Modern Data Mining - HW 4

Yang Yi
Kexin Zhu
Yifan Jiang

Contents

Overview / Instructions

This is homework #4 of STAT 471/571/701. It will be **due on 21, April, 2019 by 11:59 PM** on Canvas. You can directly edit this file to add your answers. Submit the Rmd file, a PDF or word or HTML version with **only 1 submission** per HW team. No zip files please.

Problem 0: Methods

Review the codes and concepts covered during lecture, in particular:

- Single trees, Bagging, Random Forest and Boosting
- Text Mining
- Basic Deep Learning elements

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.

The following 10 questions are answered as 1: strongly agree, 2: agree, 3: disagree, 4: strongly disagree

- Esteem 1: "I am a person of worth"

- Esteem 2: "I have a number of good qualities"
- Esteem 3: "I am inclined to feel like a failure"
- Esteem 4: "I do things as well as others"
- Esteem 5: "I do not have much to be proud of"
- Esteem 6: "I take a positive attitude towards myself and others"
- Esteem 7: "I am satisfied with myself"
- Esteem 8: "I wish I could have more respect for myself"
- Esteem 9: "I feel useless at times"
- Esteem 10: "I think I am no good at all"

Note: we will not use the Esteem scores in this homework.

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

- The first variable is the label for each person. Take that out.

```
IQ <- read.csv("IQ.Full.csv")
IQ <- IQ[,-c(1)]
```

- Take all the `Esteem` scores out for this homework

```
IQ <- IQ[,-c(22:31)]
```

- Set categorical variables as factors

```
str(IQ)
cols <- c(1:3,7)
IQ[cols] <- lapply(IQ[cols], factor)
```

- Make log transformation for Income and take the original Income out

```
IQ$Income <- log(IQ$Income2005)
IQ$Fami78 <- log(IQ$FamilyIncome78)
IQ$Fami78[which(IQ$Fami78 == -Inf)] <- 0 # There are -Inf in Income78, replace with 0
IQ <- IQ[,-c(6,21)]
```

- Take the last person out of the dataset and label it as **Michelle**.

```
Michelle <- IQ[2584,]
IQ <- IQ[-2584,]
```

- When needed, split data to three portions: training, testing and validation (70%/20%/10%)

```
set.seed(100)
train_ind <- sample(seq_len(nrow(IQ)), size = floor(0.7*nrow(IQ)))
data.train <- IQ[train_ind, ]
data <- IQ[-train_ind, ]
test_ind <- sample(seq_len(nrow(data)), size = floor(0.6667*nrow(data)))
data.test <- data[test_ind, ]
data.vali <- data[-test_ind,]
```

2. Factors affect Income

We only use linear models to answer the questions below.

i. Is there any evidence showing ASVAB test scores might affect the Income. Show your work here.

```r
fit.05 <- lm(Income~AFQT+Coding+Auto+Mechanic+Elec+Science+Math+Arith+Word+Parag+Numer, data=IQ)
summary(fit.05)
```

```
##
## Call:
## lm(formula = Income ~ AFQT + Coding + Auto + Mechanic + Elec +
##     Science + Math + Arith + Word + Parag + Numer, data = IQ)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5348 -0.3766  0.1212  0.5360  2.7763
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.544176   0.116910  81.637  < 2e-16 ***
## AFQT         0.004949   0.002629   1.882  0.05989 .
## Coding      -0.004256   0.001627  -2.616  0.00896 **
## Auto         0.041264   0.005559   7.423 1.55e-13 ***
## Mechanic    -0.001073   0.006047  -0.178  0.85911
## Elec         0.017553   0.007812   2.247  0.02472 *
## Science     -0.013725   0.007277  -1.886  0.05939 .
## Math         0.018310   0.006831   2.680  0.00740 **
## Arith        0.005332   0.005850   0.911  0.36215
## Word        -0.012099   0.005437  -2.225  0.02614 *
## Parag       -0.016594   0.011430  -1.452  0.14668
## Numer        0.011880   0.002676   4.439 9.40e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8984 on 2571 degrees of freedom
## Multiple R-squared:  0.1529, Adjusted R-squared:  0.1493
## F-statistic:  42.2 on 11 and 2571 DF,  p-value: < 2.2e-16
```

Answer: Our linear model shows that there are some grades in ASVAS which are significantly correlated with income in 2005, including `Coding`, `Auto`, `Math`, and `Numer`. The whole model has a p-value close to 0 and a F statistic of 42.2, meaning that the model has an acceptable goodness of fit.

ii. Is there any evidence to show that there is gender bias against either male or female in terms of income. Once again show your work here.

```r
fit.g <- lm(Income~Gender, data=IQ)
summary(fit.g)
```

```
##
## Call:
## lm(formula = Income ~ Gender, data = IQ)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6034 -0.4048  0.1124  0.5432  2.7175
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.12160    0.02582  391.99   <2e-16 ***
## Gendermale   0.62496    0.03631   17.21   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9227 on 2581 degrees of freedom
## Multiple R-squared:  0.1029, Adjusted R-squared:  0.1026
## F-statistic: 296.2 on 1 and 2581 DF,  p-value: < 2.2e-16
```

Answer: There is a gender bias exists in terms of income in 2005, where males are more likely to have higher income (0.6249 of logIncome) than females.

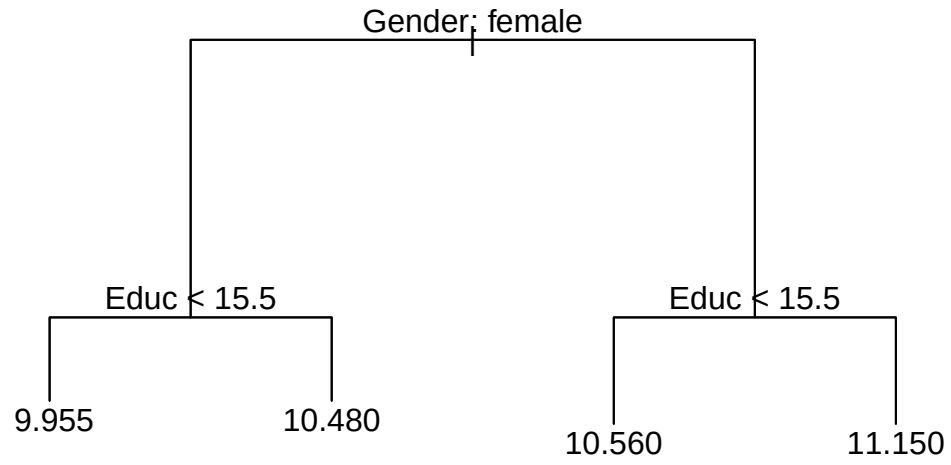
We next build a few models for the purpose of prediction using all the information available. From now on you may use the three data sets setting (training/testing/validation) when it is appropriate. **You should not use Validation set in the process of building classifer ever. This should only be used to get the testing error at the end.**

