Will it Burn?

Pablo Portillo Garrigues

May 20, 2019

Contents

1 Introduction

The main goal on this notebook is to predict on real time wheter or not a post of reddit will get to the hot sections of its subreddit.

Reddit is a popular social network with millions of daily users and hundred of thousands of posts. It would bee interesting to develope a tool that allowed us to differentiate in real time between potential succesfull post and forgotten posts. Reddit has an easy to use API called Praw.

Due to the reasons described above we have chosen reddit as the playground for our machine learning activities.

We will also build a bot who will comment on a post that we predict to be a potenital hot post.

2 Read the data

In the notebook we will preprocess just a sample of the data so that we can upload it to gitHub

```
dat <- read.csv("data/dat.csv")
```

3 Exploratory Analysis

```
dat <- dat[!(is.na(dat$score)), ]
dim(dat)
```

```
## [1] 1364   16
```

```
colnames(dat)
```

```
##  [1] "title"        "author"       "edited"       "is_self"
##  [5] "locked"       "spoiler"      "stickied"     "score"
##  [9] "upvote_ratio" "id"           "subreddit"    "url"
## [13] "num_comments" "body"         "created"      "measured"
```

```
length(unique(dat$id))
```

```
## [1] 155
```

```
dat[1, ]
```

```
##        title        author edited is_self locked spoiler stickied score
## 1 Thats deep EveKilledAdam  False   FALSE  FALSE   FALSE    FALSE    17
##   upvote_ratio     id subreddit                                 url
## 1          0.9 c1kurf     funny https://i.redd.it/p8s2vngshv431.jpg
##   num_comments body    created   measured
## 1            1    0 1560758153 1560759475
```

4 Preprocessing the data

4.1 Creating new time variables

Lets add some varaibles related to time such as: time since creation, hour of the day and day of the week.

```
dat$tsc <- dat$measured - dat$created
dat$hc <- substr(anytime(dat$created), 12, 13)
dat$wd <- weekdays(as.Date(substr(anytime(dat$created), 1, 20),
    "%Y-%m-%d"), abbreviate = TRUE)
dat$is_self <- ifelse(dat$is_self == "FALSE", "False", dat$is_self)
dat$is_self <- ifelse(dat$is_self == "TRUE", "True", dat$is_self)
```

4.2 Dealing with the URLs

The objective of this function is to classify the file that its beeing posted. The alternatives are: images, videos, gifs, news, self and other.

```r
whatIsIt <- function(dat) {
    dat$isImage <- rep(0, nrow(dat))
    dat[grep("(.png)|(.jpg)", dat$url, ignore.case = TRUE), ]$isImage <- 1
    dat$isVideo <- rep(0, nrow(dat))
    dat[grep("(youtu.be)|(v.redd)|(.mp4)", dat$url, ignore.case = TRUE),
        ]$isVideo <- 1
    dat$isNews <- rep(0, nrow(dat))
    dat[grep("news", dat$url, ignore.case = TRUE), ]$isNews <- 1
    dat$isGif <- rep(0, nrow(dat))
    dat[grep("gif", dat$url, ignore.case = TRUE), ]$isGif <- 1
    dat$isOther <- rep(1, nrow(dat))
    dat$isOther <- ifelse((dat$isImage + dat$isVideo + dat$isNews +
        dat$isSelf + dat$isGif) > 0, 0, 1)
    dat
}
```

4.3 Grouping rows of the same post

We are going to take the measurement score and number of comments closer to 5 and 10 minutes since the creation of the post. The following function does the former description. If your computer is able it is useful to compute it in parallel.

```r
findCloser <- function(dat, time, paralelo = 0) {
    ids <- unique(dat$id)
    if (paralelo == 1) {
        library(foreach)
        library(doParallel)
        no_cores <- detectCores() - 2
        splits <- split(ids, ceiling(seq_along(ids)/no_cores))
        cl <- makeCluster(no_cores)
        registerDoParallel(cl)
        aux <- foreach(splite = splits, .combine = rbind) %dopar%
            {
                library(dplyr)
                ids <- splite
                res <- lapply(ids, function(x) {
                  aux <- dat[dat$id == x, ]
                  auxT <- lapply(time, function(y) {
                    row <- which.min(abs(aux$tsc - y))
                    dist <- ((aux[row, ]$tsc - y))
                    auxW <- aux[row, colnames(aux) %in% c("id",
                      "score", "upvote_ratio", "num_comments")]
                    auxW <- auxW[, c(3, 1, 2, 4)]
                    auxW$dist <- dist
                    colnames(auxW) <- c("id", paste("score",
                      (y/60), sep = "_"), paste("upvote_ratio",
                      (y/60), sep = "_"), paste("num_comments",
                      (y/60), sep = "_"), paste("dist", (y/60),
                      sep = "_"))
                    auxW
                  })
                  aux <- bind_cols(auxT)
                aux
            })
            res <- bind_rows(res)
            res
        }
        stopCluster(cl)
        res <- aux
    } else {
        res <- lapply(ids, function(x) {
            aux <- dat[dat$id == x, ]
            auxT <- lapply(time, function(y) {
                row <- which.min(abs(aux$tsc - y))
                dist <- ((aux[row, ]$tsc - y))
                auxW <- aux[row, colnames(aux) %in% c("id", "score",
                  "upvote_ratio", "num_comments")]
                auxW <- auxW[, c(3, 1, 2, 4)]
                auxW$dist <- dist
                colnames(auxW) <- c("id", paste("score", (y/60),
                  sep = "_"), paste("upvote_ratio", (y/60), sep = "_"),
                  paste("num_comments", (y/60), sep = "_"), paste("dist",
                    (y/60), sep = "_"))
                auxW
            })
            aux <- bind_cols(auxT)
            aux
        })
        res <- bind_rows(res)
    }
    res
}
```

