Modern Data Mining - HW 4

Graham Branscom Mahika Calyanakoti Andrew Raine

Due: 11:59Pm, April 7th, 2024

Contents

Overview **1**

Objectives . . . 2

Problem 1: IQ and successes **2**

Background: Measurement of Intelligence . . . 2

1. EDA: Some cleaning work is needed to organize the data. . . . 4

2. Factors affect Income . . . 5

3. Trees . . . 6

Problem 2: Yelp challenge 2019 **13**

Goal of the study . . . 14

1. JSON data and preprocessing data . . . 14

Analysis . . . 18

2. LASSO . . . 19

3. Random Forest . . . 22

4. Boosting . . . 25

5. Ensemble model . . . 25

6. Final model . . . 25

Overview

In this homework, we will explore the transition from linear models to more flexible, tree-based methods and ensemble techniques in predictive modeling. Unlike linear models, a model-free approach, such as binary decision trees, offers a more intuitive understanding by illustrating direct relationships between predictors and responses. Although simple, binary decision trees are highly interpretable and can unveil valuable insights.

However, to harness greater predictive power, we can extend beyond a single decision tree. By aggregating multiple models, particularly those that are uncorrelated, we significantly enhance our predictive accuracy. A prime example of this concept is the RandomForest algorithm. Here, we create a multitude of decision

trees through bootstrap sampling – a method where each tree is built from a random subset of data and variables. The aggregation of these diverse trees results in a robust final prediction model.

Ensemble methods extend this idea further by combining various models to improve predictive performance. This could involve averaging or taking a weighted average of numerous distinct models. Often, this approach surpasses the predictive capability of any individual model at hand, providing a powerful tool for tackling complex data mining challenges.

Boosting, particularly Gradient Boosting Machines, stands out as another potent predictive method. Unlike traditional ensemble techniques that build models independently, boosting focuses on sequentially improving the prediction by specifically targeting the errors of previous models. Each new model incrementally reduces the errors, leading to a highly accurate combined prediction.

All the methods mentioned above can handle diverse types of data and predict outcomes ranging from continuous to categorical responses, including multi-level categories.

In Homework 4, we will delve into these advanced techniques, moving beyond the limitations of linear models and exploring the expansive potential of trees, ensembles, and boosting in modern data mining. This journey will provide you with a solid foundation in leveraging sophisticated algorithms to uncover deeper insights and achieve superior predictive performance.

Objectives

- Understand trees
  - single tree/displaying/pruning a tree
  - RandomForest
  - Ensemble idea
  - Boosting
- R functions/Packages
  - `tree`, `RandomForest`, `ranger`
  - Boosting functions
- Json data format
- text mining
  - bag of words

Data needed:

- `IQ.Full.csv`
- `yelp_review_20k.json`

Problem 1: IQ and successes

Background: Measurement of Intelligence

Case Study: how intelligence relates to one's future successes?

Data needed: `IQ.Full.csv`

ASVAB (Armed Services Vocational Aptitude Battery) tests have been used as a screening test for those who want to join the army or other jobs.

Our data set IQ.csv is a subset of individuals from the 1979 National Longitudinal Study of Youth (NLSY79) survey who were re-interviewed in 2006. Information about family, personal demographic such as gender, race and education level, plus a set of ASVAB (Armed Services Vocational Aptitude Battery) test scores are available. It is STILL used as a screening test for those who want to join the army! ASVAB scores were 1981 and income was 2005.

Our goals:

- Is IQ related to one's successes measured by Income?
- Is there evidence to show that Females are under-paid?
- What are the best possible prediction models to predict future income?

The ASVAB has the following components:

- Science, Arith (Arithmetic reasoning), Word (Word knowledge), Parag (Paragraph comprehension), Numer (Numerical operation), Coding (Coding speed), Auto (Automative and Shop information), Math (Math knowledge), Mechanic (Mechanic Comprehension) and Elec (Electronic information).
- AFQT (Armed Forces Qualifying Test) is a combination of Word, Parag, Math and Arith.
- Note: Service Branch requirement: Army 31, Navy 35, Marines 31, Air Force 36, and Coast Guard 45,(out of 100 which is the max!)

The detailed variable definitions:

Personal Demographic Variables:

- Race: 1 = Hispanic, 2 = Black, 3 = Not Hispanic or Black
- Gender: a factor with levels "female" and "male"
- Educ: years of education completed by 2006

Household Environment:

- Imagazine: a variable taking on the value 1 if anyone in the respondent's household regularly read magazines in 1979, otherwise 0
- Inewspaper: a variable taking on the value 1 if anyone in the respondent's household regularly read newspapers in 1979, otherwise 0
- Ilibrary: a variable taking on the value 1 if anyone in the respondent's household had a library card in 1979, otherwise 0
- MotherEd: mother's years of education
- FatherEd: father's years of education

Variables Related to ASVAB test Scores in 1981 (Proxy of IQ's)

- AFQT: percentile score on the AFQT intelligence test in 1981
- Coding: score on the Coding Speed test in 1981
- Auto: score on the Automotive and Shop test in 1981
- Mechanic: score on the Mechanic test in 1981
- Elec: score on the Electronics Information test in 1981
- Science: score on the General Science test in 1981
- Math: score on the Math test in 1981

- Arith: score on the Arithmetic Reasoning test in 1981
- Word: score on the Word Knowledge Test in 1981
- Parag: score on the Paragraph Comprehension test in 1981
- Numer: score on the Numerical Operations test in 1981

Variable Related to Life Success in 2006

- Income2005: total annual income from wages and salary in 2005. We will use a natural log transformation over the income.

Note: All the Esteem scores shouldn't be used as predictors to predict income

1. EDA: Some cleaning work is needed to organize the data.

- The first variable is the label for each person. Take that out.
- Set categorical variables as factors.
- Make log transformation for Income and take the original Income out
- Take the last person out of the dataset and label it as **Michelle**.
- When needed, split data to three portions: training, testing and validation (70%/20%/10%)
  - training data: get a fit
  - testing data: find the best tuning parameters/best models
  - validation data: only used in your final model to report the accuracy.

We remove the first col (subject id) and set some columns as factors. These are the remaining features:

```
## 'data.frame':    2584 obs. of  31 variables:
##  $ Imagazine     : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 2 2 2 ...
##  $ Inewspaper    : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Ilibrary      : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ MotherEd      : int  5 12 12 9 12 12 12 12 12 12 ...
##  $ FatherEd      : int  8 12 12 6 10 16 12 15 16 18 ...
##  $ FamilyIncome78: int  20000 35000 8502 7227 17000 20000 48000 15000 4510 50000 ...
##  $ Race          : Factor w/ 3 levels "1","2","3": 3 3 3 3 3 3 3 3 3 3 ...
##  $ Gender        : Factor w/ 2 levels "female","male": 1 2 2 1 2 2 1 2 2 1 ...
##  $ Educ          : int  12 16 12 14 14 16 13 13 13 17 ...
##  $ Science       : int  6 23 14 18 17 16 13 19 22 21 ...
##  $ Arith         : int  8 30 14 13 21 30 17 29 30 17 ...
##  $ Word          : int  15 35 27 35 28 29 30 33 35 28 ...
##  $ Parag         : int  6 15 8 12 10 13 12 13 14 14 ...
##  $ Numer         : int  29 45 32 24 40 36 49 35 48 39 ...
##  $ Coding        : int  52 68 35 48 46 30 58 58 61 54 ...
##  $ Auto          : int  9 21 13 11 13 21 11 18 21 18 ...
##  $ Math          : int  6 23 11 4 13 24 17 21 23 20 ...
##  $ Mechanic      : int  10 21 9 12 13 19 11 19 16 20 ...
##  $ Elec          : int  5 19 11 12 15 16 10 16 17 13 ...
##  $ AFQT          : num  6.84 99.39 47.41 44.02 59.68 ...
##  $ Esteem1       : int  1 2 2 1 1 1 2 2 2 1 ...
##  $ Esteem2       : int  1 1 1 1 1 1 2 2 2 1 ...
##  $ Esteem3       : int  4 4 3 3 4 4 3 3 3 3 ...
```

```
##  $ Esteem4       : int  1 2 2 2 1 1 2 2 2 1 ...
##  $ Esteem5       : int  3 4 3 3 1 4 3 3 3 3 ...
##  $ Esteem6       : int  3 2 2 2 1 1 2 2 2 2 ...
##  $ Esteem7       : int  1 2 2 3 1 1 3 2 2 1 ...
##  $ Esteem8       : int  3 4 2 3 4 4 3 3 3 3 ...
##  $ Esteem9       : int  3 3 3 3 4 4 3 3 3 3 ...
##  $ Esteem10      : int  3 4 3 3 4 4 3 3 3 3 ...
##  $ log_income    : num  8.61 11.08 9.85 10.49 11.08 ...
```

We get the last row as **Michelle** and then remove it.

```
##      Imagazine Inewspaper Ilibrary MotherEd FatherEd FamilyIncome78 Race Gender
## 2584         0          1        1        9        5          12410    3 female
##      Educ Science Arith Word Parag Numer Coding Auto Math Mechanic Elec   AFQT
## 2584   13      16     9   27    13    24     27   10   10        4    9 31.207
##      Esteem1 Esteem2 Esteem3 Esteem4 Esteem5 Esteem6 Esteem7 Esteem8 Esteem9
## 2584       1       2       3       2       3       2       2       3       3
##      Esteem10 log_income
## 2584        3   10.71442
```

We split the data into training (70%), testing (20%), and validation (10%) sets. Then print out the dimensions of the train, test, and validation dataframes.

```
## [1] 1808   31
```

```
## [1] 516  31
```

```
## [1] 259  31
```

2. Factors affect Income

We start with linear models to answer the questions below.

i. To summarize ASVAB test scores, create PC1 and PC2 of 10 scores of ASVAB tests and label them as ASVAB_PC1 and ASVAB_PC2. Give a quick interpretation of each ASVAB_PC1 and ASVAB_PC2 in terms of the original 10 tests.