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 over Income?
d) Predict Michelle's income

```
fit1 <- tree(Income ~ Educ + Gender, data.train)
plot(fit1)
text(fit1, pretty=0)
```




```
summary(fit1)  # Residual mean deviance(MSE) = 0.822
```

```
##
## Regression tree:
## tree(formula = Income ~ Educ + Gender, data = data.train)
## Number of terminal nodes:  4
## Residual mean deviance:  0.8222 = 1483 / 1804
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -7.0040 -0.3394  0.1433  0.0000  0.5238  2.3170
```

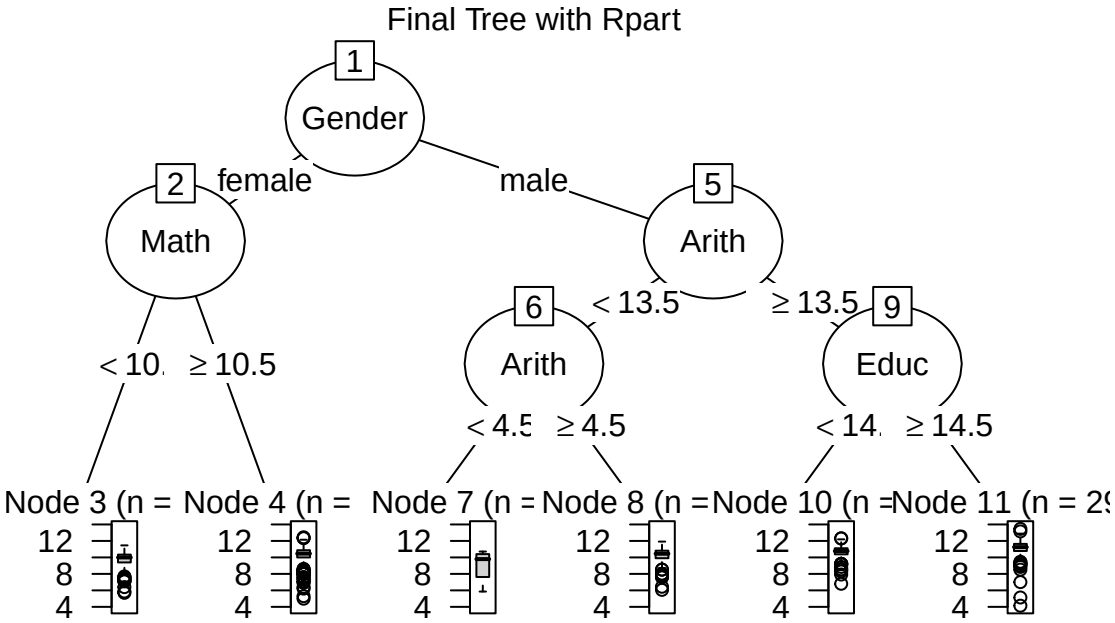
Answer: The tree is displayed as above.

- There are four end nodes in our tree. First, the classification is determined by gender, if the subject was female and if her years of education completed by 2006 was under 15.5, she would be likely to logIncome around 9.955, if she had over 15 years of education, her average logIncome would be around 10.48. Similarly, a male with under 15.5 years of education would be likely to have logIncome around 10.56, otherwise (with over 15 years of education) around 11.15.
- Yes. We observed that males tend to have more income than females regardless of the years of education obtained by 2006.
- Michelle is a female, and she completed 13 years of education by 2006. As such, she would be likely to have the logIncome around 9.95.

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 tune the fit2 settings to get a tree with small testing error (no need to try too hard.)

```
fit2 <- rpart(Income ~., data.train, minsplit=20, cp=.009)
plot(as.party(fit2), main="Final Tree with Rpart")
```




```
summary(fit2) # x=6, xerror = 0.8145
```

```
# testing error
fit1.pred <- predict(fit1, data.test)
fit1.error <- mean((data.test$Income-fit1.pred)^2)
fit1.error
```

```
## [1] 0.7102723
```

```
fit2.pred <- predict(fit2, data.vali)
fit2.error <- mean((data.test$Income-fit2.pred)^2)
fit2.error
```

```
## [1] 1.083843
```

```r
# tune fit2
fit2.1 <- rpart(Income ~., data.train, minsplit=20, cp=.004)
fit21.test <- predict(fit2.1, data.test)
fit2.error.tune <- mean((data.test$Income-fit21.test)^2)
fit2.error.tune
```

```
## [1] 0.7806682
```

Answer: The tree is displayed as above.

- Our tree has 6 end nodes, 3 splits with 4 variables involved. First, we still consider the gender of our subjects. For females with a Math score under 10.5, they would have a logIncome around 9.76, otherwise around 10.31. For males with Arithmetic reasoning score under 4.5, the logIncome is expected to be around 8.9; with Arith score between 4.5 and 13.5, the logIncome is expected to be around 10.26. For those males with Arith score higher than 13.5, if they had under 14.5 years of education completed by 2006, they would be expected to have a logIncome around 10.72; else, if they had over 14.5 years of education, they would be likely to have a logIncome around 11.15.
- The training error for fit1 is 0.8222, and training error for fit2 is 0.8145. Testing error for fit1 is 0.7102723, and testing error for fit2 is 1.083843 As we can see, fit1 has a larger training error, while fit2 has a larger testing error.
- We tried `cp = .004`, and we get an mse around 0.7806.

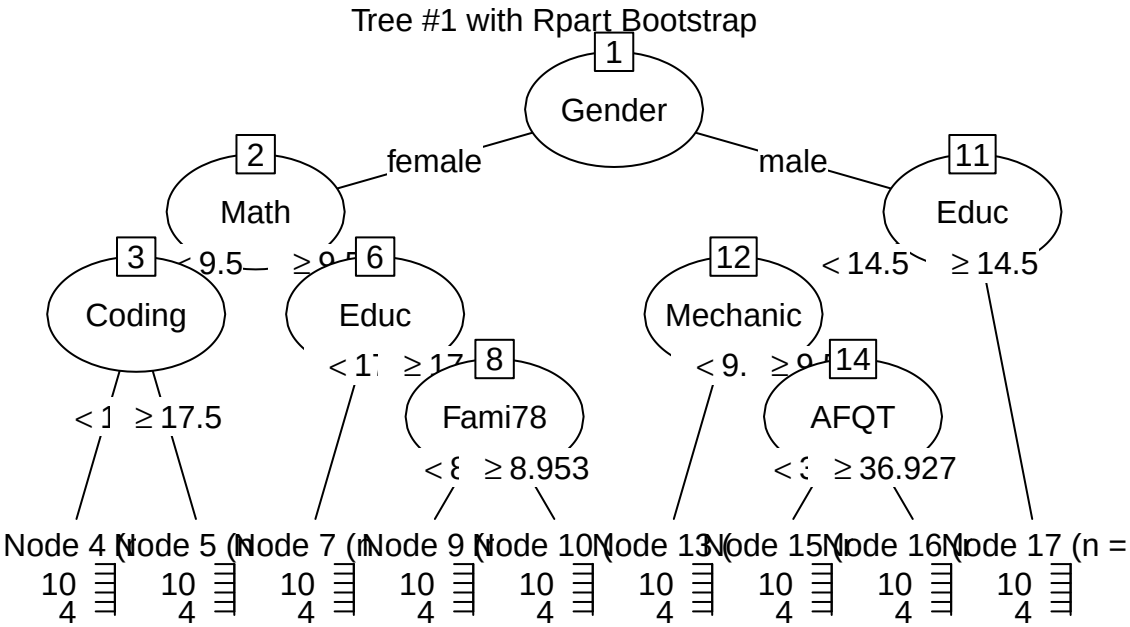
iii. Bag two trees: fit3

a) Take 2 bootstrap training samples and build two trees using the rpart(Income2005 ~., data.train.b, minsplit=20, cp=.009). Display both trees.

