Actual Final

Zoe Wang

12/11/2020

Step 0: Importing Data

```
yt_train_uncleaned <- read.csv("training.csv")
test <- read.csv("test.csv")
```

Step 1: Cleaning the Data

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
any(apply(yt_train_uncleaned, 2, function(x) sum(is.na(x))) > 0) # No NA values found
```

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

```
any(apply(yt_train_uncleaned, 2, function(x) sum(is.infinite(x))) > 0) # No infinite values found
```

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

```
any(apply(yt_train_uncleaned, 2, function(x) sum(is.nan(x))) > 0) # No NaN values found
```

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

```
sum(duplicated(yt_train_uncleaned[, -1])) # No duplicated rows
```

```
## [1] 0
```

```
summary(yt_train_uncleaned$Duration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   121.0   292.0   712.5   576.8 42895.0
```

```
# Nothing unusual with the duration data (such as 0 sec videos).
# A video of 42895 seconds (or ~12 hours) isn't an unusual thing to find on YT
```

```r
summary(yt_train_uncleaned$views_2_hours)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##        1      644     3016    27598    13750 13365727
```

```r
# Nothing too unusual here either unless you count the outliers

useless_variables <- which(apply(yt_train_uncleaned[, -c(1, 2, dim(yt_train_uncleaned)[2])], 2, sum) ==
# Grabs the indices that have nothing but zeroes
yt_train <- yt_train_uncleaned[, -c(useless_variables + 2)] # Remove useless variables
test <- test[,-c(useless_variables + 2)]


binary_v <- 236:247 # Indices for predictors that are binary
yt_train_scaled <- yt_train %>%
mutate_if(is.numeric, scale)
test_scaled <- test %>%
mutate_if(is.numeric, scale)
# Scales data in case scaling is needed for model
#yt_train_scaled[, binary_v] <- yt_train[, binary_v]
# Restores the scaled binary variables to their unscaled state

write.csv(yt_train, "training_clean.csv")
write.csv(test, "test_clean.csv")
```

```r
#Loading in the cleaned data
training <- read.csv("training_clean.csv")
training <- training[, -1] #Removing X column

test <- read.csv("test_clean.csv")
test <- test[,-1] #Removing X column

library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.2
```

```
## 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
```

```r
set.seed(1)

#Removing Publication date because it's not a quantitative predictor
training <- training[,-2]
test <- test[,-2]

#Saving the test id's for later
testid <- cbind(test$id)

#Removing the id's
```

```r
training <- training[,-1]
test <- test[,-1]

#Creating a correlation matrix for use in findCorrelation
corrmx <- cor(training)

#Finding and removing variables with correlations > 0.8
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

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

```
## Loading required package: ggplot2
```

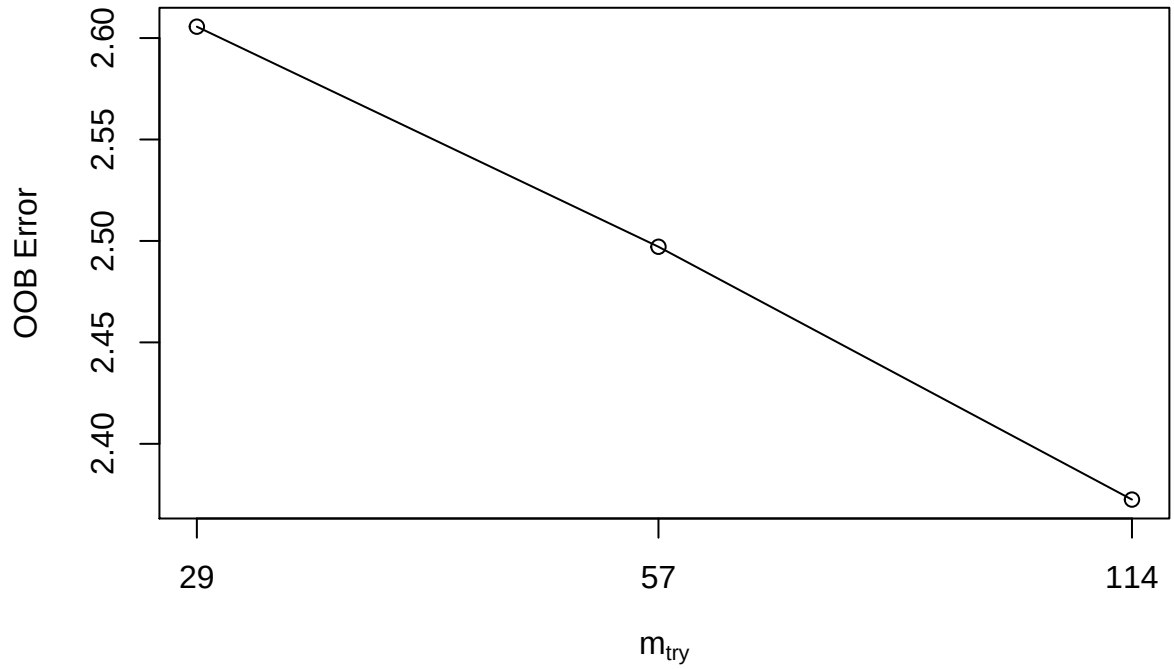
```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
##
## Attaching package: 'ggplot2'
```