We will proceed to group our data by posts and we will take the following variables: the hour the post was created (hc), the day that was created (wd), the number of character of the title (title), the number of character of the body and the subreddit.

We will have 2 datasets, one for the classification task of predict wheter or not a post will get to hot; and another one to predict when will it get the hot section.

```r
wholeDat <- dat
dat <- dat[dat$tsc < 900, ]
hot <- read.csv("data/whosHot.csv")
datTimes <- as.data.frame(dat %>% group_by(id) %>% summarise(max = max(tsc),
    min = min(tsc)))
datTimes <- merge(datTimes, hot, by = "id", all.x = TRUE)
aux <- as.data.frame(dat %>% group_by(id) %>% summarise(title = length(nchar(as.character(unique(title)
datM <- as.data.frame(dat[!(dat$id %in% aux[aux$title > 1, ]$id),
    ] %>% group_by(id) %>% summarise(hc = unique(hc), wd = unique(wd),
    title = nchar(as.character(unique(title))), body = unique(body)[1],
    subreddit = unique(subreddit), url = unique(url), isSelf = ifelse(unique(is_self) ==
        "True", 1, 0)))
aux <- data.frame(id = datM$id, sep = sample(c(0, 1, 2), nrow(datM),
    replace = TRUE))
datT1 <- findCloser(dat[dat$id %in% aux[aux$sep == 0, ]$id, ],
    c(60, 120, 180, 240, 300, 360, 420, 480, 540, 600, 660, 720,
        780, 840, 900), paralelo = 1)
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
datT2 <- findCloser(dat[dat$id %in% aux[aux$sep == 1, ]$id, ],
    c(60, 120, 180, 240, 300, 360, 420, 480, 540, 600, 660, 720,
        780, 840, 900), paralelo = 1)
datT3 <- findCloser(dat[dat$id %in% aux[aux$sep == 2, ]$id, ],
    c(60, 120, 180, 240, 300, 360, 420, 480, 540, 600, 660, 720,
        780, 840, 900), paralelo = 1)
datT1 <- datT1[, -c(6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56,
    61, 66, 71)]
datT2 <- datT2[, -c(6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56,
    61, 66, 71)]
datT3 <- datT3[, -c(6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56,
    61, 66, 71)]
datT <- rbind(datT1, datT2, datT3)
```

We will calculate from the score and upvote_ratio variables the number of upvotes and downvotes each post have.

Once we have grouped our data by posts we will get the information from the url described before aswell as som statistics from each subreddit.

```
datM <- whatIsIt(datM)

datMT <- datM[datM$id %in% datTimes[datTimes$min < 120, ]$id,
    ]

datF <- merge(datT, datMT[, -7], by = "id")


newVariables <- function(dat) {

    dat0 <- dat[dat$target == 0, ]
    dat <- dat[, !(colnames(dat) %in% "target")]
    vars <- c("score", "upvote_ratio", "num_comments", "posScore",
        "negScore", "comments_votes_ratio")
    for (i in vars) {
        variant <- colnames(dat)[grep(pattern = i, colnames(dat))]
        for (j in variant) {
            # print(j)
            names <- c(paste("min", j, sep = "_"), paste("max",
                j, sep = "_"), paste("mean", j, sep = "_"), paste("sd",
                j, sep = "_"), paste("quantiles", j, "0.25",
                sep = "_"), paste("quantiles", j, "0.5", sep = "_"),
                paste("quantiles", j, "0.75", sep = "_"))
            aux <- dat0[, c("subreddit", j)]
            aux <- as.data.frame(aux %>% group_by(subreddit) %>%
                summarise(min(aux[aux$subreddit == subreddit,
                  2]), max(aux[aux$subreddit == subreddit, 2]),
```

```r
                  mean(aux[aux$subreddit == subreddit, 2]), sd(aux[aux$subreddit ==
                    subreddit, 2]), quantile(aux[aux$subreddit ==
                    subreddit, 2], 0.25), quantile(aux[aux$subreddit ==
                    subreddit, 2], 0.5), quantile(aux[aux$subreddit ==
                    subreddit, 2], 0.75)))
            colnames(aux)[2:ncol(aux)] <- names
            dat <- merge(dat, aux, by = "subreddit")

        }
    }
    dat
}



likesDislikes <- function(dat) {
    names <- colnames(dat)[grep("score", colnames(dat))]
    names <- substr(names, 7, nchar(names))
    for (i in 1:length(names)) {

        dat[, paste("posScore", names[i], sep = "_")] <- round((dat[,
            paste("score", names[i], sep = "_")] * dat[, paste("upvote_ratio",
            names[i], sep = "_")])/(2 * datF[, paste("upvote_ratio",
            names[i], sep = "_")] - 1))
        dat[, paste("posScore", names[i], sep = "_")] <- ifelse(is.na(dat[,
            paste("posScore", names[i], sep = "_")]), 0, dat[,
            paste("posScore", names[i], sep = "_")])
        dat[, paste("negScore", names[i], sep = "_")] <- dat[,
            paste("posScore", names[i], sep = "_")] - dat[, paste("score",
            names[i], sep = "_")]
        votes <- (dat[, paste("posScore", names[i], sep = "_")] +
            dat[, paste("negScore", names[i], sep = "_")])

    }
    dat
}

datF <- likesDislikes(datF)


aux <- merge(datF, hot, by = "id", all.x = TRUE)
colnames(aux)[length(aux)] <- "target"
aux$target <- ifelse(is.na(aux$target), 0, 1)
datF <- suppressWarnings(newVariables(aux))


## For test data##

# hist <- train[match(unique(train$subreddit),
# train$subreddit),] hist<-read.csv('hist.csv')
# hist10<-read.csv('hist10.csv')
# hist15<-read.csv('hist15.csv')
# datF<-merge(datF,hist,by='subreddit')
# datF10<-merge(datF10,hist10,by='subreddit')
# datF15<-merge(datF15,hist15,by='subreddit')

## For test data##
```