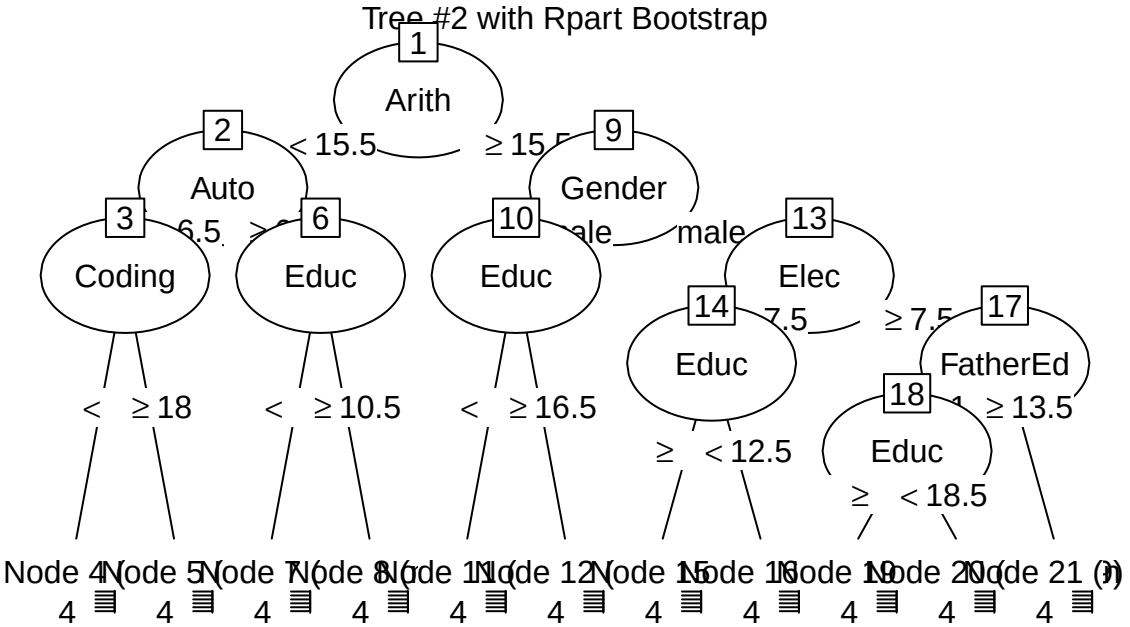
```r
set.seed(5)
ind.1 <- sample(seq_len(nrow(data.train)), replace = T)
data.train.1 <- IQ[ind.1, ]

set.seed(55)
ind.2 <- sample(seq_len(nrow(data.train)), replace = T)
data.train.2 <- IQ[ind.2, ]

tree1 <- rpart(Income ~., data.train.1, minsplit=20, cp=.009)
plot(as.party(tree1), main="Tree #1 with Rpart Bootstrap")
```




```
tree2 <- rpart(Income ~., data.train.2, minsplit=20, cp=.009)
plot(as.party(tree2), main="Tree #2 with Rpart Bootstrap")
```



b) Explain how to get fitted values for Michelle by bagging the two trees obtained above. Do not use the

Answer: From fit1, we can infer that, Michelle is a female with Math score over 9.5, education years below 17 years. So she is supposed to have logIncome around 10.176. From fit2, Michelle had Arith score under 15.5, Auto score over 6.5, and years of education over 10.5. So here she is supposed to have the logIncome of 10.25ish. We take an average of these two numbers, so overall, Michelle would be have logIncome of 10.21.

c) What is the testing error for the bagged tree. Is it guaranteed that the testing error by bagging the

```
# test error bagged tree
fit3 <- (predict(tree1, data.test) + predict(tree2, data.test)) / 2
fit3.error <- mean((data.test$Income-fit3)^2)
fit3.error
```

```
## [1] 0.6941175
```

```
# test error tree1
tree1.pred <- predict(tree1, data.test)
tree1.error <- mean((data.test$Income-tree1.pred)^2)
tree1.error
```

```
## [1] 0.7089325
```

```
# test error tree1
tree2.pred <- predict(tree2, data.test)
tree2.error <- mean((data.test$Income-tree2.pred)^2)
tree2.error
```

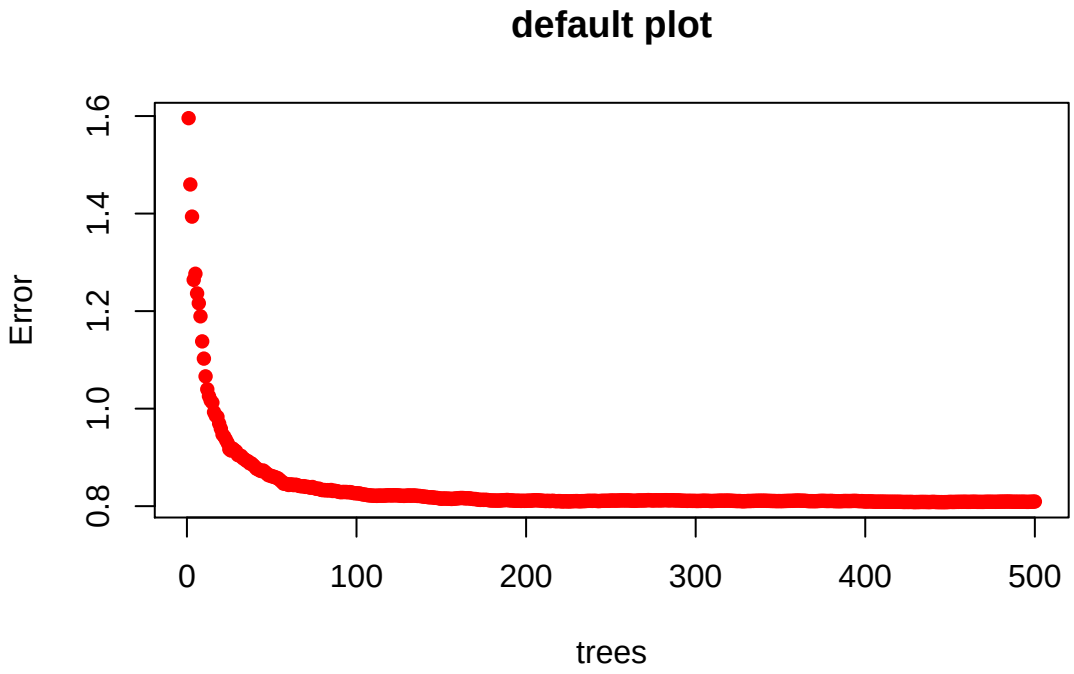
```
## [1] 0.7906924
```

Answer: The testing error for the bagged tree is 0.6941. Yes, we actually take the average of two trees by bagging them, thus we reduce the variance while maintain similar bias.

