kaggle report

Yiqun Xia

March 17, 2016

Preparation

We firstly load the data and remove non-predictors.

```
library(caret)

#read data and remove non-predictors
training <- read.csv("news_popularity_training.csv")[,-c(1,2,3)]
test.real <- read.csv("news_popularity_test.csv")[,-c(1,2,3)]
training <- training[-22686,]
```

We remove row 22686 as it has strange scale: ratios over 1000. Then we do some preprocess. We assume data are class balanced.

```
#remove n_non_stop_words
#since its low variance and 0 has already been included in other #variables

table(round(training$n_non_stop_words,4))
```

```
##
##     0     1
##   882 29117
```

```
table(round(test.real$n_non_stop_words,4))
```

```
##
##    0    1
##  299 9345
```

```
training$n_non_stop_words <- NULL
test.real$n_non_stop_words <- NULL
```

We cancel repetative variables(linear combination of other variables)

```
#cancel repetative variables
rep.dummy <-findLinearCombos(training)$remove
training[,rep.dummy] <-data.frame(NULL)
test.real[,rep.dummy] <- data.frame(NULL)
```

We then transfer population and num_keywords from integer to factor. We also make the two series of dummy variables to two factor. We omit these verbose code.

Finally we normalize the non-categorical predictors.

```
category.name <- c("weekday", "data_channel", "num_keywords")

training.pp <- preProcess(training[, !names(training) %in% c(category.name,"popularity")],
                          method = c("center", "scale"))

training <- predict(training.pp, newdata = training)
test.real <- predict(training.pp, newdata = test.real)

dim(training)
```

```
## [1] 29999    45
```

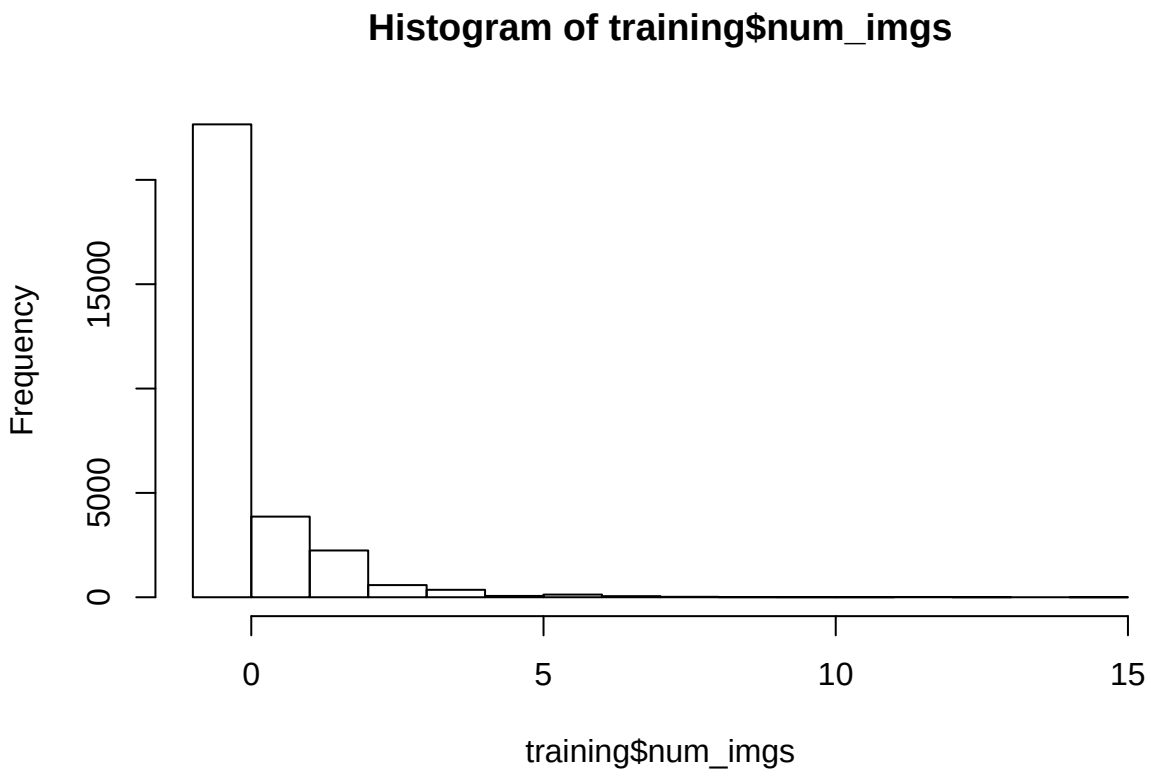
```
dim(test.real)
```

```
## [1] 9644   44
```