We will now find what type of post is the most common in each subreddit.

We will transform each column to the type we are interested and we willl merge the data with the target variabe "hot".

```
tip <- c("gif", "self", "image", "video", "news", "other")

aux <- as.data.frame(datF %>% group_by(subreddit) %>% summarise(tipFav = tip[which.max(c(sum(isGif),
    sum(isSelf), sum(isImage), sum(isVideo), sum(isNews), sum(isOther)))]))

datF <- merge(datF, aux, by = "subreddit")

datF$subreddit <- as.factor(datF$subreddit)
datF$hc <- as.integer(datF$hc)
datF$wd <- as.factor(datF$wd)
datF$isImage <- as.factor(as.character(datF$isImage))
datF$isSelf <- as.factor(as.character(datF$isSelf))
datF$isVideo <- as.factor(as.character(datF$isVideo))
datF$isNews <- as.factor(as.character(datF$isNews))
datF$isOther <- as.factor(as.character(datF$isOther))
datF$isGif <- as.factor(as.character(datF$isGif))
datF$tipFav <- as.factor(datF$tipFav)


train <- datF
train <- merge(train, hot, by = "id", all.x = TRUE)
colnames(train)[length(train)] <- "target"
train$target <- ifelse(is.na(train$target), 0, train$target)
```

We would need to save the classes of our variables aswell as the levels of the factors so that when we do live predictions we can easily transform the new data into the format of our trainning.

We now have our training data.

5 Processed training data

But first, lets take a look at the training data.

```
train <- fread("data/train.csv", data.table = FALSE)
dim(train)
```

```
## [1] 241889    630
```

As we can see this dataset includes iformation from the user who posted the pst. We have decided not using this information due to the time it takes to scrap it.