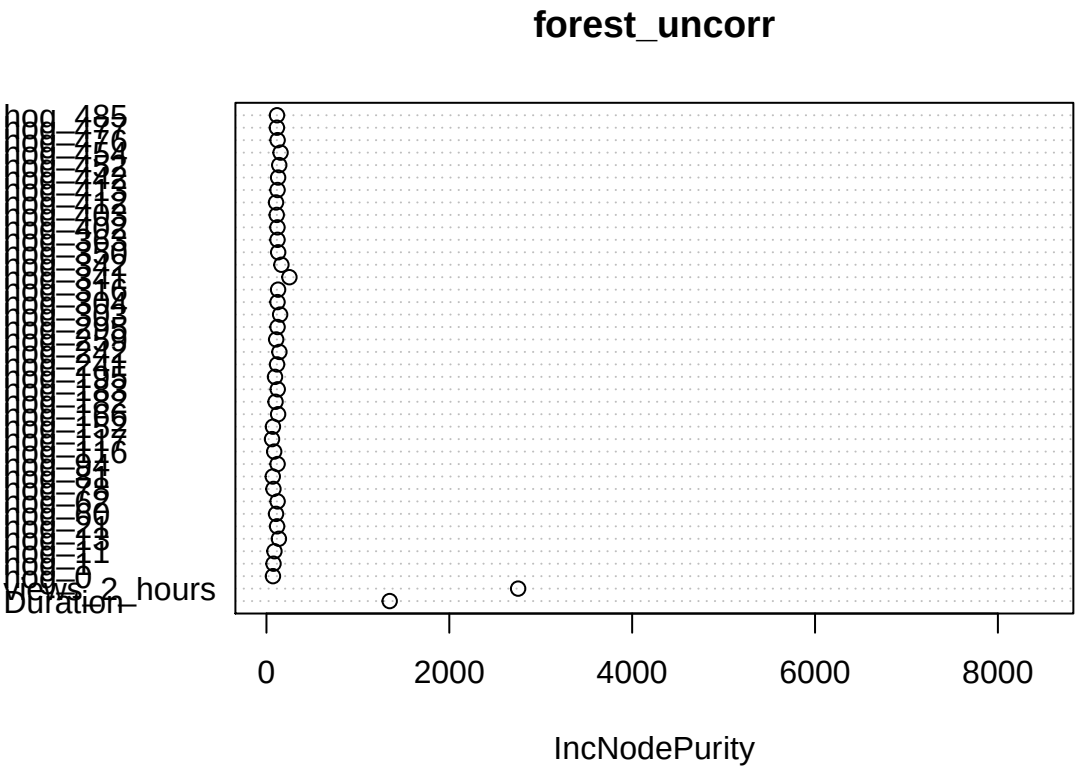
```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```r
correlated_vars <- findCorrelation(corrmx, cutoff = 0.8)
training_uncorr <- training[,-correlated_vars]
test_uncorr <- test[,-correlated_vars]

#Finding the optimal mtry Random Forest model using the new data without the correlated variables
forest_uncorr <- tuneRF(x = training_uncorr[,-172], y = training_uncorr$growth_2_6, plot = TRUE, doBest
```

```
## mtry = 57  OOB error = 2.497131
## Searching left ...
## mtry = 29    OOB error = 2.605604
## -0.04343894 0.05
## Searching right ...
## mtry = 114   OOB error = 2.372518
## 0.04990237 0.05
```