iv. Build a best possible RandomForest, call it fit4

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

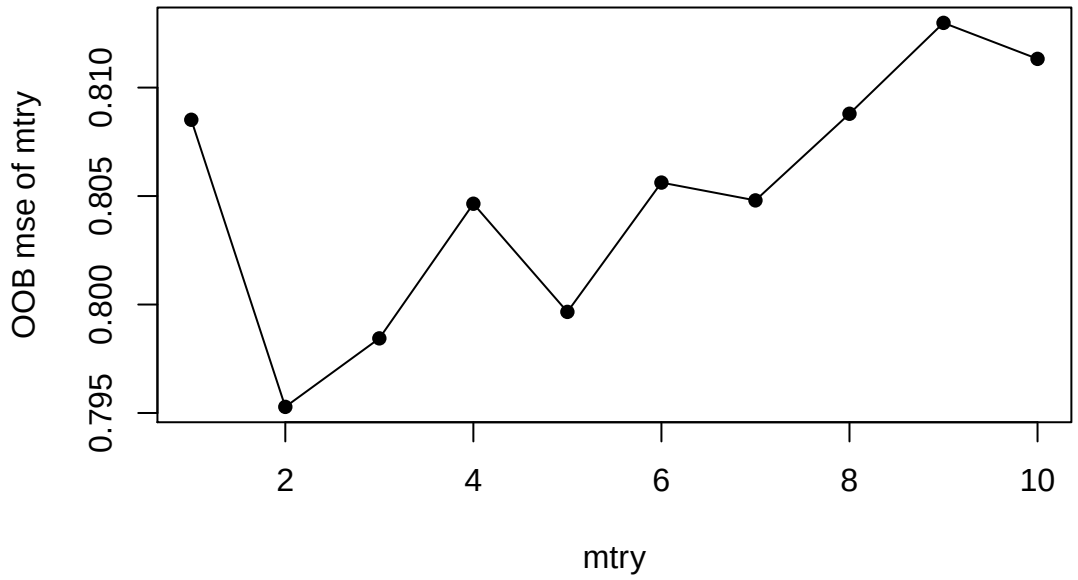
```
## tune mtry and number of trees
fit4.0 <- randomForest(Income~., data.train, mtry=10, ntree=500)
plot(fit4.0, col="red", pch=16, type="p", main="default plot")
```




```
rf.error.p <- 1:10
for (p in 1:10)
{
  fit.rf <- randomForest(Income~., data.train, mtry=p, ntree=300)
  #plot(fit.rf, col= p, lwd = 3)
  rf.error.p[p] <- fit.rf$mse[300]  # collecting oob mse
}

plot(1:10, rf.error.p, pch=16,
```

```
     xlab="mtry",
     ylab="OOB mse of mtry")
lines(1:10, rf.error.p)
```




```
fit4 <- randomForest(Income~., data.train, mtry=3, ntree=300)
fit4.pred <- predict(fit4, data.test)
fit4.error <- mean((data.test$Income-fit4.pred)^2)
fit4.error
```

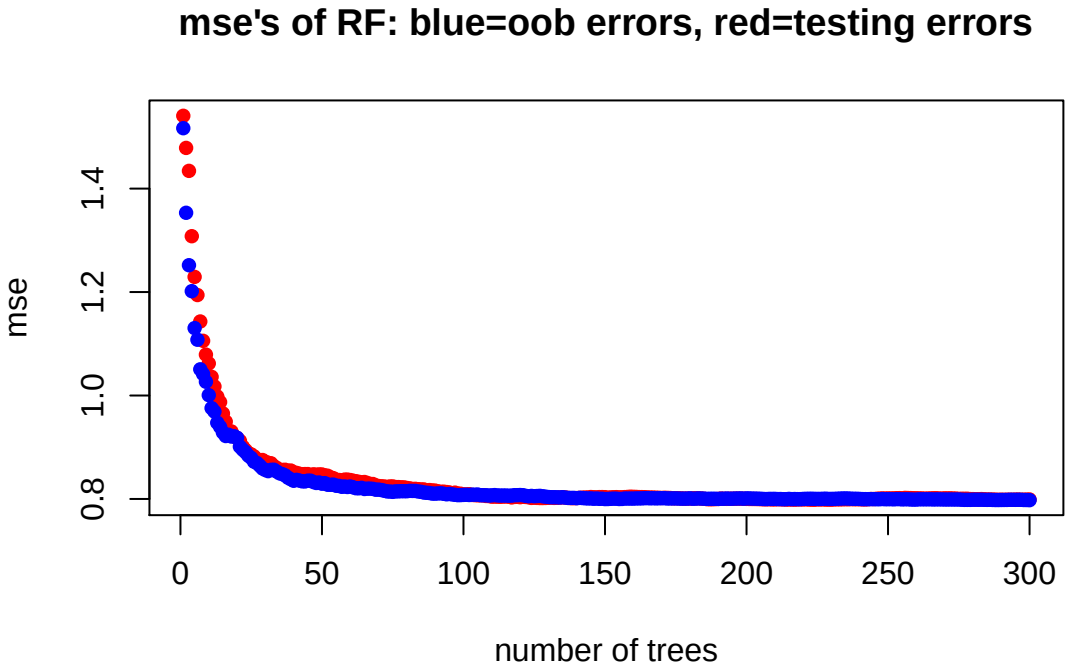
```
## [1] 0.7165338
```

Answer: First see how many trees we need. We observe 300 trees seem to be good, so we loop 10 times. Finally, we decide to use 3 try (varies between 2-5, 6 is not good). Given 300 trees and mtry=3, we build `fit4` using `randomForest()`.

b) Compare the OOB errors form fit4 to the testing errors using your testing data. Are you convinced tha

```
fit4.testing <- randomForest(Income~., data.train, xtest=data.test[, -20],
                          ytest=data.test[,20], mtry=3, ntree=300)

plot(1:300, fit4.testing$mse, col="red", pch=16,
     xlab="number of trees",
     ylab="mse",
     main="mse's of RF: blue=oob errors, red=testing errors")
points(1:300, fit4$mse, col="blue", pch=16)
```



Answer: As we can see from the plot above, the OOB errors are generally lower than testing errors, although the difference is not very salient. However, we are still convinced that OOB errors estimate testing error quite well.

c) What is the predicted value for Michelle?