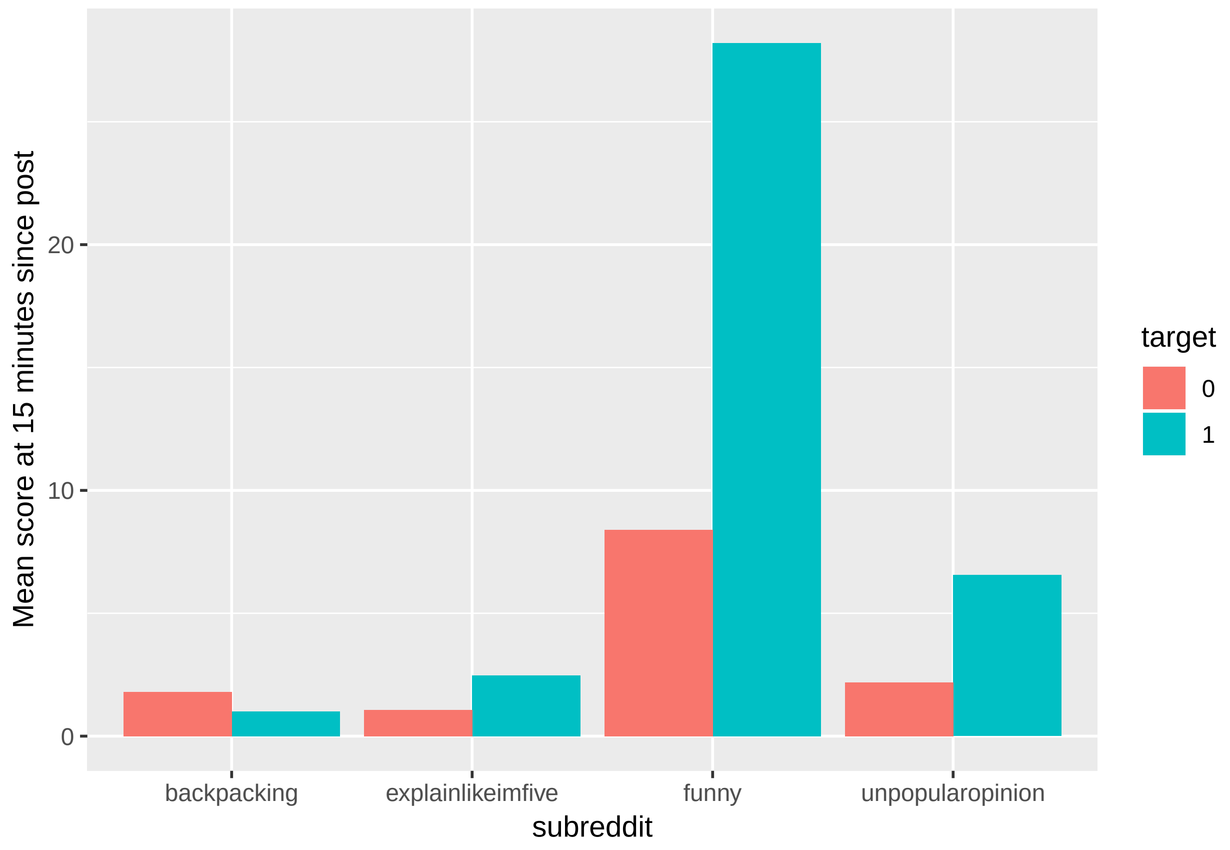
As we have discussed before we have had to kept the classes and levels of our variables so that we always process our data in the same way.

```r
changeClass <- function(dat, vars, classes) {
    for (i in 1:length(classes)) {
        if (!(vars[i] %in% colnames(dat)))
            next
        if (classes[i] == "numeric") {
            dat[, vars[i]] <- as.numeric(as.character(dat[, vars[i]]))
        }
        if (classes[i] == "integer") {
            dat[, vars[i]] <- as.integer(as.character(dat[, vars[i]]))
        }
        if (classes[i] == "factor") {
            dat[, vars[i]] <- as.factor(as.character(dat[, vars[i]]))
        }
        if (classes[i] == "character") {
            dat[, vars[i]] <- as.character(dat[, vars[i]])
        }
    }
    dat
}
addLvlTrain <- function() {
    gifLvl <- readLines("data/isGiflevels.txt")
    imgLvl <- readLines("data/isImagelevels.txt")
    newLvl <- readLines("data/isNewslevels.txt")
    othLvl <- readLines("data/isOtherlevels.txt")
    slfLvl <- readLines("data/isSelflevels.txt")
    vidLvl <- readLines("data/isVideolevels.txt")
    subLvl <- readLines("data/subredditlevels.txt")
    tpfLvl <- readLines("data/tipFavlevels.txt")
    wkdLvl <- readLines("data/wdlevels.txt")

    res <- list(isGif = gifLvl, isImage = imgLvl, isNews = newLvl,
        isOther = othLvl, isSelf = slfLvl, isVideo = vidLvl,
        subreddit = subLvl, tipFav = tpfLvl, wd = wkdLvl)


    res

}


vars <- readLines("data/dataVariables.txt")
classes <- readLines("data/dataClasses.txt")
train <- changeClass(train, vars, classes)
trainC <- train
trainC$target <- as.factor(ifelse(trainC$target > 0, 1, 0))
```