PC1 is a weighted sum of all of the 10 scores. All of PC1's coefficients are around 0.30, with some variation. PC2 is proportional to the sum of all scores except science, auto, mechanic, and elec, which are weighted negatively. See the coefficients below:

```
##                PC1         PC2
## Science  0.3518528 -0.14181863
## Arith    0.3543093  0.04873546
## Word     0.3496059  0.06147760
## Parag    0.3262639  0.18966524
## Numer    0.2748967  0.45177204
## Coding   0.2338653  0.51465945
## Auto     0.2722293 -0.47336402
## Math     0.3359736  0.14256444
## Mechanic 0.3160660 -0.33469628
## Elec     0.3237995 -0.33521321
```

ii. Is there any evidence showing ASVAB test scores in terms of ASVAB_PC1 and ASVAB_PC2, might affect the Income? Show your work here. You may control a few other variables, including gender.

We added the PC1 and PC2 scores for each row in `iq_df`.

We run a linear model of log_income (response variable) vs ASVAB_PC1 and ASVAB_PC2. We control for gender, education, and race. ASVAB_PC1 and ASVAB_PC2 both have small p-values <0.001, and both have positive coefficients of about 0.06. This implies that higher PC1 and PC2 scores on average are correlated with a higher log(income).

```
##
## Call:
## lm(formula = log_income ~ ASVAB_PC1 + ASVAB_PC2 + Gender + Educ +
##     Race, data = iq_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.0619 -0.3240  0.1402  0.5042  2.5417
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.100289   0.138580  65.668  < 2e-16 ***
## ASVAB_PC1    0.069519   0.009224   7.537 6.62e-14 ***
## ASVAB_PC2    0.062726   0.018366   3.415 0.000647 ***
## Gendermale   0.670306   0.043652  15.356  < 2e-16 ***
## Educ         0.073495   0.008398   8.751  < 2e-16 ***
## Race2       -0.031677   0.088096  -0.360 0.719198
## Race3       -0.023264   0.072104  -0.323 0.746987
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8631 on 2576 degrees of freedom
## Multiple R-squared:  0.2167, Adjusted R-squared:  0.2148
## F-statistic: 118.7 on 6 and 2576 DF,  p-value: < 2.2e-16
```

iii. Is there any evidence to show that there is gender bias against either male or female in terms of income in the above model?

Yes, because Genderfemale is the base value, and Gendermale's beta-value is 0.67 higher than the base value. This suggests that males have higher log(income) on average than females. See the summary of the lm above.

We next build a few models for the purpose of prediction using all the information available. From now on you may use the three data sets setting (training/testing/validation) when it is appropriate.

3. Trees

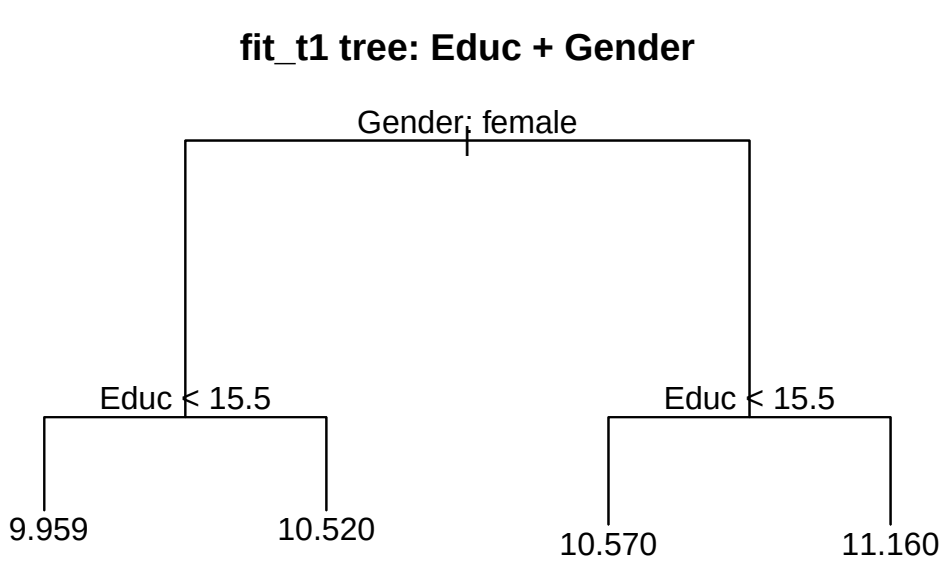
i. fit1: tree(Income ~ Educ + Gender, data.train) with default set up

a) Display the tree
b) How many end nodes? Briefly explain how the estimation is obtained in each end nodes and deescribe the prediction equation
c) Does it show interaction effect of Gender and Educ on Income?
d) Predict Michelle's income

We find the tree for Educ and Gender's effect on income. There are 4 end nodes. If female, go to the left subtree. Otherwise, go to right subtree. Then in the left sub-tree, go left if Educ < 15.5; otherwise go right. The same is true for the right-subtree. This brings you to both end nodes. The prediction values in the end nodes are the average log_income values for the datapoints in data.train that fall into that partition. The prediction equation: If female and educ < 15.5, then log_income 9.959. If female and educ > 15.5, then log_income 10.520. If male and educ < 15.5, then log_income 10.570. If male and educ > 15.5, then log_income 11.160.

There is an interactive effect of gender and education on log_income. The decision tree analysis suggests that for females, higher education (Educ > 15.5) is associated with higher income, while for males, higher education also leads to higher income, but the effect is not as pronounced as for females.

Our tree model predicts that **Michelle** has log_income 9.959322 because she is a female with educ $= 13 < 15.5$.

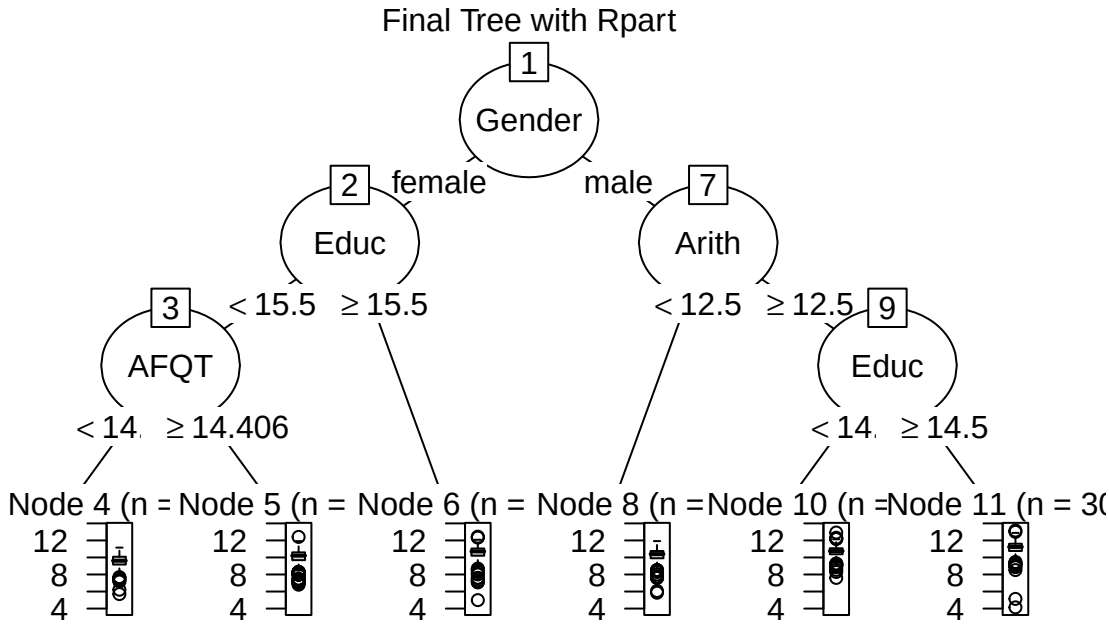



```
##      2584
## 9.959322
```

ii. fit2: fit2 <- rpart(Income2005 ~., data.train, minsplit=20, cp=.009)

a) Display the tree using plot(as.party(fit2), main="Final Tree with Rpart")
b) A brief summary of the fit2
c) Compare testing errors between fit1 and fit2. Is the training error from fit2 always less than that from fit1? Is the testing error from fit2 always smaller than that from fit1?
d) You may prune the fit2 to get a tree with small testing error.

