STATS 101C FINAL - REPORT

Team: VAT

Introduction

The goal of our project was to predict the percentage growth in the number of views of a Youtube video from the second hour to the sixth hour and determine which predictors affect this growth the most. Our training data set had a total of 259 features and 7,242 observations and our testing dataset had 3,105 observations. All the predictors were divided into four broad categories including thumbnail image features, video title features, channel features and other features.

Preprocessing

Once we loaded our training and testing data sets to our global environment, we checked for "NA's" as our first preprocessing step. Checking and dealing with missing values is very important to make sure that the data set is complete and all the models can run smoothly. After checking that there were no "NA's" in the data set we decided to convert the "PublishedDate" feature to "POSIXct", which would make it numeric. Converting it to numeric would ensure that we could use the feature in the PCR and Elastic Net models since both need numeric values as input. We used the "mdy_hms" function from the Lubridate library to change the type to numeric. We then moved on to check and remove predictors with either zero variance or just a few unique values relative to the number of samples. Removing such predictors was important in order to avoid our models from becoming unstable and eventually crash. Having only a few unique values relative to the number of samples might have an undue influence on the model when we split our data into training and validation sets for cross validation.

After examining the data set, there were a total of 259 features including the response variable. On digging deeper, we saw that around 151 predictors belonged to the same broad "HOG" category. To train our model without overfitting, we had to remove some predictors. As our first step to remove predictors, we decided to reduce the dimensions using Partial Least Squares regression (PLS). We used PLS to reduce the predictors by performing least squares regression on a smaller set of uncorrelated components. We did not run the PLS model on the entire set, but just to reduce the dimensions of the HOG predictors since they belonged to the same broad category and would help us continue to be able to interpret which predictors are important. Thus, we extracted the HOG variables from the training data and performed PLS on them. This helped us reduce 151 predictors to 13 predictors, which was a large reduction in dimensions. We then attached the HOG principal components back to our dataset with the other predictors and ran Elastic Net on it. Elastic Net combines the two penalties of regularization, LASSO and Ridge Regression. We tuned the alpha as well as lambda parameters in order to assign weight to each of the penalties. Running the Elastic Net model helped us finally reduce our number of predictors down to 68, which we used to train our bagging model (refer to figure 1).

Statistical Model

We decided to use bagging as our statistical model to make decisions. A good motivation for why we used this model is because it outperformed all other models we tried by a large margin.

One of the models that did not perform as well was when we used LASSO instead of elastic net. The underperformance of the model is probably because the penalty is higher and therefore might be removing some predictors that affected the growth_2_6 variable. We also initially used PCA instead of PLS for dimension reduction. PCA is unsupervised and therefore doesn't consider the response variable growth_2_6. PLS on the other hand does not ignore the response variable and therefore worked better for our model.

In order to find the best parameters for our bagging model, we ran a nested for loop. We conducted 5-fold cross validation on 70% of the training data and tuned the alpha and lambda parameters for elastic net as well as the ntree, mtry and nodesize for bagging. We found that the best value for the number of trees was 1500, the number of variables randomly sampled at each split was the total number of predictors and minimum size of terminal nodes was 8. Since our mtry value is the same as the number of our predictors, bagging was the best model. This nested for loop showed us the best values that outperformed all other models as this combination gave us the lowest RMSE (refer to figure 2).

Bagging worked best for our model as it constructs deep trees with low bias. It has the same low bias as decision trees and should combat variance. Bagging averages predictions from decision trees over a collection of bootstrap samples. Averaging reduces variance and therefore our predictions are improved. There is no risk of overfitting and therefore, we looped through 1000 to 2000 trees and found 1500 trees to work the best. Because bagging has low bias and we combat variance through methods like elastic net and averaging predictions, our prediction errors were low.

Once we reduced our set of predictors using the tuned alpha and lambda parameters, we proceeded to fit the bagging model using the tuned ntree, mtry and nodesize parameters on the entire training data. This is when using bagging came to our advantage as the "out-of-bag" (OOB) error estimate gave us an indicator of the model's performance. Although the OOB estimate is a good initial validation check, we still need an independent test set. Since we did not know the true values of our given test dataset, we ran elastic net and the bagging model with our tuned parameters on 70% of the training data set and made predictions on the remaining 30%. We compared these predictions with the true values of the growth_2_6 variable in the 30% dataset by calculating the RMSE. For our final model, the RMSE for the 30% data set was 1.499321.

Results

After performing PLS for dimension reduction on all the HOG features and elastic net, we chose the important predictors. From figure 3, we found that the total views of the channel that was between mid and high, high level image features extracted from the thumbnail image of video (cnn_10), average growth among all videos on the channel that was between low and mid were our most important predictors. We fit a bagging model on the subsetted training dataset with all observations and predicted growth_2_6 values using this trained model. These predictions got a score of 1.40488 on the public leaderboard.

Conclusions

We initially performed dimension reduction using PLS, which transformed the variables into principal components that are orthogonal to each other. This benefited our next step, using an elastic net for driving the coefficients for the less significant predictors down to small values or zero. Finally we fit bagging on the training dataset with the selected predictors. Bagging averages the predictions of deeply constructed decision trees with low bias. Also, using a large number of trees gives more accurate predictions and there is no risk of overfitting with bagging.

Our private leaderboard RMSE was 1.41523, which was a slight increase from the public leaderboard RMSE. This could have been because when we fit the data on a bagging model a large number of trees tend to become “correlated”. Important predictors will be on top of several trees, which will make them all vote in a similar way. Using 60% of the test data in the private leaderboard versus 40% could have increased this correlation between the trees further, causing the predictive accuracy to reduce and thus result in an increase in the RMSE.

Taking measures to reduce this correlation between trees is something that would improve the model. We could potentially use a wider range of values to tune the mtry parameter, which we had not done while fitting our current model as it became computationally complex. This could have in turn changed our other parameters such as ntree and nodesize. In our methodology, we used only 70% of the training data in the nested for loop to tune the parameters. Instead, using the entire training set could have adjusted our tuning parameters and improved our model. We could also use different methods of transforming variables such as running a PLS by extracting both HOG and CNN variables into one group.

Appendix - Graphs

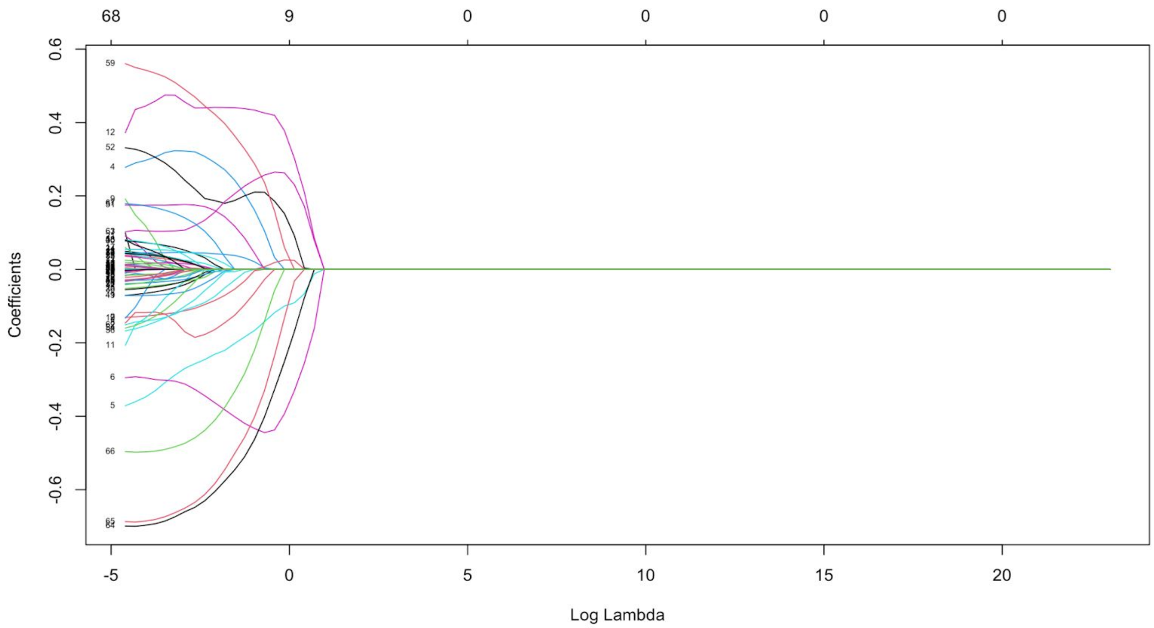


Figure 1: Coefficients that have reduced to zero due to elastic net.

alpha <dbl>	lambda <dbl>	ntree <dbl>	mtry <dbl>	nodesize <dbl>	RMSE <dbl>	n <dbl>
0.5	0.1258925	1500	52	8	1.667341	2

1 row

Figure 2: Data frame showing the best parameters for the model based on cv.

