Howell Su, Ziqi Wang, Yuchen Zhang
Stats 101C Final Project
Lecture 3
7 December 2020

Stats101C Final report

- **Introduction**

In this project, we investigated the statistical regression of different predictors against the percentage change of views for YouTube videos from 2-6 hours since publication. Using 24 predictors, we constructed a random forest model that outperforms other models in efficiency and accuracy.

- **Methodology**

Preprocessing of the data

We started off by transforming the variable "publish date" into Month, Day, and Hour. We believe that the time of YouTube video publication is highly influential on the user views. We expected a positive correlation between the increment of views and level of synchronization with the users' work cycle. Also, certain periods of the year may lead to a spike in video views, an example being holidays. These possibilities prompted us to identify month, day and hour within the character type "publish date" using the "lubridate" package, which produced highly significant variables of "publish month" and "publish hour" to be used in later regression models.

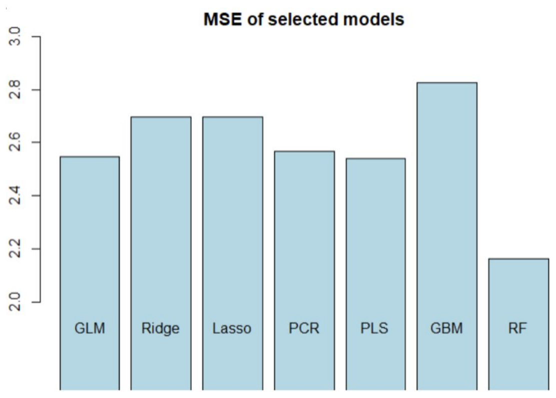
We then split the raw dataset into training and testing sets by 0.8 to 0.2 ratio.

Statistical model

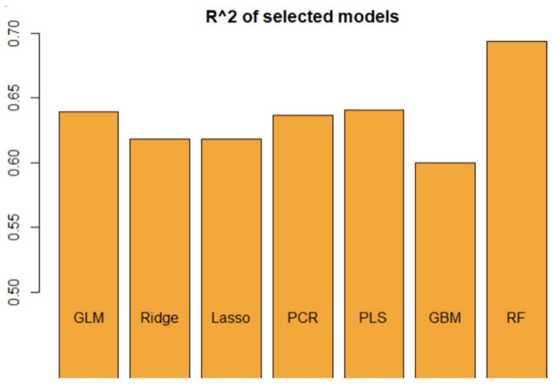
1. model selection

We tried to identify the best model for this dataset. We had the following model candidates:

1) RIDGE/ LASSO regression: Because of the multitude of predictors that are highly correlated, such as the "cnn_*" group and the "punc_num_*" group, we believe that ridge regression can be effective in checking the potential multicollinearity problem.
2) PCR and PLS: Because of the large number of predictors, we believe that a dimension reduction can address model complexity while maintaining similar predictability.
3) Random Forest regression: We included the random forest model because it is a powerful tool to handle big data with numerous variables.



(Figure 1.1)



(Figure 1.2)

By fitting the full model upon the potential regression method choices mentioned above, we analyzed their respective testing MSE **(Figure 1.1)** and R^2 **(Figure 1.2)**, and decided that the random forest model performs significantly better than other models. Thus, we decide to use it for further analysis.

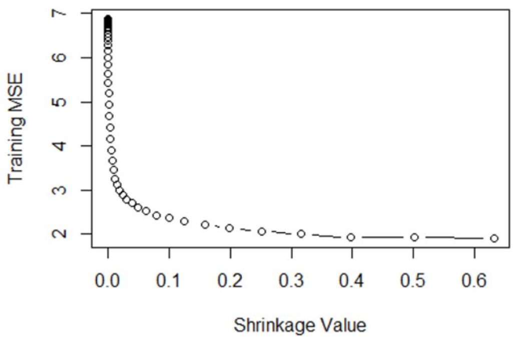
2. predictor selection

We try to identify the best predictors with the following predictor selection methods:

1) **correlation plot / matrix**: By convention, we draw the correlation matrix of predictors against the response variable to skim for significant predictors.
2) **subset selection**: We tried to use AIC and BIC to identify the best predictors. Limited by the data volume and predictor sizes, this is proven to be unattainable time-wize.
3) **boosting**: Given the large amount of predictors, we decided to use boosting to combine some of the weak predictors into a strong one to reduce the time consumption of the model.
4) **importance function**: Because the grading criteria of this competition is MSE, we decided to include the importance function for prediction selection. In regression, the importance function values both the MSE of out-of-bag data proportion and the total decrease in node impurities, which is crucial for the concision and efficiency of random forest base model.

