# Assignment_4

*Emily*

*10/31/2019*

## R Markdown

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.5.3
```

```
#install.packages()
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.5.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

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

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

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.3
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

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

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.5.3
```

```
## Loaded gbm 2.1.5
```

```
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 3.5.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

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

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

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

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

```
library(ggplot2)
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3
```

```
## -- Attaching packages ---------------------------------------------------------------------------
------------------------------ tidyverse 1.2.1 --
```

```
## v tibble  2.1.3     v purrr   0.2.5
## v tidyr   1.0.0     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
```

```
## Warning: package 'tibble' was built under R version 3.5.3
```

```
## Warning: package 'tidyr' was built under R version 3.5.3
```

```
## -- Conflicts ------------------------------------------------------------------------------------
------------------------ tidyverse_conflicts() --
## x dplyr::combine()       masks randomForest::combine()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x purrr::lift()          masks caret::lift()
## x randomForest::margin() masks ggplot2::margin()
```

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.5.3
```

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

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

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

```
library(ngram)
#install.packages('ngram')
#Sys.setenv(JAVA_HOME="C:/Program Files/Java/jdk1-11.0.4/bin")
#install.packages("tm.plugin.webmining")
#library(tm.plugin.webmining)

#install.packages("boot")
library(boot)
```

```
## Warning: package 'boot' was built under R version 3.5.3
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:lattice':
##
##     melanoma
```

# Question 1a: Cleaning Data

The purpose of this section is to turn the original data set containing three fields (title, HTML formatted body, and score of stack exchange questions) into a dataframe which describes these features in a format upon which a variety of models may be trained to determine whether or not a question will be good. I will acheive this primarily through converting the body and title fields into a series of features describing the frequency of occurrence of key words in each.

```
Clean <- function(HTML){
  return(gsub("<.*?>", "", HTML))
}
```

The Clean function will allow me to remove parse out the html components to change the result into text.

```
# read in the Stack Exchange questions
questions <- read.csv(file = "ggplot2questions2016_17.csv", stringsAsFactors = FALSE)

# removing the HTML components from the body portion
questions_html <- questions %>%
  mutate(Body2 = Clean(Body))

# creating fields which count the number of words in the body and head in case these are predictive of good/
bad questions
questions_html$bodyword = unlist(lapply(questions_html$Body2, wordcount))
questions_html$titleword = unlist(lapply(questions_html$Title, wordcount))

# removing parantheses to deal with common r syntax of function(data) and parse the components out into indi
vidual words
questions_html= questions_html %>%
  mutate(Body2 = gsub('[[:digit:]]+', '', Body2)) %>%
  mutate(Body2 = gsub("*\\(", " ", Body2)) %>%
  mutate(Body2 = gsub("*\\)", "", Body2)) %>%
  mutate(Body2 = gsub("[[:punct:]]", " ", Body2))

# creating a field simplifying the scores into a logical of whether or not the question is good
questions_html$goodq = as.factor(as.logical(questions_html$Score >= 1))

table(questions_html$goodq)
```

```
##
## FALSE  TRUE
##  3791  3677
```

Note that according to this table, the number of good questions and bad questions are approximately equal.

```
# parsing out the body by casting to corpus
corpus = Corpus(VectorSource(questions_html$Body2))

corpus[[1]]
```

```
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 538
```

```
strwrap(corpus[[5]])
```

```
##  [1] "I am trying to make a bar graph with error bars and I would like"
##  [2] "to also add letter labels to indicate statistical significant"
##  [3] "between each group over the bars However when I add the code for"
##  [4] "the labeling I get the following message"
##  [5] ""
##  [6] "Error in eval expr envir enclos object treatment area not found"
##  [7] ""
##  [8] "Here is the code I am using to produce the graph"
##  [9] ""
## [10] "This data frame calculates statistics for each treatment data"
## [11] "summary area lt data frame treatment area levels shapes a Exposure"
## [12] "mean a tapply shapes Area shapes Exposure mean n a tapply shapes"
## [13] "Area shapes Exposure length sd a tapply shapes Area shapes"
## [14] "Exposure sd"
## [15] ""
## [16] "Precalculate standard error of the mean SEM data summary area sem"
## [17] "a lt data summary area sd a sqrt data summary area n a"
## [18] ""
## [19] "Precalculate margin of error for confidence interval data summary"
## [20] "area me lt qt alpha df data summary area n a data summary area sem"
## [21] "Get some stats label area cor lt data frame Group c CON CCCP CON"
## [22] "DMSO HF CCCP HF DMSO Value c Make the plot require ggplot"
## [23] ""
## [24] "png barplot sem mito area lab stat png ggplot data summary area"
## [25] "aes x treatment area y mean a geom bar position position dodge"
## [26] "stat identity fill c red red forestgreen darkgreen geom errorbar"
## [27] "aes ymin mean a sem a ymax mean a sem a ggtitle Mitochondria Area"
## [28] "after h CCCP exposure GV oocytes xlab Exposure ylab expression"
## [29] "paste Area u theme bw geom text data label area cor aes label c a"
## [30] "b c d theme panel grid major element blank dev off"
```

```
# Change all the text to lower case.
corpus = tm_map(corpus, tolower)
```

```
## Warning in tm_map.SimpleCorpus(corpus, tolower): transformation drops
## documents
```

```
# tolower is a function
# Lets check:
strwrap(corpus[[5]])
```

```
##  [1] "i am trying to make a bar graph with error bars and i would like"
##  [2] "to also add letter labels to indicate statistical significant"
##  [3] "between each group over the bars however when i add the code for"
##  [4] "the labeling i get the following message"
##  [5] ""
##  [6] "error in eval expr envir enclos object treatment area not found"
##  [7] ""
##  [8] "here is the code i am using to produce the graph"
##  [9] ""
## [10] "this data frame calculates statistics for each treatment data"
## [11] "summary area lt data frame treatment area levels shapes a exposure"
## [12] "mean a tapply shapes area shapes exposure mean n a tapply shapes"
## [13] "area shapes exposure length sd a tapply shapes area shapes"
## [14] "exposure sd"
## [15] ""
## [16] "precalculate standard error of the mean sem data summary area sem"
## [17] "a lt data summary area sd a sqrt data summary area n a"
## [18] ""
## [19] "precalculate margin of error for confidence interval data summary"
## [20] "area me lt qt alpha df data summary area n a data summary area sem"
## [21] "get some stats label area cor lt data frame group c con cccp con"
## [22] "dmso hf cccp hf dmso value c make the plot require ggplot"
## [23] ""
## [24] "png barplot sem mito area lab stat png ggplot data summary area"
## [25] "aes x treatment area y mean a geom bar position position dodge"
## [26] "stat identity fill c red red forestgreen darkgreen geom errorbar"
## [27] "aes ymin mean a sem a ymax mean a sem a ggtitle mitochondria area"
## [28] "after h cccp exposure gv oocytes xlab exposure ylab expression"
## [29] "paste area u theme bw geom text data label area cor aes label c a"
## [30] "b c d theme panel grid major element blank dev off"
```

```
#Remove all punctuation
#corpus = tm_map(corpus, removePunctuation)
# Take a look:
#strwrap(corpus[[5]])

# Remove stop words
# First, take a look at tm's stopwords:

stopwords("english")[1:40]
```

```
##  [1] "i"          "me"         "my"         "myself"     "we"
##  [6] "our"        "ours"       "ourselves"  "you"        "your"
## [11] "yours"      "yourself"   "yourselves" "he"         "him"
## [16] "his"        "himself"    "she"        "her"        "hers"
## [21] "herself"    "it"         "its"        "itself"     "they"
## [26] "them"       "their"      "theirs"     "themselves" "what"
## [31] "which"      "who"        "whom"       "this"       "that"
## [36] "these"      "those"      "am"         "is"         "are"
```

```
#length(stopwords("english"))

# Just remove stopwords:
corpus = tm_map(corpus, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(corpus, removeWords, stopwords("english")):
## transformation drops documents
```

```
# Remove stopwords and "apple" - this is a word common to all of our tweets
#corpus = tm_map(corpus, removeWords, c("ggplot2", stopwords("english")))
# Take a look:
strwrap(corpus[[5]])
```

```
##  [1] "trying make bar graph error bars like also add letter labels"
##  [2] "indicate statistical significant group bars however add code"
##  [3] "labeling get following message"
##  [4] ""
##  [5] "error eval expr envir enclos object treatment area found"
##  [6] ""
##  [7] "code using produce graph"
##  [8] ""
##  [9] "data frame calculates statistics treatment data summary area lt"
## [10] "data frame treatment area levels shapes exposure mean tapply"
## [11] "shapes area shapes exposure mean n tapply shapes area shapes"
## [12] "exposure length sd tapply shapes area shapes exposure sd"
## [13] ""
## [14] "precalculate standard error mean sem data summary area sem lt data"
## [15] "summary area sd sqrt data summary area n"
## [16] ""
## [17] "precalculate margin error confidence interval data summary area lt"
## [18] "qt alpha df data summary area n data summary area sem get stats"
## [19] "label area cor lt data frame group c con cccp con dmso hf cccp hf"
## [20] "dmso value c make plot require ggplot"
## [21] ""
## [22] "png barplot sem mito area lab stat png ggplot data summary area"
## [23] "aes x treatment area y mean geom bar position position dodge stat"
## [24] "identity fill c red red forestgreen darkgreen geom errorbar aes"
## [25] "ymin mean sem ymax mean sem ggtitle mitochondria area h cccp"
## [26] "exposure gv oocytes xlab exposure ylab expression paste area u"
## [27] "theme bw geom text data label area cor aes label c b c d theme"
## [28] "panel grid major element blank dev"
```