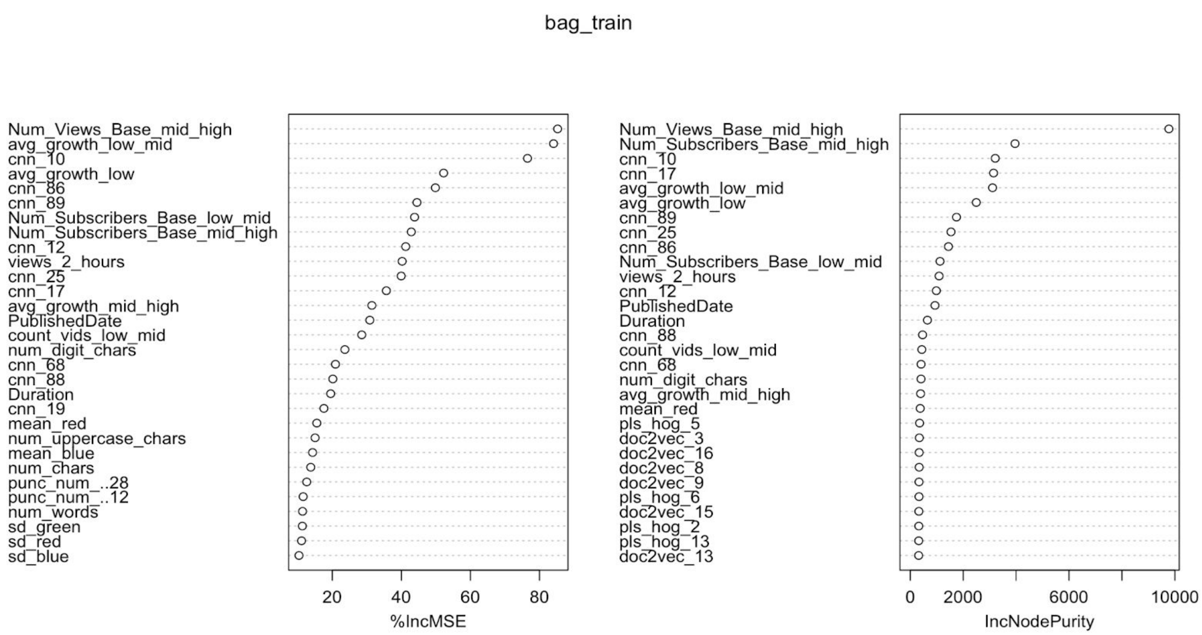


Figure 3: Important predictors after fitting bagging model.

Appendix

Team: VAT

Loading libraries

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(leaps)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.0-2
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
```

```
##
##     date, intersect, setdiff, union
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
library(dplyr)
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:caret':
##
##     R2

## The following object is masked from 'package:stats':
##
##     loadings
```

Loading the data

```r
yt_train <- read.csv("training.csv")
yt_test <- read.csv("test.csv")
any(is.na(yt_test))
```

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

```r
#summary(yt_train)

head(yt_train)
```

```
##   id   PublishedDate Duration views_2_hours       hog_0      hog_1     hog_4
## 1  0 4/17/2020 10:38      483           560 0.363904987 0.13136215 0.3639050
## 2  1  8/31/2020 9:56      226           640 0.000000000 0.00000000 0.4181439
## 3  2 8/16/2020 12:15      358         41139 0.010141719 0.10962013 0.2096886
## 4  3  8/22/2020 9:00     1426          6540 0.000000000 0.00000000 0.4026477
```

```
## 5  4 8/22/2020 14:18       61          340 0.008917475 0.00000000 0.4744185
## 6  5 7/24/2020 21:16      361          544 0.069992325 0.09796317 0.2860610
##       hog_11    hog_13     hog_21    hog_40      hog_60      hog_61     hog_62
## 1 0.02007512 0.3639050 0.00000000 0.3806942 0.006648475 0.008208492 0.37701375
## 2 0.04720125 0.4181439 0.27562590 0.4571939 0.205168972 0.061163339 0.19246280
## 3 0.21220054 0.2754997 0.00000000 0.3001838 0.268383841 0.300183765 0.00520225
## 4 0.00000000 0.4026477 0.17664695 0.6098272 0.009633553 0.014632669 0.04596554
## 5 0.00000000 0.4744185 0.06104809 0.4349003 0.024832150 0.000000000 0.00000000
## 6 0.03787452 0.2860610 0.13776670 0.3138993 0.313899277 0.221301132 0.17928536
##       hog_78     hog_81      hog_89     hog_94    hog_105     hog_106   hog_108
## 1 0.01054524 0.02245025 0.008478517 0.20013316 0.02297729 0.003158798 0.0206199
## 2 0.00000000 0.00000000 0.000000000 0.21208159 0.04315798 0.018174355 0.0000000
## 3 0.00000000 0.00000000 0.033150043 0.09929306 0.20219804 0.121749034 0.0000000
## 4 0.00000000 0.00000000 0.000000000 0.05969038 0.00000000 0.017155598 0.0000000
## 5 0.00000000 0.00000000 0.000000000 0.40971005 0.02135419 0.000000000 0.0000000
## 6 0.00000000 0.00000000 0.000000000 0.11410080 0.08545540 0.132925984 0.0000000
##       hog_116    hog_117   hog_125    hog_132     hog_133    hog_138   hog_139
## 1 0.007787269 0.02051376 0.0427406 0.02110396 0.002901263 0.06514851 0.2755926
## 2 0.000000000 0.00000000 0.0000000 0.04272667 0.017992726 0.29159418 0.2948962
## 3 0.035428447 0.00000000 0.0000000 0.21609513 0.130116852 0.04857156 0.2438648
## 4 0.000000000 0.00000000 0.0000000 0.00000000 0.012591171 0.34433641 0.3443364
## 5 0.000000000 0.00000000 0.0000000 0.02235686 0.000000000 0.22361809 0.4614272
## 6 0.000000000 0.00000000 0.0000000 0.08400773 0.130674126 0.22879120 0.1460956
##      hog_144    hog_152     hog_155    hog_156    hog_165   hog_166    hog_177
## 1 0.01783314 0.03715551 0.002600624 0.11347600 0.05663529 0.2395798 0.33387810
## 2 0.00000000 0.00000000 0.000000000 0.00000000 0.23531988 0.2379847 0.28343946
## 3 0.00000000 0.00000000 0.000000000 0.11222762 0.03747545 0.1881542 0.32885642
## 4 0.00000000 0.00000000 0.000000000 0.00000000 0.34958508 0.3495851 0.34958508
## 5 0.00000000 0.00000000 0.000000000 0.00000000 0.22167263 0.4456293 0.08409504
## 6 0.00000000 0.00000000 0.056175131 0.02933069 0.21730815 0.1387631 0.18457467
##       hog_178    hog_182   hog_183    hog_195    hog_204    hog_205     hog_214
## 1 0.255760608 0.00305819 0.1334415 0.01174540 0.39109909 0.30076033 0.006000555
## 2 0.197008231 0.00000000 0.0000000 0.00000000 0.31973908 0.22223875 0.032717570
## 3 0.328856420 0.00000000 0.1008535 0.08506021 0.31978485 0.32200162 0.322001624
## 4 0.241869281 0.00000000 0.0000000 0.02902601 0.35627788 0.33977920 0.356277884
## 5 0.142558919 0.00000000 0.0000000 0.00000000 0.08660713 0.14681744 0.021266153
## 6 0.004918933 0.05567969 0.0290720 0.00000000 0.18294679 0.00487555 0.046751875
##      hog_215    hog_241   hog_242     hog_259    hog_271   hog_279   hog_287
## 1 0.03485480 0.07110250 0.2999989 0.007788384 0.24392556 0.2439256 0.2014168
## 2 0.01323855 0.08759324 0.1152865 0.060942968 0.08780390 0.2362150 0.2545562
## 3 0.08194776 0.06631212 0.1164950 0.254603444 0.25460344 0.2546034 0.2546034
## 4 0.19378854 0.10794800 0.1432395 0.008938517 0.24884192 0.1027678 0.3076729
## 5 0.16312774 0.00000000 0.0000000 0.000000000 0.05756343 0.3097512 0.3097512
## 6 0.01292583 0.11240806 0.2197816 0.186659335 0.07356951 0.2143093 0.2559957
##     hog_295    hog_303     hog_304    hog_306   hog_314    hog_316    hog_330
## 1 0.1366898 0.02008512 0.002761197 0.26096476 0.1959832 0.08672688 0.01718608
## 2 0.1810236 0.05088650 0.021428931 0.30449937 0.3044994 0.03164357 0.12103254
## 3 0.1572710 0.16041073 0.096587737 0.27977235 0.2797724 0.07127928 0.18960375
## 4 0.1973520 0.00000000 0.014008921 0.09017672 0.2763246 0.19534938 0.00000000
## 5 0.1064054 0.01341816 0.000000000 0.33181176 0.3318118 0.12394039 0.01288768
## 6 0.2532564 0.07210237 0.112155318 0.20792430 0.2523767 0.12164616 0.07193927
##      hog_336   hog_337     hog_341    hog_342    hog_350   hog_351   hog_359
## 1 0.05305390 0.2244297 0.118098361 0.20863512 0.25302465 0.2530247 0.2530247
## 2 0.35011138 0.3501114 0.164897221 0.04971412 0.04964305 0.0000000 0.0000000
```