Answer: The predicted value for Michelle is: 9.8189188

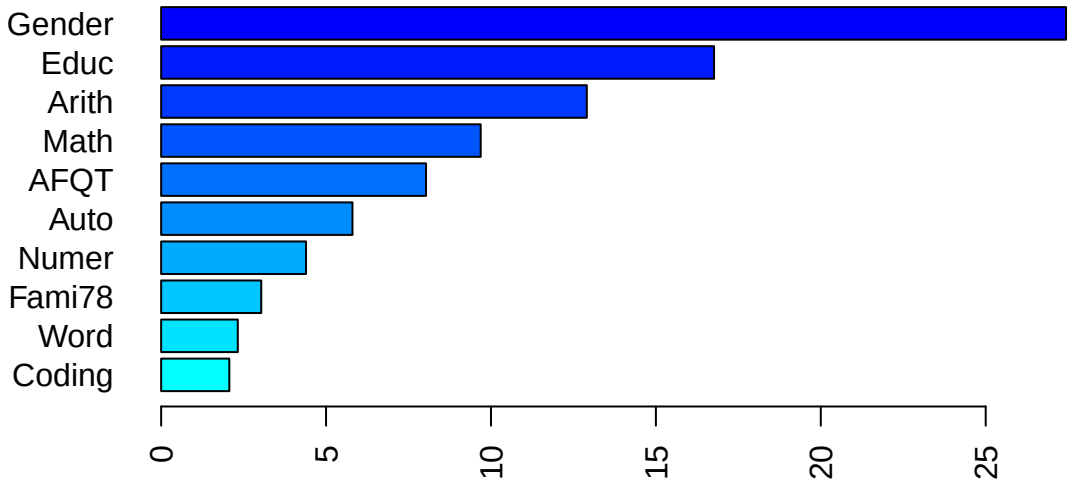
v. Build a best possible boosting tree, call it fit5

a) Once again show the process of how fit5 is built

```
# For regression problem, we use GBM
fit5 <- gbm(
  formula = Income ~.,
  distribution = "gaussian",
  data = data.train,
  n.trees = 10000,
  interaction.depth = 1,
  shrinkage = 0.001,
  cv.folds = 5,
  n.cores = NULL,
  verbose = FALSE
  )
print(fit5)
```

```
## gbm(formula = Income ~ ., distribution = "gaussian", data = data.train,
##     n.trees = 10000, interaction.depth = 1, shrinkage = 0.001,
##     cv.folds = 5, verbose = FALSE, n.cores = NULL)
## A gradient boosted model with gaussian loss function.
## 10000 iterations were performed.
## The best cross-validation iteration was 9600.
## There were 20 predictors of which 20 had non-zero influence.
```

```
summary(fit5, cBars = 10, method = relative.influence, las = 2)
```




```
##                   var      rel.inf
## Gender         Gender 27.435227634
## Educ             Educ 16.763993465
## Arith           Arith 12.905759476
## Math             Math  9.688700323
## AFQT             AFQT  8.031310182
## Auto             Auto  5.801433311
## Numer           Numer  4.392068519
## Fami78         Fami78  3.034271484
## Word             Word  2.323675901
## Coding         Coding  2.066968270
## Science       Science  1.927492447
## FatherEd     FatherEd  1.634943870
## Parag           Parag  1.077597638
## Elec             Elec  0.893587545
## MotherEd     MotherEd  0.800679821
## Imagazine   Imagazine  0.768296102
## Mechanic     Mechanic  0.347949483
## Race             Race  0.097566089
## Ilibrary     Ilibrary  0.004844356
## Inewspaper Inewspaper  0.003634084
```

Answer: We use gbm to build the boosting tree. For the parameters, we select gaussian as distribution, and create 10000 trees using the training data, and apply 5 fold cross validation. All cores we use are by default.

b) Report the testing error

```
fit5.pred <- predict(fit5, n.trees = fit5$n.trees, data.test)
fit5.mse.test <- mean((data.test$Income-fit5.pred)^2)
fit5.mse.test
```

```
## [1] 0.6944255
```

Answer: The testing error is 0.6945.

c) Report the predicted value for Michelle.

Answer: The predicted value for Michelle is: 9.8626045

vi. Now you have built so many predicted models (fit1 through fit5 in this section). 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.

```
data.frame(fit1.error, fit2.error, fit2.error.tune, fit3.error, tree1.error, tree2.error, fit4.error, fi
```

```
##   fit1.error fit2.error fit2.error.tune fit3.error tree1.error tree2.error
## 1  0.7102723   1.083843       0.7806682  0.6941175   0.7089325   0.7906924
##   fit4.error fit5.mse.test
## 1  0.7165338     0.6944255
```

```
prediction.bag <- (predict(tree1, data.vali) + predict(tree2, data.vali)) / 2
prediction.bag.error <- mean((data.vali$Income-prediction.bag)^2)
prediction.bag.error
```

```
## [1] 0.6358247
```

```
prediction.boost <- predict(fit5, data.vali)
```

```
## Using 9600 trees...
```

```
prediction.boost.error <- mean((data.vali$Income-prediction.boost)^2)
prediction.boost.error
```

```
## [1] 0.6166512
```

Answer: We bring all result together shown in a dataframe. As we can see, bagging and boosting ahve smaller error than other methods. So the final model can use bagged or boosting tree to predict income. Using ensemble methods to make prediction, we reduce the variance while maintain similar bias. The prediction error for bagged and boosting tree are 0.6358 and 0.6173 respectively.

Problem 2: Yelp review

(for more information check their website: http://www.yelp.com/dataset_challenge)

It is unlikely we will win the $5000 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.

Data needed: yelp_subset.csv available in Canvas. . . /Data/

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. Data and data split: Take a random sample of 20000 reviews (set.seed(1)) from our original data set. 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.

```
library(data.table)
data.all <- fread("yelp_subset.csv", stringsAsFactors = FALSE)
set.seed(1)
yelp_ind <- sample(seq_len(nrow(data.all)), size = 20000, replace = F)
data <- data.all[yelp_ind,]
data$rating <- c(0)
data$rating[data$stars >= 4] <- 1
data$rating <- as.factor(data$rating)
weekdays <- weekdays(as.Date(data$date)) # get weekdays for each review
months <- months(as.Date(data$date))
```

```
# remove irrelevant words
yelp.text <- data$text   # take the text out
mycorpus1 <- VCorpus( VectorSource(yelp.text))
mycorpus2 <- tm_map(mycorpus1, content_transformer(tolower))
mycorpus3<- tm_map(mycorpus2, removeWords, stopwords("english"))
mycorpus4 <- tm_map(mycorpus3, removePunctuation)
mycorpus5 <- tm_map(mycorpus4, removeNumbers)
mycorpus6 <- tm_map(mycorpus5, stemDocument, lazy = TRUE)
dtm1 <- DocumentTermMatrix(mycorpus6)   ## library = collection of words for all documents
as.matrix(dtm1[1, 1:50])
```

```
##     Terms
## Docs aaa aaaaaa aaaah aaaand aaaawweeesssooommme aaaawwweesssommme aaah
##    1   0      0     0      0                   0                  0    0
##     Terms
## Docs aaahhhhh aaargh aadladjac aag aah aahh aahhhhh aammmaaazzzinnnnggg
##    1        0      0         0   0   0    0       0                   0
##     Terms
## Docs aangan aardvark aaron aarp ababa abacaxi aback abalon abandon abasi
##    1      0        0     0    0     0       0     0      0       0     0
##     Terms
## Docs abat abbey abbi abbott abbrevi abc abdomen abdomin abdomnioplasti
##    1    0     0    0      0       0   0       0       0              0
##     Terms
## Docs abduct abdul abduljabbar abe abel aber abercrombi aberdeen aberr
##    1      0     0           0   0    0    0          0        0     0
##     Terms
## Docs abhor abhorr abig abigail abil abit abita
##    1     0      0    0       0    0    0     0
```

```
# set threshold
threshold <- .005*length(mycorpus6)   # .5% of the total documents
words.05 <- findFreqTerms(dtm1, lowfreq=threshold)
dtm.05 <- DocumentTermMatrix(mycorpus6, control = list(dictionary = words.05))
dtm.05.2 <- removeSparseTerms(dtm1, 1-.01)  # sparsity < .99  another way of removing sparsity
```