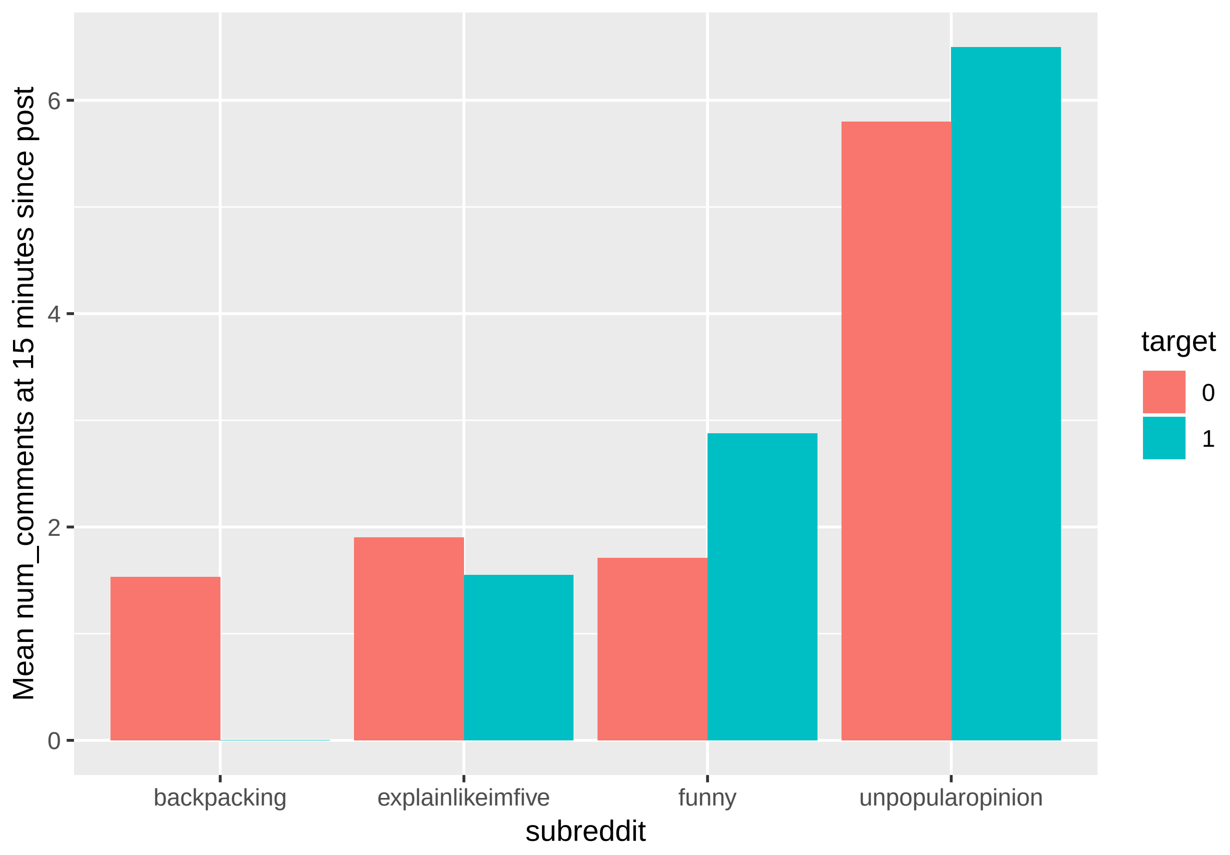
5.1 Plots of interest of processed data

Lets explore a little bit more our proccesed data. We will use 4 subreddits based on the number of suscribers: unpopularopinion with 600K, funny with 21M, explainlikeimfive with 16M and backpacking with 1.3M.

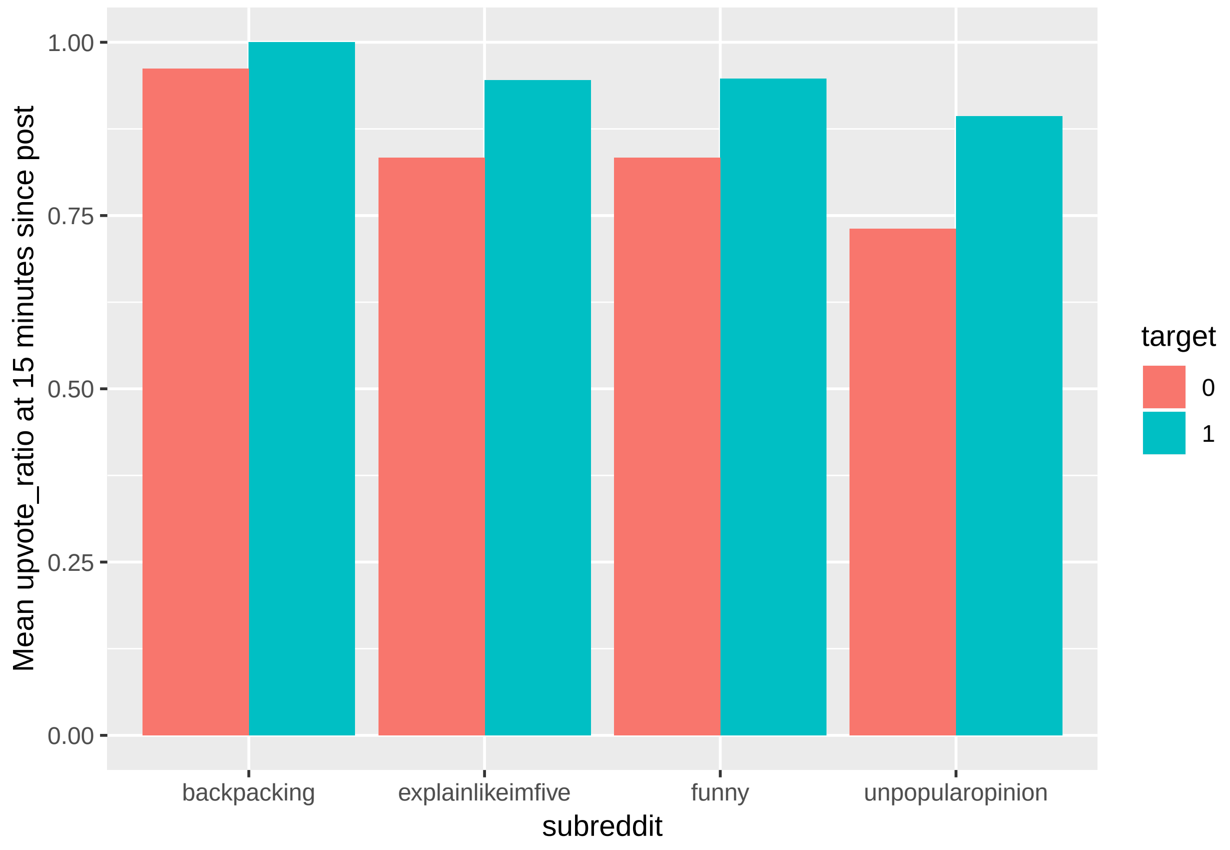
```r
trainC[trainC$subreddit %in% c("unpopularopinion", "funny", "backpacking",
    "explainlikeimfive"), ] %>% group_by(subreddit, target) %>%
    summarise(value = mean(score_15)) %>% ggplot(aes(x = subreddit,
    y = value, fill = target)) + geom_bar(stat = "identity",
    position = "dodge") + labs(y = "Mean score at 15 minutes since post")
```