```
## 3 0.04261711 0.2139691 0.062821012 0.24534765 0.05434617 0.2225514 0.2453477
## 4 0.26363705 0.2636370 0.036024322 0.14584377 0.26363705 0.2636370 0.1684228
## 5 0.12890534 0.3301406 0.008597386 0.33014056 0.33014056 0.3301406 0.3301406
## 6 0.19592332 0.1251077 0.226837882 0.08221304 0.13403466 0.1057285 0.0231181
##      hog_363   hog_364     hog_368    hog_375     hog_376   hog_378    hog_386
## 1 0.05496436 0.2325114 0.122351074 0.23617902 0.236179018 0.2361790 0.23617902
## 2 0.25370148 0.2565744 0.050647082 0.30557985 0.212397190 0.0000000 0.00000000
## 3 0.03645657 0.1830387 0.053739888 0.27899269 0.278992687 0.1903804 0.27899269
## 4 0.27468894 0.2746889 0.028531694 0.27468894 0.152606822 0.2746889 0.13339288
## 5 0.12861630 0.2871364 0.008578108 0.04879264 0.082713868 0.2871364 0.28713639
## 6 0.20580515 0.1314178 0.238278952 0.17480439 0.004658554 0.1110612 0.02428411
##      hog_402     hog_403     hog_412     hog_413    hog_442     hog_452
## 1 0.25874232 0.258742324 0.006748444 0.039198982 0.27117657 0.109520602
## 2 0.23974935 0.166640858 0.024532553 0.009926638 0.07343120 0.066958692
## 3 0.26234134 0.287342966 0.287342966 0.067227339 0.24573392 0.008157832
## 4 0.30628788 0.191079435 0.238226316 0.108979607 0.13702115 0.038927039
## 5 0.05693884 0.096523396 0.013981182 0.107246414 0.07754143 0.000000000
## 6 0.23593222 0.006287616 0.060292248 0.016669439 0.10024280 0.050697643
##      hog_453    hog_454    hog_469    hog_476    hog_477   hog_485     hog_492
## 1 0.27117657 0.11201937 0.24395456 0.24395456 0.24395456 0.2134310 0.151123181
## 2 0.14358713 0.29806349 0.06434858 0.06374154 0.17311417 0.2891409 0.049672262
## 3 0.00000000 0.16184243 0.24675894 0.07969636 0.28473734 0.2847373 0.218814156
## 4 0.11345092 0.09048481 0.13176597 0.09735204 0.05441726 0.1864844 0.161066487
## 5 0.01987548 0.03489127 0.05459071 0.27314712 0.27314712 0.2731471 0.016934118
## 6 0.03627184 0.03806346 0.06640257 0.01294812 0.19343183 0.2347858 0.000417111
##      hog_495    hog_499   hog_512    hog_514    hog_522    hog_523    hog_526
## 1 0.07655389 0.05487324 0.2144883 0.09491579 0.07693313 0.09939283 0.05514507
## 2 0.14622173 0.23834126 0.3239109 0.01520791 0.13217459 0.14102669 0.21544445
## 3 0.10759088 0.01274450 0.3257021 0.06938175 0.13337389 0.01003957 0.01579859
## 4 0.19098398 0.24919721 0.2261409 0.14295222 0.23159733 0.26507333 0.26507333
## 5 0.04214436 0.27314712 0.2847834 0.10236314 0.04117631 0.10025433 0.28478342
## 6 0.17170835 0.40249726 0.2355032 0.11351311 0.17223299 0.02906247 0.44684426
##     hog_549    hog_557    hog_576    hog_584    hog_640     hog_641     hog_643
## 1 0.2628976 0.26289756 0.28418901 0.28418901 0.11712066 0.002332154 0.220550073
## 2 0.0000000 0.00000000 0.00000000 0.00000000 0.17840567 0.164264405 0.303071487
## 3 0.2254513 0.26354553 0.19174607 0.28445581 0.29919695 0.246657012 0.011117740
## 4 0.2707373 0.15434366 0.25074826 0.10886071 0.25597889 0.000000000 0.083826453
## 5 0.2890204 0.28902040 0.27213172 0.27213172 0.09128912 0.005339895 0.110247017
## 6 0.0954334 0.02086702 0.09102241 0.01990253 0.26618339 0.105537740 0.009174139
##       hog_649    hog_651     hog_655   hog_657     hog_658    hog_659
## 1 0.254412307 0.25441231 0.061455585 0.2163748 0.254412307 0.07079814
## 2 0.139983243 0.11985586 0.099803161 0.1079381 0.083864041 0.08260458
## 3 0.092699162 0.00000000 0.097671186 0.2711694 0.199473178 0.07018102
## 4 0.133215969 0.11277030 0.263295914 0.0748389 0.162208724 0.15048744
## 5 0.003052375 0.02584037 0.004160804 0.3466963 0.005505511 0.02404884
## 6 0.027097642 0.03571173 0.044480216 0.2816694 0.281669424 0.19634727
##      hog_665    hog_666    hog_668    hog_669     hog_673     hog_674
## 1 0.22861232 0.04288962 0.16126242 0.12463751 0.021884223 0.130730959
## 2 0.13744628 0.05111571 0.06433913 0.07624181 0.000000000 0.084901056
## 3 0.26318867 0.24778295 0.17326016 0.05754314 0.133899846 0.116682710
## 4 0.24957344 0.02838621 0.03745664 0.10116321 0.000000000 0.088079325
## 5 0.18775854 0.00000000 0.01093186 0.03405510 0.000867329 0.002762731
## 6 0.05620833 0.28166942 0.10919894 0.13948403 0.101224689 0.269693250
##      hog_675    hog_676     hog_677    hog_678    hog_686    hog_697    hog_698
```

