# Notes on the Buzz data

Nina Zumel and John Mount

Win-Vector LLC

10-25-2013

To run this example you need a system with R installed (see `http://cran.r-project.org`), Latex (see `http://tug.org`) and data from `https://github.com/WinVector/zmPDSwR/tree/master/Buzz`.

To run this example:

1. Download buzz.Rns and TomsHardware-Relative-Sigma-500.data.txt from the github URL.
2. Start a copy of R, use `setwd()` to move to the directory you have stored the files.
3. Make sure knitr is loaded into R ( `install.packages('knitr')` and `library(knitr)` ).
4. In R run: (produces buzz.tex from buzz.Rnw).

```
knit('buzz.Rnw')
system('pdflatex buzz.tex')
```

Now you can run the following data prep steps:

```
buzzdata <- read.table("TomsHardware-Relative-Sigma-500.data.txt",
                       header=F, sep=",")
makevars <- function(colname, ndays=7) {
  paste(colname, 0:ndays, sep='')
}
varnames <- c("num.new.disc",
             "burstiness",
             "number.total.disc",
             "auth.increase",
             "atomic.containers", # not documented
             "num.displays", # number of times topic displayed to user (measure of interest,
             "contribution.sparseness", # not documented
             "avg.auths.per.disc",
             "num.authors.topic", # total authors on the topic
             "avg.disc.length",
             "attention.level.author",
             "attention.level.contrib"
)
colnames <- as.vector(sapply(varnames, FUN=makevars))
colnames <-  c(colnames, "buzz")
colnames(buzzdata) <- colnames
# Split into training and test
set.seed(2362690L)
rgroup <- runif(dim(buzzdata)[1])
buzztrain <- buzzdata[rgroup > 0.1,]
buzztest <- buzzdata[rgroup <=0.1,]
```

This currently returns a training set with 7114 rows and a test set with 791 rows, which is the same as when this document was prepared.

Notice we have exploded the basic column names into the following:

```
print(colnames)

##  [1] "num.new.disc0"            "num.new.disc1"
##  [3] "num.new.disc2"            "num.new.disc3"
##  [5] "num.new.disc4"            "num.new.disc5"
##  [7] "num.new.disc6"            "num.new.disc7"
##  [9] "burstiness0"              "burstiness1"
## [11] "burstiness2"              "burstiness3"
## [13] "burstiness4"              "burstiness5"
## [15] "burstiness6"              "burstiness7"
## [17] "number.total.disc0"       "number.total.disc1"
## [19] "number.total.disc2"       "number.total.disc3"
## [21] "number.total.disc4"       "number.total.disc5"
## [23] "number.total.disc6"       "number.total.disc7"
## [25] "auth.increase0"           "auth.increase1"
## [27] "auth.increase2"           "auth.increase3"
## [29] "auth.increase4"           "auth.increase5"
## [31] "auth.increase6"           "auth.increase7"
## [33] "atomic.containers0"       "atomic.containers1"
## [35] "atomic.containers2"       "atomic.containers3"
## [37] "atomic.containers4"       "atomic.containers5"
## [39] "atomic.containers6"       "atomic.containers7"
## [41] "num.displays0"            "num.displays1"
## [43] "num.displays2"            "num.displays3"
## [45] "num.displays4"            "num.displays5"
## [47] "num.displays6"            "num.displays7"
## [49] "contribution.sparseness0" "contribution.sparseness1"
## [51] "contribution.sparseness2" "contribution.sparseness3"
## [53] "contribution.sparseness4" "contribution.sparseness5"
## [55] "contribution.sparseness6" "contribution.sparseness7"
## [57] "avg.auths.per.disc0"      "avg.auths.per.disc1"
## [59] "avg.auths.per.disc2"      "avg.auths.per.disc3"
## [61] "avg.auths.per.disc4"      "avg.auths.per.disc5"
## [63] "avg.auths.per.disc6"      "avg.auths.per.disc7"
## [65] "num.authors.topic0"       "num.authors.topic1"
## [67] "num.authors.topic2"       "num.authors.topic3"
## [69] "num.authors.topic4"       "num.authors.topic5"
## [71] "num.authors.topic6"       "num.authors.topic7"
## [73] "avg.disc.length0"         "avg.disc.length1"
## [75] "avg.disc.length2"         "avg.disc.length3"
## [77] "avg.disc.length4"         "avg.disc.length5"
## [79] "avg.disc.length6"         "avg.disc.length7"
## [81] "attention.level.author0"  "attention.level.author1"
## [83] "attention.level.author2"  "attention.level.author3"
## [85] "attention.level.author4"  "attention.level.author5"
## [87] "attention.level.author6"  "attention.level.author7"
## [89] "attention.level.contrib0" "attention.level.contrib1"
## [91] "attention.level.contrib2" "attention.level.contrib3"
## [93] "attention.level.contrib4" "attention.level.contrib5"
## [95] "attention.level.contrib6" "attention.level.contrib7"
## [97] "buzz"
```