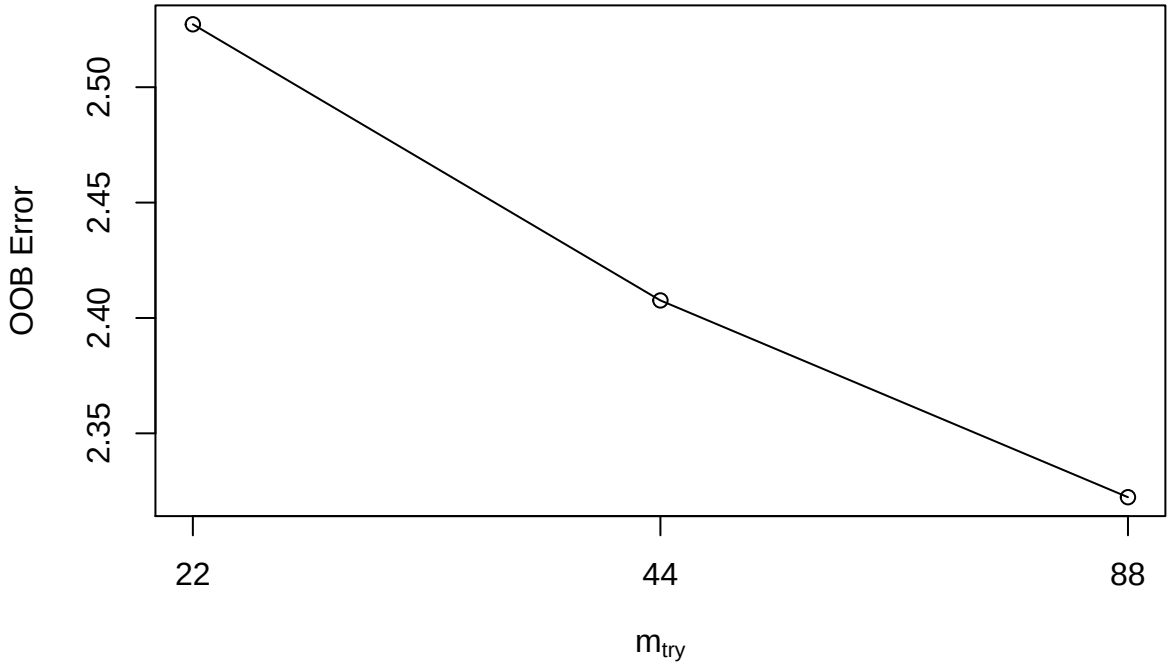

```
#Plotting to see what variables are unimportant
varImpPlot(forest_uncorr, sort = FALSE, n.var = 40)
```