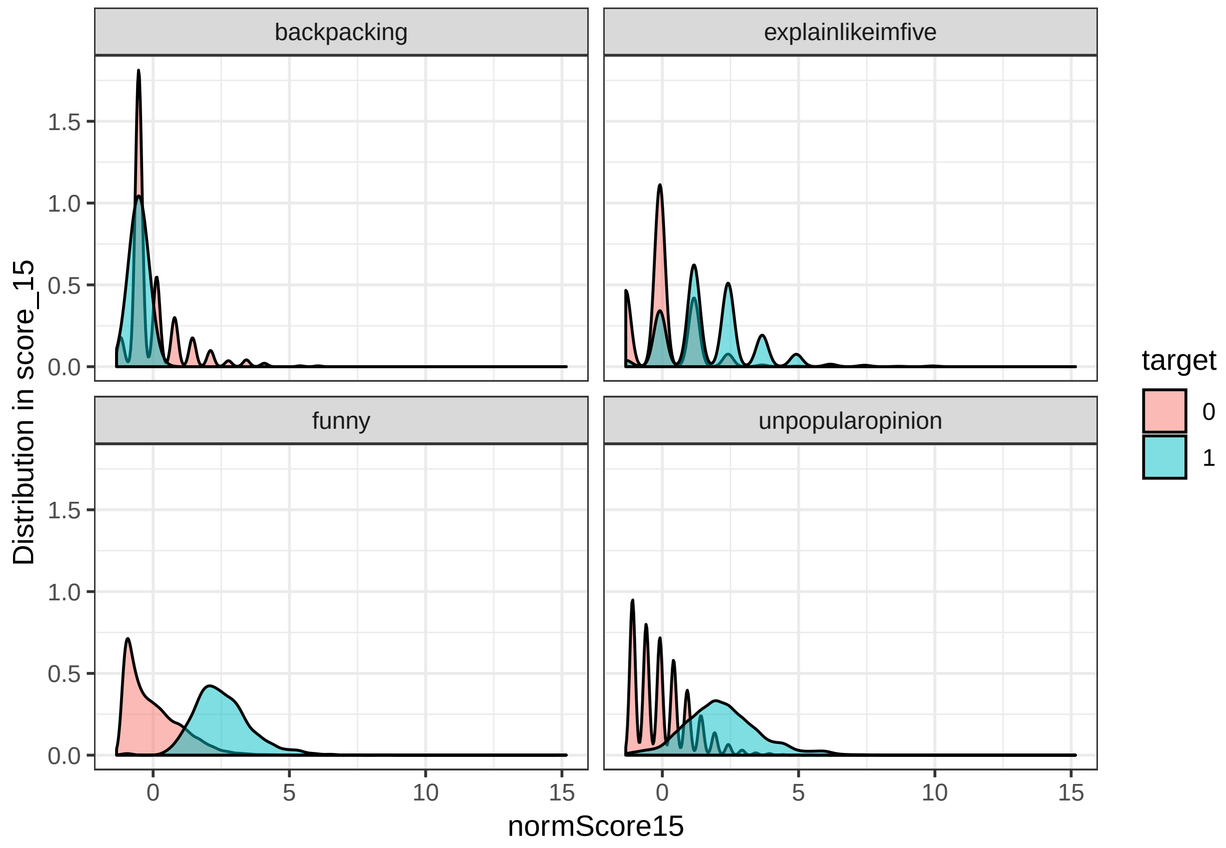
```r
trainC[trainC$subreddit %in% c("unpopularopinion", "funny", "backpacking",
    "explainlikeimfive"), ] %>% group_by(subreddit, target) %>%
    summarise(value = mean(num_comments_15)) %>% ggplot(aes(x = subreddit,
    y = value, fill = target)) + geom_bar(stat = "identity",
    position = "dodge") + labs(y = "Mean num_comments at 15 minutes since post")
```



```
trainC[trainC$subreddit %in% c("unpopularopinion", "funny", "backpacking",
    "explainlikeimfive"), ] %>% group_by(subreddit, target) %>%
    summarise(value = mean(upvote_ratio_15)) %>% ggplot(aes(x = subreddit,
    y = value, fill = target)) + geom_bar(stat = "identity",
    position = "dodge") + labs(y = "Mean upvote_ratio at 15 minutes since post")
```



```
aux <- trainC
aux$normScore15 <- (trainC$score_15 - trainC$mean_score_15)/trainC$sd_score_15
ggplot(aux[aux$subreddit %in% c("unpopularopinion", "funny",
    "backpacking", "explainlikeimfive"), ], aes(x = normScore15,
    fill = target)) + geom_density(alpha = 0.5) + theme_bw() +
    labs(y = "Distribution in score_15") + facet_wrap(~subreddit)
```



6 Creating the models

Having processed our data we are ready to search the parameters that best fit our data!