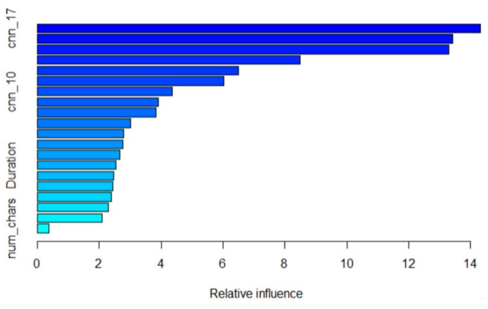
With the above methods and motivation established, we started with the construction of a correlation matrix. Because of the large data set, we made a rudimentary cut-off at 0.2 correlation to eliminate predictors that are minimally correlated to the response variable, it left us with a set of 19 variables that are predictive for the percentage growth of a YouTube video to some degree. We collected these variables in as a list of "high correlation"

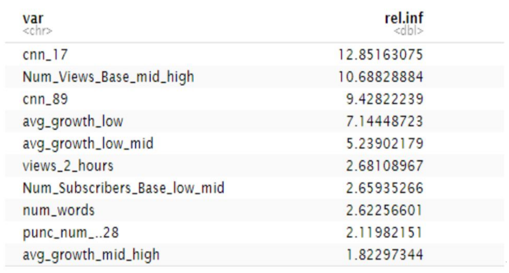
Then, we performed boosting in an attempt to refine the predictor group for a more time-efficient model. We adopted a shrinkage value from 10^-10 to 10^-0.2, grew 1000 trees for every shrinkage value, and implemented a five-fold cross-validation **(Figure 1.3)**.



(Figure 1.3)

By using the lambda value that generated the best training MSE, we performed the gaussian elimination on the training set to obtain the optimal model with refined number of predictors.



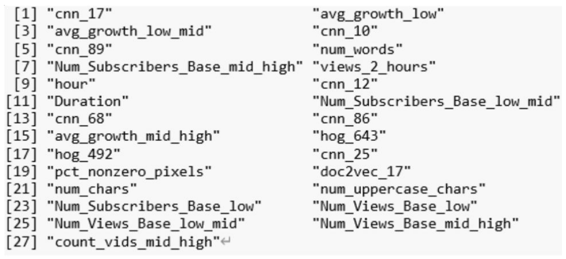


var <chr>	rel.inf <dbl>
cnn_17	12.85163075
Num_Views_Base_mid_high	10.68828884
cnn_89	9.42822239
avg_growth_low	7.14448723
avg_growth_low_mid	5.23902179
views_2_hours	2.68108967
Num_Subscribers_Base_low_mid	2.65935266
num_words	2.62256601
punc_num_..28	2.11982151
avg_growth_mid_high	1.82297344

(Figure 1.4) (Figure 1.5)

The summary **(Figure 1.4 & 1.5)** outputs the list of variables in descending order regarding the relative importance they possess individually.

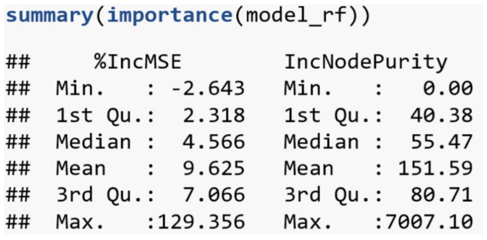
We chose the top 20 predictors in the summary graph and made them into a list of high relative importance. By cross-referencing this list and the previously selected list of predictors with high correlation, we combine the two lists of predictors to ensure both their relative significance in the model as well as their relevance in determining the response variable, yielding 27 predictors **(Figure 1.6)** to fit the random forest model, which yields a **MSE of 2.0896**.



```
 [1] "cnn_17"                        "avg_growth_low"
 [3] "avg_growth_low_mid"            "cnn_10"
 [5] "cnn_89"                        "num_words"
 [7] "Num_Subscribers_Base_mid_high" "views_2_hours"
 [9] "hour"                          "cnn_12"
[11] "Duration"                      "Num_Subscribers_Base_low_mid"
[13] "cnn_68"                        "cnn_86"
[15] "avg_growth_mid_high"           "hog_643"
[17] "hog_492"                       "cnn_25"
[19] "pct_nonzero_pixels"            "doc2vec_17"
[21] "num_chars"                     "num_uppercase_chars"
[23] "Num_Subscribers_Base_low"      "Num_Views_Base_low"
[25] "Num_Views_Base_low_mid"        "Num_Views_Base_mid_high"
[27] "count_vids_mid_high"
```

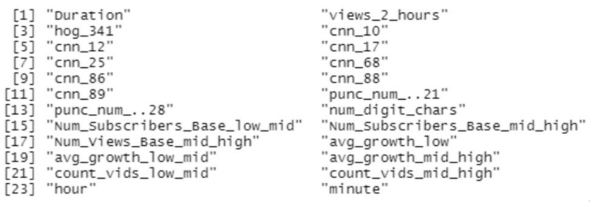
(Figure 1.6)

Aside from the combination of high-correlation/ high relative importance, we also used the random forest importance function to identify significant predictors judging by the MSE on the out-of-bag portion of the data and the total decrease in node impurities from splits over that variable, measured by RSS. By analyzing the distribution of %IncMSE(MSE) and IncNodePurity(RSS), we found that the mean of both indicators are significantly higher than their respective 3rd Quantile **(Figure 1.7)**, suggesting outliers with influential high MSE and RSS, so we choose the variables whose %IncMSE(MSE) and IncNodePurity(RSS) are both above their respective mean value, constructing a random forest model with 24 predictors **(Figure 1.8)** and achieves a **MSE of 1.984**.



```
summary(importance(model_rf))

##     %IncMSE          IncNodePurity
##  Min.   : -2.643   Min.   :   0.00
##  1st Qu.:  2.318   1st Qu.:  40.38
##  Median :  4.566   Median :  55.47
##  Mean   :  9.625   Mean   : 151.59
##  3rd Qu.:  7.066   3rd Qu.:  80.71
##  Max.   :129.356   Max.   :7007.10
```

(Figure 1.7)



```
 [1] "Duration"                     "views_2_hours"
 [3] "hog_341"                      "cnn_10"
 [5] "cnn_12"                       "cnn_17"
 [7] "cnn_25"                       "cnn_68"
 [9] "cnn_86"                       "cnn_88"
[11] "cnn_89"                       "punc_num_..21"
[13] "punc_num_..28"                "num_digit_chars"
[15] "Num_Subscribers_Base_low_mid" "Num_Subscribers_Base_mid_high"
[17] "Num_Views_Base_mid_high"      "avg_growth_low"
[19] "avg_growth_low_mid"           "avg_growth_mid_high"
[21] "count_vids_low_mid"           "count_vids_mid_high"
[23] "hour"                         "minute"
```

(Figure 1.8)

Judging by the result, the predictors selected by importance function outperforms the result of high correlation and boosting. So we chose the above 24 predictors as the final model.