fit_t2 first splits by Gender, then each subtree to the left and right is split differently. The left splits by Education and then AFQT. The right splits by Arith then Educ. There are 6 leaf nodes, each of which has the average log_income for that respective partition of the data.




```
## n= 1808
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 1808 1649.84100 10.439400
##    2) Gender=female 903  822.07480 10.126200
##      4) Educ< 15.5 635  515.80820  9.959322
##        8) AFQT< 14.4055 89   83.65769  9.445045 *
##        9) AFQT>=14.4055 546  404.77470 10.043150 *
##      5) Educ>=15.5 268  246.68180 10.521610 *
##    3) Gender=male 905  650.80200 10.751910
##      6) Arith< 12.5 200  155.98050 10.215150 *
##      7) Arith>=12.5 705  420.85310 10.904180
##       14) Educ< 14.5 400  154.46400 10.710650 *
##       15) Educ>=14.5 305  231.75960 11.157990 *
```

The training and testing errors for fit1 and fit2 are printed below. Fit2 has better training error and testing error than fit1, which makes sense since fit2 has more variables, so its tree can be more precise.

```
## [1] "train error t1: 1344.62012263094"
```

```
## [1] "train error t2: 1277.31834193723"
```

```
## [1] "test error t1: 438.983281447547"
```

```
## [1] "test error t2: 423.709128678244"
```

We predict **Michelle** using fit2 below.

```
##     2584
## 10.04315
```

With a lower cp score assigned while pruning, we cannot reach a training and testing error below what was achieved with the initial fit_t2 tree.

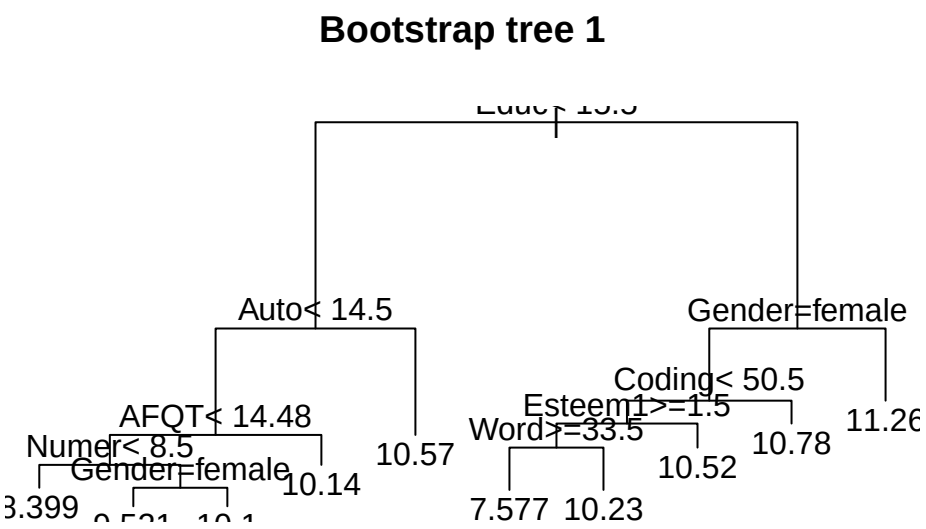
```
## [1] "train error t2 pruned: 1277.31834193723"
```

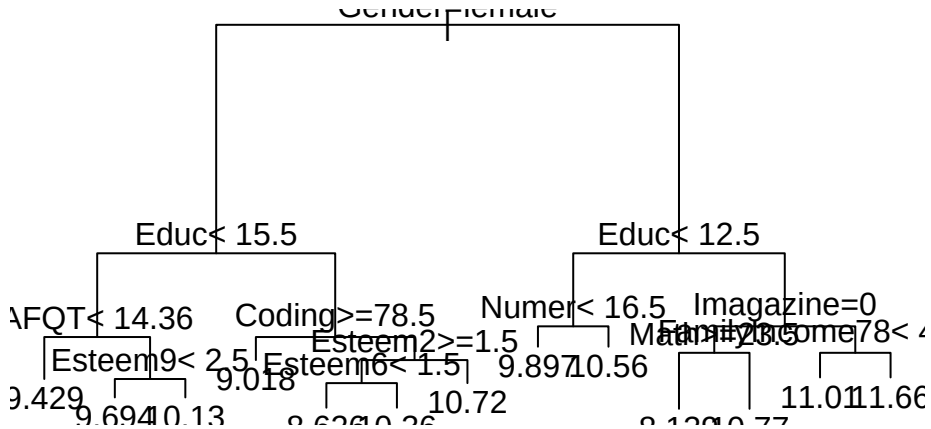
```
## [1] "test error t2 pruned: 423.709128678244"
```

iii. fit3: bag two trees

```
a) Take 2 bootstrap training samples and build two trees using the
rpart(Income2005 ~., data.train.b, minsplit=20, cp=.009). Display both trees.
b) Explain how to get fitted values for Michelle by bagging the two trees obtained above. Do not use th
c) What is the testing error for the bagged tree. Is it guaranteed that the testing error by bagging th
```

We take 2 bootstrap training samples and build two trees. The trees are clearly different.





To find the average, we followed each tree boot1 and boot2 separately for **Michelle**'s variables and then found the value at the leaf node for each tree. We then averaged those two values: (10.1449 + 10.13356)/2 = 10.13923. Lastly, we verified our answer by using predict to bag the two trees, and we obtained the value below:

```
##        fitted     obsy
## 2584 10.13923 10.71442
```

We find the testing error for the bagged tree. The testing errors of the boot1, boot2, and bagged trees are shown below. Our bagged error was less than that of the individual times, which was the case for several times we ran it, but not always.

```
## [1] "test error boot1: 485.086621938308"
```

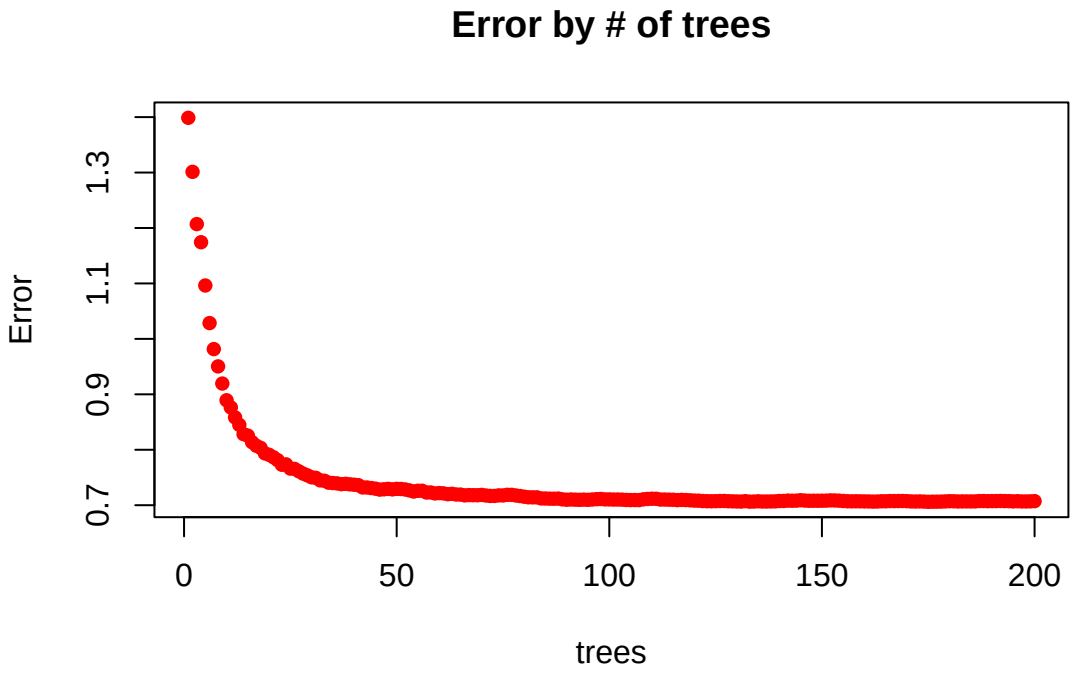
```
## [1] "test error boot2: 476.655336747513"
```

```
## [1] "test error bagged boot1 and boot2: 456.140290845032"
```