```
## 1 0.25441231 0.16223893 0.254412307 0.08165763 0.07056590 0.05627575 0.12355030
## 2 0.07001527 0.00967384 0.015016901 0.05110693 0.07906292 0.21729685 0.16682885
## 3 0.06628204 0.10357897 0.076301512 0.07688950 0.07338882 0.01354202 0.00000000
## 4 0.02071629 0.05110896 0.016298874 0.07883244 0.14528207 0.31454701 0.14717483
## 5 0.02755429 0.01051655 0.008073653 0.02148929 0.01706260 0.38190849 0.04255054
## 6 0.08540257 0.01146606 0.202733912 0.12299700 0.12333900 0.45100352 0.08037225
##      hog_702     hog_703     hog_704    hog_705    hog_711    hog_716
## 1 0.25484451 0.161706739 0.254844515 0.08138976 0.25484451 0.09584749
## 2 0.06701338 0.009259076 0.014373054 0.04891573 0.03664890 0.23218778
## 3 0.06931163 0.108313307 0.079789065 0.08040392 0.18233935 0.16061174
## 4 0.01999971 0.049341099 0.015735095 0.07610562 0.06794750 0.05142435
## 5 0.01954971 0.007461469 0.005728241 0.01524660 0.08083339 0.05676763
## 6 0.05364713 0.007202606 0.127350878 0.07726273 0.00000000 0.02463826
##       hog_719    hog_724    hog_725    hog_738    hog_743     hog_746
## 1 0.084405393 0.06349573 0.13940138 0.25887388 0.10814439 0.095234316
## 2 0.039840559 0.19856676 0.15244889 0.03348992 0.21217415 0.036406467
## 3 0.183799968 0.01696534 0.00000000 0.22843325 0.20121308 0.230263104
## 4 0.007377114 0.29905367 0.20802066 0.09603873 0.07268449 0.010427001
## 5 0.026467990 0.29696193 0.03312166 0.06292132 0.04418835 0.020602884
## 6 0.010927905 0.46828672 0.06851574 0.00000000 0.02100362 0.009315822
##       hog_747     hog_755     hog_774     hog_782     hog_783     hog_788
## 1 0.258873877 0.194022617 0.266547077 0.266547077 0.029357545 0.047250268
## 2 0.001312701 0.000936375 0.001660698 0.001184607 0.005832644 0.005422278
## 3 0.129052848 0.223580870 0.141472109 0.245096933 0.262978449 0.007539162
## 4 0.011713294 0.022441708 0.010631591 0.020369254 0.195482343 0.038278323
## 5 0.258285407 0.126191557 0.275869322 0.134782602 0.048006637 0.039628434
## 6 0.003597218 0.008283069 0.003701904 0.008524122 0.000000000 0.005768534
##      hog_791     hog_797     hog_810     hog_815    hog_818     hog_819
## 1 0.02908354 0.072662604 0.044720698 0.071976896 0.04430331 0.255807299
## 2 0.00000000 0.007947899 0.005992112 0.005570527 0.00000000 0.303397040
## 3 0.30496155 0.010910407 0.207797062 0.005957202 0.28209705 0.282097046
## 4 0.05238421 0.082960376 0.120260010 0.023548682 0.03222657 0.244132426
## 5 0.06693068 0.081682786 0.062659360 0.051723938 0.08735945 0.000000000
## 6 0.00000000 0.225702045 0.000000000 0.005458077 0.00000000 0.001616307
##      hog_825     hog_827   hog_828   hog_829    hog_831    hog_832    hog_844
## 1 0.07053871 0.167316626 0.0631726 0.0684044 0.05178900 0.07014128 0.26262166
## 2 0.01022349 0.091772423 0.3056788 0.2642341 0.01504479 0.14053405 0.04692200
## 3 0.08847386 0.282097046 0.2820029 0.2224627 0.06737951 0.02943483 0.02166157
## 4 0.24433295 0.244332954 0.2156757 0.1535288 0.06653925 0.02735927 0.01136469
## 5 0.01061410 0.001578179 0.3600457 0.1145999 0.05694702 0.45917989 0.00504594
## 6 0.00000000 0.000000000 0.0000000 0.2230130 0.16849948 0.42609239 0.13154558
##      hog_849     hog_852    hog_855     hog_856     hog_857     hog_858
## 1 0.08336813 0.039741401 0.21246651 0.150124994 0.226998367 0.262621662
## 2 0.09104079 0.008375604 0.05502668 0.005633907 0.080557616 0.154455445
## 3 0.06013976 0.070365108 0.28200287 0.282002875 0.147328590 0.164591257
## 4 0.13742590 0.250733717 0.09234138 0.037636275 0.097262714 0.250733717
## 5 0.02480824 0.013116956 0.00000000 0.000000000 0.039051597 0.061210629
## 6 0.11429000 0.000000000 0.01294061 0.058445416 0.008277068 0.001400258
##     hog_859    hog_860     hog_863 cnn_0 cnn_9    cnn_10     cnn_12   cnn_17
## 1 0.2626217 0.21569001 0.262621662     0     0 4.1840990 2.72285080 1.898573
## 2 0.3056788 0.06175981 0.045091900     0     0 2.2659173 3.73716260 2.908048
## 3 0.2820029 0.05795232 0.077544898     0     0 2.3478550 4.40463070 3.382686
## 4 0.2507337 0.07973226 0.183962072     0     0 2.2632027 2.85689930 1.914747
## 5 0.4591799 0.01067384 0.001630227     0     0 2.8848767 5.18916030 3.982236
```

```
## 6 0.4260924 0.01546657 0.019207519     0     0 0.3315478 0.08884464 3.779148
##      cnn_19 cnn_20   cnn_25 cnn_35 cnn_36 cnn_37 cnn_39 cnn_65    cnn_68
## 1 12.734538      0 3.254870      0      0      0      0      0 12.982931
## 2  7.454588      0 3.863644      0      0      0      0      0  8.817122
## 3  8.929517      0 4.280438      0      0      0      0      0 11.239214
## 4  5.158570      0 2.734765      0      0      0      0      0  7.541817
## 5  8.992725      0 4.837797      0      0      0      0      0 11.760019
## 6 16.922571      0 4.776662      0      0      0      0      0 14.115436
##     cnn_86   cnn_88   cnn_89 pct_nonzero_pixels mean_pixel_val sd_pixel_val
## 1 4.026877 6.336098 5.929763          0.7682639      104.57191     92.30211
## 2 1.839474 4.251849 1.707943          0.7600733      117.72881     96.04252
## 3 2.056389 4.865271 2.959120          0.7558160       89.28390     82.39884
## 4 1.185343 3.315254 2.113211          0.7190683       55.93395     64.98214
## 5 2.183827 5.806765 2.710464          0.7658526      119.70391     92.96901
## 6 3.187032 9.653324 7.190603          0.7576447       66.29625     79.29316
##   min_red max_red  mean_red   sd_red min_green max_green mean_green sd_green
## 1       0     255 110.92255 94.08831         0       255  116.17091 97.27989
## 2       0     255 114.58758 98.91974         0       255  118.72323 95.39359
## 3       0     255  95.96794 87.51983         0       255   80.33281 78.28456
## 4       0     255  24.75016 57.95460         0       255   59.34641 57.84243
## 5       0     237 109.91665 90.31101         0       239  118.98117 91.32717
## 6       0     255  59.90086 87.17359         0       255   65.12867 76.97497
##   min_blue max_blue mean_blue  sd_blue edge_avg_value doc2vec_0  doc2vec_1
## 1        0      255  86.62227 82.13400       54.42700 0.7385172 -1.8451594
## 2        0      255 119.87561 93.65884       29.08058 0.7263064  0.4667385
## 3        0      255  91.55094 80.31548       52.39182 0.7691737 -0.6027893
## 4        0      255  83.70527 64.87088       45.48204 0.7125310  1.1055785
## 5        0      255 130.21390 96.05314       34.89519 0.2541418  1.1195441
## 6        0      255  73.85921 72.37658       66.69895 0.2081554 -1.1638162
##    doc2vec_2  doc2vec_3   doc2vec_4 doc2vec_5  doc2vec_6  doc2vec_7  doc2vec_8
## 1 -0.7249853  0.7187983 -0.96192199 0.2882334  0.9457743 -1.6059402 -0.9845942
## 2  0.8805618  1.4257150 -0.08911733 0.5973507 -0.3647046 -1.1683331  0.5213282
## 3 -0.2253620  0.9164937 -0.09241677 0.3671916 -0.5539510 -0.4632470  0.4254579
## 4  0.3923650  1.7253712 -0.15009935 0.1896803 -0.5363334 -0.4925366 -0.7128234
## 5  0.2873214 -1.0048057  0.84798855 1.3320868  0.3845531 -1.6523771 -0.9727676
## 6  0.7917331  0.7660719 -0.53810847 0.6544388 -1.5690154 -0.2481944 -0.3846805
##    doc2vec_9  doc2vec_10 doc2vec_11  doc2vec_12  doc2vec_13  doc2vec_14
## 1 -0.2712932  0.09369647  0.5497547 -0.79252029 -0.03722652 -0.41607961
## 2 -0.5628252 -0.37260357 -0.4226983 -0.65327191 -0.16355355  0.39088678
## 3  0.6853577 -0.97245491  0.1984160 -0.45096496 -0.11303196 -0.51042193
## 4 -0.4851938 -0.83699703 -0.2529746 -0.60007596 -0.43487957 -0.14205198
## 5  1.6761322 -1.10823762  0.7174380  0.02657769 -0.52754843 -0.01327521
## 6  0.6253583 -1.09426785  0.7116535  0.21089082 -0.79745215 -0.57939583
##     doc2vec_15 doc2vec_16   doc2vec_17  doc2vec_18  doc2vec_19 punc_num_..1
## 1 -0.768596172 -1.2182463 -0.006610689  1.04154646 -0.14433199            0
## 2  0.006349204 -0.8302318 -0.313354313 -0.09304868 -0.16404946            0
## 3 -0.764861822 -0.4986935  0.048435513 -0.23351046 -0.78382277            0
## 4 -1.176180840 -0.3243907 -0.267352909 -0.42827782  1.06583571            0
## 5 -0.787316978 -0.8174562 -0.248511821  0.72031134 -0.69862354            0
## 6 -0.479990095 -0.3646929 -0.114320882  0.20121610 -0.01517609            0
##   punc_num_..2 punc_num_..3 punc_num_..4 punc_num_..5 punc_num_..6 punc_num_..7
## 1            0            0            0            0            0            0
## 2            0            0            0            0            0            0
## 3            0            0            0            0            0            0
```