We are now ready to create a simple model predicting “buzz” as function of the other columns.

```
# build a model
# let's use all the input variables
nlist = varnames
varslist = as.vector(sapply(nlist, FUN=makevars))
# these were defined previously,  in Chapter 9
loglikelihood <- function(y, py) {
  pysmooth <- ifelse(py==0, 1e-12,
                     ifelse(py==1, 1-1e-12, py))
  sum(y * log(pysmooth) + (1-y)*log(1 - pysmooth))
}
accuracyMeasures <- function(pred, truth, threshold=0.5, name="model") {
  dev.norm <- -2*loglikelihood(as.numeric(truth), pred)/length(pred)
  ctable = table(truth=truth,
                 pred=pred)
  accuracy <- sum(diag(ctable))/sum(ctable)
  precision <- ctable[2,2]/sum(ctable[,2])
  recall <- ctable[2,2]/sum(ctable[2,])
  f1 <- precision*recall
  print(paste("precision=", precision, "; recall=" , recall))
  print(ctable)
  data.frame(model=name, accuracy=accuracy, f1=f1, dev.norm)
}
library(randomForest)
```

```
## randomForest 4.6-7
## Type rfNews() to see new features/changes/bug fixes.
```

```
bzFormula <- paste('as.factor(buzz) ~ ',paste(varslist,collapse=' + '))
fmodel <- randomForest(as.formula(bzFormula),
                      data=buzztrain,
                      ntree=101,
                      mtry=floor(sqrt(length(varslist))),
                      importance=T)
rframe <- data.frame(truth=buzztrain$buzz, pred=predict(fmodel, newdata=buzztrain))
print(with(rframe,table(truth=truth,pred=pred)))
```

```
##      pred
## truth    0    1
##     0 5550    0
##     1    1 1563
```

```
rtest <- data.frame(truth=buzztest$buzz, pred=predict(fmodel, newdata=buzztest))
print(with(rtest,table(truth=truth,pred=pred)))
```

```
##      pred
## truth   0   1
##     0 584  30
##     1  29 148
```

```
print(accuracyMeasures(rframe$pred, rframe$truth))
```

```
## [1] "precision= 1 ; recall= 0.999360613810742"
##      pred
## truth    0    1
##     0 5550    0
##     1    1 1563
##   model accuracy     f1 dev.norm
## 1 model   0.9999 0.9994 0.007768
```

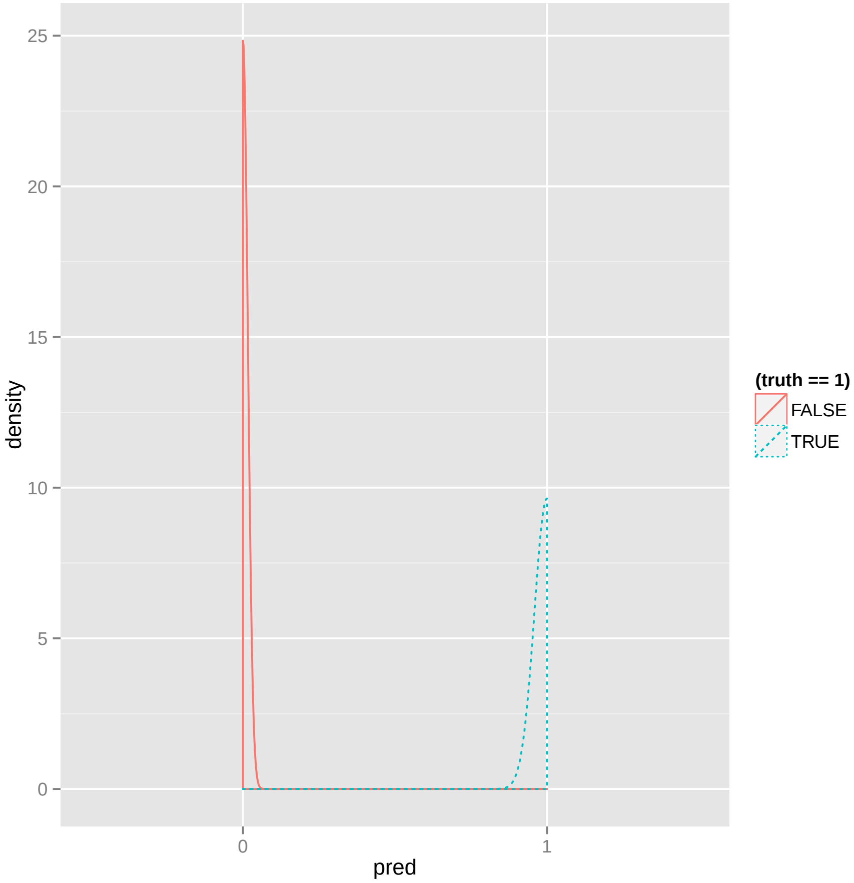
```
print(accuracyMeasures(rtest$pred, rtest$truth))
```

```
## [1] "precision= 0.831460674157303 ; recall= 0.836158192090395"
##      pred
## truth   0   1
##     0 584  30
##     1  29 148
##   model accuracy     f1 dev.norm
## 1 model   0.9254 0.6952    4.122
```