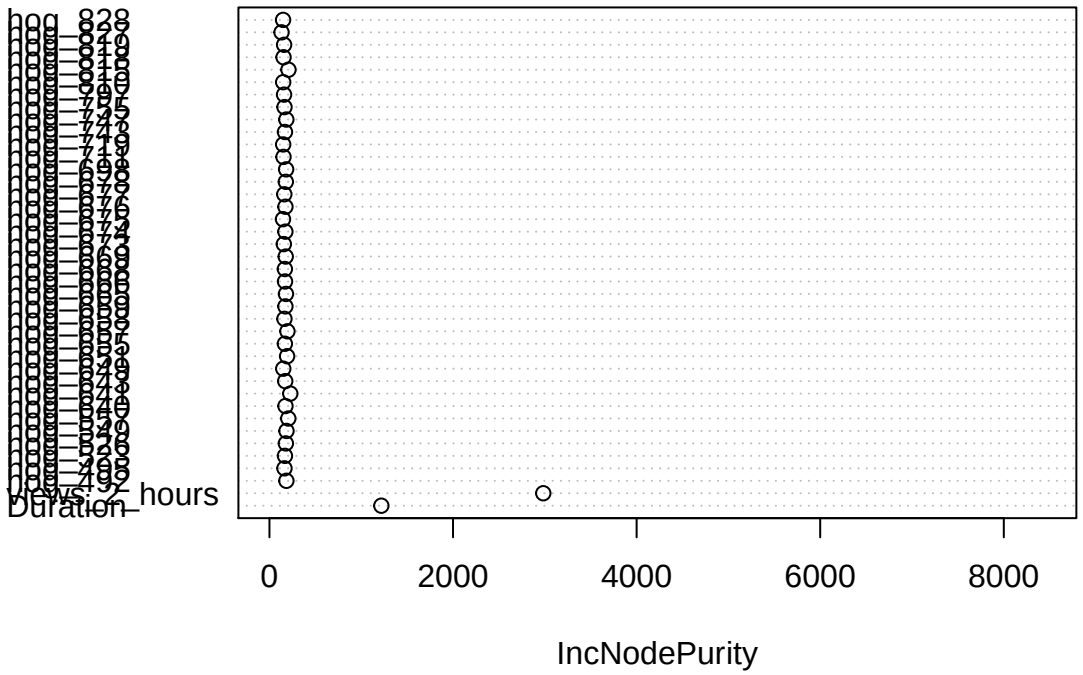

```
#Removing all hog variables up until hog_485
training_uncorr <- training_uncorr[,-(3:40)]
test_uncorr <- test_uncorr[,-(3:40)]

#Fitting a new RF to this new dataset
```

```
without_hog485 <- tuneRF(x = training_uncorr[,-134], y = training_uncorr$growth_2_6, plot = TRUE, doBes
```

```
## mtry = 44  OOB error = 2.407601
## Searching left ...
## mtry = 22    OOB error = 2.527262
## -0.04970106 0.05
## Searching right ...
## mtry = 88    OOB error = 2.322329
## 0.03541797 0.05
```




```
varImpPlot(without_hog485, sort = FALSE, n.var = 40)
```




```
#Remove the rest of the hog variables
training_uncorr <- training_uncorr[,-(3:53)]
test_uncorr <- test_uncorr[,-(3:53)]

ncol(training_uncorr)
```

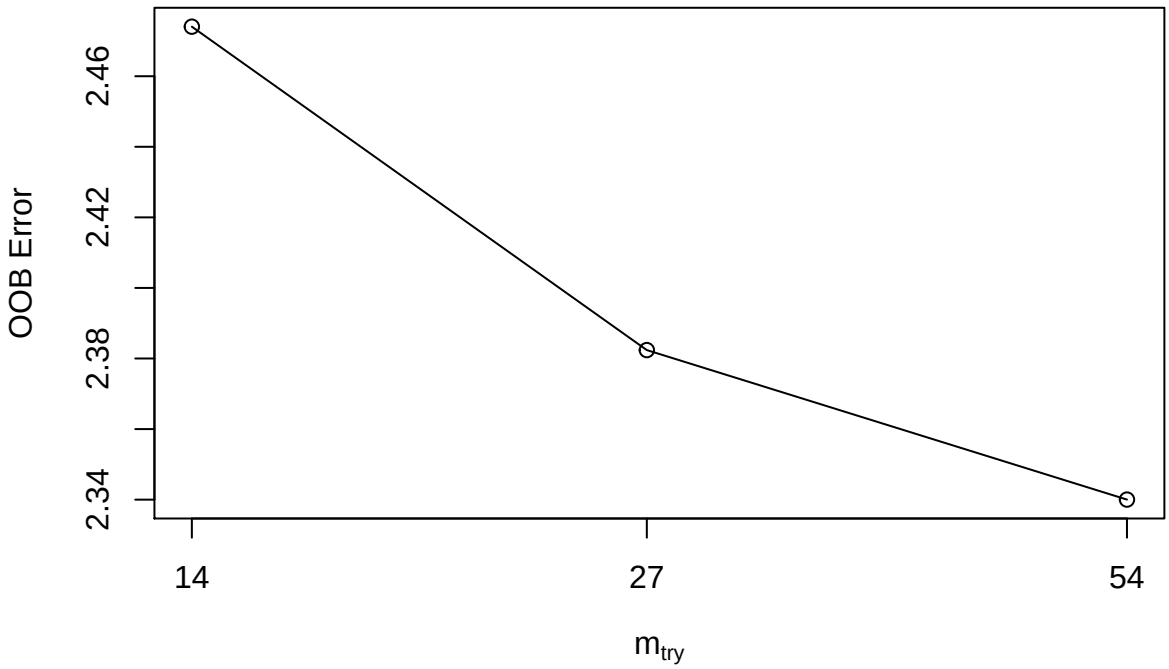
```
## [1] 83
```

```
ncol(test_uncorr)
```