- **Result**

Our final model is Random Forest with 24 predictors **(Figure 1.8)** with **MSE is 1.988**. It earns us a root square mean error of 1.41873 and 1.41325 in the private and public leaderboard.

- **Conclusion**

1. Model advantage

With over 7200 observations in training data, Random Forest guarantees a large flexibility and low bias by taking the average multiple decision trees.

Another advantage of random forest is its unique ability to handle binary features, which are abundant in this dataset with examples of the “cnn” group. It ensures the information held by these binary predictors when used in a model.

Compared to the bagging approach, random forest does not consider all the available predictors at each tree split. Since there are several strong predictors, the bagged trees tend to be similar to each other as most of them put the same predictors in the top split. By decorrelation with random forest, each tree will be less correlated, making the resulting trees less variable. Moreover, considering fewer predictors in each split saves computing power, making the model more efficient.

2. Drawbacks and potential improvements

A random forest model with large trees takes up lots of working memory. Lowering the number of tries per node and the trees or having a faster device might help us attempt more models.

Another potential disadvantage is that none of us have any experience with YouTube channels, so our predictor selection is rather random. A random forest method further limits the model’s interpretability and checks its real-life implication. If we have had previous experience with YouTube, we might be able to handle this task with the aid of some first-hand knowledge about it.

- **Contribution Statement**
  Team WDNMD
  Eric Ziqi Wang (905198360): model tuning/ model testing/ report writing/ presentation
  Howell Haoyu Su (505117747): predictor selection/ report writing/ presentation
  Yuchen Zhang (405107678): model comparison/ report writing/ presentation

- **Appendix**

Report

2020/12/13

```r
library(corrplot)

library(car)

library(knitr)
library(plot.matrix)
library(class)

library(MASS)

library(leaps)

library(caret)

library(nnet)

library(dplyr)

library(randomForest)

library(gbm)

library(glmnet)

library(lubridate)
```

Preprocessing of the Data

```r
testing <- read.csv("test.csv")
training <- read.csv("training.csv")
training$PublishedDate <- mdy_hm(training$PublishedDate)
training$month <- month(training$PublishedDate)
training$day<- day(training$PublishedDate)
training$hour<- hour(training$PublishedDate)
training$minute<- minute(training$PublishedDate)

testing$PublishedDate <- mdy_hm(testing$PublishedDate)
testing$month <- month(testing$PublishedDate)
testing$day<- day(testing$PublishedDate)
testing$hour<- hour(testing$PublishedDate)
testing$minute<- minute(testing$PublishedDate)


set.seed(123456)
index <- sample(seq_len(nrow(training)), size = 0.8 * nrow(training))
train <- training[index,-c(1,2)]
train <- na.omit(train)
test <- training[-index,-c(1,2)]
```

Statisctic Model Selection

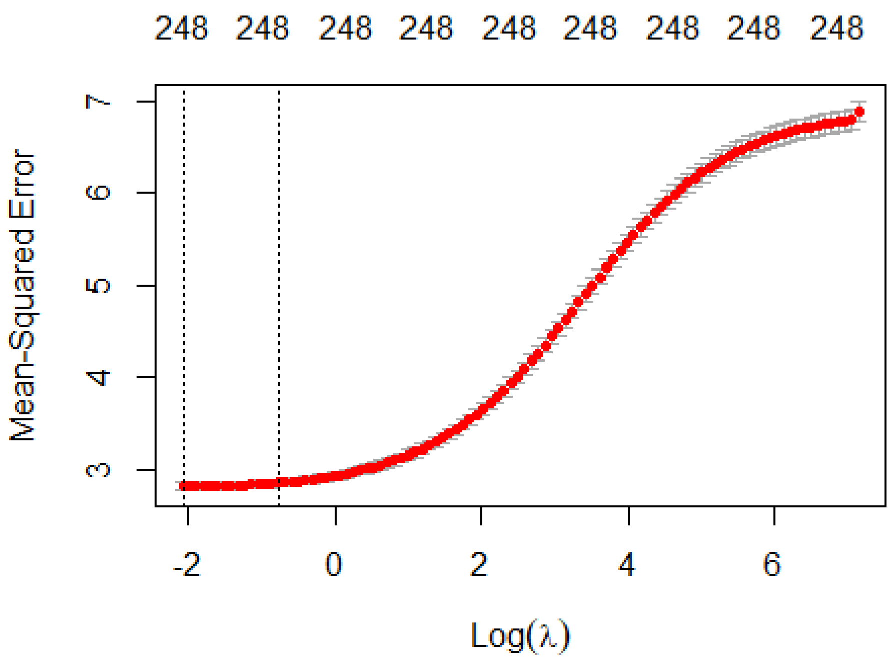
GLM

```
model_glm <- glm(train$growth_2_6 ~ ., data = train)
yhat.glm <- predict(model_glm, newdata = test)

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if
(type == :
## prediction from a rank-deficient fit may be misleading

glm.err <- mean((yhat.glm - test$growth_2_6)^2)
```