And we can also make plots (though in this case the classification scores are so concentrated near zero and one the plot's smoothing makes for a slightly deceptive presentation).

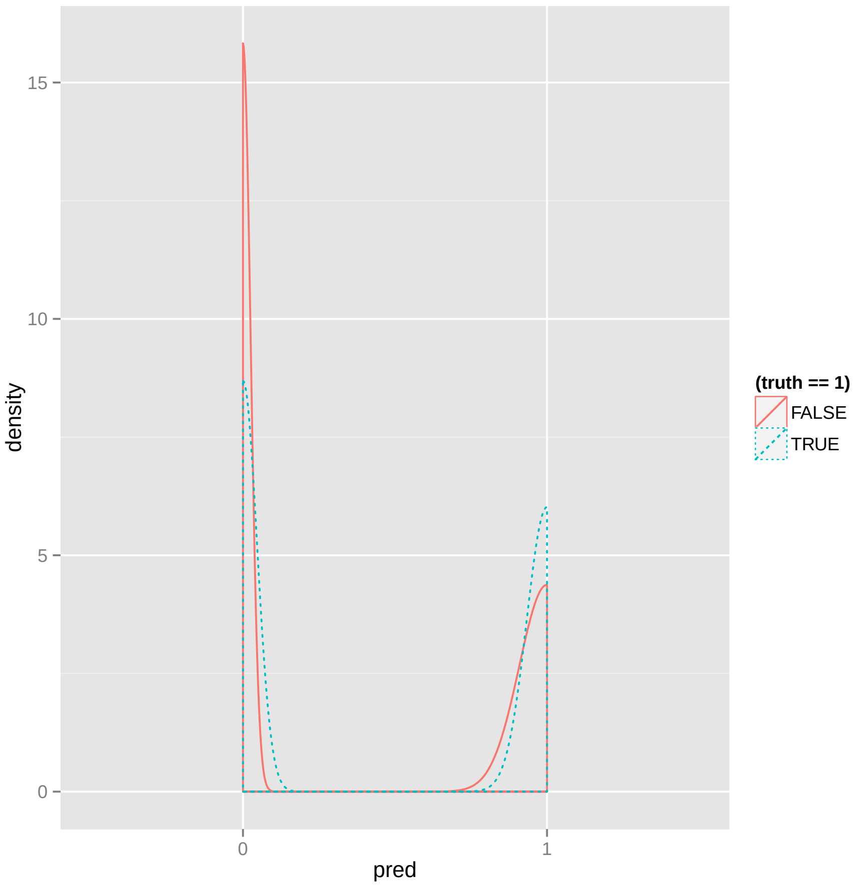
Training performance:

```
library(ggplot2)
ggplot(rframe, aes(x=pred, color=(truth==1),linetype=(truth==1))) +
   geom_density(adjust=0.1,)
```



Test performance:

```
ggplot(rtest, aes(x=pred, color=(truth==1),linetype=(truth==1))) +
   geom_density(adjust=0.1)
```


15
10
5
0
density
0
1
pred
(truth == 1)
FALSE
TRUE