(i) Briefly explain what does this matrix record? What is the cell number at row 100 and column 405? Wha

```
# inspect(dtm.05[1:5, 100:405])
dtm.05[100,405]
```

```
## <<DocumentTermMatrix (documents: 1, terms: 1)>>
## Non-/sparse entries: 0/1
## Sparsity           : 100%
## Maximal term length: 4
## Weighting          : term frequency (tf)
```

```
dim(dtm.05)
```

```
## [1] 20000  1691
```

Answer: The occurance of each word in the whole documents. The cell number is $1691 * 100 + 405 = 169505$, represents the number 169505 word, its terms and its sparsity.

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

Answer: 97%. So there is some words that occured in different reviews.

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

```
names(data)
```

```
##  [1] "user_id"      "review_id"    "text"         "votes.cool"
##  [5] "business_id"  "votes.funny"  "stars"        "date"
##  [9] "type"         "votes.useful" "rating"
```

```
## Combine the original data with the text matrix
data1.temp <- data.frame(data,as.matrix(dtm.05))
dim(data1.temp)
```

```
## [1] 20000  1702
```

```
names(data1.temp)[1:30]
```

```
##  [1] "user_id"      "review_id"    "text"         "votes.cool"
##  [5] "business_id"  "votes.funny"  "stars"        "date"
##  [9] "type"         "votes.useful" "rating"       "abl"
## [13] "absolut"      "accept"       "access"       "accommod"
## [17] "accompani"    "account"      "across"       "act"
## [21] "actual"       "add"          "addict"       "addit"
## [25] "admit"        "ador"         "adult"        "advantag"
## [29] "adventur"     "advic"
```

```
# data2 consists of date, rating and all the top 1% words
data2 <- data1.temp[, c(1,7, 8,11, 14:ncol(data1.temp))]
names(data2)[1:20]
```

```
##  [1] "user_id"   "stars"     "date"      "rating"    "accept"
##  [6] "access"    "accommod"  "accompani" "account"   "across"
## [11] "act"       "actual"    "add"       "addict"    "addit"
## [16] "admit"     "ador"      "adult"     "advantag"  "adventur"
```

```
dim(data2)
```

```
## [1] 20000  1693
```

```
# Set the stars as a two category response variable
data2$stars = as.factor(ifelse(data2$stars > 3, "1","0"))
```

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

```
data2.train = data2[c(1:13000),-c(1,3,4)]
data2.test = data2[c(13001:18000),-c(1,3,4)]
data2.validaion = data2[c(18001:20000),-c(1,3,4)]
```

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

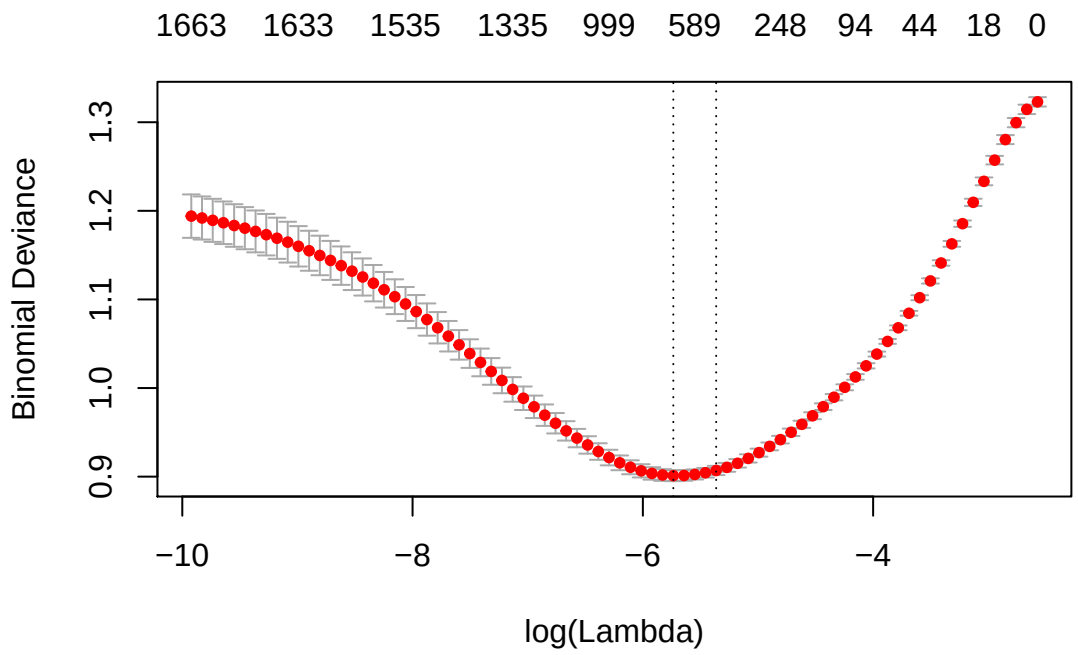
```
y <- data2.train$stars
X <- as.matrix(data2.train[, -c(1)])

set.seed(2)
X <- sparse.model.matrix(stars~., data=data2.train)[, -1]

result.lasso <- cv.glmnet(X, y, alpha=.99, family="binomial")
dim(X)
```

```
## [1] 13000  1689
```

```
plot(result.lasso)
```




```
beta.lasso <- coef(result.lasso, s="lambda.1se")   # output lasso estimates
beta <- beta.lasso[which(beta.lasso !=0),] # non zero beta's
beta <- as.matrix(beta);
beta <- rownames(beta)
beta[2:50]
```

```
##  [1] "accommod" "act"      "actual"   "addict"   "admit"    "ador"
##  [7] "advantag" "adventur" "afford"   "age"      "agre"     "air"
## [13] "almost"   "alright"  "also"     "alway"    "amaz"     "ambianc"
## [19] "anim"     "annoy"    "anoth"    "anymor"   "anyon"    "anywher"
## [25] "apolog"   "appar"    "appeal"   "art"      "ask"      "ass"
## [31] "attempt"  "attitud"  "austin"   "authent"  "avail"    "averag"
## [37] "avoid"    "awesom"   "babi"     "back"     "bad"      "bake"
## [43] "balanc"   "ball"     "bare"     "barista"  "base"     "basement"
## [49] "bathroom"
```