GLM model MSE: 2.5475668

Ridge

```
library(glmnet)
xtrain <- model.matrix(growth_2_6~., data = train)
ytrain <- train$growth_2_6
xtest <- model.matrix(growth_2_6~., data = test)
ytest <- test$growth_2_6

ridge.fit <- cv.glmnet(xtrain,ytrain,alpha = 0)
plot(ridge.fit)
```

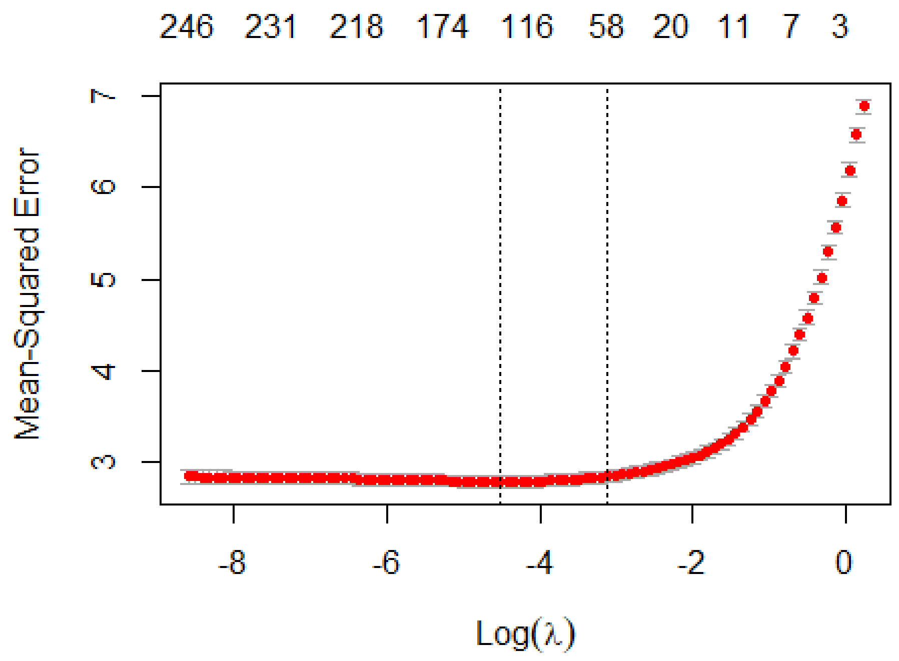


```
ridge.lambda <- ridge.fit$lambda.min

ridge.pred <- predict(ridge.fit, s = ridge.lambda, newx = xtest)
ridge.err <- mean((ridge.pred - ytest)^2)
```

Ridge test MSE: 2.5400127.

Lasso

```
lasso.fit <- cv.glmnet(xtrain,ytrain,alpha = 1)
plot(lasso.fit)
```

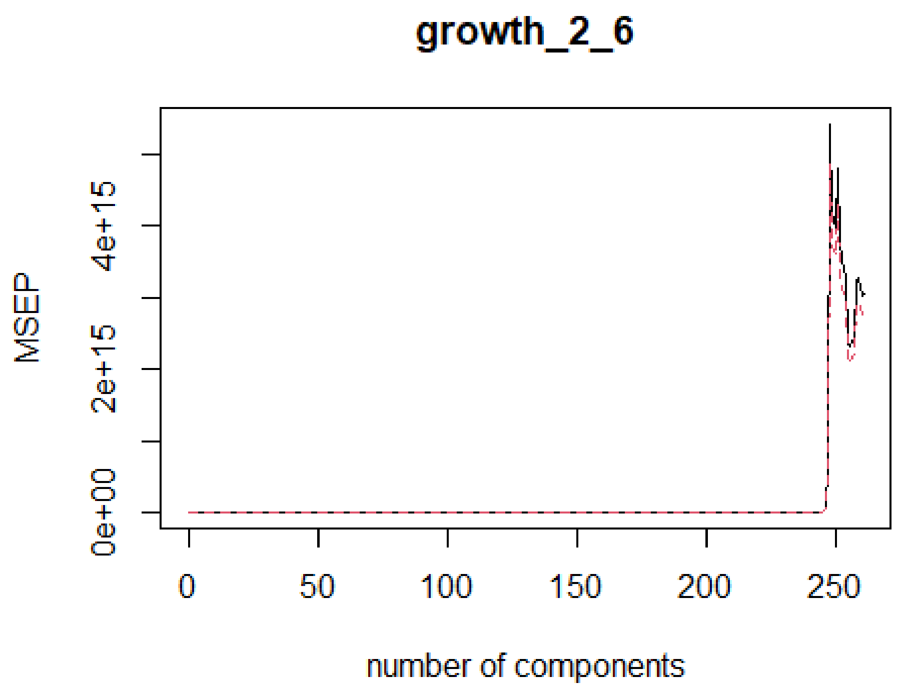


```
lasso.lambda <- lasso.fit$lambda.min
#lasso.lambda

lasso.pred <- predict(lasso.fit, s = lasso.lambda, newx = xtest)
lasso.err <- mean((lasso.pred - ytest)^2)
```