We will use an Extreme Gradient Boosting which hyperparameters will be searched using a genetic algorithm called Ant Colony Optimization.

The hyperparameters obtained are:

We will train the model:

```
paramList<-c(0.0379022295040808,1437,4,0.0438020526291911,0.772582709563033,0.25)
 xgb_params <- list("objective" = "binary:logistic",

                    "eval_metric" = "auc",

                    eta=paramList[1], gamma=paramList[4], max_depth=paramList[3], subsample=paramList[
 trainC<-train
 trainC$target=ifelse(trainC$target>0,1,0)
lvls<-addLvlTrain()
```

```
## Warning in readLines("data/subredditlevels.txt"): incomplete final line
## found on 'data/subredditlevels.txt'
```

```r
  new_tr <- model.matrix(~.+0,data = trainC[,!(colnames(trainC) %in% c("target","id","targetScore"))],x


  dtrain <- xgb.DMatrix(data = new_tr,label = trainC$target)

    xgbC <- xgb.train(params= xgb_params, data = dtrain, nrounds = paramList[2], print.every.n = 10)
```

```
## Warning: 'print.every.n' is deprecated.
## Use 'print_every_n' instead.
## See help("Deprecated") and help("xgboost-deprecated").
```

6.1 Testing the model on new data

We will test our model on new data:

```r
dat <- fread("data/datTest.csv", data.table = FALSE)

dat$tsc <- dat$measured - dat$created
dat$hc <- substr(anytime(dat$created), 12, 13)
dat$wd <- weekdays(as.Date(substr(anytime(dat$created), 1, 20),
    "%Y-%m-%d"), abbreviate = TRUE)
dat$is_self <- ifelse(dat$is_self == "FALSE", "False", dat$is_self)
dat$is_self <- ifelse(dat$is_self == "TRUE", "True", dat$is_self)
wholeDat <- dat
dat <- dat[dat$tsc < 900, ]
hot <- read.csv("data/whosHot.csv")
datTimes <- as.data.frame(dat %>% group_by(id) %>% summarise(max = max(tsc),
    min = min(tsc)))
datTimes <- merge(datTimes, hot, by = "id", all.x = TRUE)
aux <- as.data.frame(dat %>% group_by(id) %>% summarise(title = length(nchar(as.character(unique(title)
datM <- as.data.frame(dat[!(dat$id %in% aux[aux$title > 1, ]$id),
    ] %>% group_by(id) %>% summarise(hc = unique(hc), wd = unique(wd),
    title = nchar(as.character(unique(title))), body = unique(body)[1],
    subreddit = unique(subreddit), url = unique(url), isSelf = ifelse(unique(is_self) ==
        "True", 1, 0)))
aux <- data.frame(id = datM$id, sep = sample(c(0, 1, 2), nrow(datM),
    replace = TRUE))
datT1 <- findCloser(dat[dat$id %in% aux[aux$sep == 0, ]$id, ],
    c(60, 120, 180, 240, 300, 360, 420, 480, 540, 600, 660, 720,
        780, 840, 900), paralelo = 1)
datT2 <- findCloser(dat[dat$id %in% aux[aux$sep == 1, ]$id, ],
    c(60, 120, 180, 240, 300, 360, 420, 480, 540, 600, 660, 720,
        780, 840, 900), paralelo = 1)
datT3 <- findCloser(dat[dat$id %in% aux[aux$sep == 2, ]$id, ],
    c(60, 120, 180, 240, 300, 360, 420, 480, 540, 600, 660, 720,
        780, 840, 900), paralelo = 1)
datT1 <- datT1[, -c(6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56,
    61, 66, 71)]
datT2 <- datT2[, -c(6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56,
    61, 66, 71)]
datT3 <- datT3[, -c(6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56,
    61, 66, 71)]
```

```r
datT <- rbind(datT1, datT2, datT3)
datM <- whatIsIt(datM)

datMT <- datM[datM$id %in% datTimes[datTimes$min < 125, ]$id,
    ]

datF <- merge(datT, datMT[, -7], by = "id")


datF <- likesDislikes(datF)
hist <- fread("data/hist.csv", data.table = FALSE)

datF <- merge(datF, hist, by = "subreddit")

test <- datF
test <- merge(test, hot, by = "id", all.x = TRUE)
colnames(test)[length(test)] <- "target"
test$target <- ifelse(is.na(test$target), 0, test$target)

any(train$id %in% test$id)
```

```
## [1] FALSE
```

```r
test <- test[test$subreddit %in% unique(train$subreddit), ]
test <- changeClass(test, vars, classes)
testC <- test
testC$target <- ifelse(testC$target > 0, 1, 0)
new_ts <- model.matrix(~. + 0, data = testC[, !(colnames(testC) %in%
    c("target", "id"))], xlev = lvls)


dtest <- xgb.DMatrix(data = new_ts, label = testC$target)
preds <- predict(xgbC, dtest)
```