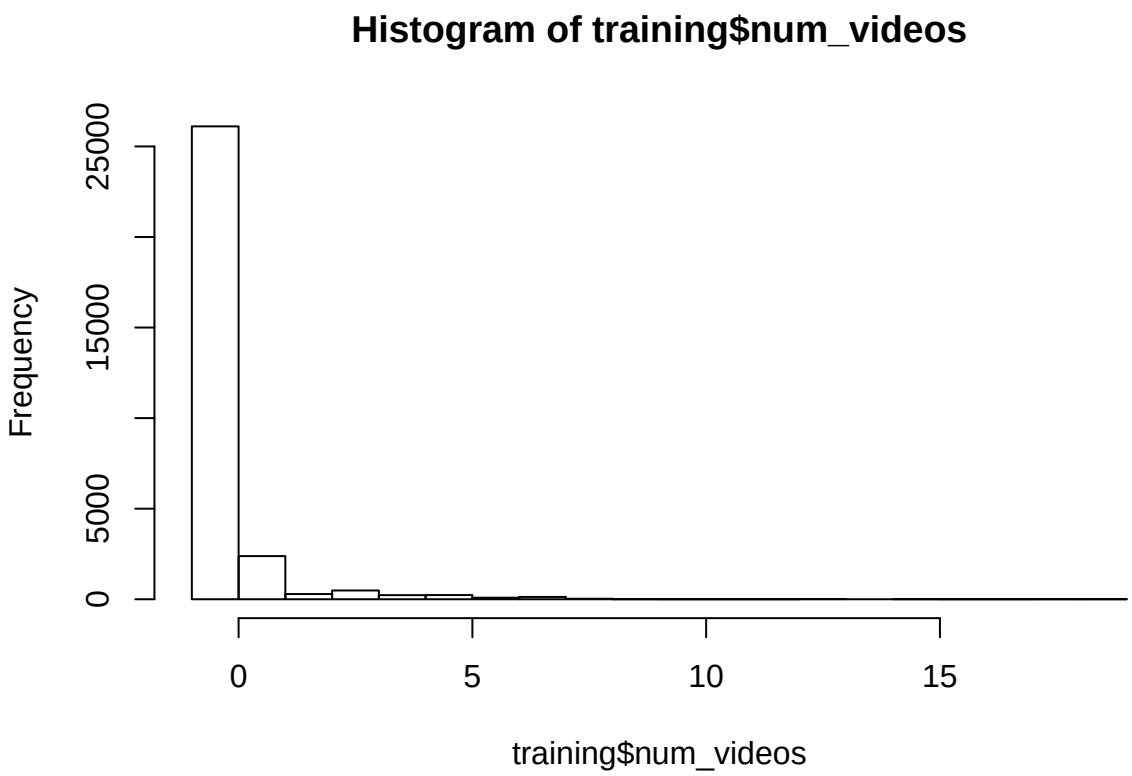
We found that many predictors are seriously right tail and we have both continuous and catigorical predictors, so we didn't think SVM might do a good job especially for multiclass.

```
hist(training$num_hrefs)
```


```
hist(training$num_imgs)
```




```
hist(training$num_videos)
```



S1

We then do multinomial logistic regression and some ordinal logistical regression like proportional odds, it seems diffcult to improve the fits(0.508 and 0.500). And becasue there are bunch of outliers, these parametric models might not be very robust.

```
library(nnet)

multiNom <- multinom(popularity~., data = training, maxit = 1000)

library(ordinal)

propOdds <- clm(popularity ~ ., data = training, link = "logit")
```

S2

We also consider a boosted MARS with only 1 order interactions between predictors using mboost package. We found the fit still not satisfactory(0.512) even we expand the iterations to 10000. So we guess we should include more interactions for each weak learner. That means the weak leaner should not be too weak.

```
#In this case we trasform the categorical variables to dummy variables
item <- character(0)

  for(i in dummy.name){
```