```
# Step 5: Stem our document
# Recall, this means chopping off the ends of words that aren't maybe
# as necessary as the rest, like 'ing' and 'ed'
corpus = tm_map(corpus, stemDocument)
```

```
## Warning in tm_map.SimpleCorpus(corpus, stemDocument): transformation drops
## documents
```

```
# Take a look:
strwrap(corpus[[5]])
```

```
##  [1] "tri make bar graph error bar like also add letter label indic"
##  [2] "statist signific group bar howev add code label get follow messag"
##  [3] "error eval expr envir enclo object treatment area found code use"
##  [4] "produc graph data frame calcul statist treatment data summari area"
##  [5] "lt data frame treatment area level shape exposur mean tappli shape"
##  [6] "area shape exposur mean n tappli shape area shape exposur length"
##  [7] "sd tappli shape area shape exposur sd precalcul standard error"
##  [8] "mean sem data summari area sem lt data summari area sd sqrt data"
##  [9] "summari area n precalcul margin error confid interv data summari"
## [10] "area lt qt alpha df data summari area n data summari area sem get"
## [11] "stat label area cor lt data frame group c con cccp con dmso hf"
## [12] "cccp hf dmso valu c make plot requir ggplot png barplot sem mito"
## [13] "area lab stat png ggplot data summari area ae x treatment area y"
## [14] "mean geom bar posit posit dodg stat ident fill c red red"
## [15] "forestgreen darkgreen geom errorbar ae ymin mean sem ymax mean sem"
## [16] "ggtitl mitochondria area h cccp exposur gv oocyt xlab exposur ylab"
## [17] "express past area u theme bw geom text data label area cor ae"
## [18] "label c b c d theme panel grid major element blank dev"
```

```
# Seems we didn't catch all of the ggplot
#corpus = tm_map(corpus, removeWords, c("ggplot"))

# Step 6: Create a word count matrix (rows are tweets, columns are words)
# We've finished our basic cleaning, so now we want to calculate frequencies
# of words across the tweets
frequencies = DocumentTermMatrix(corpus)
# We can get some summary information by looking at this structure
frequencies
```

```
## <<DocumentTermMatrix (documents: 7468, terms: 25407)>>
## Non-/sparse entries: 456260/189283216
## Sparsity           : 100%
## Maximal term length: 983
## Weighting          : term frequency (tf)
```

```
# Step 7: Account for sparsity
# We currently have way too many words, which will make it hard to train
# our models and may even lead to overfitting.
# Use findFreqTerms to get a feeling for which words appear the most
# currently commented out because otherwise the document would be enormous, but they were used for examinati
on

# Words that appear at least 50 times:
#findFreqTerms(frequencies, lowfreq=1000)
# Words that appear at least 20 times:
#findFreqTerms(frequencies, lowfreq=500)

# Our solution to the possibility of overfitting is to only keep terms
# that appear in x% or more of the tweets. For example:
#examinint number of terms in 1%
sparse = removeSparseTerms(frequencies, 0.99)

# in .5% or more
sparse = removeSparseTerms(frequencies, 0.995)
# How many did we keep?
#sparse

# Let's keep it at the 20% (otherwise it takes FOREVER to run)
sparse = removeSparseTerms(frequencies, 0.8)
sparse
```

```
## <<DocumentTermMatrix (documents: 7468, terms: 49)>>
## Non-/sparse entries: 127908/238024
## Sparsity           : 65%
## Maximal term length: 8
## Weighting          : term frequency (tf)
```

```
# Create data frame from the document-term matrix
QsTM = as.data.frame(as.matrix(sparse))
# We have some variable names that start with a number,
# which can cause R some problems. Let's fix this before going
# any further
colnames(QsTM) = make.names(colnames(QsTM))
# This isn't our original dataframe, so we need to bring that column
# with the dependent variable into this new one
QsTM$goodq = questions_html$goodq


# also need to bring over the fields with the body word and title word fields
QsTM$bodyword = questions_html$bodyword

QsTM$titleword = questions_html$titleword



# Bonus: make a cool word cloud!
#wordcloud(corpus, max.words = 200, random.order = FALSE, rot.per = .1,
#          colors = brewer.pal(8, "Dark2"))
```

In the following section I parse out the header, using similar steps to the body cleaning.

```
corpus = Corpus(VectorSource(questions_html$Title))

corpus[[1]]
```

```
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 25
```

```
strwrap(corpus[[1]])
```

```
## [1] "Missing Ribbon in ggplot2"
```

```
# Step 2: Change all the text to lower case.
# tm_map applies an operation to every document in our corpus
# Here, that operation is 'tolower', i.e., 'to lowercase'
corpus = tm_map(corpus, tolower)
```

```
## Warning in tm_map.SimpleCorpus(corpus, tolower): transformation drops
## documents
```

```
# tolower is a function

# Lets check:
strwrap(corpus[[1]])
```

```
## [1] "missing ribbon in ggplot2"
```

```
# Step 3: Remove all punctuation
corpus = tm_map(corpus, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(corpus, removePunctuation): transformation
## drops documents
```

```
# Take a look:
strwrap(corpus[[1]])
```

```
## [1] "missing ribbon in ggplot2"
```

```
# Step 4: Remove stop words
# First, take a look at tm's stopwords:
stopwords("english")[1:10]
```

```
##  [1] "i"         "me"        "my"        "myself"    "we"
##  [6] "our"       "ours"      "ourselves" "you"       "your"
```

```
length(stopwords("english"))
```

```
## [1] 174
```

```
# Just remove stopwords:
# corpus = tm_map(corpus, removeWords, stopwords("english"))
# Remove stopwords and "apple" - this is a word common to all of our tweets
corpus = tm_map(corpus, removeWords,  stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(corpus, removeWords, stopwords("english")):
## transformation drops documents
```

```
# Take a look:
strwrap(corpus[[1]])
```

```
## [1] "missing ribbon ggplot2"
```

```
# Step 5: Stem our document
# Recall, this means chopping off the ends of words that aren't maybe
# as necessary as the rest, like 'ing' and 'ed'
corpus = tm_map(corpus, stemDocument)
```

```
## Warning in tm_map.SimpleCorpus(corpus, stemDocument): transformation drops
## documents
```

```
# Take a look:
strwrap(corpus[[1]])
```

```
## [1] "miss ribbon ggplot2"
```

```
# might try dropping ggplot2(although this could actually be helpful since really bad questions might be mis
labaled and not have ggplot2)
#corpus = tm_map(corpus, removeWords, c("ggplot", "ggplot2"))

frequencies = DocumentTermMatrix(corpus)
# We can get some summary information by looking at this structure
frequencies
```

```
## <<DocumentTermMatrix (documents: 7468, terms: 3351)>>
## Non-/sparse entries: 41782/24983486
## Sparsity           : 100%
## Maximal term length: 53
## Weighting          : term frequency (tf)
```

```
# Words that appear at least 7000 times:
findFreqTerms(frequencies, lowfreq=2000)
```

```
## [1] "ggplot2" "ggplot"  "plot"
```

```
# Words that appear at least 500 times:
findFreqTerms(frequencies, lowfreq=500)
```

```
##  [1] "ggplot2" "ggplot"  "label"   "legend"  "plot"    "use"     "data"
##  [8] "line"    "color"   "bar"
```

```
# Our solution to the possibility of overfitting is to only keep terms
# that appear in x% or more of the tweets. For example:
# 1% of the tweets or more (= 12 or more)
#sparse = removeSparseTerms(frequencies, 0.99)

# 0.5% of the tweets or more (= 6 or more)
#sparse = removeSparseTerms(frequencies, 0.995)
# How many did we keep?
#sparse

# Let's keep it at the 4%
sparse = removeSparseTerms(frequencies, 0.96)
sparse
```

```
## <<DocumentTermMatrix (documents: 7468, terms: 22)>>
## Non-/sparse entries: 15340/148956
## Sparsity           : 91%
## Maximal term length: 7
## Weighting          : term frequency (tf)
```

```
# Step 8: Create data frame from the document-term matrix
QsTM_t = as.data.frame(as.matrix(sparse))
# We have some variable names that start with a number,
# which can cause R some problems. Let's fix this before going
# any further
colnames(QsTM_t) = make.names(colnames(QsTM_t))

# create unique column names so we can keep track of the title words as opposed to the body words
colnames(QsTM_t) <- paste("T", colnames(QsTM_t), sep = "_")
```

Now that we have two matricies describing the body and text (with unique names for the word frequencies), they must be brought together into a single data frame upon which we can train models.

```
# giving both the title matrix and the body matrix id fields so that they can be joined
id <- rownames(QsTM)
QsTM <- cbind(id=id, QsTM)

id <- rownames(QsTM_t)

QsTM_t <- cbind(id=id, QsTM_t)

#joining and dropping the id field so that it does not create problems
QsTM <- QsTM %>%
  left_join(QsTM_t) %>%
  dplyr::select(-id)
```

```
## Joining, by = "id"
```

# Question 1 part b: Splitting and training

First I split the data set into a training and test set, with 70% of the data in the training set and 30% in the test set. The ratio of good to bad questions in each is maintained.