Lasso test MSE: 2.5599742

PCR

```
library(pls)

pcr.fit <- pcr(growth_2_6~.,data = train, scale= FALSE, validation = "C
V")
validationplot(pcr.fit, val.type = "MSEP")
```

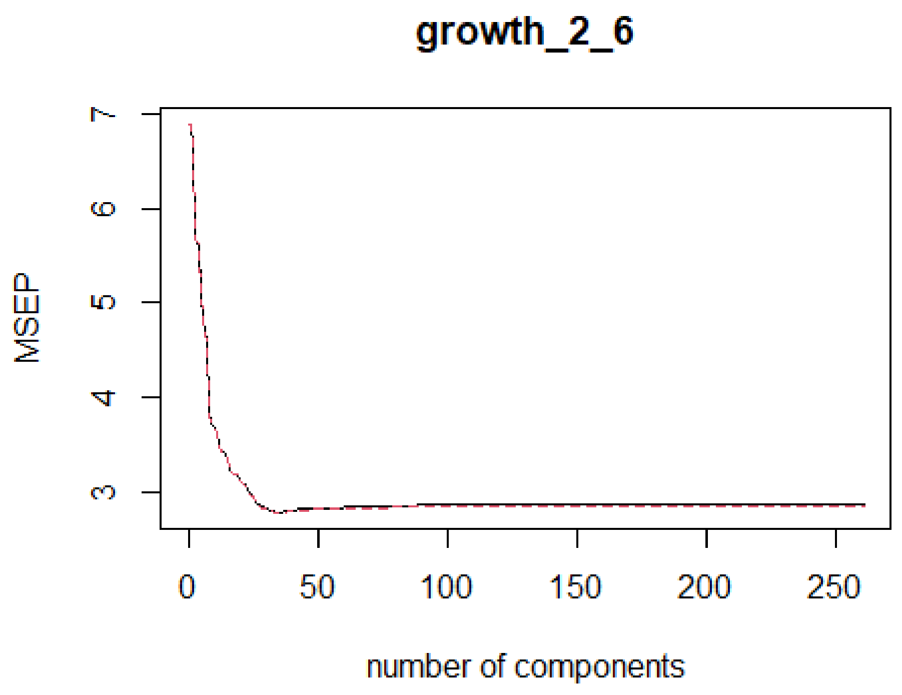


```
#summary(pcr.fit)
pcr.pred <- predict(pcr.fit, test, ncomp = 109)
pcr.err = mean((pcr.pred - test$growth_2_6)^2)
```

PCR test error rate : 2.5657244.

PLS

```
pls.fit <- plsr(growth_2_6~.,data = train, scale= FALSE, validation = "
CV")
validationplot(pls.fit, val.type = "MSEP")
```

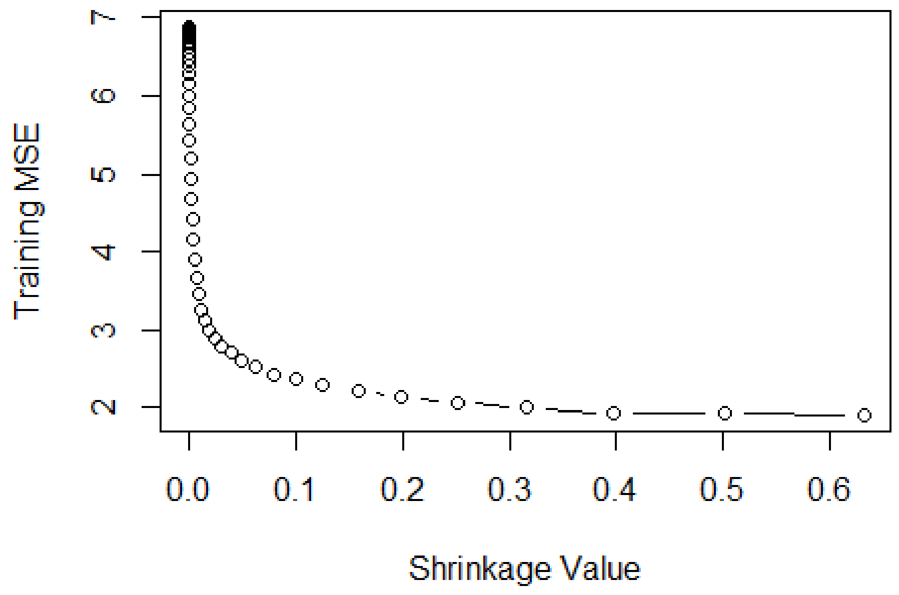


```r
#summary(pls.fit)
pls.pred <- predict(pls.fit, test, ncomp = 34)
pls.err = mean((pls.pred - test$growth_2_6)^2)
```