```
## 4           0            0            0            0            0            0
## 5           0            0            0            0            0            0
## 6           0            0            0            0            0            0
##   punc_num_..8 punc_num_..9 punc_num_..10 punc_num_..11 punc_num_..12
## 1            0            0             0             0             0
## 2            0            0             0             0             0
## 3            0            0             0             0             0
## 4            0            0             0             0             0
## 5            0            0             0             0             1
## 6            0            0             0             0             0
##   punc_num_..13 punc_num_. punc_num_..14 punc_num_..15 punc_num_..16
## 1             1          0             0             0             0
## 2             0          0             0             0             0
## 3             0          0             0             1             0
## 4             1          0             0             1             0
## 5             0          0             0             0             0
## 6             0          0             0             1             0
##   punc_num_..17 punc_num_..18 punc_num_..19 punc_num_..20 punc_num_..21
## 1             0             0             0             0             0
## 2             0             0             0             0             0
## 3             0             0             0             0             0
## 4             0             0             0             1             0
## 5             0             0             0             0             0
## 6             0             0             0             0             0
##   punc_num_..22 punc_num_..23 punc_num_..24 punc_num_..25 punc_num__
## 1             0             0             0             0          0
## 2             0             0             0             0          0
## 3             0             0             0             0          0
## 4             0             0             0             0          0
## 5             0             0             0             0          0
## 6             0             0             0             0          0
##   punc_num_..26 punc_num_..27 punc_num_..28 punc_num_..29 punc_num_..30
## 1             0             0             0             0             0
## 2             0             0             1             0             0
## 3             0             0             0             0             0
## 4             0             0             0             0             0
## 5             0             0             0             0             0
## 6             0             0             0             0             0
##   num_words num_chars num_stopwords num_uppercase_chars num_uppercase_words
## 1        10        79             1                   9                   5
## 2         9        43             3                   5                   4
## 3         5        41             0                   5                   5
## 4        14        67             6                  10                  10
## 5        10        51             4                   4                   4
## 6        13        67             5                   3                   3
##   num_digit_chars Num_Subscribers_Base_low Num_Subscribers_Base_low_mid
## 1               2                        0                            0
## 2               0                        1                            0
## 3               0                        0                            1
## 4               0                        0                            1
## 5               0                        1                            0
## 6               6                        0                            0
##   Num_Subscribers_Base_mid_high Num_Views_Base_low Num_Views_Base_low_mid
## 1                             0                  0                      0
```

```
## 2                               0                  0                      1
## 3                               0                  0                      0
## 4                               0                  1                      0
## 5                               0                  0                      1
## 6                               0                  0                      0
##   Num_Views_Base_mid_high avg_growth_low avg_growth_low_mid avg_growth_mid_high
## 1                       1              0                  0                   0
## 2                       0              0                  0                   0
## 3                       1              0                  1                   0
## 4                       0              0                  1                   0
## 5                       0              0                  0                   0
## 6                       1              0                  0                   0
##   count_vids_low count_vids_low_mid count_vids_mid_high growth_2_6
## 1              0                  0                   0  5.1553571
## 2              0                  0                   0  2.7906250
## 3              0                  0                   0  2.0796568
## 4              0                  0                   1  0.6452599
## 5              0                  0                   0  2.3558824
## 6              0                  0                   0  3.5128676
```

```r
yt_train <- yt_train[,-1]

#Converting PublishedDate to DateTime format
yt_train$PublishedDate <- paste(yt_train$PublishedDate, ":00", sep = "")
yt_train$PublishedDate <- mdy_hms(yt_train$PublishedDate)


#Converting PublishedDate to DateTime format
yt_test$PublishedDate <- paste(yt_test$PublishedDate, ":00", sep = "")
yt_test$PublishedDate <- mdy_hms(yt_test$PublishedDate)
```

Removing the highly correlated predictors

```r
# Near-Zero-Variance
nzv <- nearZeroVar(yt_train, saveMetrics= TRUE)
nzv[nzv$nzv,][,]
```

```
##                freqRatio percentUnique zeroVar  nzv
## cnn_0         7241.00000    0.02761668   FALSE TRUE
## cnn_9           85.01190    0.26235846   FALSE TRUE
## cnn_20        7172.00000    0.98039216   FALSE TRUE
## cnn_35           0.00000    0.01380834    TRUE TRUE
## cnn_36        7241.00000    0.02761668   FALSE TRUE
## cnn_37           0.00000    0.01380834    TRUE TRUE
## cnn_39        7212.00000    0.42805855   FALSE TRUE
## cnn_65        7241.00000    0.02761668   FALSE TRUE
## min_red          0.00000    0.01380834    TRUE TRUE
## max_red       1440.00000    0.33140017   FALSE TRUE
## min_green        0.00000    0.01380834    TRUE TRUE
## max_green      295.70833    0.49710025   FALSE TRUE
## min_blue         0.00000    0.01380834    TRUE TRUE
## max_blue       713.00000    0.64899199   FALSE TRUE
## punc_num_..2   138.21154    0.04142502   FALSE TRUE
```

```
## punc_num_..3      45.12739    0.02761668   FALSE TRUE
## punc_num_..4     400.94444    0.05523336   FALSE TRUE
## punc_num_..5     313.86957    0.02761668   FALSE TRUE
## punc_num_..8      30.07725    0.04142502   FALSE TRUE
## punc_num_..9      30.34632    0.04142502   FALSE TRUE
## punc_num_..10    149.79167    0.08285004   FALSE TRUE
## punc_num_..11    135.60377    0.05523336   FALSE TRUE
## punc_num_..14     68.97087    0.05523336   FALSE TRUE
## punc_num_..16    361.10000    0.02761668   FALSE TRUE
## punc_num_..17      0.00000    0.01380834    TRUE TRUE
## punc_num_..18   2413.00000    0.02761668   FALSE TRUE
## punc_num_..19      0.00000    0.01380834    TRUE TRUE
## punc_num_..21    115.75806    0.05523336   FALSE TRUE
## punc_num_..22    153.08511    0.02761668   FALSE TRUE
## punc_num_..23      0.00000    0.01380834    TRUE TRUE
## punc_num_..24    149.87500    0.02761668   FALSE TRUE
## punc_num_..25      0.00000    0.01380834    TRUE TRUE
## punc_num__         0.00000    0.01380834    TRUE TRUE
## punc_num_..26   7241.00000    0.02761668   FALSE TRUE
## punc_num_..27   7241.00000    0.02761668   FALSE TRUE
## punc_num_..29      0.00000    0.01380834    TRUE TRUE
## punc_num_..30      0.00000    0.01380834    TRUE TRUE
## count_vids_low    24.77224    0.02761668   FALSE TRUE
```

```
dim(yt_train)
```

```
## [1] 7242  259
```

```
nzv <- nearZeroVar(yt_train)
filtered_training <- yt_train[, -nzv]
dim(filtered_training)
```

```
## [1] 7242  221
```

```
yt_train <- filtered_training
filtered_training_names <- names(yt_train)[names(yt_train) != 'growth_2_6']

yt_test <- yt_test[, filtered_training_names]
```

Splitting our training data to 70 and 30

```
#70% of data for train and 30% of data for test
train_size = floor(0.7 * nrow(yt_train))

#set the seed
set.seed(123)

#get training indices
train_ind = sample(seq_len(nrow(yt_train)), size = train_size)

data_train = yt_train[train_ind, ]
data_test = yt_train[-train_ind, ]

X_train = model.matrix(growth_2_6 ~., data_train)[,-1]
y_train = data_train$growth_2_6
```

```r
X_test = model.matrix(growth_2_6 ~., data_test)[,-1]
y_test = data_test$growth_2_6
```

Grouping the hog variables & transforming with PLS

```r
hog_data <- yt_train[,4:155]
hog_data$growth_2_6 <- yt_train$growth_2_6


pls_model = plsr(growth_2_6 ~ ., data = hog_data , scale = TRUE, validation = "CV")
model_pls_mse <- MSEP(pls_model, estimate = "CV")$val %>%
  reshape2::melt() %>%
  mutate(M = 0:(nrow(.)-1)) %>%
  select(M, value) %>%
  rename(CV_MSE = value)

ncomps_hog <- model_pls_mse[which.min(model_pls_mse$CV_MSE),] #Finding the min number of components
ncomps_hog
```