```
split = sample.split(QsTM$goodq, SplitRatio = 0.7)

# what is a split?
q.train <- filter(QsTM, split == TRUE) # is split a variable in loans?
q.test <- filter(QsTM, split == FALSE)
```

```
table(q.train$goodq)
```

```
##
## FALSE  TRUE
##  2654  2574
```

```
accuracy_b = length(q.train$goodq[q.train$goodq == FALSE])/nrow(q.train)
accuracy_b
```

```
## [1] 0.5076511
```

```
table(q.test$goodq)
```

```
##
## FALSE  TRUE
##  1137  1103
```

```
ac_bt = length(q.test$goodq[q.test$goodq == FALSE])/nrow(q.test)
ac_bt
```

```
## [1] 0.5075893
```

Based on the test set tale, slightly more than half of the questions are bad. Therefore, it follows that the baseline assumption is that every question is bad. The accuracy of a baseline model predicting that all questions are bad is 50.7%.

The first model I will train on this data is a logistic model.

```
mod1 <- glm(goodq ~., data=q.train, family="binomial")

summary(mod1)
```

```
##
## Call:
## glm(formula = goodq ~ ., family = "binomial", data = q.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0883  -1.1083  -0.7582   1.1581   1.8674
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.4121310  0.0996590  -4.135 3.54e-05 ***
## code        -0.0738857  0.0302196  -2.445 0.014487 *
## color       -0.0050977  0.0177284  -0.288 0.773697
## data         0.0013808  0.0097852   0.141 0.887785
## geom        -0.0032983  0.0186708  -0.177 0.859780
## get          0.0167970  0.0371356   0.452 0.651042
## ggplot       0.1462531  0.0213903   6.837 8.07e-12 ***
## group        0.0197447  0.0137785   1.433 0.151855
## line        -0.0002994  0.0185223  -0.016 0.987102
## point       -0.0314626  0.0226044  -1.392 0.163959
## set          0.1349536  0.0448907   3.006 0.002645 **
## tri         -0.0382951  0.0351977  -1.088 0.276596
## bar         -0.0138915  0.0238794  -0.582 0.560746
## fill        -0.0112177  0.0217697  -0.515 0.606349
## frame        0.0925132  0.0289215   3.199 0.001380 **
## graph       -0.0194382  0.0286795  -0.678 0.497915
## label        0.0130184  0.0162843   0.799 0.424032
## librari      0.1059406  0.0239500   4.423 9.72e-06 ***
## scale       -0.0214312  0.0204246  -1.049 0.294047
## stat         0.0154743  0.0305104   0.507 0.612028
## text         0.0156650  0.0209280   0.749 0.454150
## use          0.0009442  0.0251900   0.037 0.970098
## variabl     -0.0087481  0.0234411  -0.373 0.709002
## axi         -0.0317236  0.0216459  -1.466 0.142766
## colour      -0.0259392  0.0213852  -1.213 0.225148
## creat       -0.0839301  0.0386629  -2.171 0.029945 *
## differ       0.0308153  0.0405401   0.760 0.447183
## exampl       0.0824210  0.0349308   2.360 0.018297 *
## grid         0.0085982  0.0187227   0.459 0.646061
## legend      -0.0035246  0.0204120  -0.173 0.862910
## like         0.0612949  0.0308534   1.987 0.046961 *
## name        -0.0452534  0.0190792  -2.372 0.017698 *
## plot         0.0057422  0.0104641   0.549 0.583179
## size         0.0279743  0.0199023   1.406 0.159847
## want        -0.0070253  0.0349607  -0.201 0.840738
## can          0.0759238  0.0337785   2.248 0.024596 *
## follow       0.0190178  0.0412605   0.461 0.644857
## list        -0.0168034  0.0218557  -0.769 0.441991
## look        -0.0289835  0.0522738  -0.554 0.579267
## thank       -0.1378562  0.0657779  -2.096 0.036101 *
## element     -0.0101462  0.0231361  -0.439 0.660992
## make         0.0427225  0.0464451   0.920 0.357652
## posit        0.0149334  0.0247548   0.603 0.546340
## theme       -0.0089977  0.0251484  -0.358 0.720507
## valu        -0.0162403  0.0149057  -1.090 0.275916
## work         0.0201300  0.0392101   0.513 0.607679
## factor      -0.0085690  0.0190638  -0.449 0.653076
## function.    0.0100837  0.0215567   0.468 0.639945
## one         -0.0566514  0.0323193  -1.753 0.079625 .
## way         -0.0457479  0.0503612  -0.908 0.363669
## bodyword     0.0002029  0.0001257   1.615 0.106350
## titleword    0.0093621  0.0091926   1.018 0.308467
## T_ggplot2    0.0460583  0.0696569   0.661 0.508473
## T_ggplot    -0.1592461  0.0715572  -2.225 0.026052 *
## T_label      0.1584289  0.1154182   1.373 0.169861
## T_add        0.0075481  0.1358367   0.056 0.955686
```

```
## T_legend     0.1652369  0.1294401   1.277 0.201761
## T_plot      -0.1353374  0.0623188  -2.172 0.029879 *
## T_use       -0.0870853  0.0837475  -1.040 0.298407
## T_error     -0.4655479  0.1318121  -3.532 0.000413 ***
## T_group     -0.3433621  0.1405632  -2.443 0.014575 *
## T_axi       -0.0531910  0.1228957  -0.433 0.665150
## T_chang     -0.1185345  0.1411247  -0.840 0.400949
## T_data      -0.2598153  0.1051783  -2.470 0.013502 *
## T_graph     -0.2044351  0.1358018  -1.505 0.132223
## T_line      -0.0159054  0.1105214  -0.144 0.885569
## T_multipl   -0.2880851  0.1211537  -2.378 0.017414 *
## T_facet      0.0259766  0.1355594   0.192 0.848036
## T_color      0.1294443  0.1223432   1.058 0.290036
## T_variabl   -0.1642393  0.1292307  -1.271 0.203764
## T_valu      -0.1957147  0.1199543  -1.632 0.102769
## T_point      0.3290815  0.1493671   2.203 0.027583 *
## T_differ    -0.1211128  0.1288623  -0.940 0.347288
## T_bar       -0.0089855  0.1144445  -0.079 0.937419
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7246.3  on 5227  degrees of freedom
## Residual deviance: 6949.2  on 5154  degrees of freedom
## AIC: 7097.2
##
## Number of Fisher Scoring iterations: 4
```

```
q_log_b = predict(mod1, newdata=q.test, type="response")
summary(q_log_b)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07178 0.41306 0.48419 0.49513 0.56573 0.93061
```

```
pred_log = as.factor(as.logical(q_log_b >= .5))

t1 = table(q.test$goodq, q_log_b > 0.5)
t1
```

```
##
##         FALSE TRUE
##   FALSE   703  434
##   TRUE    551  552
```

```
table(q.test$goodq)
```

```
##
## FALSE  TRUE
##  1137  1103
```

```
accuracy_isb = (t1[1,1]+t1[2,2])/nrow(q.test)
accuracy_isb
```

```
## [1] 0.5602679
```

The accuracy of this model is better than the baseline. its accuracy is 0.5602679.

The sumary of the model indicates tha many of the features are insignificant; in order to reduce overfitting and improve the model, here we restrict the condidered features to only those significant in the original model.

```
q.train_log <- q.train %>%
  dplyr::select("color", "get", "ggplot", "set", "frame", "librari", "stat", "scale", "differ", "exampl", "
name", "list", "thank", "one", "T_legend", "T_error", "T_group", "goodq")

q.test_log <- q.test %>%
  dplyr::select("color", "get", "ggplot", "set", "frame", "librari", "stat", "scale", "differ", "exampl", "
name", "list", "thank", "one", "T_legend", "T_error", "T_group", "goodq")

mod2 <- glm(goodq ~., data=q.train_log, family="binomial")
summary(mod2)
```

```
##
## Call:
## glm(formula = goodq ~ ., family = "binomial", data = q.train_log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1713  -1.1112  -0.8455   1.1759   1.6787
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.44908    0.05628  -7.979 1.48e-15 ***
## color        0.00236    0.01522   0.155 0.876754
## get          0.01677    0.03493   0.480 0.631203
## ggplot       0.13387    0.01873   7.145 8.97e-13 ***
## set          0.11998    0.04352   2.757 0.005836 **
## frame        0.08665    0.02592   3.343 0.000828 ***
## librari      0.11150    0.02297   4.855 1.20e-06 ***
## stat         0.02360    0.02732   0.864 0.387844
## scale       -0.02103    0.01775  -1.185 0.236091
## differ       0.01463    0.03780   0.387 0.698815
## exampl       0.09087    0.03562   2.551 0.010752 *
## name        -0.04342    0.01803  -2.408 0.016025 *
## list        -0.01447    0.02056  -0.704 0.481675
## thank       -0.14619    0.06439  -2.270 0.023181 *
## one         -0.06341    0.03300  -1.921 0.054683 .
## T_legend     0.18087    0.10635   1.701 0.088988 .
## T_error     -0.46257    0.12632  -3.662 0.000251 ***
## T_group     -0.30468    0.13300  -2.291 0.021974 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7246.3  on 5227  degrees of freedom
## Residual deviance: 7041.1  on 5210  degrees of freedom
## AIC: 7077.1
##
## Number of Fisher Scoring iterations: 4
```

```
q_log_b2 = predict(mod2, newdata=q.test_log, type="response")
summary(q_log_b2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.08904 0.42594 0.48027 0.49463 0.54843 0.92072
```

```
t1 = table(q.test$goodq, q_log_b2 > 0.5)
t1
```

```
##
##         FALSE TRUE
##   FALSE   738  399
##   TRUE    579  524
```

```
table(q.test_log$goodq)
```

```
##
## FALSE  TRUE
##  1137  1103
```

```
accuracy_isb2 = (t1[1,1]+t1[2,2])/nrow(q.test)
accuracy_isb2
```

```
## [1] 0.5633929
```

The accuracy of the trimmed logistic model is actually very  slightly better: 0.5633929. From here we proceed to other types of models.

Next we explore the ability of a CART model to more accurately predict good questions, using k-fold cross validation to determine the ideal cp value.