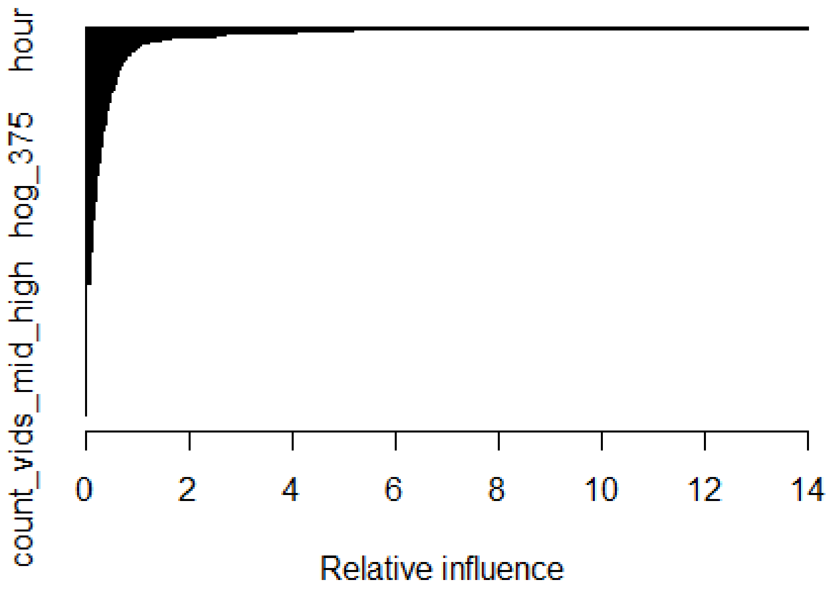
PLS test error rate : 2.5387545.

Boosting

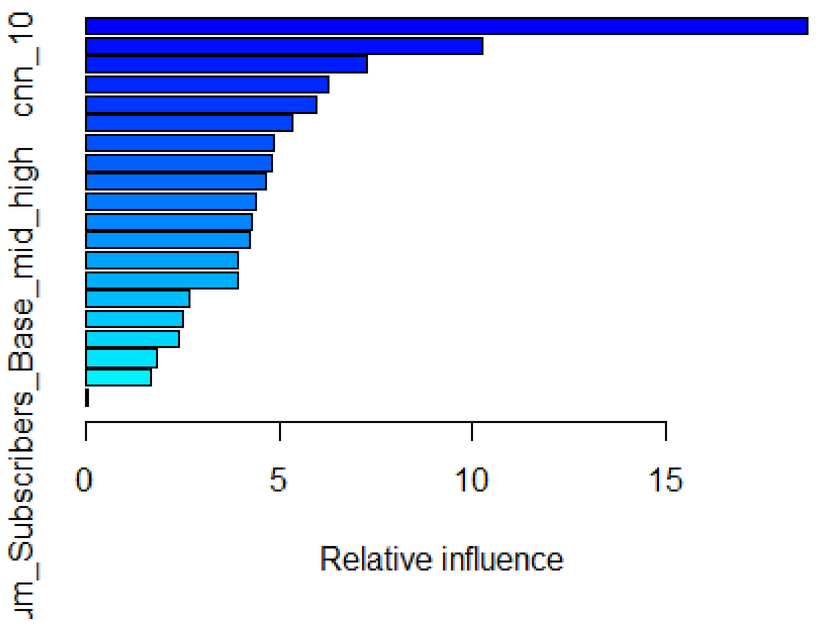
```r
library(gbm)
set.seed(123)
power <- seq(-10, -0.2, by = 0.1)
lambda <- 10^power
trainMSE <- rep(NA, length(lambda))
for (i in 1:length(lambda)){
  boost <- gbm(growth_2_6~., data = train, distribution = "gaussian",
n.trees = 500,verbose = FALSE, shrinkage = lambda[i])
  pred.train <- predict(boost, train, n.trees = 1000)
  trainMSE[i] <- mean((pred.train - train$growth_2_6)^2)
}

plot(lambda, trainMSE, type = "b", xlab = "Shrinkage Value", ylab = "Tr
aining MSE")
```



```
#min(trainMSE)
#lambda[which.min(trainMSE)]

model_gbm <- gbm(growth_2_6~., data = train, distribution = "gaussian",
 n.trees = 500,  shrinkage = lambda[which.min(trainMSE)])

yhat.gbm <- predict(model_gbm, newdata = test)

## Using 500 trees...

gbm.err <- mean((yhat.gbm - test$growth_2_6)^2)

a <- summary(model_gbm)
```



```
gbm_x <- head(a,20)[,1]
model_gbm1 <- gbm(growth_2_6~., data = train[,c(gbm_x, "growth_2_6")],
distribution = "gaussian", n.trees = 500,  shrinkage = lambda[which.min
(trainMSE)])

yhat.gbm1 <- predict(model_gbm1, newdata = test)

gbm1.err <- mean((yhat.gbm1 - test$growth_2_6)^2)

summary(model_gbm1)
```



```
##                                                            var     rel.
inf
## cnn_17                                                  cnn_17 18.64845
190
## Num_Views_Base_mid_high                Num_Views_Base_mid_high 10.22589
007
## cnn_10                                                  cnn_10  7.27887
473
## avg_growth_low                                  avg_growth_low  6.26273
596
## cnn_89                                                  cnn_89  5.94869
388
## pct_nonzero_pixels                          pct_nonzero_pixels  5.33038
857
## hog_643                                                hog_643  4.87490
171
## avg_growth_low_mid                          avg_growth_low_mid  4.82927
820
## num_words                                            num_words  4.63475
139
## cnn_68                                                  cnn_68  4.39411
512
## views_2_hours                                    views_2_hours  4.30614
076
## cnn_25                                                  cnn_25  4.26066
982
## Duration                                              Duration  3.93968
```

```
681
## cnn_12                                                   cnn_12  3.91091
219
## hog_492                                                 hog_492  2.65284
802
## hour                                                       hour  2.50540
482
## cnn_86                                                   cnn_86  2.38794
966
## Num_Subscribers_Base_low_mid   Num_Subscribers_Base_low_mid  1.84216
823
## avg_growth_mid_high                         avg_growth_mid_high  1.70365
198
## Num_Subscribers_Base_mid_high Num_Subscribers_Base_mid_high  0.06248
617
```

Boosted model MSE: 3.078268.