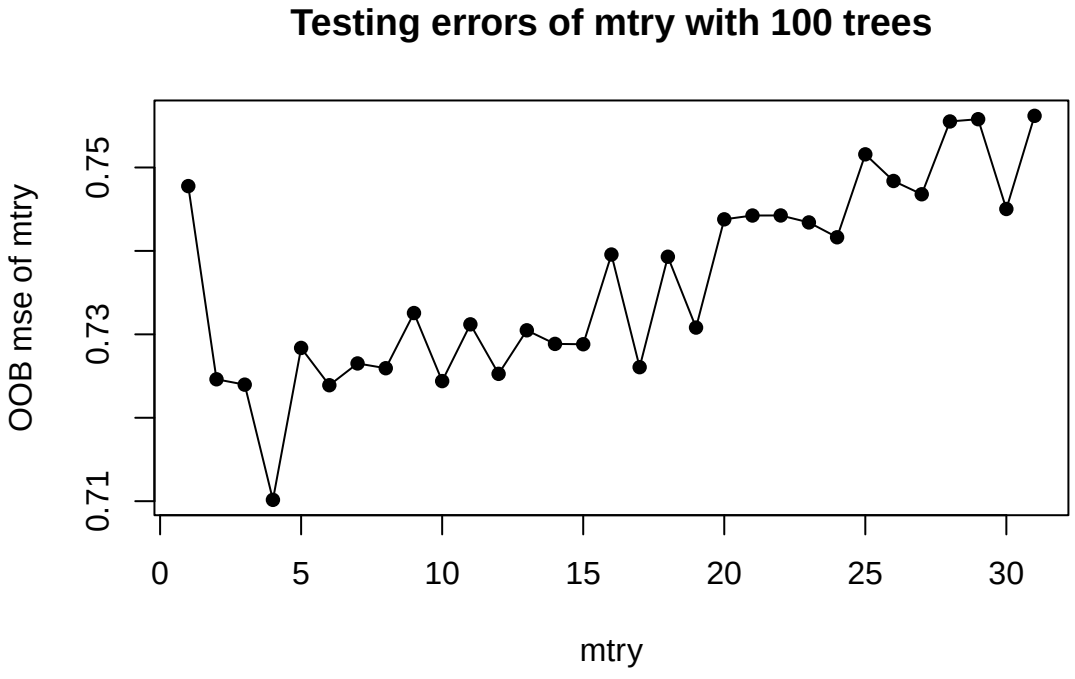
iv. fit4: Build a best possible RandomForest

a) Show the process how you tune mtry and number of trees. Give a very high level explanation how fit4 is built.

Based on the plot below we see that the error plateaus after 100 trees.



Based on the plot of the testing errors of mtry with 100 trees, optimal mtry is 4.



Now we run our fine-tuned random forest with the optimal parameters found above.

fit4 is created by taking 100 random trees and choosing only 4 parameters when splitting at each node. Then, using bagging, the output values of each tree are averaged.

```
##
## Call:
##  randomForest(formula = log_income ~ ., data = data.train, mtry = 4,      ntree = 100)
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 0.7263784
##                     % Var explained: 20.4
```

b) Compare the oob errors from fit4 to the testing errors using your testing data. Are you convinced that oob errors estimate testing error reasonably well.

```
## [1] "Out-of-Bag (OOB) Error: 0.605007033576919"
```

```
## [1] "Mean Absolute Error (MAE): 0.617412424108033"
```

```
## [1] "Residual Sum of Squares (RSS): 404.705361710032"
```

The OOB error is quite similar to the testing MAE error (see values above), so we can conclude that it does in fact estimate testing error well.

c) What is the predicted value for Michelle?

```
##    2584
## 9.82308
```

The predicted value for Michelle is printed above, 9.823.

v. Now you have built so many predicted models (fit1 through fit4 in this section). What about build a fit5 which bags fit1 through fit4. Does fit5 have the smallest testing error?

```
##        fitted     obsy
## 2584 9.991196 10.71442
```

Above we see fit5's prediction for Michelle versus the actual value. fit5 does not have the smallest testing error, but it is still pretty good.

iv. Now use `XGBoost` to build the fit6 predictive equation. Evaluate its testing error. Also briefly explain how it works.

```
## [1] "Residual Sum of Squares (RSS) for fit6: 0.658902683786994"
```

XGBoost uses boosting, which builds an ensemble of decision trees sequentially. Each model after that corrects the errors made by the previous models, leading to a stronger overall model.

v. Summarize the results and nail down one best possible final model you will recommend to predict income. Explain briefly why this is the best choice. Finally for the first time evaluate the prediction error using the validating data set.

Note that the testing errors are RSS for all of the models.

fit1 is a single decision tree based on the Educ + Gender variables. Testing error for fit1: 438.98

fit2 is a single decision tree based on all of the variables. Testing error for fit2: 423.70

fit3 is a bagging of two bootstrapped trees. Testing error for fit3: 456.14

fit4 is a random forest of 100 trees with mtry=4. Testing error for fit4: 404.71

fit5 is a bagging of fit 1 through 4. Testing error for fit5: 422.59

fit6 is a XGBoost model. Testing error for fit6: 0.65

fit6 beats all the other models in terms of testing loss, so that is the best choice.

```
## [1] "RSS for fit1: 228.401465303496"
```

```
## [1] "RSS for fit2: 223.224382454451"
```

```
## [1] "RSS for fit3: 226.566641847104"
```

```
## [1] "RSS for fit4: 218.066272590957"
```

```
## [1] "RSS for fit5: 218.053019508847"
```

```
## [1] "RSS for fit6: 0.188232392838687"
```

Based on the validation set, fit6 remains the best performing model.

vi. Use your final model to predict Michelle's income.

```
## [1] 10.71299
```

The final mode, fit6, predicts Michelle's log income as 10.71299 which is extremely close to her actual income (10.71442).

Problem 2: Yelp challenge 2019

Yelp has made their data available to public and launched Yelp challenge. More information. It is unlikely we will win the $5,000 prize posted but we get to use their data for free. We have done a detailed analysis in our lecture. This exercise is designed for you to get hands on the whole process.

For this case study, we downloaded the data and took a 20k subset from **review.json**. *json* is another format for data. It is flexible and commonly-used for websites. Each item/subject/sample is contained in a brace *{}*. Data is stored as **key-value** pairs inside the brace. *Key* is the counterpart of column name in *csv* and *value* is the content/data. Both *key* and *value* are quoted. Each pair is separated by a comma.

Data needed: yelp_review_20k.json available in Canvas.

yelp_review_20k.json contains full review text data including the user_id that wrote the review and the business_id the review is written for.

Goal of the study

The goals are

1) Try to identify important words associated with positive ratings and negative ratings. Collectively we have a sentiment analysis.
2) To predict ratings using different methods.

1. JSON data and preprocessing data

i. Load *json* data

The *json* data provided is formatted as newline delimited JSON (ndjson). It is relatively new and useful for streaming.

We use `stream_in()` in the `jsonlite` package to load the JSON data (of ndjson format) as `data.frame`. (For the traditional JSON file, use `fromJSON()` function.)

```
## 'data.frame':    19999 obs. of  9 variables:
##  $ review_id  : chr  "Q1sbwvVQXV2734tPgoKj4Q" "GJXCdrto3ASJOqKeVWPi6Q" "2TzJjDVDEuAW6MR5Vuc1ug" "yi0
##  $ user_id    : chr  "hG7b0MtEbXx5QzbzE6C_VA" "yXQM5uF2jS6es16SJzNHfg" "n6-Gk65cPZL6Uz8qRm3NYw" "dac
##  $ business_id: chr  "ujmEBvifdJM6h6RLv4wQIg" "NZnhc2sEQy3RmzKTZnqtwQ" "WTqjgwHlXbSFevF32_DJVw" "ikC
##  $ stars      : num  1 5 5 5 1 4 3 1 2 3 ...
##  $ useful     : int  6 0 3 0 7 0 5 3 1 1 ...
##  $ funny      : int  1 0 0 0 0 0 4 1 0 0 ...
##  $ cool       : int  0 0 0 0 0 0 5 1 0 1 ...
##  $ text       : chr  "Total bill for this horrible service? Over $8Gs. These crooks actually had the
##  $ date       : chr  "2013-05-07 04:34:36" "2017-01-14 21:30:33" "2016-11-09 20:09:03" "2018-01-09 2
```

Write a brief summary about the data:

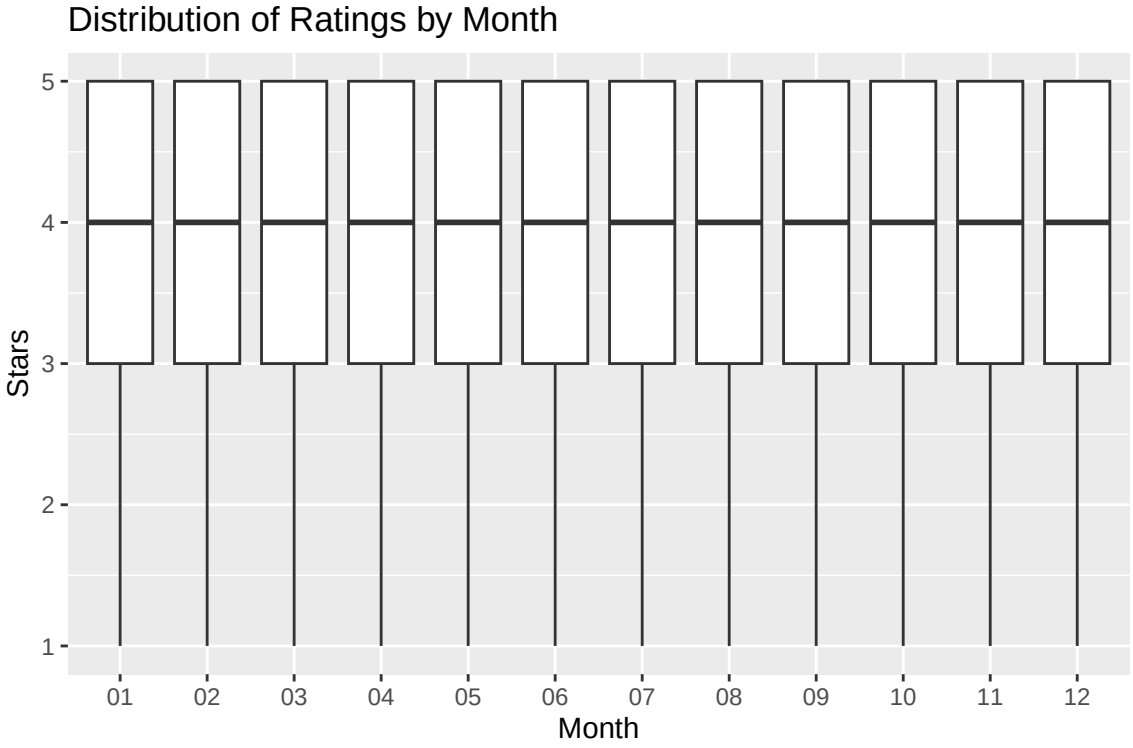
```
##   review_id           user_id          business_id            stars      
##  Length:19999       Length:19999       Length:19999       Min.   :1.000  
##  Class :character   Class :character   Class :character   1st Qu.:3.000  
##  Mode  :character   Mode  :character   Mode  :character   Median :4.000  
##                                                           Mean   :3.742  
##                                                           3rd Qu.:5.000  
##                                                           Max.   :5.000  
##      useful            funny              cool             text          
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000   Length:19999      
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.0000   Class :character  
##  Median : 0.000   Median : 0.0000   Median : 0.0000   Mode  :character  
##  Mean   : 1.252   Mean   : 0.4384   Mean   : 0.5313                     
##  3rd Qu.: 1.000   3rd Qu.: 0.0000   3rd Qu.: 0.0000                     
##  Max.   :91.000   Max.   :42.0000   Max.   :86.0000                     
##       date                    
##  Min.   :2004-10-19 03:05:42.00  
##  1st Qu.:2014-05-07 19:43:19.50  
##  Median :2016-02-12 02:40:46.00  
##  Mean   :2015-08-19 03:31:26.96  
##  3rd Qu.:2017-06-28 00:11:17.50  
##  Max.   :2018-10-04 18:20:45.00
```

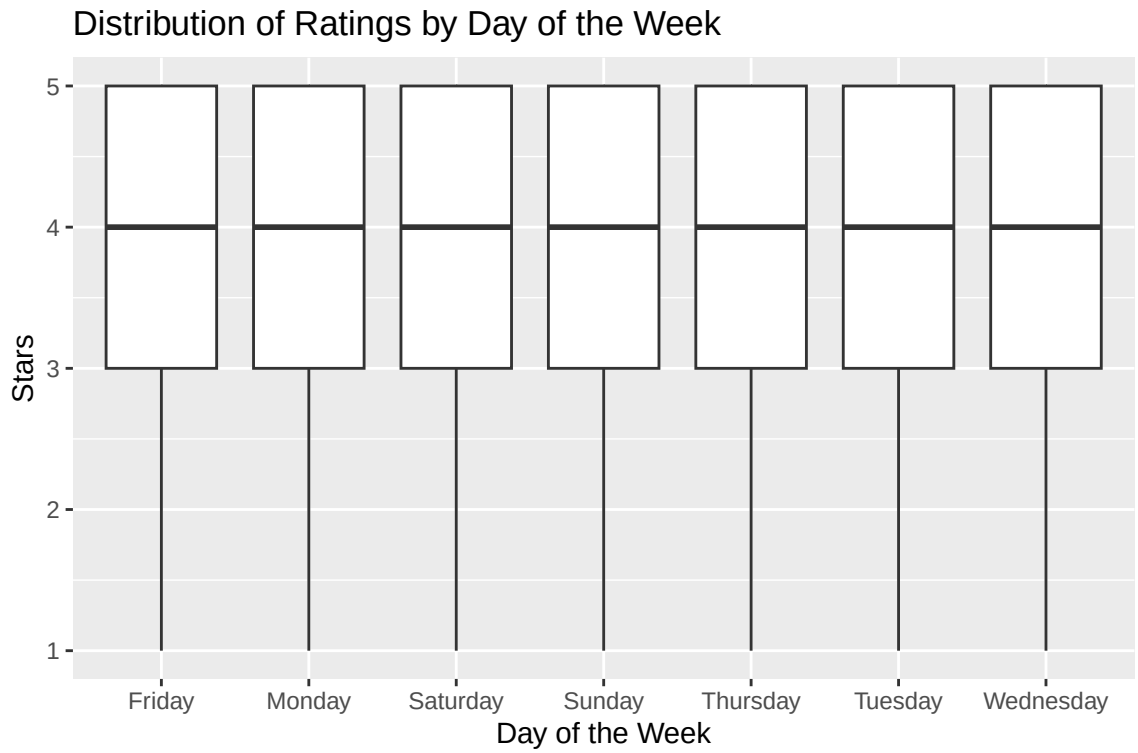
a) Which time period were the reviews collected in this data?

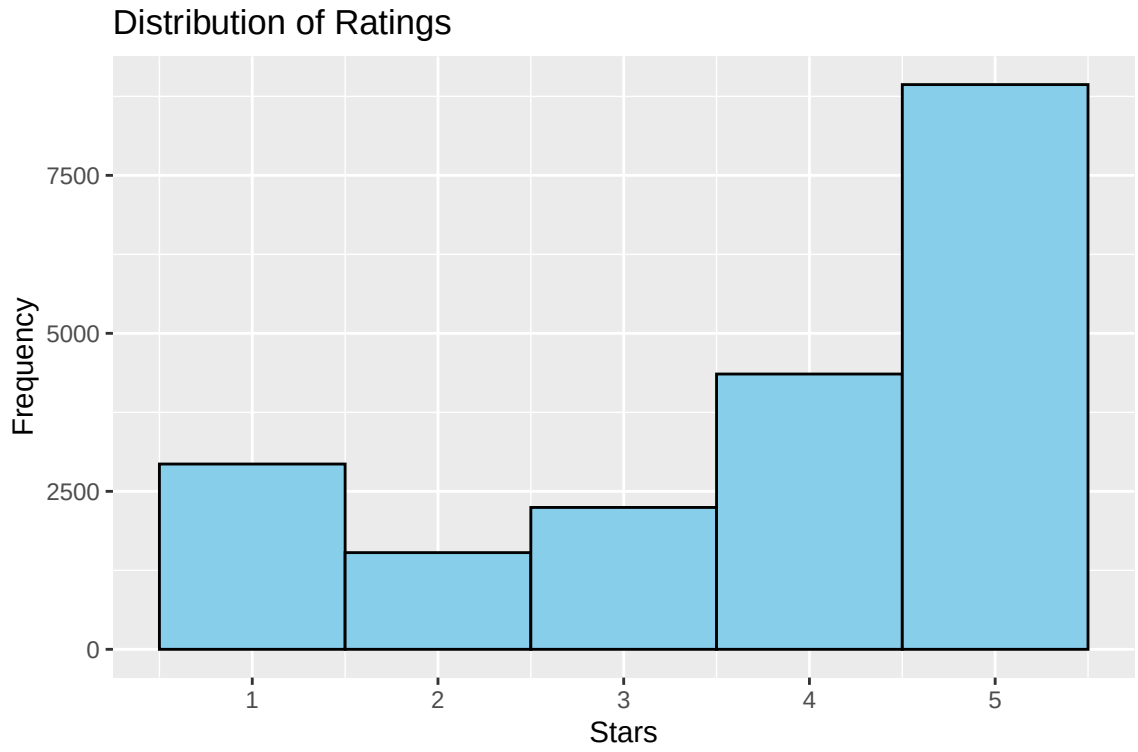
The oldest review was 2004-10-19 03:05:42.00, and the newest review was 2018-10-04 18:20:45.00. So between 2004 and 2018.

b) Are ratings (with 5 levels) related to month of the year or days of the week? Only address this through EDA please.