```
modCART <- rpart(goodq ~.,
            data = q.train, method="class",
            minbucket=5, cp = 0.001)
modCART
```

```
## n= 5228
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##        1) root 5228 2574 FALSE (0.50765111 0.49234889)
##          2) librari< 0.5 3317 1457 FALSE (0.56074766 0.43925234)
##            4) ggplot< 2.5 2511 1037 FALSE (0.58701712 0.41298288)
##              8) exampl< 0.5 2028  792 FALSE (0.60946746 0.39053254)
##               16) frame< 0.5 1151  415 FALSE (0.63944396 0.36055604)
##                 32) grid< 2.5 1121  396 FALSE (0.64674398 0.35325602)
##                   64) can< 1.5 1028  350 FALSE (0.65953307 0.34046693)
##                    128) bodyword< 168.5 830  266 FALSE (0.67951807 0.32048193) *
##                    129) bodyword>=168.5 198   84 FALSE (0.57575758 0.42424242)
##                      258) data>=0.5 170   65 FALSE (0.61764706 0.38235294)
##                        516) posit< 0.5 121   39 FALSE (0.67768595 0.32231405) *
##                        517) posit>=0.5 49   23 TRUE (0.46938776 0.53061224)
##                         1034) T_use>=0.5 7    0 FALSE (1.00000000 0.00000000) *
##                         1035) T_use< 0.5 42   16 TRUE (0.38095238 0.61904762)
##                           2070) get>=0.5 18    6 FALSE (0.66666667 0.33333333) *
##                           2071) get< 0.5 24    4 TRUE (0.16666667 0.83333333) *
##                      259) data< 0.5 28    9 TRUE (0.32142857 0.67857143)
##                        518) group>=0.5 5    1 FALSE (0.80000000 0.20000000) *
##                        519) group< 0.5 23    5 TRUE (0.21739130 0.78260870) *
##                   65) can>=1.5 93   46 FALSE (0.50537634 0.49462366)
##                    130) colour< 0.5 69   29 FALSE (0.57971014 0.42028986)
##                      260) graph< 1.5 62   23 FALSE (0.62903226 0.37096774) *
##                      261) graph>=1.5 7    1 TRUE (0.14285714 0.85714286) *
##                    131) colour>=0.5 24    7 TRUE (0.29166667 0.70833333) *
##                 33) grid>=2.5 30   11 TRUE (0.36666667 0.63333333)
##                   66) T_use< 0.5 21   10 FALSE (0.52380952 0.47619048)
##                    132) size>=0.5 14    4 FALSE (0.71428571 0.28571429) *
##                    133) size< 0.5 7    1 TRUE (0.14285714 0.85714286) *
##                   67) T_use>=0.5 9    0 TRUE (0.00000000 1.00000000) *
##               17) frame>=0.5 877  377 FALSE (0.57012543 0.42987457)
##                 34) axi>=6.5 13    1 FALSE (0.92307692 0.07692308) *
##                 35) axi< 6.5 864  376 FALSE (0.56481481 0.43518519)
##                   70) T_axi< 0.5 820  348 FALSE (0.57560976 0.42439024)
##                    140) fill>=4.5 8    0 FALSE (1.00000000 0.00000000) *
##                    141) fill< 4.5 812  348 FALSE (0.57142857 0.42857143)
##                      282) name>=4.5 40   10 FALSE (0.75000000 0.25000000) *
##                      283) name< 4.5 772  338 FALSE (0.56217617 0.43782383)
##                        566) factor< 3.5 743  318 FALSE (0.57200538 0.42799462)
##                         1132) tri>=2.5 14    1 FALSE (0.92857143 0.07142857) *
##                         1133) tri< 2.5 729  317 FALSE (0.56515775 0.43484225)
##                           2266) work>=1.5 29    7 FALSE (0.75862069 0.24137931) *
##                           2267) work< 1.5 700  310 FALSE (0.55714286 0.44285714)
##                             4534) geom>=4.5 47   14 FALSE (0.70212766 0.29787234) *
##                             4535) geom< 4.5 653  296 FALSE (0.54670750 0.45329250)
##                               9070) color< 1.5 547  237 FALSE (0.56672761 0.43327239)
##                                 18140) titleword>=4.5 519  219 FALSE (0.57803468 0.42196532)
```

```
##                                 36280) legend< 3.5 507  210 FALSE (0.58579882 0.41420118)
##                                   72560) T_differ>=0.5 24    5 FALSE (0.79166667 0.20833333) *
##                                   72561) T_differ< 0.5 483  205 FALSE (0.57556936 0.42443064)
##                                    145122) T_add>=0.5 27    6 FALSE (0.77777778 0.22222222) *
##                                    145123) T_add< 0.5 456  199 FALSE (0.56359649 0.43640351)
##                                      290246) posit< 0.5 355  144 FALSE (0.59436620 0.40563380)
##                                        580492) bar< 3.5 350  139 FALSE (0.60285714 0.39714286)
##                                          1160984) set< 0.5 294  108 FALSE (0.63265306 0.36734694) *
##                                          1160985) set>=0.5 56   25 TRUE (0.44642857 0.55357143)
##                                            2321970) T_multipl>=0.5 7    1 FALSE (0.85714286 0.14285714)
*
##                                            2321971) T_multipl< 0.5 49   19 TRUE (0.38775510 0.61224490)

##                                              4643942) titleword>=12.5 12    4 FALSE (0.66666667 0.333333
33) *
##                                              4643943) titleword< 12.5 37   11 TRUE (0.29729730 0.7027027
0) *
##                                        580493) bar>=3.5 5    0 TRUE (0.00000000 1.00000000) *
##                                      290247) posit>=0.5 101   46 TRUE (0.45544554 0.54455446)
##                                        580494) T_bar>=1.5 5    0 FALSE (1.00000000 0.00000000) *
##                                        580495) T_bar< 1.5 96   41 TRUE (0.42708333 0.57291667)
##                                          1160990) text>=4.5 8    1 FALSE (0.87500000 0.12500000) *
##                                          1160991) text< 4.5 88   34 TRUE (0.38636364 0.61363636)
##                                            2321982) stat>=1.5 12    4 FALSE (0.66666667 0.33333333) *
##                                            2321983) stat< 1.5 76   26 TRUE (0.34210526 0.65789474)
##                                              4643966) ggplot< 1.5 54   23 TRUE (0.42592593 0.57407407)
##                                                9287932) T_ggplot2>=0.5 16    5 FALSE (0.68750000 0.31250
000) *
##                                                9287933) T_ggplot2< 0.5 38   12 TRUE (0.31578947 0.684210
53)
##                                                 18575866) titleword< 6.5 10    3 FALSE (0.70000000 0.300
00000) *
##                                                 18575867) titleword>=6.5 28    5 TRUE (0.17857143 0.8214
2857) *
##                                              4643967) ggplot>=1.5 22    3 TRUE (0.13636364 0.86363636) *
##                                   36281) legend>=3.5 12    3 TRUE (0.25000000 0.75000000) *
##                                 18141) titleword< 4.5 28   10 TRUE (0.35714286 0.64285714)
##                                   36282) bar>=1.5 5    1 FALSE (0.80000000 0.20000000) *
##                                   36283) bar< 1.5 23    6 TRUE (0.26086957 0.73913043) *
##                                9071) color>=1.5 106   47 TRUE (0.44339623 0.55660377)
##                                 18142) data>=8.5 9    1 FALSE (0.88888889 0.11111111) *
##                                 18143) data< 8.5 97   39 TRUE (0.40206186 0.59793814)
##                                   36286) group>=1.5 18    6 FALSE (0.66666667 0.33333333)
##                                     72572) label< 0.5 11    1 FALSE (0.90909091 0.09090909) *
##                                     72573) label>=0.5 7    2 TRUE (0.28571429 0.71428571) *
##                                   36287) group< 1.5 79   27 TRUE (0.34177215 0.65822785)
##                                     72574) element>=4.5 7    2 FALSE (0.71428571 0.28571429) *
##                                     72575) element< 4.5 72   22 TRUE (0.30555556 0.69444444) *
##                           567) factor>=3.5 29    9 TRUE (0.31034483 0.68965517)
##                            1134) label>=5.5 6    1 FALSE (0.83333333 0.16666667) *
##                            1135) label< 5.5 23    4 TRUE (0.17391304 0.82608696) *
##                       71) T_axi>=0.5 44   16 TRUE (0.36363636 0.63636364)
##                        142) bodyword< 105.5 16    6 FALSE (0.62500000 0.37500000) *
##                        143) bodyword>=105.5 28    6 TRUE (0.21428571 0.78571429) *
##                   9) exampl>=0.5 483  238 TRUE (0.49275362 0.50724638)
##                    18) bodyword< 132.5 224   99 FALSE (0.55803571 0.44196429)
##                      36) bodyword>=43 213   90 FALSE (0.57746479 0.42253521)
##                        72) work< 1.5 208   85 FALSE (0.59134615 0.40865385)
##                         144) T_legend< 0.5 197   77 FALSE (0.60913706 0.39086294)
##                           288) can< 0.5 128   42 FALSE (0.67187500 0.32812500)
##                             576) data< 3.5 111   32 FALSE (0.71171171 0.28828829)
##                              1152) line< 0.5 79   17 FALSE (0.78481013 0.21518987) *
##                              1153) line>=0.5 32   15 FALSE (0.53125000 0.46875000)
##                                2306) want< 0.5 19    5 FALSE (0.73684211 0.26315789) *
##                                2307) want>=0.5 13    3 TRUE (0.23076923 0.76923077) *
##                             577) data>=3.5 17    7 TRUE (0.41176471 0.58823529)
##                              1154) data>=4.5 9    3 FALSE (0.66666667 0.33333333) *
##                              1155) data< 4.5 8    1 TRUE (0.12500000 0.87500000) *
##                           289) can>=0.5 69   34 TRUE (0.49275362 0.50724638)
##                             578) axi< 1.5 61   28 FALSE (0.54098361 0.45901639)
##                              1156) T_multipl>=0.5 6    0 FALSE (1.00000000 0.00000000) *
##                              1157) T_multipl< 0.5 55   27 TRUE (0.49090909 0.50909091)
##                                2314) can>=1.5 14    4 FALSE (0.71428571 0.28571429) *
```