6.2 Results

```r
res <- data.frame(subreddit = testC$subreddit, target = testC$target,
    pred = preds)
as.data.frame(res %>% group_by(subreddit, target) %>% summarise(n(),
    thres_0.5 = sum(pred > 0.5), thres_0.6 = sum(pred > 0.6),
    thres_0.7 = sum(pred > 0.7), thres_0.8 = sum(pred > 0.8)))
```

```
##              subreddit target  n() thres_0.5 thres_0.6 thres_0.7 thres_0.8
## 1                  Art      0  522        12         5         3         1
## 2                  Art      1   50        29        19        15         8
## 3                  aww      0 3366         8         1         0         0
## 4                  aww      1   71         7         3         1         0
## 5          backpacking      0   49         0         0         0         0
## 6          backpacking      1    1         0         0         0         0
## 7                books      0  148         1         0         0         0
## 8                books     1   10          1          1          0          0
## 9       ChoosingBeggars    0  179          2          0          0          0
## 10      ChoosingBeggars    1   21          6          2          2          0
## 11               cringe    0   78          3          2          1          1
## 12               cringe    1    6          2          2          1          0
## 13          datascience    0   35          0          0          0          0
## 14                  DIY    0  132          4          4          1          1
## 15                  DIY    1    5          1          1          1          1
## 16            EarthPorn    0  299         13          9          6          1
## 17            EarthPorn    1   32         14         10          4          3
## 18 EatCheapAndHealthy      0   35          0          0          0          0
## 19 EatCheapAndHealthy      1    5          1          1          0          0
## 20   explainlikeimfive     0  279         22         16         11          6
## 21   explainlikeimfive     1   75         44         38         31         23
## 22               funny     0 2320         18          9          3          0
## 23               funny     1  119         30         21          6          1
## 24              gaming     0 1223         20         16          9          2
## 25              gaming     1   55         25         20          9          2
## 26                gifs     0  180          1          1          0          0
## 27                gifs     1    9          2          2          0          0
## 28              horror     0  201          1          0          0          0
## 29              horror     1   26          6          5          3          1
## 30         JapanTravel     0   67          0          0          0          0
## 31         JapanTravel     1    6          1          1          1          0
## 32               Jokes     0  757          2          2          2          1
## 33               Jokes     1   38         10          4          3          0
## 34         LifeProTips     0  145         11          6          3          1
## 35         LifeProTips     1   12          9          6          4          2
## 36                math     0   57          0          0          0          0
## 37                math     1    1          1          0          0          0
## 38              movies     0  402         14         10          5          0
## 39              movies     1   55         26         20         13          9
## 40               Music     0  814          9          3          3          0
## 41               Music     1   67         33         26         20         11
## 42 nevertellmetheodds      0   86          0          0          0          0
## 43                news     0  340          7          4          2          1
## 44                news     1   29          4          4          3          2
## 45        OutOfTheLoop     0   56          0          0          0          0
## 46            pcgaming     0  132          4          2          1          0
## 47            pcgaming     1   16          7          4          3          2
## 48                pics     0 1604         10          5          2          1
## 49                pics     1   42         13         12          4          2
## 50      PublicFreakout     0  200          3          1          0          0
## 51      PublicFreakout     1   24         12          8          4          1
## 52   quityourbullshit      0  135          1          0          0          0
## 53             science     0  123          0          0          0          0
## 54                 sex     0  543         33         16          2          1
## 55                 sex     1   82         28         25         23         10
## 56     Showerthoughts      0 3749          9          7          4          1
## 57     Showerthoughts      1   68         11          9          4          1
## 58              sports     0   49         15         12          5          1
## 59              sports     1   32         19         17         15         11
## 60            StarWars     0  112          3          3          0          0
## 61            StarWars     1   21          8          7          4          2
## 62         television     0  108         5         2         1         1
## 63         television     1   17         5         3         1         1
## 64      todayilearned     0  375        16         9         6         4
## 65      todayilearned     1   41        24        19        10         4
## 66             trashy     0  530         0         0         0         0
## 67             trashy     1   15         3         1         0         0
## 68   unpopularopinion     0 1582         5         2         2         1
## 69   unpopularopinion     1   52         8         3         0         0
## 70             videos     0 1419         1         1         0         0
## 71             videos     1   48         2         1         1         0
## 72          worldnews     0  572         8         6         1         0
## 73          worldnews     1   55        17        12         5         3
```

From the prediction table above we can see that the algorithm performs pretty different in function of the subreddit. It seems that the bigger a subreddit is the most likely it will have false positives. It might be interesting to use a bigger thresholds than 0.5 in osme subreddit such as "aww" where if th threshold is 0.5 it has 8FP/7TP and in 0.6 it has 1FP/3TP