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

```
glm.input <- as.formula(paste("stars", "~", paste(beta[-1], collapse = "+"))) # prepare the formulae
result.glm <- glm(glm.input, family=binomial, data2.train )
```

(i) 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.

```
result.glm.coef <- coef(result.glm)
good.glm <- result.glm.coef[which(result.glm.coef > 0)]
names(good.glm)  # which words are positively associated with good ratings
```

```
##   [1] "(Intercept)" "accommod"    "addict"      "admit"       "ador"
##   [6] "adventur"    "afford"      "age"         "air"         "almost"
##  [11] "also"        "alway"       "amaz"        "anim"        "anyon"
##  [16] "anywher"     "art"         "ass"         "austin"      "authent"
```

```
##  [21] "avail"       "awesom"      "babi"        "bake"        "balanc"
##  [26] "barista"     "beat"        "beauti"      "belli"       "berkeley"
##  [31] "best"        "book"        "bookstor"    "boot"        "boston"
##  [36] "bring"       "budget"      "butter"      "can"         "caramel"
##  [41] "cash"        "casual"      "central"     "char"        "charm"
##  [46] "cheesecak"   "chipotl"     "chocol"      "choos"       "citi"
##  [51] "classic"     "club"        "collect"     "combin"      "come"
##  [56] "comfi"       "comfort"     "comment"     "complaint"   "compliment"
##  [61] "concern"     "cooki"       "cozi"        "crab"        "craft"
##  [66] "crazi"       "cream"       "cucumb"      "daili"       "damn"
##  [71] "darn"        "day"         "decad"       "def"         "definit"
##  [76] "delici"      "delight"     "diddi"       "die"         "discov"
##  [81] "divers"      "downsid"     "downstair"   "dream"       "earlier"
##  [86] "earth"       "easi"        "effici"      "enjoy"       "equal"
##  [91] "ever"        "everi"       "everyon"     "everyth"     "excel"
##  [96] "extra"       "fabul"       "fantast"     "far"         "fast"
## [101] "favorit"     "five"        "flower"      "fluffi"      "four"
## [106] "frame"       "free"        "fresh"       "friend"      "fun"
## [111] "funki"       "funni"       "garlic"      "gem"         "glad"
## [116] "goat"        "good"        "gorgeous"    "grab"        "great"
## [121] "greet"       "haircut"     "hat"         "heaven"      "help"
## [126] "hesit"       "hidden"      "high"        "highlight"   "hipster"
## [131] "hold"        "home"        "hot"         "huge"        "imagin"
## [136] "incred"      "indian"      "inexpens"    "ithaca"      "juici"
## [141] "know"        "knowledg"    "laid"        "learn"       "level"
## [146] "life"        "light"       "live"        "local"       "love"
## [151] "lover"       "lucki"       "man"         "mani"        "memori"
## [156] "modern"      "mom"         "mouth"       "movi"        "muffin"
## [161] "museum"      "music"       "name"        "new"         "nice"
## [166] "notch"       "often"       "omelet"      "omg"         "outstand"
## [171] "pair"        "pant"        "park"        "patio"       "perfect"
## [176] "play"        "pleasant"    "pleasur"     "plenti"      "princeton"
## [181] "process"     "profession"  "prompt"      "provid"      "pud"
## [186] "quiet"       "readi"       "reason"      "recommend"   "refresh"
## [191] "regret"      "regular"     "relax"       "right"       "satisfi"
## [196] "share"       "shop"        "show"        "simpl"       "slice"
## [201] "smart"       "smooth"      "soon"        "sooo"        "sport"
## [206] "spot"        "stuf"        "stuff"       "summer"      "super"
## [211] "surpris"     "tasti"       "tempura"     "tend"        "thank"
## [216] "thorough"    "tonight"     "tradit"      "travel"      "treat"
## [221] "trip"        "truck"       "true"        "trust"       "uniqu"
## [226] "usual"       "varieti"     "vegan"       "version"     "vintag"
## [231] "wax"         "weather"     "week"        "well"        "west"
## [236] "whenev"      "whip"        "winter"      "wish"        "wonder"
## [241] "wood"        "world"       "worri"       "wow"         "yelp"
## [246] "yes"         "yum"         "yummi"
```

```
good.fre <- sort(good.glm, decreasing = TRUE) # sort the coef's
round(good.fre, 4)[1:2] # leading 2 positive words
```

```
##  notch  hesit
## 1.6714 1.4424
```

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

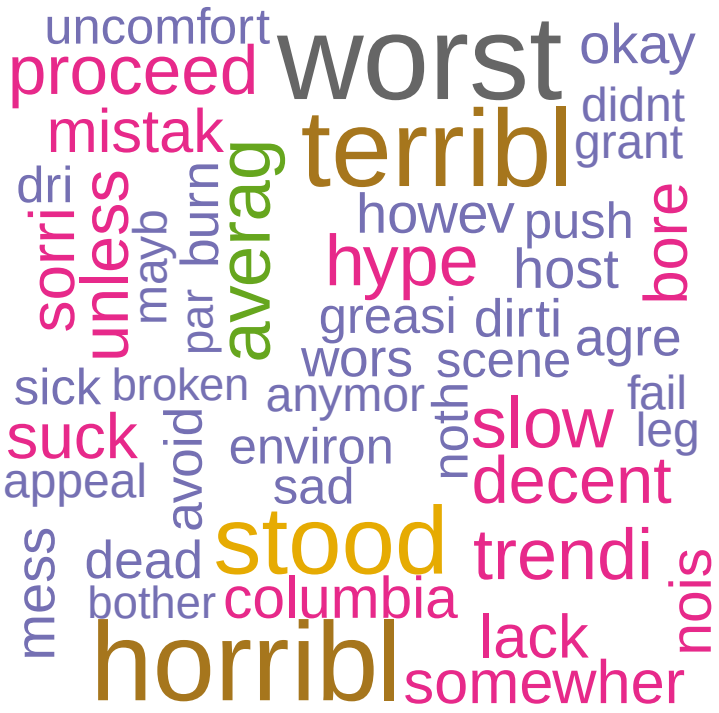
```
cor.special <- brewer.pal(8, "Dark2")  # set up a pretty color scheme
good.word <- names(good.fre)  # good words with a decreasing order in the coeff's
wordcloud(good.word[1:100], good.fre[1:100], colors=cor.special, ordered.colors=F)
```

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

```
bad.glm <- result.glm.coef[which(result.glm.coef < 0)]
bad.fre <- sort(-bad.glm, decreasing = TRUE) # sort the coef's
round(bad.fre, 4)[1:2] # leading 2 negative words
```

```
## mediocr   worst
##  1.9460  1.8125
```

```
bad.word <- names(bad.fre)  # good words with a decreasing order in the coeff's
wordcloud(bad.word[1:100], bad.fre[1:100],colors=cor.special, ordered.colors=F)
```



(iv) Summarize the findings.

```
# Compare the bad words and good words:
par(mfrow=c(1,2))
cor.special <- brewer.pal(8,"Dark2")
wordcloud(good.word[1:100], good.fre[1:100], colors=cor.special, ordered.colors=F)
wordcloud(bad.word[1:100], bad.fre[1:100], color="darkgreen", ordered.colors=F)
```

Answer: There are some common words in positive reviews as well as some negative words in negative reviews. We can predict the stars by the nature of each word in the review.