```
##                          2315) can< 1.5 41   17 TRUE (0.41463415 0.58536585) *
##                        579) axi>=1.5 8    1 TRUE (0.12500000 0.87500000) *
##                      145) T_legend>=0.5 11    3 TRUE (0.27272727 0.72727273) *
##                     73) work>=1.5 5    0 TRUE (0.00000000 1.00000000) *
##                   37) bodyword< 43 11    2 TRUE (0.18181818 0.81818182) *
##                 19) bodyword>=132.5 259  113 TRUE (0.43629344 0.56370656)
##                   38) creat>=2.5 5    0 FALSE (1.00000000 0.00000000) *
##                   39) creat< 2.5 254  108 TRUE (0.42519685 0.57480315)
##                     78) titleword>=19.5 5    0 FALSE (1.00000000 0.00000000) *
##                     79) titleword< 19.5 249  103 TRUE (0.41365462 0.58634538)
##                      158) titleword< 7.5 89   42 FALSE (0.52808989 0.47191011)
##                        316) way>=0.5 31    9 FALSE (0.70967742 0.29032258) *
##                        317) way< 0.5 58   25 TRUE (0.43103448 0.56896552)
##                          634) fill>=2.5 14    4 FALSE (0.71428571 0.28571429) *
##                          635) fill< 2.5 44   15 TRUE (0.34090909 0.65909091)
##                           1270) legend>=0.5 7    2 FALSE (0.71428571 0.28571429) *
##                           1271) legend< 0.5 37   10 TRUE (0.27027027 0.72972973) *
##                      159) titleword>=7.5 160   56 TRUE (0.35000000 0.65000000)
##                        318) thank>=0.5 50   24 TRUE (0.48000000 0.52000000)
##                          636) get< 0.5 32   12 FALSE (0.62500000 0.37500000)
##                           1272) bodyword>=156.5 24    5 FALSE (0.79166667 0.20833333) *
##                           1273) bodyword< 156.5 8    1 TRUE (0.12500000 0.87500000) *
##                          637) get>=0.5 18    4 TRUE (0.22222222 0.77777778) *
##                        319) thank< 0.5 110   32 TRUE (0.29090909 0.70909091)
##                          638) tri>=1.5 9    3 FALSE (0.66666667 0.33333333) *
##                          639) tri< 1.5 101   26 TRUE (0.25742574 0.74257426) *
##               5) ggplot>=2.5 806  386 TRUE (0.47890819 0.52109181)
##                10) frame< 0.5 365  167 FALSE (0.54246575 0.45753425)
##                  20) name>=0.5 78   24 FALSE (0.69230769 0.30769231)
##                    40) plot< 8.5 72   19 FALSE (0.73611111 0.26388889)
##                      80) bodyword>=99.5 64   13 FALSE (0.79687500 0.20312500) *
##                      81) bodyword< 99.5 8    2 TRUE (0.25000000 0.75000000) *
##                    41) plot>=8.5 6    1 TRUE (0.16666667 0.83333333) *
##                  21) name< 0.5 287  143 FALSE (0.50174216 0.49825784)
##                    42) like< 0.5 147   62 FALSE (0.57823129 0.42176871)
##                      84) make< 1.5 138   54 FALSE (0.60869565 0.39130435)
##                      168) work< 1.5 124   44 FALSE (0.64516129 0.35483871)
##                        336) axi< 1.5 95   28 FALSE (0.70526316 0.29473684)
##                          672) label< 0.5 74   17 FALSE (0.77027027 0.22972973) *
##                          673) label>=0.5 21   10 TRUE (0.47619048 0.52380952)
##                           1346) label>=3.5 8    1 FALSE (0.87500000 0.12500000) *
##                           1347) label< 3.5 13    3 TRUE (0.23076923 0.76923077) *
##                        337) axi>=1.5 29   13 TRUE (0.44827586 0.55172414)
##                          674) element>=7.5 9    2 FALSE (0.77777778 0.22222222) *
##                          675) element< 7.5 20    6 TRUE (0.30000000 0.70000000) *
##                      169) work>=1.5 14    4 TRUE (0.28571429 0.71428571) *
##                      85) make>=1.5 9    1 TRUE (0.11111111 0.88888889) *
##                    43) like>=0.5 140   59 TRUE (0.42142857 0.57857143)
##                      86) axi>=0.5 48   20 FALSE (0.58333333 0.41666667)
##                      172) geom>=3.5 15    2 FALSE (0.86666667 0.13333333) *
##                      173) geom< 3.5 33   15 TRUE (0.45454545 0.54545455)
##                        346) want>=0.5 15    5 FALSE (0.66666667 0.33333333) *
##                        347) want< 0.5 18    5 TRUE (0.27777778 0.72222222) *
##                      87) axi< 0.5 92   31 TRUE (0.33695652 0.66304348)
##                      174) set>=0.5 21    8 FALSE (0.61904762 0.38095238) *
##                      175) set< 0.5 71   18 TRUE (0.25352113 0.74647887) *
##                11) frame>=0.5 441  188 TRUE (0.42630385 0.57369615)
##                  22) titleword>=14.5 42   15 FALSE (0.64285714 0.35714286)
##                    44) use< 2.5 33    8 FALSE (0.75757576 0.24242424) *
##                    45) use>=2.5 9    2 TRUE (0.22222222 0.77777778) *
##                  23) titleword< 14.5 399  161 TRUE (0.40350877 0.59649123)
##                    46) group< 4.5 373  158 TRUE (0.42359249 0.57640751)
##                      92) name>=1.5 101   48 FALSE (0.52475248 0.47524752)
##                      184) posit>=1.5 19    4 FALSE (0.78947368 0.21052632) *
##                      185) posit< 1.5 82   38 TRUE (0.46341463 0.53658537)
##                        370) bodyword< 166.5 20    5 FALSE (0.75000000 0.25000000)
##                          740) work< 0.5 15    1 FALSE (0.93333333 0.06666667) *
##                          741) work>=0.5 5    1 TRUE (0.20000000 0.80000000) *
##                        371) bodyword>=166.5 62   23 TRUE (0.37096774 0.62903226)
##                          742) grid>=0.5 17    7 FALSE (0.58823529 0.41176471) *
##                          743) grid< 0.5 45   13 TRUE (0.28888889 0.71111111) *
##                      93) name< 1.5 272  105 TRUE (0.38602941 0.61397059)
```

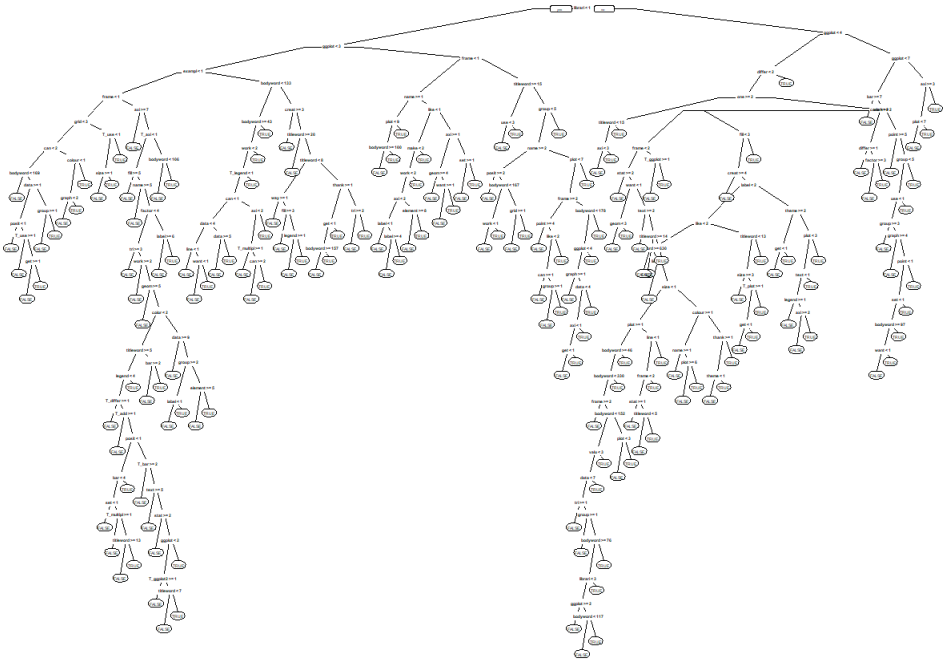
```
##                   186) plot< 6.5 247  101 TRUE (0.40890688 0.59109312)
##                     372) frame>=1.5 92   46 FALSE (0.50000000 0.50000000)
##                       744) point>=3.5 10    1 FALSE (0.90000000 0.10000000) *
##                       745) point< 3.5 82   37 TRUE (0.45121951 0.54878049)
##                         1490) like< 1.5 69   34 FALSE (0.50724638 0.49275362)
##                           2980) can>=0.5 27    8 FALSE (0.70370370 0.29629630) *
##                           2981) can< 0.5 42   16 TRUE (0.38095238 0.61904762)
##                             5962) group>=0.5 7    1 FALSE (0.85714286 0.14285714) *
##                             5963) group< 0.5 35   10 TRUE (0.28571429 0.71428571) *
##                         1491) like>=1.5 13    2 TRUE (0.15384615 0.84615385) *
##                     373) frame< 1.5 155   55 TRUE (0.35483871 0.64516129)
##                       746) bodyword< 177.5 84   37 TRUE (0.44047619 0.55952381)
##                         1492) ggplot< 3.5 57   27 FALSE (0.52631579 0.47368421)
##                           2984) graph>=0.5 14    3 FALSE (0.78571429 0.21428571) *
##                           2985) graph< 0.5 43   19 TRUE (0.44186047 0.55813953)
##                             5970) data< 3.5 34   16 FALSE (0.52941176 0.47058824)
##                               11940) axi< 0.5 23    8 FALSE (0.65217391 0.34782609)
##                                 23880) get< 0.5 18    4 FALSE (0.77777778 0.22222222) *
##                                 23881) get>=0.5 5    1 TRUE (0.20000000 0.80000000) *
##                               11941) axi>=0.5 11    3 TRUE (0.27272727 0.72727273) *
##                             5971) data>=3.5 9    1 TRUE (0.11111111 0.88888889) *
##                         1493) ggplot>=3.5 27    7 TRUE (0.25925926 0.74074074) *
##                       747) bodyword>=177.5 71   18 TRUE (0.25352113 0.74647887) *
##                     187) plot>=6.5 25    4 TRUE (0.16000000 0.84000000) *
##                   47) group>=4.5 26    3 TRUE (0.11538462 0.88461538) *
##               3) librari>=0.5 1911  794 TRUE (0.41548927 0.58451073)
##                 6) ggplot< 3.5 1309  590 TRUE (0.45072574 0.54927426)
##                  12) differ< 1.5 1235  569 TRUE (0.46072874 0.53927126)
##                    24) one>=1.5 76   28 FALSE (0.63157895 0.36842105)
##                      48) titleword< 14.5 71   23 FALSE (0.67605634 0.32394366)
##                        96) axi< 2.5 66   19 FALSE (0.71212121 0.28787879) *
##                        97) axi>=2.5 5    1 TRUE (0.20000000 0.80000000) *
##                      49) titleword>=14.5 5    0 TRUE (0.00000000 1.00000000) *
##                    25) one< 1.5 1159  521 TRUE (0.44952545 0.55047455)
##                      50) code>=1.5 137   58 FALSE (0.57664234 0.42335766)
##                       100) frame< 1.5 104   38 FALSE (0.63461538 0.36538462)
##                         200) stat>=1.5 9    0 FALSE (1.00000000 0.00000000) *
##                         201) stat< 1.5 95   38 FALSE (0.60000000 0.40000000)
##                           402) want< 0.5 54   16 FALSE (0.70370370 0.29629630)
##                             804) geom< 2.5 43    9 FALSE (0.79069767 0.20930233) *
##                             805) geom>=2.5 11    4 TRUE (0.36363636 0.63636364) *
##                           403) want>=0.5 41   19 TRUE (0.46341463 0.53658537)
##                             806) text>=1.5 15    3 FALSE (0.80000000 0.20000000) *
##                             807) text< 1.5 26    7 TRUE (0.26923077 0.73076923)
##                              1614) titleword>=13.5 7    2 FALSE (0.71428571 0.28571429) *
##                              1615) titleword< 13.5 19    2 TRUE (0.10526316 0.89473684) *
##                       101) frame>=1.5 33   13 TRUE (0.39393939 0.60606061)
##                         202) T_ggplot>=0.5 8    2 FALSE (0.75000000 0.25000000) *
##                         203) T_ggplot< 0.5 25    7 TRUE (0.28000000 0.72000000) *
##                     51) code< 1.5 1022  442 TRUE (0.43248532 0.56751468)
##                       102) fill< 2.5 923  413 TRUE (0.44745395 0.55254605)
##                         204) creat>=3.5 8    0 FALSE (1.00000000 0.00000000) *
##                         205) creat< 3.5 915  405 TRUE (0.44262295 0.55737705)
##                           410) label< 1.5 748  350 TRUE (0.46791444 0.53208556)
##                             820) like< 1.5 638  312 TRUE (0.48902821 0.51097179)
##                              1640) bodyword>=629.5 10    1 FALSE (0.90000000 0.10000000) *
##                              1641) bodyword< 629.5 628  303 TRUE (0.48248408 0.51751592)
##                                3282) list>=7 6    0 FALSE (1.00000000 0.00000000) *
##                                3283) list< 7 622  297 TRUE (0.47749196 0.52250804)
##                                  6566) size< 0.5 453  224 FALSE (0.50551876 0.49448124)
##                                   13132) plot>=0.5 319  146 FALSE (0.54231975 0.45768025)
##                                     26264) bodyword>=46 310  137 FALSE (0.55806452 0.44193548)
##                                       52528) bodyword< 329.5 300  129 FALSE (0.57000000 0.43000000)
##                                        105056) frame>=1.5 50   13 FALSE (0.74000000 0.26000000) *
##                                        105057) frame< 1.5 250  116 FALSE (0.53600000 0.46400000)
##                                          210114) bodyword< 151.5 187   79 FALSE (0.57754011 0.42245989)
##                                            420228) valu< 2.5 173   68 FALSE (0.60693642 0.39306358)
##                                              840456) data< 6.5 166   62 FALSE (0.62650602 0.37349398)
##                                               1680912) tri>=0.5 68   19 FALSE (0.72058824 0.27941176) *
##                                               1680913) tri< 0.5 98   43 FALSE (0.56122449 0.43877551)
##                                                 3361826) group>=0.5 15    3 FALSE (0.80000000 0.20000000) *
##                                                 3361827) group< 0.5 83   40 FALSE (0.51807229 0.48192771)
##                                                   6723654) bodyword>=75.5 56   22 FALSE (0.60714286 0.3928571
```