```
##     M   CV_MSE
## 14 13 5.876869
```

```r
#validationplot(pls_model, val.type = "MSEP")
#summary(pcr_model)

pls_model = plsr(growth_2_6 ~., data=hog_data, scale=TRUE, ncomp=ncomps_hog$M)
summary(pls_model)
```

```
## Data:    X dimension: 7242 152
##  Y dimension: 7242 1
## Fit method: kernelpls
## Number of components considered: 13
## TRAINING: % variance explained
##             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X             6.521    10.24    13.64    16.63    18.96    21.16    23.44
## growth_2_6    9.278    13.82    15.65    16.60    17.09    17.37    17.58
##             8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X             25.50    27.45     29.53     31.22     32.81     34.90
## growth_2_6    17.74    17.88     17.99     18.09     18.16     18.22
```

```r
# Data Storage
Z = pls_model$scores
#size_Z = object.size(Z)
#print(size_Z)

predictor_names = names(hog_data)[names(hog_data) != 'growth_2_6']
#X = as.matrix(data_train[,predictor_names])
#size_X = object.size(X)
#print(size_X)

#print(as.numeric(size_Z / size_X))
```

```r
proj = pls_model$projection
X = as.matrix(hog_data[,predictor_names])
Z = as.data.frame(scale(X) %*% proj)
#attach PC variables
pls_var_names_hog = apply(as.matrix(1:ncomps_hog$M), 2, function(s){paste('pls_hog', s, sep='_')})
colnames(Z) = pls_var_names_hog

#revert to original data frames
data_train_pls_hog = data_train[,-(4:155)]
data_test_pls_hog = data_test[,-(4:155)]


data_train_pls_hog[,pls_var_names_hog] = Z[1:nrow(data_train_pls_hog),]
data_test_pls_hog[,pls_var_names_hog] = Z[(nrow(data_train_pls_hog)+1):(nrow(yt_train)),]
```

Tuning Parameters

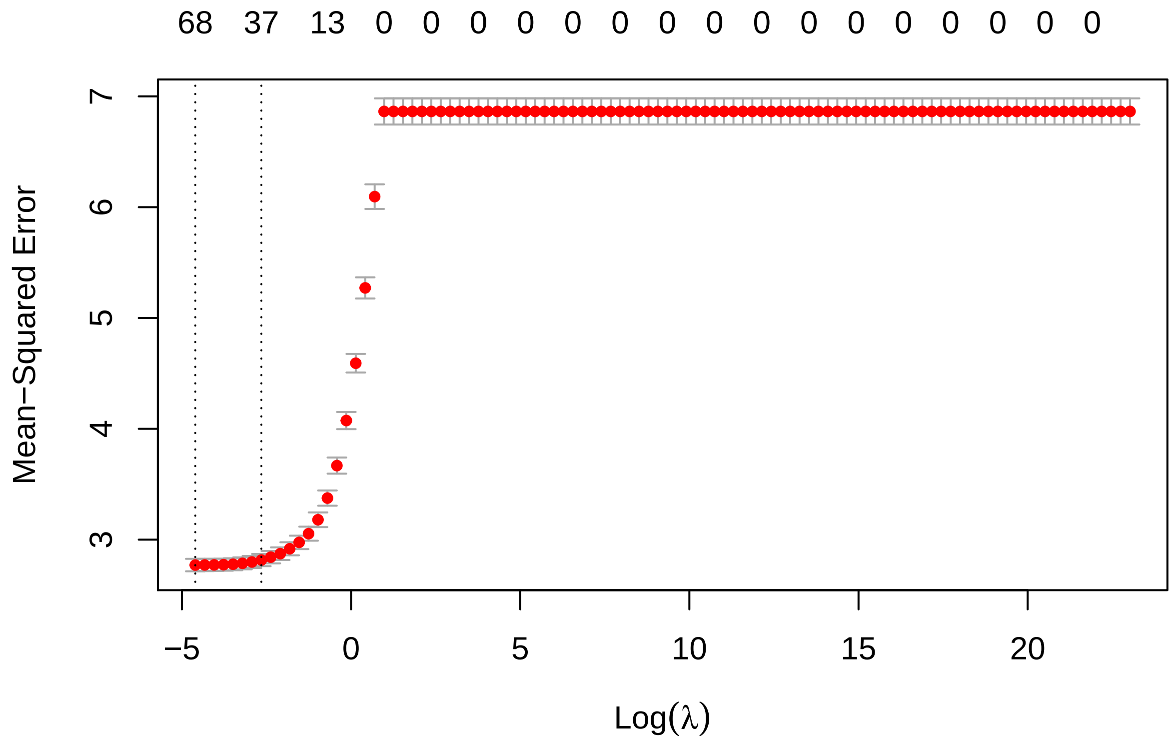
```r
# alpha_vals <- seq(0.1, 0.9, by = 0.2)
# lambda_vals <- 10^seq(-3, 0, by = 0.1)
# ntree_vals <- seq(1000, 2000, by = 500)
# nodesize_vals <- c(5, 8, 10, 25, 37, 50)
# en.best.lambda.cv <- c()
# p <- c()
# count <- 1
# rmse_vals <- c()
# df <- data.frame("alpha" = c(),
#                  "lambda" = c(),
#                  "ntree" = c(),
#                  "mtry" = c(),
#                  "nodesize" = c(),
#                  "RMSE" = c(),
#                  "n" = c())
#
# set.seed(1)
#
# # for - splitting data for cv
# for(n in 1:5){
#
#   i.train <- sample(nrow(data_train), size = round(nrow(data_train)/5))
#   training <- data_train[i.train, ]
#   validation <- data_train[-i.train, ]
#
#   for(i in 1:length(alpha_vals)){
#     # new data train
#     data_train_cv <- training
#     data_test_cv <- validation
#
#     X_train_cv = model.matrix(growth_2_6 ~., data_train_cv)[,-1]
#     y_train_cv = data_train_cv$growth_2_6
#
#     # Elastic Net
#     en.cv.output <- cv.glmnet(X_train_cv, y_train_cv, family = "gaussian",
```

```
#                                alpha = alpha_vals[i], lambda = lambda_vals,
#                                standardize = TRUE, nfolds = 10)
#
#     en.best.lambda.cv[i] <- en.cv.output$lambda.min
#
#     en.mod <- glmnet(X_train_cv, y_train_cv, family = "gaussian",
#                      alpha = alpha_vals[i], lambda = en.cv.output$lambda.min,
#                      standardize = TRUE)
#
#     # Getting predictors after Elastic Net
#     var_imp <- varImp(en.mod, lambda = en.cv.output$lambda.min)
#     var_imp <- as.data.frame(var_imp)
#     rows <- row.names(var_imp)
#     var_imp_df <- data.frame("Predictors" = rows, var_imp$Overall)
#     predictors_en <- ifelse(var_imp$Overall > 0, T, F)
#     predictors_en_names <- var_imp_df[predictors_en,]
#
#     predictors <- c(predictors_en_names$Predictors, "growth_2_6")
#
#     data_train_cv <- data_train_cv[,predictors]
#     data_test_cv <- data_test_cv[,predictors]
#
#     p[i] <- length(predictors) - 1
#     mtry_vals <- c(p[i]/3, p[i])
#
#     # update data with new preds
#
#     for(j in 1:length(ntree_vals)){
#       for(k in 1:length(mtry_vals)){
#         for(l in 1:length(nodesize_vals)){
#
#           # Fitting Random Forest
#           rf_model <- randomForest(growth_2_6 ~ ., data = data_train_cv,
#                                    ntree = ntree_vals[j], mtry = mtry_vals[k],
#                                    nodesize = nodesize_vals[l], importance = TRUE)
#
#           #predict with RF
#           rf_preds = predict(rf_model, data_test_cv)
#
#           rmse_vals[count] <- RMSE(rf_preds, data_test_cv$growth_2_6)
#
#           df[count, 1] <- alpha_vals[i]
#           df[count, 2] <- en.cv.output$lambda.min
#           df[count, 3] <- ntree_vals[j]
#           df[count, 4] <- mtry_vals[k]
#           df[count, 5] <- nodesize_vals[l]
#           df[count, 6] <- RMSE(rf_preds, data_test_cv$growth_2_6)
#           df[count, 7] <- n
#
#           count <- count + 1
#         } # nodesize
#       } # mtry
#     } # ntree
```

