = TRUE,
    strata = noBID$click, sampsize = c(10, 10))
rf.model5
```

```
##
## Call:
##  randomForest(formula = click ~ ., data = noBID, importance =
TRUE,      prox = TRUE, strata = noBID$click, sampsize = c(10, 10))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 45.76%
## Confusion matrix:
##     0   1 class.error
## 0 772 679       0.468
## 1  49  91       0.350
```

```
result <- importance(rf.model5, )[, "MeanDecreaseAccuracy"]
importance(rf.model5)[order(result, decreasing = TRUE), ]
```

```
##                0      1 MeanDecreaseAccuracy MeanDecreaseGini
## srcs       3.1369  4.142               3.8784         1.536105
## referrals  1.0602  5.298               2.0870         1.871308
## views      1.2387  4.362               1.9245         2.019903
## t_id       1.1309 -1.811               0.9382         1.149385
## state      0.9391 -1.888               0.8159         0.278711
## v_id       0.4579  1.626               0.7614         0.439901
## wday      -0.3111 -2.171              -0.8717         0.930050
## country   -1.0010  0.000              -1.0010         0.003867
## hour      -0.7312 -2.829              -1.4010         1.650569
```

The important variables are now: srcs, referrals, views, t_id, and state. This seems to make more intuitive sense.

Now let's see how much accuracy we get on the test set.

```
# drop seller, b_id from the test set
test <- test[, !(names(test) %in% c("seller", "b_id"))]
# make prediction from the test set
test.pred5 <- predict(rf.model5, test[, -10])
# compare it to the actual outcome
confusionMatrix <- table(observed = test$click, predicted = test.pred5)
confusionMatrix
```

```
##         predicted
## observed   0   1
##        0 382 337
##        1  26  51
```

```
# class error rate of click=1
confusionMatrix[2, 1]/sum(confusionMatrix[2, ])
```

```
## [1] 0.3377
```

Now let's make prediction with validation set

```
# drop seller, b_id from the validation set
validation <- validation[, !(names(validation) %in% c("seller",
"b_id"))]
# make prediction from the validation set
validation.pred5 <- predict(rf.model5, validation[, -10])
# compare it to the actual outcome
confusionMatrix <- table(observed = validation$click, predicted =
validation.pred5)
confusionMatrix
```

```
##         predicted
## observed    0    1
##        0 8582 7072
##        1    1    3
```

```
# class error rate of click=1
confusionMatrix[2, 1]/sum(confusionMatrix[2, ])
```

```
## [1] 0.25
```

So we didn't lose our predictive power, either.