```
4)
##                                           13447308) librari< 2.5 50   17 FALSE (0.66000000 0.3400000
0)
##                                             26894616) ggplot>=1.5 34    8 FALSE (0.76470588 0.235294
12) *
##                                             26894617) ggplot< 1.5 16    7 TRUE (0.43750000 0.5625000
0)
##                                               53789234) bodyword< 116.5 8    2 FALSE (0.75000000 0.
25000000) *
##                                               53789235) bodyword>=116.5 8    1 TRUE (0.12500000 0.8
7500000) *
##                                           13447309) librari>=2.5 6    1 TRUE (0.16666667 0.83333333)
*
##                                         6723655) bodyword< 75.5 27    9 TRUE (0.33333333 0.66666667
) *
##                                       840457) data>=6.5 7    1 TRUE (0.14285714 0.85714286) *
##                                     420229) valu>=2.5 14    3 TRUE (0.21428571 0.78571429) *
##                                   210115) bodyword>=151.5 63   26 TRUE (0.41269841 0.58730159)
##                                     420230) plot< 2.5 35   14 FALSE (0.60000000 0.40000000) *
##                                     420231) plot>=2.5 28    5 TRUE (0.17857143 0.82142857) *
##                                  52529) bodyword>=329.5 10    2 TRUE (0.20000000 0.80000000) *
##                                26265) bodyword< 46 9    0 TRUE (0.00000000 1.00000000) *
##                              13133) plot< 0.5 134   56 TRUE (0.41791045 0.58208955)
##                                26266) line< 0.5 103   49 TRUE (0.47572816 0.52427184)
##                                  52532) frame< 1.5 93   45 FALSE (0.51612903 0.48387097)
##                                   105064) stat>=0.5 23    6 FALSE (0.73913043 0.26086957) *
##                                   105065) stat< 0.5 70   31 TRUE (0.44285714 0.55714286)
##                                     210130) titleword< 4.5 8    1 FALSE (0.87500000 0.12500000) *
##                                     210131) titleword>=4.5 62   24 TRUE (0.38709677 0.61290323) *
##                                  52533) frame>=1.5 10    1 TRUE (0.10000000 0.90000000) *
##                                26267) line>=0.5 31    7 TRUE (0.22580645 0.77419355) *
##                             6567) size>=0.5 169   68 TRUE (0.40236686 0.59763314)
##                              13134) colour>=0.5 71   35 FALSE (0.50704225 0.49295775)
##                                26268) name>=0.5 22    5 FALSE (0.77272727 0.22727273) *
##                                26269) name< 0.5 49   19 TRUE (0.38775510 0.61224490)
##                                  52538) plot>=5.5 6    1 FALSE (0.83333333 0.16666667) *
##                                  52539) plot< 5.5 43   14 TRUE (0.32558140 0.67441860) *
##                              13135) colour< 0.5 98   32 TRUE (0.32653061 0.67346939)
##                                26270) thank>=0.5 22    9 FALSE (0.59090909 0.40909091)
##                                  52540) theme< 0.5 13    3 FALSE (0.76923077 0.23076923) *
##                                  52541) theme>=0.5 9    3 TRUE (0.33333333 0.66666667) *
##                                26271) thank< 0.5 76   19 TRUE (0.25000000 0.75000000) *
##                           821) like>=1.5 110   38 TRUE (0.34545455 0.65454545)
##                            1642) titleword< 12.5 92   36 TRUE (0.39130435 0.60869565)
##                              3284) size>=2.5 6    0 FALSE (1.00000000 0.00000000) *
##                              3285) size< 2.5 86   30 TRUE (0.34883721 0.65116279)
##                                6570) T_plot>=0.5 28   13 FALSE (0.53571429 0.46428571)
##                                 13140) get< 0.5 20    7 FALSE (0.65000000 0.35000000) *
##                                 13141) get>=0.5 8    2 TRUE (0.25000000 0.75000000) *
##                                6571) T_plot< 0.5 58   15 TRUE (0.25862069 0.74137931) *
##                            1643) titleword>=12.5 18    2 TRUE (0.11111111 0.88888889) *
##                         411) label>=1.5 167   55 TRUE (0.32934132 0.67065868)
##                           822) theme>=1.5 35   17 FALSE (0.51428571 0.48571429)
##                            1644) get< 0.5 24    7 FALSE (0.70833333 0.29166667) *
##                            1645) get>=0.5 11    1 TRUE (0.09090909 0.90909091) *
##                           823) theme< 1.5 132   37 TRUE (0.28030303 0.71969697)
##                            1646) plot< 2.5 104   35 TRUE (0.33653846 0.66346154)
##                              3292) text< 0.5 48   22 TRUE (0.45833333 0.54166667)
##                                6584) legend>=0.5 8    0 FALSE (1.00000000 0.00000000) *
##                                6585) legend< 0.5 40   14 TRUE (0.35000000 0.65000000)
##                                 13170) axi>=1.5 9    3 FALSE (0.66666667 0.33333333) *
##                                 13171) axi< 1.5 31    8 TRUE (0.25806452 0.74193548) *
##                              3293) text>=0.5 56   13 TRUE (0.23214286 0.76785714) *
##                            1647) plot>=2.5 28    2 TRUE (0.07142857 0.92857143) *
##                     103) fill>=2.5 99   29 TRUE (0.29292929 0.70707071) *
##                  13) differ>=1.5 74   21 TRUE (0.28378378 0.71621622) *
##               7) ggplot>=3.5 602  204 TRUE (0.33887043 0.66112957)
##                14) ggplot< 6.5 484  177 TRUE (0.36570248 0.63429752)
##                  28) bar>=6.5 12    3 FALSE (0.75000000 0.25000000) *
##                  29) bar< 6.5 472  168 TRUE (0.35593220 0.64406780)
##                    58) work>=1.5 59   29 FALSE (0.50847458 0.49152542)
##                     116) differ>=0.5 14    2 FALSE (0.85714286 0.14285714) *
##                     117) differ< 0.5 45   18 TRUE (0.40000000 0.60000000)
```