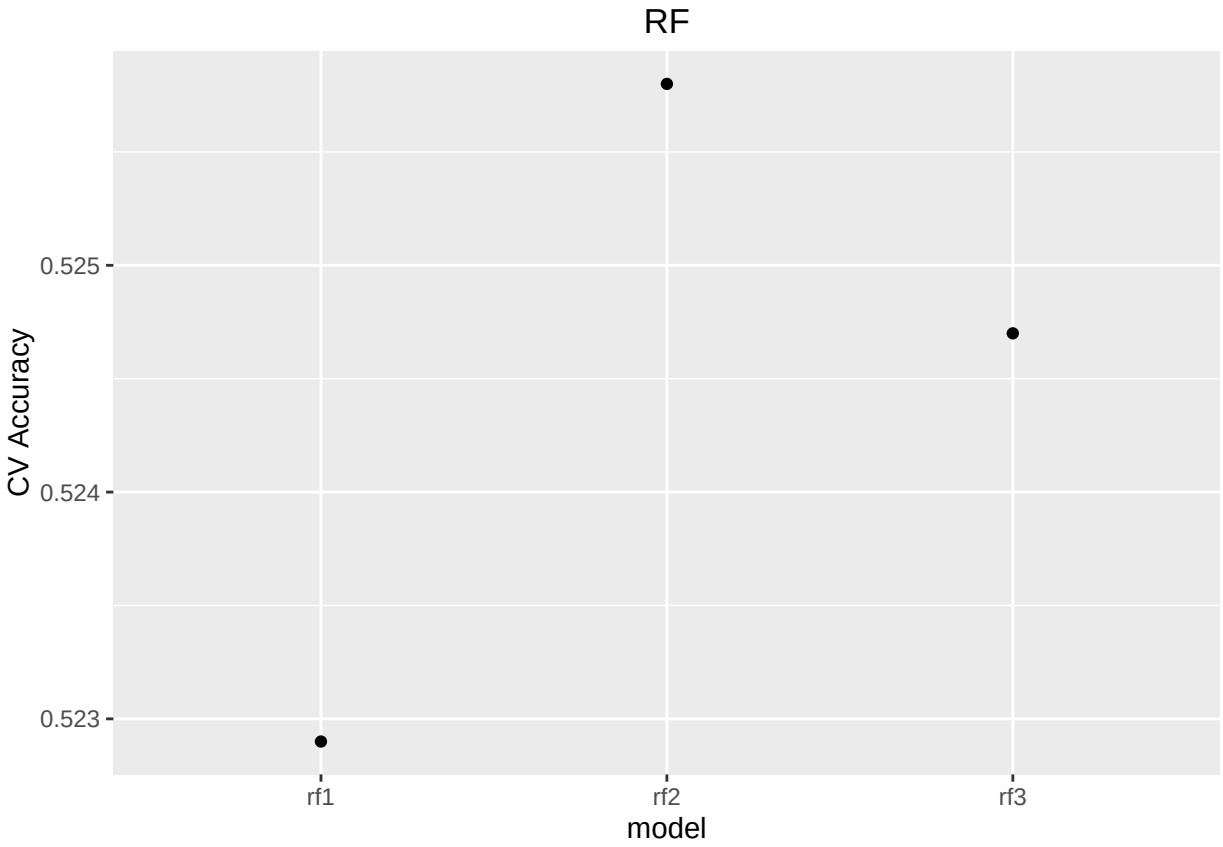
```
    item<-paste(item, paste0("bols(", i , ", intercept = FALSE, df = NULL)"),
                sep = " + ")

  }

  item.nonDummy <- character(0)
  for(j in setdiff(names(training)[-c(1,ncol(training))], dummy.name)){
    item.nonDummy <- paste(item.nonDummy, paste0("bols(", j, ", intercept = FALSE)",
                                                 " + ", "bbs(", j, ", center = TRUE, df = 3)"),
                           sep = " + ")

  }

  pop.formula <- as.formula(pop.formula)

  ctrl <- boost_control(mstop = 20000, nu = 0.13)
  MARS <- gamboost(pop.formula, data = training,
                   family = PropOdds(), control = ctrl)
```