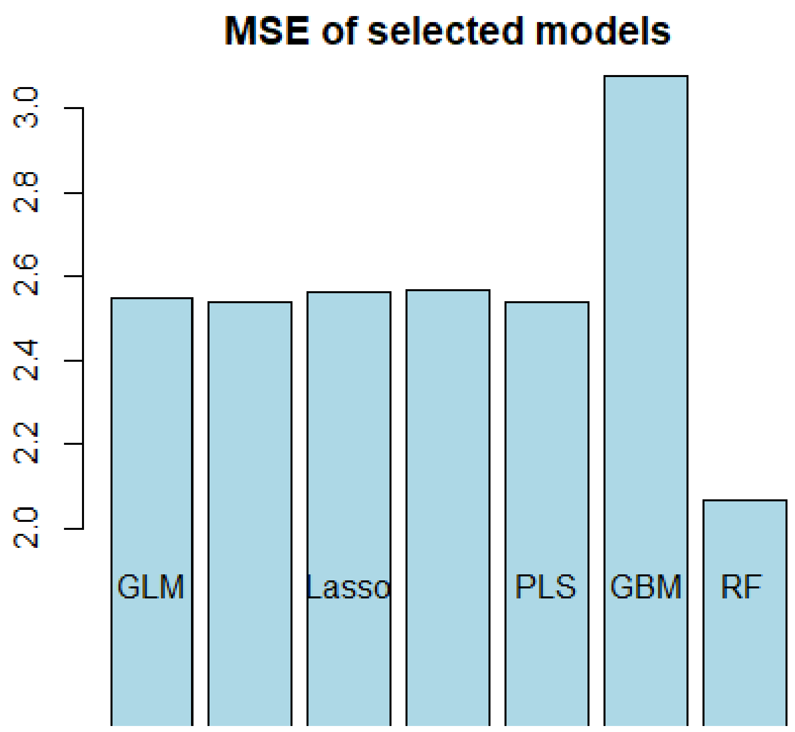
Random Forest

```
library(randomForest)
model_rf <- randomForest(growth_2_6~., data = train, mtry = 262/3, ntre
e= 2000, importance = TRUE) # 2.10

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to w
ithin valid
## range

yhatrf <- predict(model_rf, newdata = test)
rf.err <- mean((yhatrf - test$growth_2_6)^2)
```

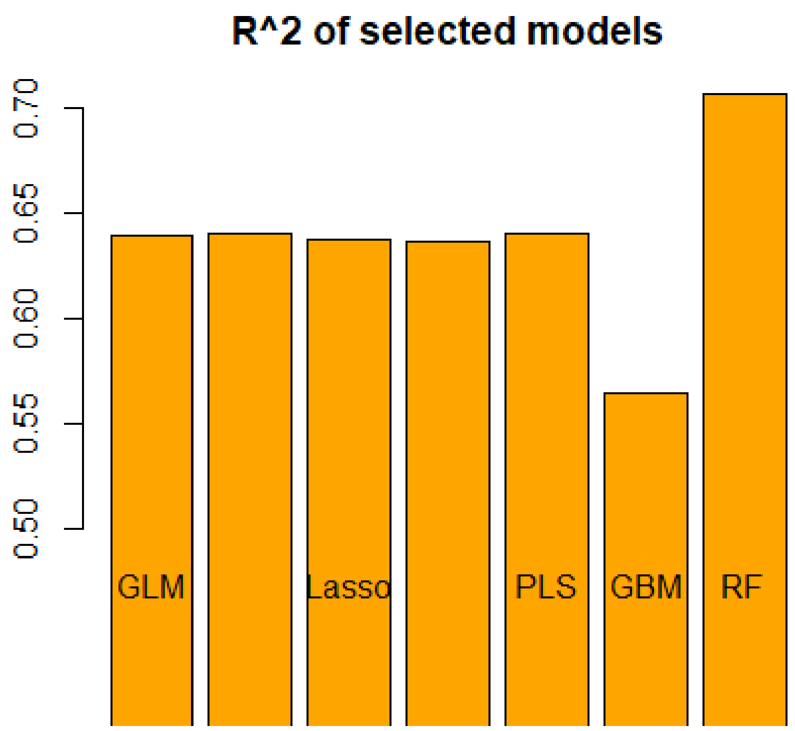
Random forest model MSE: 2.067357.

Summary

```
result <- c(glm.err, ridge.err,lasso.err,pcr.err,pls.err,gbm.err, rf.er
r)
barplot(result,
        names.arg = c("GLM", "Ridge","Lasso", "PCR", "PLS", "GBM", "RF
"),
        ylim = c(2,3),
        col = "lightblue",
        main = "MSE of selected models",
        axes = TRUE)
```



```r
sst <- mean((mean(test$growth_2_6) - test$growth_2_6)^2)

r2 <- c()
for ( i in 1:length(result)){
  r2 <- c(r2, 1 - result[i]/sst)
}
barplot(r2,
        names.arg = c("GLM", "Ridge","Lasso", "PCR", "PLS", "GBM", "RF
"),
        col = "ORANGE",
        ylim = c(0.5,0.7),
        main = "R^2 of selected models")
```