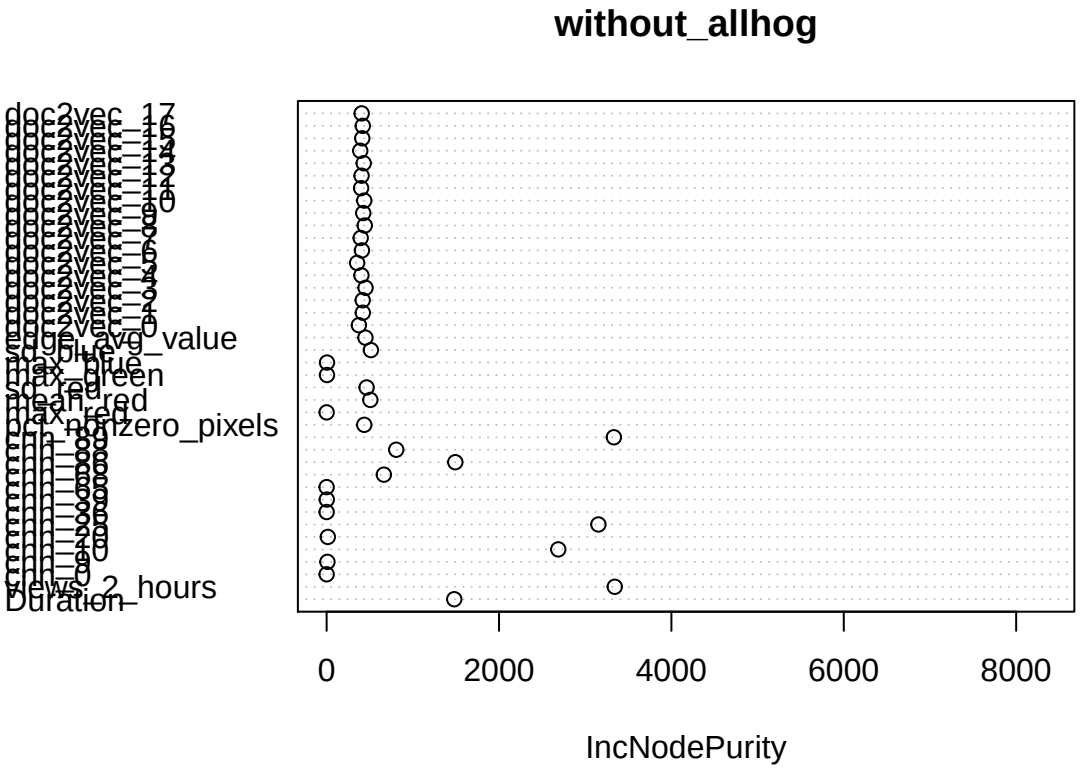
```
## [1] 82
```

```
without_allhog <- tuneRF(x = training_uncorr[,-83], y = training_uncorr$growth_2_6, plot = TRUE, doBest
```

```
## mtry = 27  OOB error = 2.382386
## Searching left ...
## mtry = 14    OOB error = 2.474041
## -0.0384717 0.05
## Searching right ...
## mtry = 54    OOB error = 2.340016
## 0.01778454 0.05
```




```
#Remove more variables deemed unimportant from varImpPlot
varImpPlot(without_allhog, sort = FALSE, n.var = 40)
```




```
#The cnn and max variables are found to be unimportant
unnecessary <- c(3, 4, 6, 8, 9, 10, 16, 19, 20) #removing max_Blue, green, red, and cnn_0,9,20,36,39,65

training_uncorr <- training_uncorr[,-unnecessary]
test_uncorr <- test_uncorr[,-unnecessary]
```

```r
ncol(training_uncorr)
```