Based on the below plots of the distribution of ratings by day and month, it seems that the ratings have the same distribution regardless of the day of the week or the month, so they are NOT related. Because the boxplot is calculated using medians (and quartiles), they output whole numbers. And since all of the months/days have the same distribution curve (if you looked at a histogram), they all have the same medians/quartiles, and thus they all have the same boxplots (though the means would show slight variations).







We also made a histogram of the ratings across all of the yelp reviews, and when filtering by month or day, we saw the same distribution across all months and days, roughly.

ii. Document term matrix (dtm)

Extract document term matrix for texts to keep words appearing at least .5% of the time among all 20000 documents. Go through the similar process of cleansing as we did in the lecture.

First, we extract the text column:

```
## [1] 19999
```

```
## [1] "character"
```

Then, we make the corpus, clean the text (punctuation, lowercase, stemming, etc.), and set a spare threshold for words appearing 0.5% of the time, and we inspect the document term matrix:

```
## Installing package into 'C:/Users/Andrew/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
## package 'tm' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Andrew\AppData\Local\Temp\RtmpUnsg17\downloaded_packages
```

```
## Installing package into 'C:/Users/Andrew/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
## package 'SnowballC' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Andrew\AppData\Local\Temp\RtmpUnsg17\downloaded_packages
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

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

a) Briefly explain what does this matrix record? What is the cell number at row 100 and column 405? What does it represent?

A document term matrix records at position i, j, where rows are the documents and columns are the terms/words, the frequency of the word j in document i. In this case, at row 100 column 405, we see a value of 0, which means that in the 100th document/yelp review the 405th word appeared that many times. We printed the 405th word, which appears to be "except". After checking the actual review, we verified that "except" wasn't found in the text.

```
## [1] 19999  1337
```

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

```
## [1] "except"
```

```
## [1] "Hands down best Bloody Mary ever. So many things in my BM, veggies  mmm bacon.  \"Nailed it\" i
```

b) What is the sparsity of the dtm obtained here? What does that mean?

```
## <<DocumentTermMatrix (documents: 19999, terms: 1337)>>
## Non-/sparse entries: 713107/26025556
## Sparsity           : 97%
## Maximal term length: 13
## Weighting          : term frequency (tf)
```

As seen from the output above, the sparsity is 97%, which is the proportion of cells in the matrix that are empty/0. This is a pretty sparse matrix.

iii. Set the stars as a two category response variable called rating to be "1" = 5,4 and "0"= 1,2,3. Combine the variable rating with the dtm as a data frame called data2.

We create the rating column as a factor as seen below:

```
## 'data.frame':    19999 obs. of  12 variables:
##  $ review_id  : chr  "Q1sbwvVQXV2734tPgoKj4Q" "GJXCdrto3ASJOqKeVWPi6Q" "2TzJjDVDEuAW6MR5Vuc1ug" "yi0
##  $ user_id    : chr  "hG7b0MtEbXx5QzbzE6C_VA" "yXQM5uF2jS6es16SJzNHfg" "n6-Gk65cPZL6Uz8qRm3NYw" "dac
##  $ business_id: chr  "ujmEBvifdJM6h6RLv4wQIg" "NZnhc2sEQy3RmzKTZnqtwQ" "WTqjgwHlXbSFevF32_DJVw" "ikC
##  $ stars      : num  1 5 5 5 1 4 3 1 2 3 ...
##  $ useful     : int  6 0 3 0 7 0 5 3 1 1 ...
##  $ funny      : int  1 0 0 0 0 0 4 1 0 0 ...
##  $ cool       : int  0 0 0 0 0 0 5 1 0 1 ...
##  $ text       : chr  "Total bill for this horrible service? Over $8Gs. These crooks actually had the
##  $ date       : POSIXct, format: "2013-05-07 04:34:36" "2017-01-14 21:30:33" ...
##  $ month      : chr  "05" "01" "11" "01" ...
##  $ day_of_week: chr  "Tuesday" "Saturday" "Wednesday" "Tuesday" ...
##  $ rating     : Factor w/ 2 levels "0","1": 1 2 2 2 1 2 1 1 1 1 ...
```

We use cbind to combine the dtm.10 data frame with the newly created ratings column, and call it data2.

Analysis

Get a training data with 13000 reviews and the 5000 reserved as the testing data. Keep the rest (2000) as our validation data set.

Reserve 13000 randomly chosen rows as our test data (`data2.test`) and of the remaining 7000, reserve 5000 as the training data (`data2.train`) and the rest as validation.

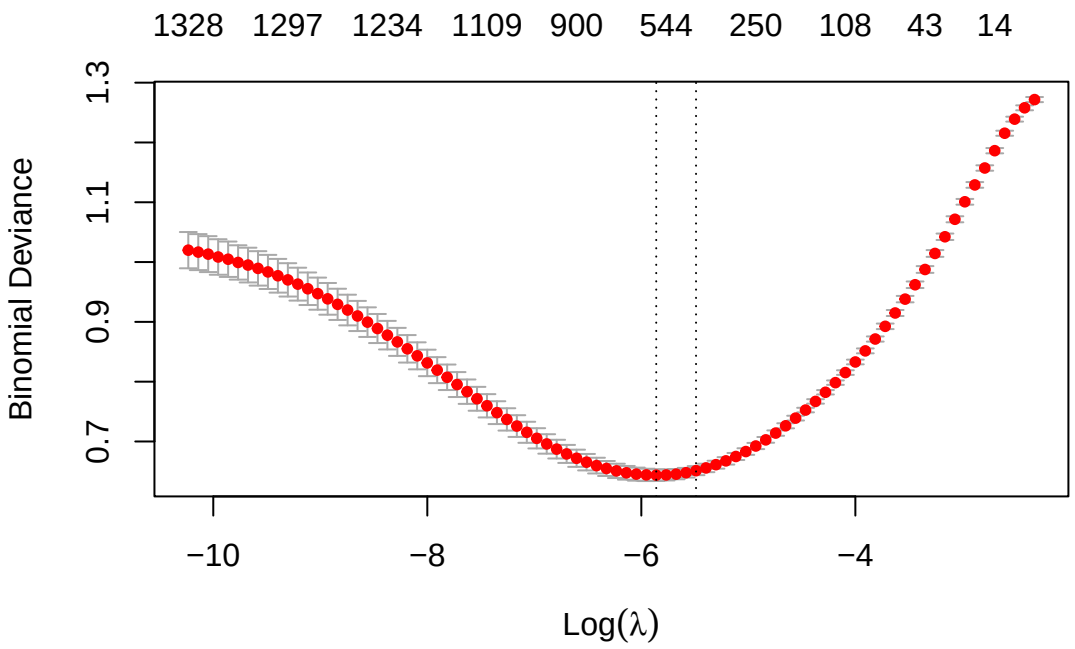
```
## [1] 13000  1338
```

```
## [1] 5000 1338
```

```
## [1] 1999 1338
```

2. LASSO

i. Use the training data to get Lasso fit. Choose lambda.1se. Keep the result here.




```
## [1] 426
```

Using lambda 1se got us 426 variables in the LASSO model.

ii. Feed the output from Lasso above, get a logistic regression.

We get a logistic regression with the code below:

```
glm.input <- as.formula(paste("rating_c", "~", paste(var.1se, collapse = "+")))
fit.1se.glm <- glm(glm.input, family = binomial(logit), data=data.train)  # debiased or relaxed LASSO
```

a) Pull out all the positive coefficients and the corresponding words. Rank the coefficients in a decreasing order. Report the leading 2 words and the coefficients. Describe briefly the interpretation for those two coefficients.

The leading two words with the highest positive coefficients are gem and bomb, with 2.36898934 and 2.36822048 respectively.The positive coefficients for the words "gem" and "bomb" suggest that reviews mentioning these terms are associated with significantly higher chances of receiving a positive rating. Specifically, each occurrence of the word "gem" and "bomb" in a review is associated with approximately 2.37 times higher chances of receiving a positive rating.

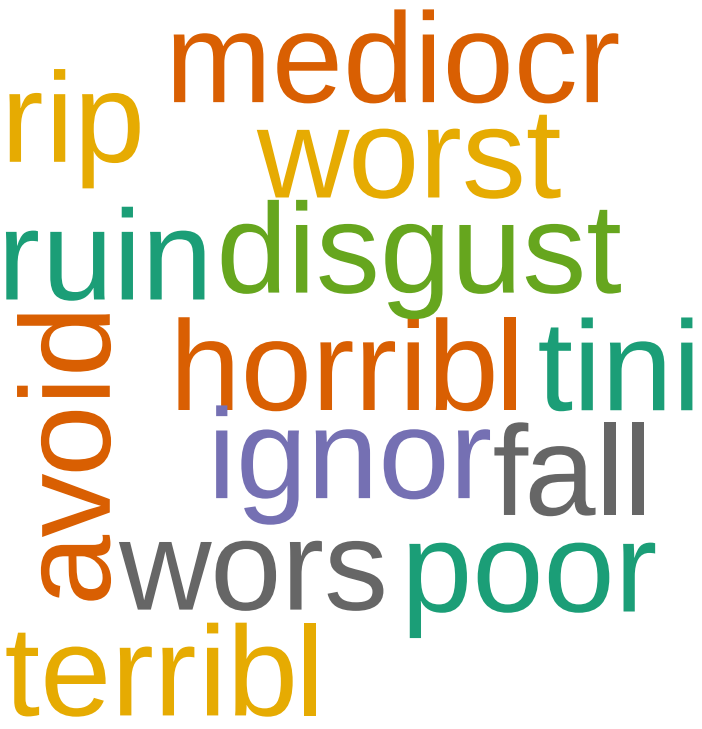
b) Make a word cloud with the top 100 positive words according to their coefficients. Interpret the cloud briefly.

The word cloud prints the top positive coefficient words, with bigger words for bigger positive coefficients. This is a way to visualize the relative coefficients among the positive coefficient words.

c) Repeat i) and ii) for the bag of negative words.

The leading two words with the most negative coefficients are unprofession and horribl, with -4.51791831 and -3.06445611 respectively. The negative coefficients for the words “unprofession” and “horribl” suggest that reviews mentioning these terms are associated with significantly lower chances of receiving a positive rating. Specifically, each occurrence of the word “unprofession” and “horribl” in a review is associated with approximately 4.52 or 3.06 times lower chances of receiving a positive rating respectively.

The word cloud prints the more negative coefficient words, with bigger words for more negative coefficients. This is a way to visualize the relative coefficients among the negative coefficient words.



d) Summarize the findings. We found that 189 words and 238 words have positive and negative coefficients respectively. Through ranking the coefficients, we found that gem and bomb have the greatest positive impact on rating, whereas unprofession and horribl have the greatest negative impact. The wordclouds above show some of the most positive and negative coefficient words.

iii. Using majority votes find the testing errors

i) From Lasso fit in 3)

The mean absolute error here is 0.1757 when using the predictions of the lasso model.

```
## [1] 0.1757431
```

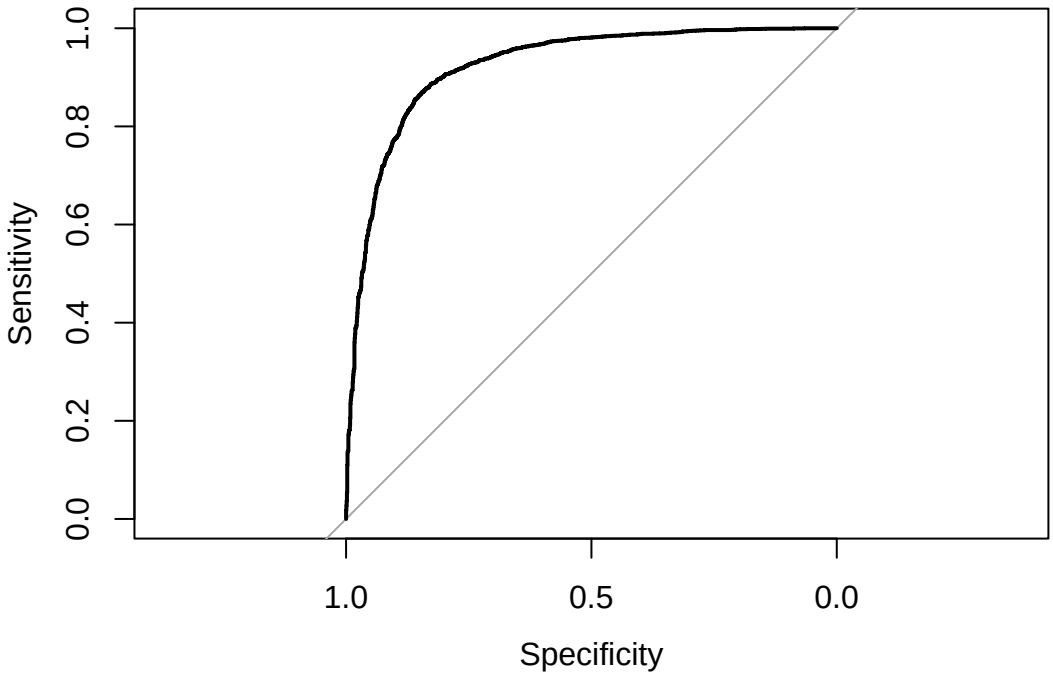
ii) From logistic regression in 4)

Use the testing data we get mis-classification errors for one rule: majority vote. We found test error (mean of misclassification errors) was 0.1332, and AUC was 0.9235, which means our model is pretty good.