Predictor Selection

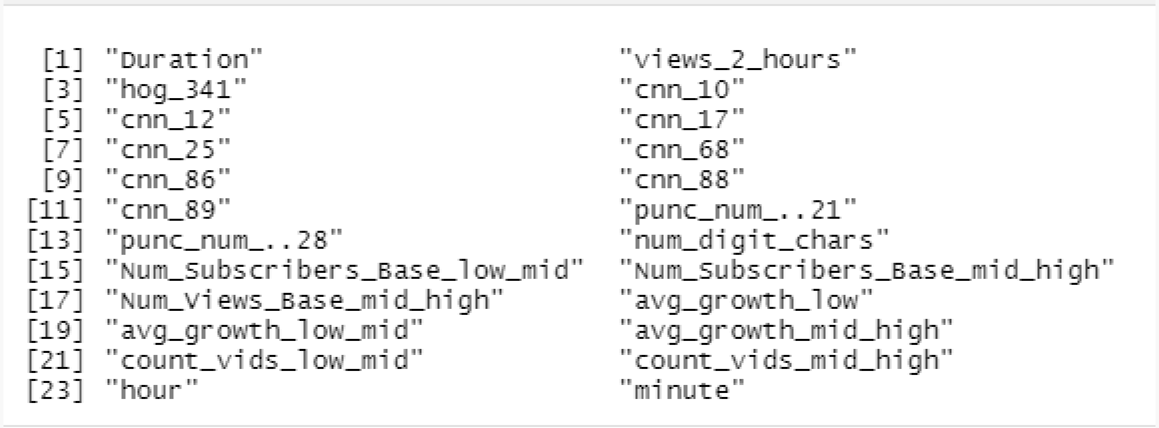
High correlation

```
cor <- abs(cor(train$growth_2_6,train[,-258]))

pick <-  which(cor > 0.2)
length(pick)

## [1] 19

high_cor <- colnames(train)[pick]

correlationMatrix <- cor(train[,pick])


x <- c(gbm_x, high_cor)
length(x)

## [1] 39

for (i in 1:length(x)){
  for (j in 1:(i-1)) {
    if (x[i] == x[j]){
      x[i] = 0
      break
    }
  }
}
x <- x[-which(x == 0)]
x

##  [1] "cnn_17"                       "avg_growth_low"
##  [3] "avg_growth_low_mid"           "cnn_10"
##  [5] "cnn_89"                       "num_words"
##  [7] "Num_Subscribers_Base_mid_high" "views_2_hours"
##  [9] "hour"                         "cnn_12"
## [11] "Duration"                     "Num_Subscribers_Base_low_mid"
## [13] "cnn_68"                       "cnn_86"
## [15] "avg_growth_mid_high"          "hog_643"
## [17] "hog_492"                      "cnn_25"
## [19] "pct_nonzero_pixels"           "doc2vec_17"
## [21] "num_chars"                    "num_uppercase_chars"
## [23] "Num_Subscribers_Base_low"     "Num_Views_Base_low"
## [25] "Num_Views_Base_low_mid"       "Num_Views_Base_mid_high"
## [27] "count_vids_mid_high"

model_1.1 <- randomForest(growth_2_6~., data = train[,c(x,"growth_2_6
")], mtry = 27/3, ntree = 500)

summary(model_1.1)

##                 Length Class  Mode
## call               5   -none- call
## type               1   -none- character
## predicted       5793   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times       5793   -none- numeric
## importance        27   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               5793   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call

yhat.1.1 <- predict(model_1.1, newdata = test)
mse1.1 <- mean((yhat.1.1 - test$growth_2_6)^2)
mse1.1

## [1] 2.089632
```

MSE: 2.0896315

Importance

```
summary(importance(model_rf))

##     %IncMSE         IncNodePurity
##  Min.   : -2.643   Min.   :   0.00
##  1st Qu.:  2.318   1st Qu.:  40.38
##  Median :  4.566   Median :  55.47
##  Mean   :  9.625   Mean   : 151.59
##  3rd Qu.:  7.066   3rd Qu.:  80.71
##  Max.   :129.356   Max.   :7007.10

rf_imp <- which(importance(model_rf)[,1]>mean(importance(model_rf)[,1])
& importance(model_rf)[,2]>mean(importance(model_rf)[,2]))
rf_imp <- rownames(importance(model_rf))[rf_imp]

rf_imp
```



```
  [1] "Duration"                        "views_2_hours"
  [3] "hog_341"                         "cnn_10"
  [5] "cnn_12"                          "cnn_17"
  [7] "cnn_25"                          "cnn_68"
  [9] "cnn_86"                          "cnn_88"
 [11] "cnn_89"                          "punc_num_..21"
 [13] "punc_num_..28"                   "num_digit_chars"
 [15] "Num_Subscribers_Base_low_mid"    "Num_Subscribers_Base_mid_high"
 [17] "Num_Views_Base_mid_high"         "avg_growth_low"
 [19] "avg_growth_low_mid"              "avg_growth_mid_high"
 [21] "count_vids_low_mid"              "count_vids_mid_high"
 [23] "hour"                            "minute"
```

```
set.seed(123)
model_1.2 <- randomForest(growth_2_6~., data = train[,c(rf_imp,"growth_
2_6")], mtry = 24/3, ntree = 500) # 1.988

yhat.1.2 <- predict(model_1.2, newdata = test)
mse1.2 <- mean((yhat.1.2 - test$growth_2_6)^2)
mse1.2

## [1] 1.984859
```