```
#   } # elastic net
# } # cv
#
# nrow(df)
#
# names(df) <- c("alpha", "lambda", "ntree", "mtry", "nodesize", "RMSE", "n")
#
# df[which.min(df$RMSE),]

#(length(alpha_vals) * 5 * length(ntree_vals) * 2 * length(nodesize_vals))

#   0.5 0.1258925   1500    52  8   1.667341    2
```

Performing Elastic Net

```
data_train <- data_train_pls_hog
data_test <- data_test_pls_hog

X_train = model.matrix(growth_2_6 ~., data_train)[,-1]
y_train = data_train$growth_2_6

X_test = model.matrix(growth_2_6 ~., data_test)[,-1]
y_test = data_test$growth_2_6

# Let's define a grid of possible values for lambda
i.exp <- seq(10, -2, length = 100)
grid <- 10^i.exp


#Need to fix it for the plot
#X_train <- scale(X_train)

en.mod <- glmnet(X_train, y_train, family = "gaussian", alpha = 0.5,
                    lambda = grid, standardize = TRUE)

# Plots of coefficients.
#plot(en.mod, xvar = "lambda", label = TRUE)

# Select the best value for lambda using K-fold cross-validation.
en.cv.output <- cv.glmnet(X_train, y_train, family = "gaussian",
                          alpha = 0.5, lambda = grid,
                          standardize = TRUE, nfolds = 10)

plot(en.cv.output)
```




```
# Retrieve the actual best value of lambda.
en.best.lambda.cv <- en.cv.output$lambda.min
en.best.lambda.cv
```

```
## [1] 0.01
```

```
var_imp <- varImp(en.mod, lambda = en.best.lambda.cv)
var_imp <- as.data.frame(var_imp)
rows <- row.names(var_imp)
var_imp_df <- data.frame("Predictors" = rows, var_imp$Overall)
predictors_en <- ifelse(var_imp$Overall > 0, T, F)
predictors_en_names <- var_imp_df[predictors_en,]

length(predictors_en_names$Predictors)
```

```
## [1] 68
```

```
predictors <- c(predictors_en_names$Predictors, "growth_2_6")
```

Using the predictors from en model and fitting trees

Fitting the model on the full train data

```
# hog transformation
yt_train_pls_hog = yt_train[,-(4:155)]
yt_train_pls_hog[,pls_var_names_hog] = Z[1:nrow(yt_train),]




yt_train_bag <- yt_train_pls_hog[,predictors]
```

```r
bag_train = randomForest(growth_2_6 ~ .,data = yt_train_bag,
                    mtry = length(predictors) - 1, importance =TRUE,
                    ntrees = 1500, nodesize = 8)

# Using OOB error estimate
RMSE(bag_train$predicted, yt_train_bag$growth_2_6)
```

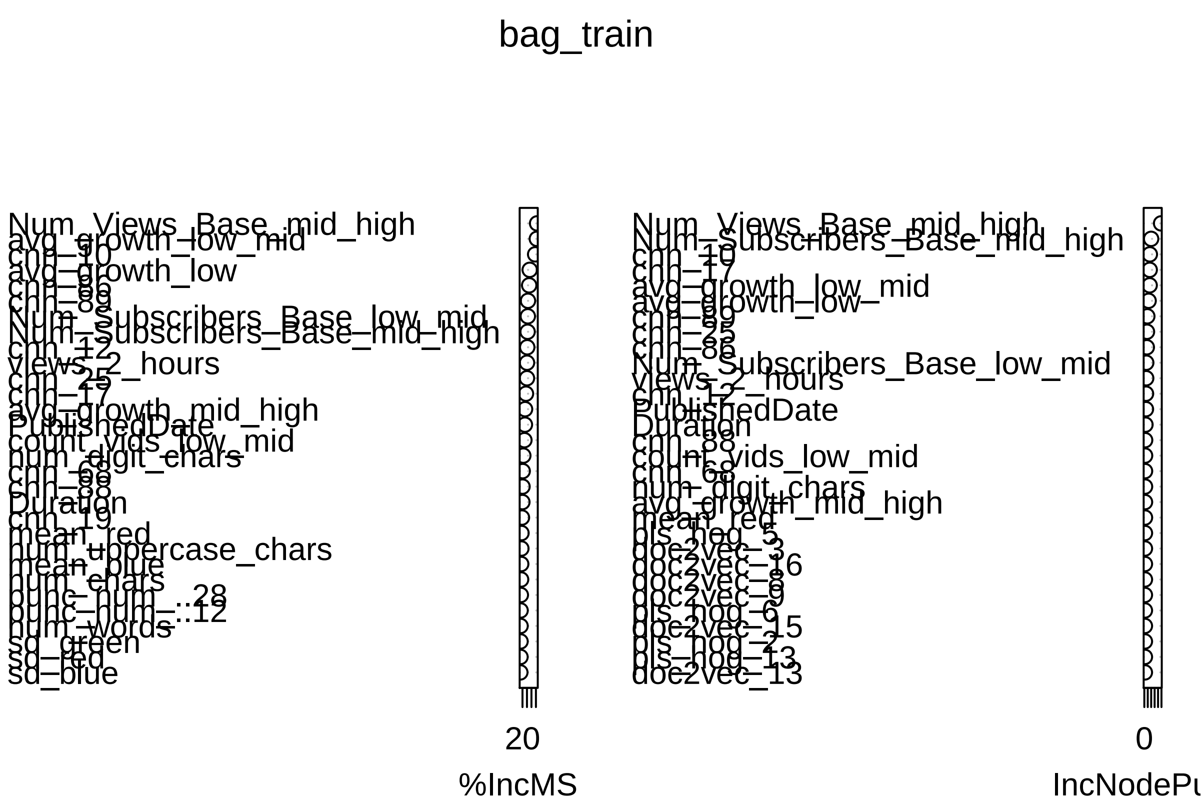
```
## [1] 1.476912
```

Bagging on 70% of training data

```r
set.seed(1)
data_train_tree <- data_train[, predictors]
data_test_tree <- data_test[, predictors]

bag = randomForest(growth_2_6 ~ .,data = data_train_tree,
                    mtry = length(predictors) - 1, importance =TRUE,
                    ntrees = 1500, nodesize = 8)

yhat.bag = predict(bag, newdata = data_test_tree)

RMSE(yhat.bag, data_test_tree$growth_2_6)
```

```
## [1] 1.499321
```

```r
varImpPlot(bag_train)
```



Fitting the model on test data

```r
# hog transformation
yt_test_pls_hog = yt_test[,4:155]

X = as.matrix(yt_test_pls_hog[,predictor_names])
Z_test = as.data.frame(scale(X) %*% proj)
pls_var_names_hog = apply(as.matrix(1:ncomps_hog$M), 2, function(s){paste('pls_hog', s, sep='_')})
colnames(Z_test) = pls_var_names_hog

yt_test_pls_hog_final = yt_test[,-(4:155)]

yt_test_pls_hog_final[,pls_var_names_hog] = Z_test[1:nrow(yt_test_pls_hog_final),]