5. Using majority votes find the testing errors
   i) From Lasso fit in 3)

```
predict.lasso <- predict(result.lasso, as.matrix(data2.test[, -1]), type = "class", s="lambda.1se")
lasso.error <- mean(data2.test$stars != predict.lasso)
```

```
lasso.error
```

```
## [1] 0.2078
```

ii) From logistic regression in 4)

```
predict.glm <- predict(result.glm, data2.test, type = "response")
class.glm <- rep("0", 10000)
class.glm[predict.glm > .5] ="1"
length(class.glm)
```

```
## [1] 10000
```

```
testerror.glm <- mean(data2.test$stars != class.glm)
testerror.glm
```

```
## [1] 0.204
```

iii) Which one is smaller?

Answer: Misclassification error for Lasso fit is 0.2078,logistic regression error is 0.2024. Logistic regression error is a little smaller.

6. Now train the data using the training data set by RF. Get the testing error. Also explain how the RF works and how you tune the tuning parameters.

```
fit.rf.ranger <- ranger::ranger(stars~., data2.train, num.trees = 200, importance="impurity") # no plot
```

```
## Growing trees.. Progress: 59%. Estimated remaining time: 21 seconds.
```

```
fit.rf.ranger
```

```
## Ranger result
##
## Call:
##  ranger::ranger(stars ~ ., data2.train, num.trees = 200, importance = "impurity")
##
## Type:                             Classification
## Number of trees:                  200
## Sample size:                      13000
## Number of independent variables:  1689
## Mtry:                             41
## Target node size:                 1
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error:             22.13 %
```

```
predict.rf <- predict(fit.rf.ranger, data=data2.test, type="response")  # output the classes by majorit
predict.rf.error <- mean(data2.test$stars != predict.rf$predictions)
predict.rf.error
```

```
## [1] 0.218
```

Answer: The testing error is 0.218. The RF works by combining the multiple singles trees, and generate the results by choosing majority vote. We tune the tune parameters by changing the value of `num.trees`.

7. Now train the data using Boosting. Get the testing error

```
fit.gbm <- gbm(stars ~ . ,data = data2.train,distribution = "gaussian", n.trees = 100,
                  shrinkage = 0.01, interaction.depth = 4)
```

```
gbm.pred <- predict(fit.gbm, data2.test, n.trees = fit5$n.trees, type="response")
class.glm.boost <- rep("0", 10000)
class.glm.boost[gbm.pred > 0.5] <- "1"
test.error.boost<-mean(data2.test$stars != class.glm.boost)
test.error.boost
```

```
## [1] 0.3756
```

Answer: The testing erorr is 0.3756.

8. If you can train a neural net with two layers with a reasonable number of neutrons in each layer (say 20). You could try a different number of layers and different number of neutrons and see how the results change. Settle down on one final architecture. Report the testing errors.

```
library(keras)

model <- keras_model_sequential() %>%
  layer_dense(units = 10, activation = "relu", input_shape = c(1689)) %>% # 1 layer with 10 neurons
  layer_dense(units = 4, activation = "relu") %>%  # layer 2 with 4 neurons
  layer_dense(units = 1, activation = "sigmoid")
print(model)
```

```
## Model
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## dense (Dense)                (None, 10)                16900
## _________________________________________________________________________
## dense_1 (Dense)              (None, 4)                 44
## _________________________________________________________________________
## dense_2 (Dense)              (None, 1)                 5
## =========================================================================
## Total params: 16,949
## Trainable params: 16,949
## Non-trainable params: 0
## _________________________________________________________________________
```

```
#Compile the model
model %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)
```

Split data from training data and testing data when train the model

```
x_train <- as.matrix(data2.train[, -c(1)])
x_test <- as.matrix(data2.test[, -c(1)])
y_train <- as.matrix(data2.train$stars)
y_test <- as.matrix(data2.test$stars)

y_train <- as.numeric(y_train)
y_test <- as.numeric(y_test)
```

```
# Set up validation set
val_indices <- 1:4000
```

```
x_val <- x_train[val_indices,]
partial_x_train <- x_train[-val_indices,]

y_val <- y_train[val_indices]
partial_y_train <- y_train[-val_indices]

y_val <- as.numeric(y_val)
partial_y_train <- as.numeric(partial_y_train)

# fit the model and tune using validate data
model %>% fit(
  partial_x_train,
  partial_y_train,
  epochs = 20,
  batch_size = 512,
  validation_data = list(x_val, y_val)
)
```

To avoid overfitting, use 4 epoches in the final model. Retrain the model using training and testing data.

```
model <- keras_model_sequential() %>%
  layer_dense(units = 10, activation = "relu", input_shape = c(1689)) %>% # 1 layer with 10 neurons
  layer_dense(units = 1, activation = "sigmoid")
print(model)
```

```
## Model
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## dense_3 (Dense)              (None, 10)                16900
## _________________________________________________________________________
## dense_4 (Dense)              (None, 1)                 11
## =========================================================================
## Total params: 16,911
## Trainable params: 16,911
## Non-trainable params: 0
## _________________________________________________________________________
```

```
#Compile the model
model %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

model %>% fit(x_train, y_train, epochs = 4, batch_size = 512)
```

See the test error for the model

```
results <- model %>% evaluate(x_test, y_test)
results
```

```
## $loss
## [1] 0.4623359
##
## $acc
```

```
## [1] 0.8048
```

9. 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 using our final model?

Answer: To choose final model among the previous few models, we find that logistic regerssion seem to produce the least testing error, 0.2024. But there are not so much difference, and we are not surprised.

```
final.model <- result.glm

predict.glm <- predict(final.model, data2.validaion, type = "response")
class.glm.final <- rep("0", 2000)
class.glm.final[predict.glm > .5] ="1"
length(class.glm.final)
```

```
## [1] 2000
```

```
testerror.glm <- mean(data2.validaion$stars != class.glm.final)
testerror.glm   # mis classification error is 0.2024
```

```
## [1] 0.2175
```

The test errror forr the final model is 0.2175. So I will prerdict a review to see each word in the reviews, whehter it is a positive or negative word. So after I trained my final model by enough training data, I will put my review into my final model, if it is positive probability, I will believe the star is 4 or 5, otherwise is 1,2 or 3.