```
##                     234) factor>=2.5 7    1 FALSE (0.85714286 0.14285714) *
##                     235) factor< 2.5 38   12 TRUE (0.31578947 0.68421053) *
##                   59) work< 1.5 413  138 TRUE (0.33414044 0.66585956)
##                    118) point>=4.5 21    8 FALSE (0.61904762 0.38095238) *
##                    119) point< 4.5 392  125 TRUE (0.31887755 0.68112245)
##                      238) group< 4.5 368  124 TRUE (0.33695652 0.66304348)
##                        476) use< 0.5 122   53 TRUE (0.43442623 0.56557377)
##                          952) group>=2.5 9    0 FALSE (1.00000000 0.00000000) *
##                          953) group< 2.5 113   44 TRUE (0.38938053 0.61061947)
##                           1906) graph>=3.5 5    0 FALSE (1.00000000 0.00000000) *
##                           1907) graph< 3.5 108   39 TRUE (0.36111111 0.63888889)
##                             3814) point< 0.5 75   34 TRUE (0.45333333 0.54666667)
##                               7628) set< 0.5 58   27 FALSE (0.53448276 0.46551724)
##                                 15256) bodyword>=97 48   19 FALSE (0.60416667 0.39583333)
##                                   30512) want< 0.5 33    9 FALSE (0.72727273 0.27272727) *
##                                   30513) want>=0.5 15    5 TRUE (0.33333333 0.66666667) *
##                                 15257) bodyword< 97 10    2 TRUE (0.20000000 0.80000000) *
##                               7629) set>=0.5 17    3 TRUE (0.17647059 0.82352941) *
##                             3815) point>=0.5 33    5 TRUE (0.15151515 0.84848485) *
##                        477) use>=0.5 246   71 TRUE (0.28861789 0.71138211) *
##                      239) group>=4.5 24    1 TRUE (0.04166667 0.95833333) *
##                15) ggplot>=6.5 118   27 TRUE (0.22881356 0.77118644)
##                  30) axi>=2.5 34   15 TRUE (0.44117647 0.55882353)
##                    60) plot< 6.5 21    7 FALSE (0.66666667 0.33333333) *
##                    61) plot>=6.5 13    1 TRUE (0.07692308 0.92307692) *
##                  31) axi< 2.5 84   12 TRUE (0.14285714 0.85714286) *
```

```
prp(modCART)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



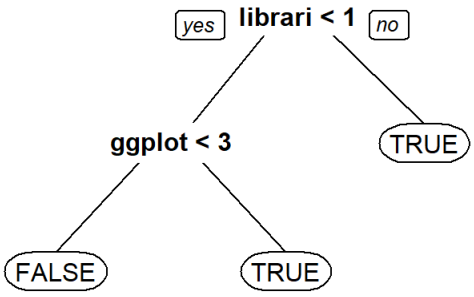
```
cpVals = data.frame(cp = seq(0, .04, by=.001))

set.seed(123)
train.cart <- train(goodq ~.,
                    data = q.train,
                    method = "rpart",
                    tuneGrid = cpVals,
                    trControl = trainControl(method = "cv", number=5),
                    metric = "Accuracy")

train.cart$bestTune
```

```
##       cp
## 14 0.013
```

```
mod123 = train.cart$finalModel
prp(mod123, digits=3)
```




```
q.test.mm = as.data.frame(model.matrix(goodq~.+0, data=q.test))
pred_cart = predict(mod123, newdata=q.test.mm, type="class")
tcart = table(q.test$goodq, pred_cart)

accuracy_isb_cart = (tcart[1,1]+tcart[2,2])/nrow(q.test)
accuracy_isb_cart
```

```
## [1] 0.5589286
```

Next I try a random forest model, using the values determined in the following cross validation analysis.

```
set.seed(144)
mod.q.rf <- randomForest(goodq ~ ., data = q.train, mtry = 2, nodesize = 5, ntree = 900)

pred.q.rf <- predict(mod.q.rf, newdata = q.test)

t_rf = table(q.test$goodq, pred.q.rf)
t_rf
```

```
##        pred.q.rf
##         FALSE TRUE
##   FALSE   742  395
##   TRUE    564  539
```

```
accuracy_isb_rf = (t_rf[1,1]+t_rf[2,2])/nrow(q.test)
accuracy_isb_rf
```

```
## [1] 0.571875
```

The accuracy of the random forest model is significantly higher than any of the preceding models; 0.571875.

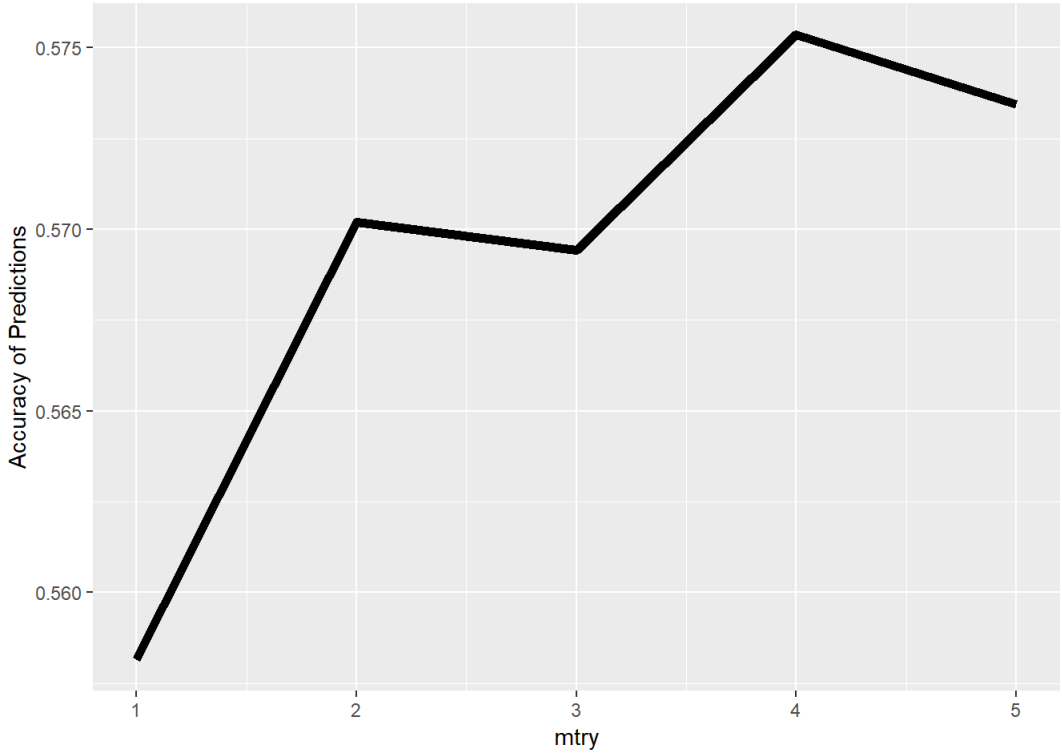
For processing time reasons, the actual cross calidation section is here commented out.

```
set.seed(144)
train.rf = train(goodq~., data = q.train, method = "rf", tuneGrid = data.frame(mtry=seq(1, 5, 1)), trControl
= trainControl(method = "cv", number = 5), metric = "Accuracy")

best.rf = train.rf$finalModel
best.rf
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 42.65%
## Confusion matrix:
##       FALSE TRUE class.error
## FALSE  1612 1042   0.3926149
## TRUE   1188 1386   0.4615385
```

```
rf.plot <- ggplot(train.rf$results, aes(x=mtry, y=Accuracy)) + geom_line(lwd=2) +
  ylab("Accuracy of Predictions")
rf.plot
```




```
q.test.mm = as.data.frame(model.matrix(goodq ~. +0, data = q.test))

set.seed(144)
pred.best.rf = predict(best.rf, newdata = q.test.mm, type = "class")

t_rf_all = table(q.test$goodq, pred.best.rf)
t_rf_all
```

```
##        pred.best.rf
##         FALSE TRUE
##   FALSE   681  456
##   TRUE    511  592
```

```
accuracy.rf = (t_rf_all[1,1]+t_rf_all[2,2])/nrow(q.test)
accuracy.rf
```

```
## [1] 0.5683036
```