print(head(yt_test_pls_hog_final))
```

```
##         PublishedDate Duration views_2_hours    cnn_10    cnn_12    cnn_17
## 1 2020-07-31 21:48:00      137           353 2.2282567 1.0552410 0.8439769
## 2 2020-04-25 08:00:00      466          7820 0.7708848 2.8084993 2.2903674
## 3 2020-08-10 23:54:00      146          1602 2.2472627 0.5593343 0.5049460
## 4 2020-07-30 08:19:00      329          4232 1.6343858 1.5885264 0.0000000
## 5 2020-08-10 06:00:00      112           204 0.8716856 0.0000000 0.2596904
## 6 2020-08-02 18:50:00       93          1900 1.5376674 2.2627034 1.5777969
##     cnn_19   cnn_25   cnn_68   cnn_86   cnn_88    cnn_89 pct_nonzero_pixels
## 1 6.618105 1.489002 5.671893 2.583460 2.593078 3.6723633          0.9226292
## 2 4.425623 3.193791 5.053873 1.780094 4.049758 0.3209431          0.7062346
## 3 8.408702 1.713054 6.192671 3.877300 4.437484 4.5279274          0.7671200
## 4 4.521935 2.426096 7.816822 1.787179 2.110607 2.6066034          0.7634722
## 5 8.059477 1.636617 3.088646 4.482940 4.013352 3.4067060          0.7659240
## 6 6.764500 3.367607 7.844359 2.655847 5.525042 2.0583134          0.7665529
##   mean_pixel_val sd_pixel_val  mean_red    sd_red mean_green sd_green mean_blue
## 1      117.61617     92.16556 106.88318  97.32201  112.58543 90.10003 133.37991
## 2      117.33567     98.73671 134.50128 103.55183  116.65834 96.08568 100.84737
## 3      118.89191     95.58527 118.33161  91.01845  121.27437 97.50293 117.06976
## 4       81.43967     77.30366  90.91508  84.56694   80.29094 75.53887  73.11300
## 5       89.54165     74.35359 116.48895  81.21838   89.37847 68.28469  62.75753
## 6       85.95384     70.25759  83.42923  72.48547   87.06824 69.05949  87.36405
##    sd_blue edge_avg_value doc2vec_0   doc2vec_1  doc2vec_2  doc2vec_3
## 1 86.63070       63.62762 0.2775769 -0.43919882  0.1745114  0.5153695
## 2 93.40448       37.02866 1.2597632 -0.91790110  0.6121376  0.5446451
## 3 98.02623       38.62724 0.7910371 -0.97162121 -0.4811322  0.2228500
## 4 70.06743       52.43571 0.7833609  0.41186795 -0.6855040 -0.7238917
## 5 62.31030       33.14582 1.1107637 -1.04251194 -0.3130709  0.6535609
## 6 69.10446       40.52465 0.5934411  0.01406487 -0.2676050  0.8075056
##     doc2vec_4   doc2vec_5  doc2vec_6   doc2vec_7  doc2vec_8   doc2vec_9
## 1 -0.28792566  0.28279692 -0.9793835 -0.34427845 -0.1602731  0.40731996
## 2 -1.67114675  0.08878684 -1.0456047 -0.24748121 -1.0496852  0.60118890
## 3  0.87183559 -0.01146263 -0.5603901 -0.66888559 -0.5410829 -0.01099389
## 4  0.40834472  0.01267466 -1.3426918 -0.83616251  0.2711654  0.89496833
## 5  0.07835364 -0.39280599 -1.1547202  0.02444992 -1.6486804  0.70537221
## 6  0.18675269  0.56679899 -1.0557666 -0.63871098  0.5288880  0.44205624
##   doc2vec_10  doc2vec_11  doc2vec_12   doc2vec_13 doc2vec_14  doc2vec_15
## 1 -0.4782745  0.415244758 -0.26429161 -0.223968372 -0.3729997 -0.35363176
## 2 -0.3273304  1.397056818  0.27543569  0.312654495  0.3690019 -0.08083221
```

```
## 3 -0.8579616  0.225897357 -0.48315492 -0.589098334 -0.2292313 -1.00645673
## 4  0.6546971  0.938541770 -0.62812215 -0.793963313 -0.9696600 -0.57603025
## 5 -0.4416643 -0.003293813  0.55067205  0.035068419 -1.0744796 -0.32095629
## 6 -0.7816467  0.328091592  0.09307414 -0.009394798 -0.3154699 -0.13214625
##    doc2vec_16  doc2vec_17  doc2vec_18  doc2vec_19 punc_num_..1 punc_num_..6
## 1 -0.45506415 -0.01216792 -0.02537384  0.02999593            0            0
## 2  0.13311517 -0.09529759 -0.23871782  0.81038314            0            0
## 3 -0.45339215  0.33566257  1.15237403 -0.72405267            0            0
## 4 -0.17138481 -0.05096229  0.92166013 -0.13050926            0            0
## 5 -0.68120474 -0.21581963  0.40686700 -0.99654311            0            0
## 6 -0.03229506 -0.32515737 -0.26799589  0.14190097            0            0
##   punc_num_..7 punc_num_..12 punc_num_..13 punc_num_. punc_num_..15
## 1            0             0             0          0             1
## 2            0             0             0          0             0
## 3            1             0             0          0             0
## 4            0             0             0          2             0
## 5            0             0             1          0             0
## 6            1             0             0          0             1
##   punc_num_..20 punc_num_..28 num_words num_chars num_stopwords
## 1             0             0         8        42             1
## 2             0             0         9        50             4
## 3             0             0         7        48             1
## 4             0             1        15        77             3
## 5             0             0        10        76             2
## 6             0             0         7        39             2
##   num_uppercase_chars num_uppercase_words num_digit_chars
## 1                   2                   2               5
## 2                   9                   9               0
## 3                   6                   2               0
## 4                  15                  10               0
## 5                   8                   4               2
## 6                   6                   6               0
##   Num_Subscribers_Base_low Num_Subscribers_Base_low_mid
## 1                        0                            0
## 2                        0                            0
## 3                        0                            0
## 4                        0                            1
## 5                        0                            0
## 6                        1                            0
##   Num_Subscribers_Base_mid_high Num_Views_Base_low Num_Views_Base_low_mid
## 1                             0                  0                      0
## 2                             0                  0                      0
## 3                             0                  0                      0
## 4                             0                  0                      0
## 5                             0                  0                      0
## 6                             0                  0                      1
##   Num_Views_Base_mid_high avg_growth_low avg_growth_low_mid avg_growth_mid_high
## 1                       1              0                  0                   0
## 2                       0              0                  0                   0
## 3                       1              0                  0                   0
## 4                       1              0                  0                   0
## 5                       1              0                  0                   0
## 6                       0              0                  0                   0
##   count_vids_low_mid count_vids_mid_high pls_hog_1    pls_hog_2  pls_hog_3
```

```
## 1                  0                   0  4.635836 -0.392650928  3.1421216
## 2                  0                   0  1.886488 -1.906926098  0.2319708
## 3                  0                   0  4.707766  0.006561441 -3.3821368
## 4                  0                   0  4.047605  2.501952269  2.6409570
## 5                  0                   0 -1.878300 -0.295700998  1.0260223
## 6                  0                   0  1.625553  1.098960055  3.3039724
##   pls_hog_4  pls_hog_5  pls_hog_6   pls_hog_7   pls_hog_8   pls_hog_9
## 1 -3.596076  2.4593354 -3.4846423  0.55384337  0.82133153 -2.32778274
## 2  1.097496  0.6650316 -1.3259187  0.65969631  1.22123111 -0.57439306
## 3  1.356320 -1.4699897  1.2759902 -0.58532559 -0.92299491  0.21492474
## 4 -1.174629  1.2106123  0.7470483 -0.08081831 -0.08493367 -0.67336269
## 5  1.400909  1.7926928  0.5400897 -0.85199069  1.47601868  0.09782922
## 6  2.101307  0.2576729  1.4114819  1.89710852  2.25736051  1.67307964
##    pls_hog_10 pls_hog_11 pls_hog_12 pls_hog_13
## 1  3.31746407 -3.1654055 -1.5016929  2.1667713
## 2  0.49033966 -0.2187207 -0.2454007  1.0431056
## 3 -0.55910492  0.3296007  0.6874205 -1.5331506
## 4 -1.43885627  2.5010307 -0.2761556  0.0478525
## 5  0.06720956  1.0042598  1.7747130  1.6774788
## 6 -1.05142264 -1.3096157 -0.5100916 -1.5085652
```

```r
# predicting with bagging
yt_test_bag <- yt_test_pls_hog_final[,predictors[-length(predictors)]]
yhat.bag.test = predict(bag_train, newdata = yt_test_bag)
range(yhat.bag.test)
```

```
## [1] 0.2510028 7.8864570
```

```r
yt_test <- read.csv("test.csv")

sol <- data.frame(yt_test$id, yhat.bag.test)
names(sol) <- c("id", "growth_2_6")
head(sol)
```

```
##     id growth_2_6
## 1 7242   5.392464
## 2 7243   2.924533
## 3 7244   3.972683
## 4 7245   6.743082
## 5 7246   4.419929
## 6 7247   2.992420
```

```r
#write.csv(sol, file = "sol_26.csv", row.names = F)
```

Team Contributions

Alekhya Vittalam (UID: 604995902)
– cleaning and preprocessing the data
– selection of features
– visualization of the data
– collaborated with other team members on selection of model and tuning parameters
– consolidating our procedures to write the report

Tanvi Pati (UID: 104901736)
– cleaning and preprocessing the data
– selection of features
– visualization of the data
– collaborated with other team members on selection of model and tuning parameters
– consolidating our procedures to write the report

Vansika Saraf (UID: 804996439)
– cleaning and preprocessing the data
– selection of features
– visualization of the data
– collaborated with other team members on selection of model and tuning parameters
– consolidating our procedures to write the report

All 3 team members collaborated on the project over zoom. We equally split the responsibilities of programming the model, doing research on various methods to make accurate predictions and selecting significant predictors. We took turns to share our screen and write the code.