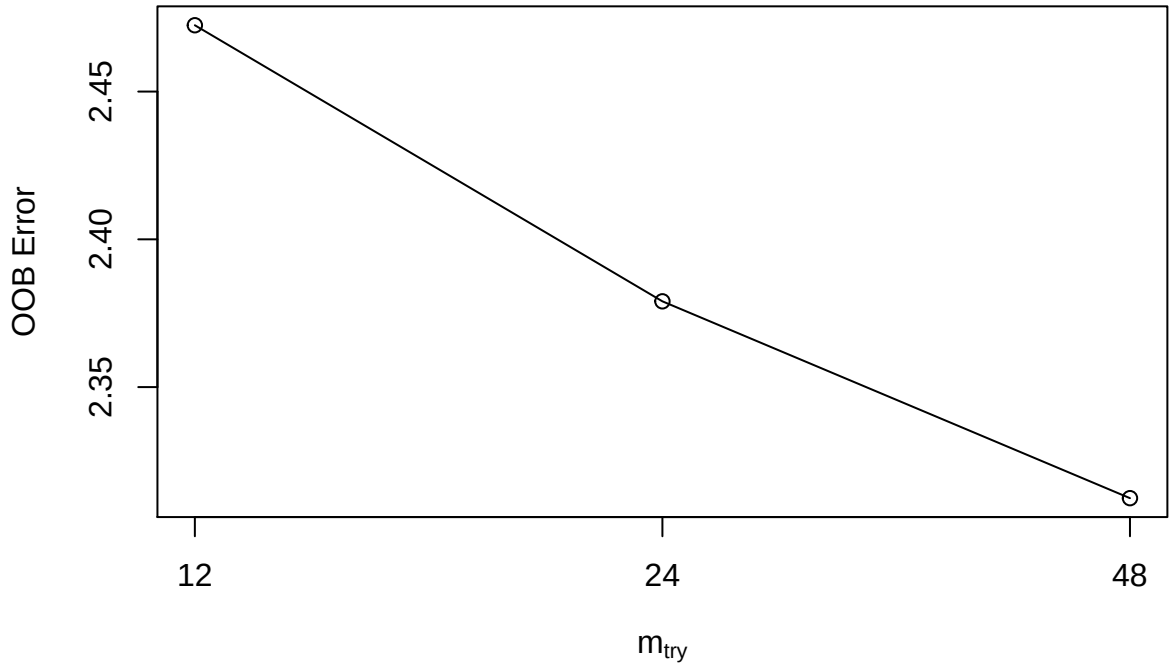
```
## [1] 74
```

```r
ncol(test_uncorr)
```

```
## [1] 73
```

```r
library(randomForest)
without_cnnmax <- tuneRF(x = training_uncorr[,-74], y = training_uncorr$growth_2_6, plot = TRUE, doBest
```

```
## mtry = 24  OOB error = 2.379013
## Searching left ...
## mtry = 12    OOB error = 2.472436
## -0.03926991 0.05
## Searching right ...
## mtry = 48    OOB error = 2.312416
## 0.02799349 0.05
```




```r
final <- cbind(testid, predict(without_cnnmax, newdata = test_uncorr))
colnames(final) <- c("id", "growth_2_6")
write.csv(final, "rmcnn_max_final.csv", row.names = FALSE)
```