Next we try a boosted model.

```
set.seed(144)
mod.boost <- gbm(goodq ~ .,
                 data = q.train,
                 distribution = "multinomial",
                 n.trees = 8000,
                 interaction.depth = 1)

set.seed(144)
pred.boost <- predict(mod.boost, newdata = q.test, n.trees=8000, type = "response")

pred_fixed = apply(pred.boost, 1, which.max)
pred_boost = factor(pred_fixed, levels = c(1,2), labels = c(FALSE, TRUE))

t_rf_all = table(q.test$goodq, pred_boost)
t_rf_all
```

```
##        pred_boost
##         FALSE TRUE
##   FALSE   668  469
##   TRUE    524  579
```

```
accuracy.boost = (t_rf_all[1,1]+t_rf_all[2,2])/nrow(q.test)
accuracy.boost
```

```
## [1] 0.5566964
```

The accuracy of the boosted model is 0.5566964. The baseline model is least accurate at 0.5076511. Second most accurate is the logistic model with an accuracy of 0.5602679. The CART and boosted models have similar accuracies of 0.5589286 and 0.5566964. The model with the highest accuracy is the random forest model which has an accuracy of 0.57.

Because the random forest model has the highest accuracy, I will apply bootstrapping to this model in order to identify the confidence interval of the boosted model.

Boostraping on random forest model:

```
mean_squared_error <- function(data, index) {
  responses <- data$response[index]
  predictions <- data$prediction[index]
  MSE <- mean((responses - predictions)^2)
  return(MSE)
}

mean_absolute_error <- function(data, index) {
  responses <- data$response[index]
  predictions <- data$prediction[index]
  MAE <- mean(abs(responses - predictions))
  return(MAE)
}

OS_R_squared <- function(data, index) {
  responses <- data$response[index]
  predictions <- data$prediction[index]
  baseline <- data$baseline[index]
  SSE <- sum((responses - predictions)^2)
  SST <- sum((responses - baseline)^2)
  r2 <- 1 - SSE/SST
  return(r2)
}

all_metrics <- function(data, index) {
  mse <- mean_squared_error(data, index)
  mae <- mean_absolute_error(data, index)
  OSR2 <- OS_R_squared(data, index)
  return(c(mse, mae, OSR2))
}

tableAccuracy <- function(test, pred) {
  t = table(test, pred)
  a = sum(diag(t))/length(test)
  return(a)
}

tableAccuracy_boot <- function(data, index) {

  test_i = data$response[index]
  pred_i = data$prediction[index]
  t = table(test_i, pred_i)
  a = sum(diag(t))/length(test_i)
  return(a)
}

Accuracy_baseline <- function(data, index) {

  test_i = data$response[index]
  pred_i = data$prediction[index]
  t = table(test_i, pred_i)
  a = t[2]/(t[1]+t[2])
  return(a)
}
```

Boostrapping appliad to the random forest model.

```
RF_test_set = data.frame(response = q.test$goodq, prediction = pred.q.rf, baseline = rep(FALSE, times = nrow
(q.test)))

set.seed(892)
RF_boot <- boot(RF_test_set, tableAccuracy_boot, R = 10000)
RF_boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = RF_test_set, statistic = tableAccuracy_boot, R = 10000)
##
##
## Bootstrap Statistics :
##     original       bias    std. error
## t1* 0.571875 3.075893e-05  0.01048866
```

```
boot.ci(RF_boot, index = 1, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = RF_boot, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%   ( 0.5513,  0.5924 )
## Calculations and Intervals on Original Scale
```

The random forest model has an extremely small bias and relatively low standard error. The Confidence 95% confidence interval of accuracies is .56-.60.

# Question 1 Part C: Improvements in Modeling for 15 best

I am selecting the most reliably accurate model as the best model for achieving an improvement in identification of good questions. Accuracy is a critical metric because it indicates both the models ability to identify good questions and its ability to identify bad questions. In addition to the simple accuracy score, I will apply bootstrapping to the 4 top candidate models to determine the confidence interval of the accuracies of each, as a model with high accuracy but relatively large error bars may not be the best option in this case.

```
boosted_test_set = data.frame(response = q.test$goodq, prediction = pred_boost, baseline = rep(FALSE, times
= nrow(q.test)))

set.seed(892)
boost_boot <- boot(boosted_test_set, tableAccuracy_boot, R = 10000)
boost_boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = boosted_test_set, statistic = tableAccuracy_boot,
##     R = 10000)
##
##
## Bootstrap Statistics :
##      original        bias    std. error
## t1* 0.5566964 -6.424107e-05  0.01046686
```

```
boot.ci(boost_boot, index = 1, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boost_boot, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%   ( 0.5362,  0.5772 )
## Calculations and Intervals on Original Scale
```

The standard error of the boosted model is slightly larger than the standard error of the rf model, confirming that the rf model is a better option for accuracy determination than the boosted model.

```
cart_test_set = data.frame(response = q.test$goodq, prediction = pred_cart, baseline = rep(FALSE, times = nr
ow(q.test)))

set.seed(892)
cart_boot <- boot(cart_test_set, tableAccuracy_boot, R = 10000)
cart_boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = cart_test_set, statistic = tableAccuracy_boot, R = 10000)
##
##
## Bootstrap Statistics :
##      original        bias    std. error
## t1* 0.5589286 -6.084821e-05  0.01050588
```

```
boot.ci(cart_boot, index = 1, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = cart_boot, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%   ( 0.5384,  0.5795 )
## Calculations and Intervals on Original Scale
```

The CART model has a similar standard error (and therefore similar 95% confidence interval error bars) as the boosted model, but its accuracy range is somewhat higher. The random forest model is still the best one.

```
log_test_set = data.frame(response = q.test$goodq, prediction = pred_log, baseline = rep(FALSE, times = nrow
(q.test)))

set.seed(892)
log_boot <- boot(log_test_set, tableAccuracy_boot, R = 10000)
log_boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = log_test_set, statistic = tableAccuracy_boot, R = 10000)
##
##
## Bootstrap Statistics :
##      original     bias    std. error
## t1* 0.5602679 5.8125e-05   0.0104311
```

```
boot.ci(log_boot, index = 1, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = log_boot, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%   ( 0.5393,  0.5804 )
## Calculations and Intervals on Original Scale
```

The logistic model performs well, but not as well as the random forest model, so the random forest model is the one which is ultimately selected as the best model to identify a good question for the lookup.

In order to determine the degree to which the random forest method of assigning top hits improves upon the “most recent” method, we must first evaluate the accuracy of the “most recent” method in a comparable manner. The “most recent” method essentially assumes that all questions are good (meaning that the most recently asked question is a good one and should be selected). A bootstrapped assessment of the accuracy of a basic model predicting that all questions are good will give a basis of comparison for evaluating the degree to which the selected random forest model from part b will improve placement of good questions in the top 15.

```
rec_test_set = data.frame(response = q.test$goodq, prediction = rep(TRUE, times = nrow(q.test)), baseline =
rep(FALSE, times = nrow(q.test)))

set.seed(592)
rec_boot <- boot(rec_test_set, Accuracy_baseline, R = 10000)
rec_boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = rec_test_set, statistic = Accuracy_baseline, R = 10000)
##
##
## Bootstrap Statistics :
##      original        bias    std. error
## t1* 0.4924107 -8.491071e-05  0.01067414
```

```
boot.ci(rec_boot, index = 1, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = rec_boot, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%   ( 0.4714,  0.5138 )
## Calculations and Intervals on Original Scale
```

```
 pp = rep(TRUE, times = nrow(q.test))
t = table(q.test$goodq, pp)
t
```

```
##        pp
##         TRUE
##   FALSE 1137
##   TRUE  1103
```

The selected model (the random forest model from part B) has a test set accuracy of .58 with 95% confidence interval of .56-.60. The “all questions are good” model (a standin for the default of choosing the most recent question) has an accuracy of .49 with a confidence interval of .47-.51. Given that the confidence intervals of the two models do not overlap, we can say with a high degree of certainty (>95%) that the random forest model will do a better job of recommeding useful questions than the default of recommending the most recent question

asked. The improvement in accuracy can be best approximted by the difference of the model accuracies, which is 9%.