```
## [1] 0.1332
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##
## Call:
## roc.default(response = data.test$rating, predictor = predict.glm,     plot = T)
##
## Data: predict.glm in 1707 controls (data.test$rating 0) < 3293 cases (data.test$rating 1).
## Area under the curve: 0.9235
```

iii) Which one is smaller?

The error is smaller for the logistic regression model, since 0.1332 is less than 0.1757.

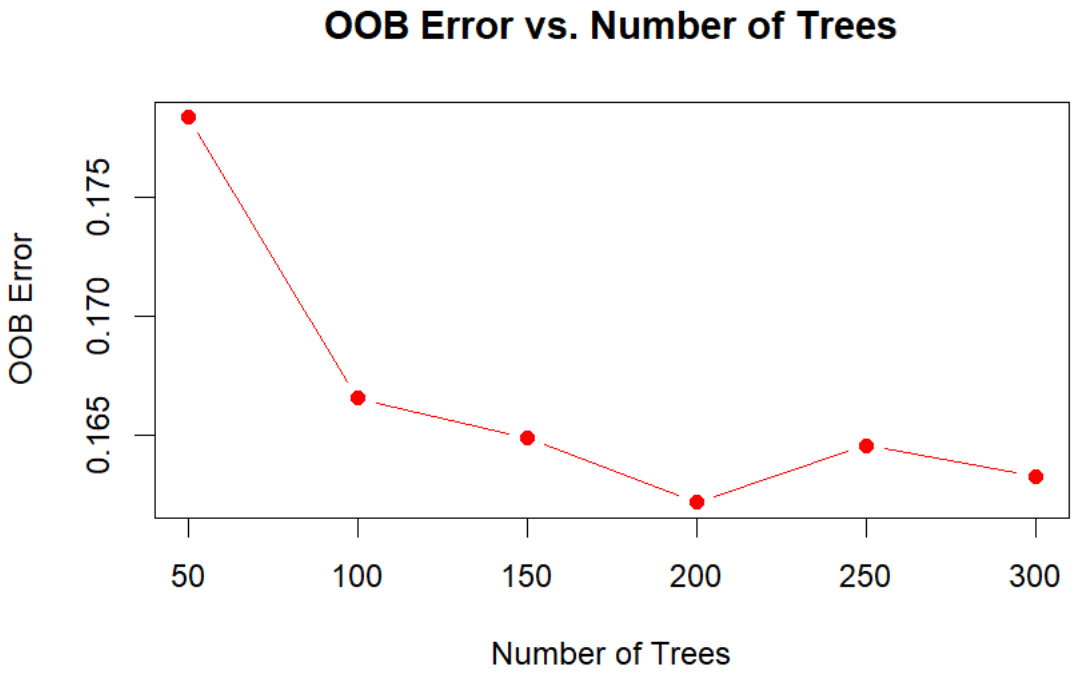
3. Random Forest

i. Briefly summarize the method of Random Forest

It is an ensemble learning method that uses multiple decision trees during training and outputs the mode/mean classification/regression of the individual trees. It introduces randomness by using different subsets of data for training each tree and by selecting a random subset of features for each split in the trees. This approach helps in reducing overfitting and improving prediction accuracy compared to a single decision tree.

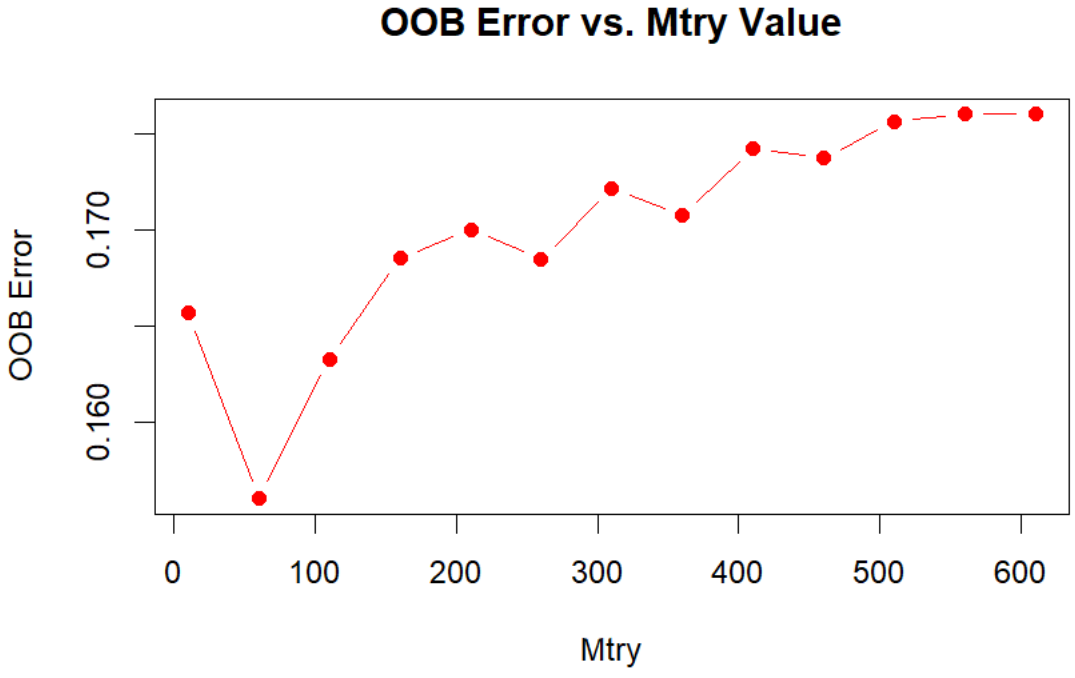
ii. Now train the data using the training data set by RF. Get the testing error of majority vote. Also explain how you tune the tuning parameters (`mtry` and `ntree`).

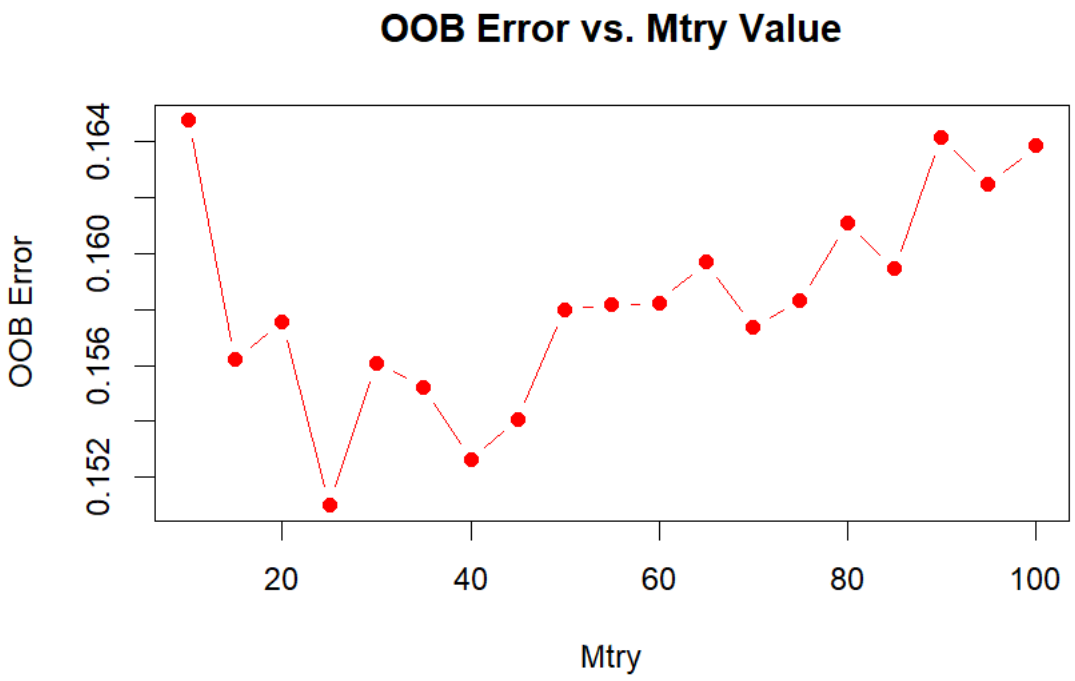
To find the optimal number of trees for our random forest, we increment the number of trees by 50 and record the OOB error. We then choose the number of trees at the local minima of the OOB vs. n_trees plot.



From the plot above (we commented out the training to reduce runtime), we can see that the optimal number of trees is 200.

Now we will do the same process to find the optimal mtry value, except setting num.trees to the optimal found above.





From the plot above the optimal mtry value is 25.

```
## Installing package into 'C:/Users/Andrew/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'doParallel' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Andrew\AppData\Local\Temp\RtmpUnsg17\downloaded_packages

## Installing package into 'C:/Users/Andrew/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'foreach' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Andrew\AppData\Local\Temp\RtmpUnsg17\downloaded_packages

## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel

## [1] "Mean Absolute Error (MAE): 0.415461538461538"

## [1] "Residual Sum of Squares (RSS): 5401"
```

4. Boosting

Now use `XGBoost` to build the fourth predictive equation. Evaluate its testing error.

```
## [1] "Mean Absolute Error (MAE): 0.264604384255409"
```

5. Ensemble model

i. Take average of some of the models built above (also try all of them) and this gives us the fifth model. Report it's testing error. (Do you have more models to be bagged, try it.)

```
## [1] "Mean Absolute Error (MAE) for Ensemble: 0.243081552438954"
```

After bagging the GLM, LASSO classifier, random forest, and XGBoost model, we found a MAE value of 0.244.

6. Final model

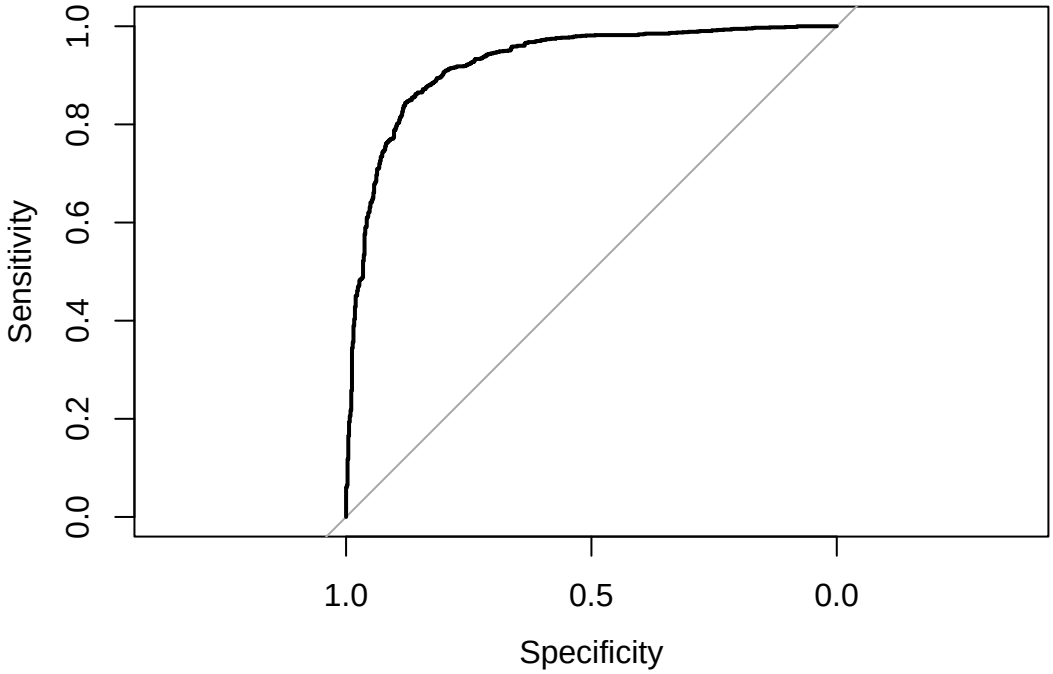
Which classifier(s) seem to produce the least testing error? Are you surprised? Report the final model and accompany the validation error. Once again this is THE only time you use the validation data set. For the purpose of prediction, comment on how would you predict a rating if you are given a review (not a tm output) using our final model?

Based on the errors found for each model, the LASSO classification model performs the best. Its MAE is 0.1332. We thought that the boosting model would do the best. Based on the validation data, we see a MAE of 0.1305653 and an AUC of 0.9253. We would look at only the words that are used by the glm, count up the frequency for those words in the review, then use those are parameters along with the beta values from our glm prediction equation, and take the output of the prediction equation and round it up/down based on a threshold of 0.5.

```
## [1] 0.1305653
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##
## Call:
## roc.default(response = data.val$rating, predictor = predict.glm,     plot = T)
##
## Data: predict.glm in 665 controls (data.val$rating 0) < 1334 cases (data.val$rating 1).
## Area under the curve: 0.9253
```