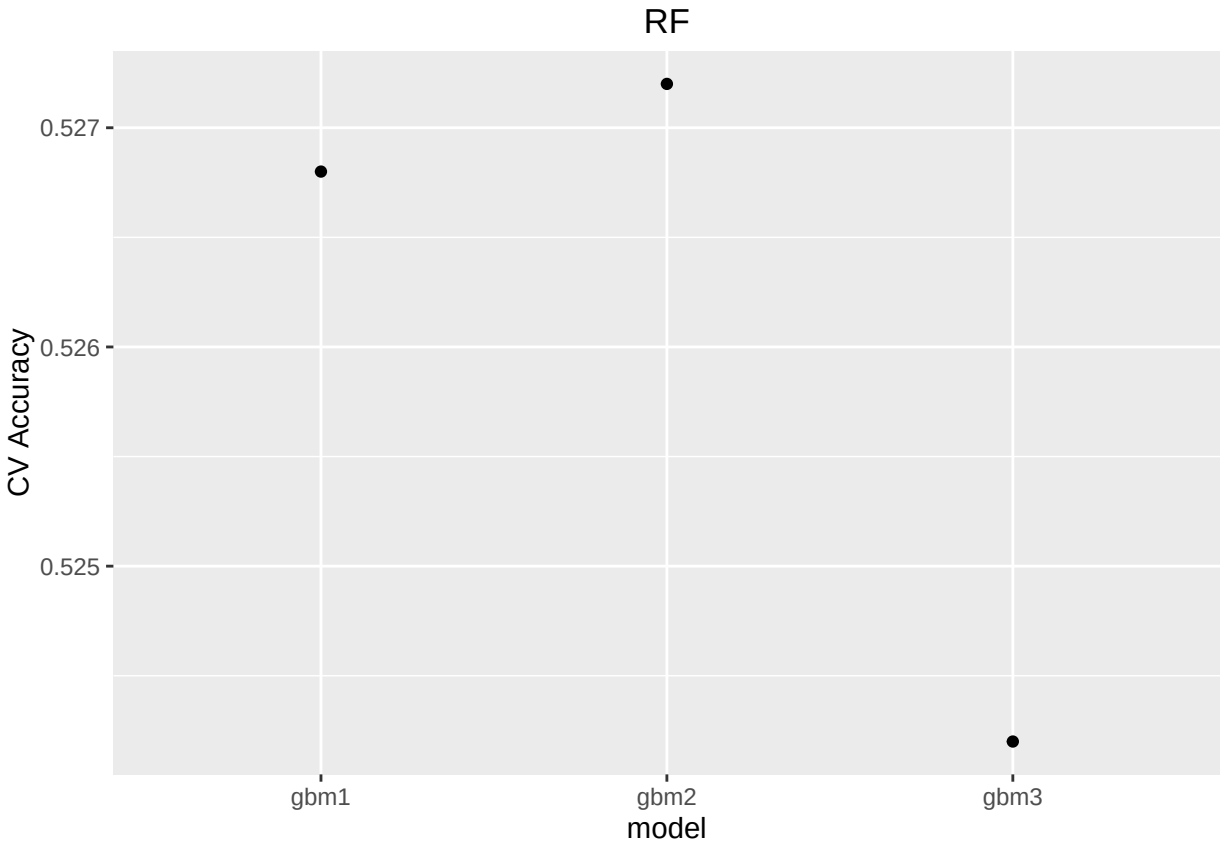
S3

Finally, we focused on tree based models with ensemble methods. We chose the most powerful two tools: randomForest and gbm. Refer to caret package, we use 3-repeat 5-fold CV to tune the paremeters. We select 3 models with the best estimated accuracy from each type. Our strategy is just to do simple model average taking majority rule.

```
set.seed(111)
  #RF1
  Grid.rf1 <-  expand.grid(mtry = 7)
  fitControl.rf1<- trainControl(method = "repeatedCV", number=5, repeats=3)
  rf1 <- train(popularity ~ ., data = training,
               ntree=750,
               method = "rf",
               trControl = fitControl.rf1,
               verbose = FALSE,
               tuneGrid = Grid.rf1
               )


  #RF2
  Grid.rf2 <-  expand.grid(mtry = 6)
  fitControl.rf2<- trainControl(method = "repeatedCV", number=5, repeats=3)
  rf2 <- train(popularity ~ ., data = training,
               ntree=1000,
               method = "rf",
               trControl = fitControl.rf2,
               verbose = FALSE,
               tuneGrid = Grid.rf2
               )


  #RF3
```

```
  Grid.rf3 <-  expand.grid(mtry = 8)
  fitControl.rf3<- trainControl(method = "repeatedCV", number=5, repeats=3)
  rf3 <- train(popularity ~ ., data = training,
               ntree=1250,
               method = "rf",
               trControl = fitControl.rf3,
               verbose = FALSE,
               tuneGrid = Grid.rf3
  )

#gbm1
  Grid.gbm1 <-  expand.grid(interaction.depth = 13,
                            n.trees = c(800),
                            shrinkage = 0.008,
                            n.minobsinnode = 10)
  fitControl.gbm1 <- trainControl(method = "repeatedCV", number=5, repeats=3)
  gbm1 <- train(popularity ~ ., data = training,
                  method = "gbm",
                  trControl = fitControl.gbm1,
                  verbose = FALSE,
                  tuneGrid = Grid.gbm1)


  #gbm2
  Grid.gbm2 <-  expand.grid(interaction.depth = 10,
                              n.trees = c(2000),
                              shrinkage = 0.003,
                              n.minobsinnode = 15)
  fitControl.gbm2 <- trainControl(method = "repeatedCV", number=5, repeats=3)
  gbm2 <- train(popularity ~ ., data = training,
                method = "gbm",
                trControl = fitControl.gbm2,
                verbose = FALSE,
                tuneGrid = Grid.gbm2)


  #gbm3
  Grid.gbm3 <-  expand.grid(interaction.depth = 8,
                              n.trees = c(1000),
                              shrinkage = 0.01,
                              n.minobsinnode = 15)
  fitControl.gbm3 <- trainControl(method = "none")
  gbm3 <- train(popularity ~ ., data = training,
                method = "gbm",
                trControl = fitControl.gbm3,
                verbose = FALSE,
                tuneGrid = Grid.gbm3)
```


RF
CV Accuracy
0.525
0.524
0.523
rf1
rf2
rf3
model



After we do model average, the CV accuracy achieve to 0.532.