Tabula statement

We're part of an academic community at Warwick.

Whether studying, teaching, or researching, we're all taking part in an expert conversation which must meet standards of academic integrity. When we all meet these standards, we can take pride in our own academic achievements, as individuals and as an academic community.

Academic integrity means committing to honesty in academic work, giving credit where we've used others' ideas and being proud of our own achievements.

In submitting my work I confirm that:

1. I have read the guidance on academic integrity provided in the Student Handbook and understand the University regulations in relation to Academic Integrity. I am aware of the potential consequences of Academic Misconduct.

2. I declare that the work is all my own, except where I have stated otherwise.

3. No substantial part(s) of the work submitted here has also been submitted by me in other credit bearing assessments courses of study (other than in certain cases of a resubmission of a piece of work), and I acknowledge that if this has been done this may lead to an appropriate sanction.

4. Where a generative Artificial Intelligence such as ChatGPT has been used I confirm I have abided by both the University guidance and specific requirements as set out in the Student Handbook and the Assessment brief. I have clearly acknowledged the use of any generative Artificial Intelligence in my submission, my reasoning for using it and which generative AI (or AIs) I have used. Except where indicated the work is otherwise entirely my own.

5. I understand that should this piece of work raise concerns requiring investigation in relation to any of points above, it is possible that other work I have submitted for assessment will be checked, even if marks (provisional or confirmed) have been published.

6. Where a proof-reader, paid or unpaid was used, I confirm that the proofreader was made aware of and has complied with the University's proofreading policy.

7. I consent that my work may be submitted to Turnitin or other analytical technology. I understand the use of this service (or similar), along with other methods of maintaining the integrity of the academic process, will help the University uphold academic standards and assessment fairness.

Privacy statement

The data on this form relates to your submission of coursework. The date and time of your submission, your identity, and the work you have submitted will be stored. We will only use this data to administer and record your coursework submission.

Related articles

Reg. 11 Academic Integrity (from 4 Oct 2021)

Guidance on Regulation 11

Proofreading Policy

Education Policy and Quality Team

Academic Integrity (warwick.ac.uk)

Predictions on Ratings of Stars to A Restaurant- Regression and Decision Trees Analysis

Lina Tang u2144108

December 5, 2023

Email: u2144108@live.warwick.ac.uk

Github ID: LinaaaaaainL

All files for this project are updated to the Github, in the 'EC349 Assignment' repository.

Methodology

This project will use John Rollin's General DS Methodology: BAD_RCUP-MED/F to produce accurate predictions about the "stars" ratings since it is concise, easy to grasp and apply. Firstly, predictive analytic on "stars" on user reviews and businesses is the problem definition. Understanding and organizing the data are in the second phase. Summary and graph descriptive statistics will be investigated. Third, in order to find the best one, a range of analysis approaches will be utilized, such as regression and decision trees. Based on the results of model development and cross-validation, I will construct exhaustive analyses and evaluate the predictability of each model.

Problem Understanding & Analytic Approach

The objective of this research is to forecast star ratings by utilizing user and review data. Both regression and decision tree analysis will be used in order to identify the most suitable model. The analysis will specifically focus on OLS, ridge, and LASSO regression. Furthermore, with decision trees, enhancement techniques such as bagging, random forests, and boosting will be adopted. Each approach will be analyzed and examined.

Data Understanding & Organization

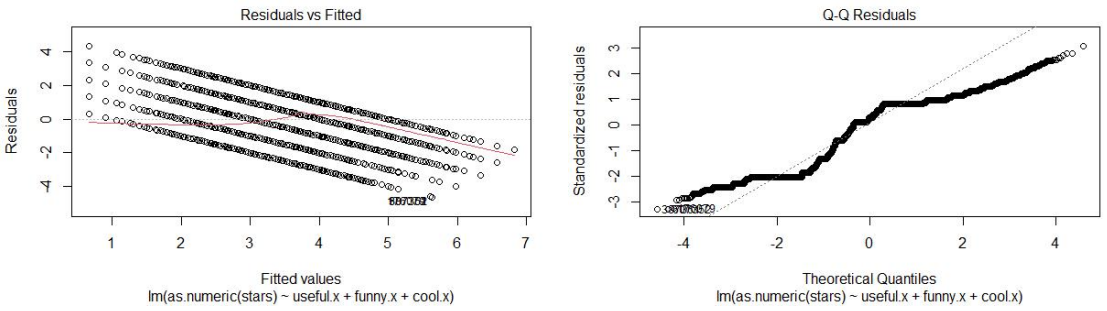
In the review sample, review_id, text and date will not be included as stars prediction factors because to their specificity and difficulty in definition. Meanwhile, in the user sample, name, yelping_since and elite will not be involved for the same reasons. From the brief inspection of data, the presence of extreme values implies that the sample suffers from outlier problems. Initially, we will join the two samples based on the user_id and thereafter do data cleansing.

While stars may appear to be a categorical variable, R Studio does not recognize factors as a response variable. All main variables will be designated as numerical values in the regression analysis but some of them will be placed as factors in the decision trees. The combined sample has a significant amount of missing data. Following Zhang's (2015) approach, the missing values are excluded from the data. While scholars (Emmanuel et al., 2021) may argue the presence of bias, it is important to note that R Studio does not handle missing values in either regression or decision tree analyses.

Review data involves the up-to-date evaluation of users to the restaurant. By contrast, user data contains previous accumulative reviews of users. In general, human or mechanical mistake, respondents' reluctance to answer specific questions, study dropout, merging unrelated data, and errors due to malfunctioning equipment are common causes of missing values and sampling errors (Emmanuel et al., 2021). The histograms demonstrated that a majority data of user.x, funny.x and cool.x are below 50. As stars are clearly categorical values ranging from 1 to 5, values for these variables are restricted to 5, to exclude outliers. Additionally, data is randomly split into training and test samples, with probability of 0.8 and 0.2 respectively.

Regression Analysis & Evaluation

$$Stars_{ij} = \beta_1 cool.x_{ij} + \beta_2 funny.x_{ij} + \beta_3 useful.x_{ij} + \varepsilon_{ij} \qquad (1)$$



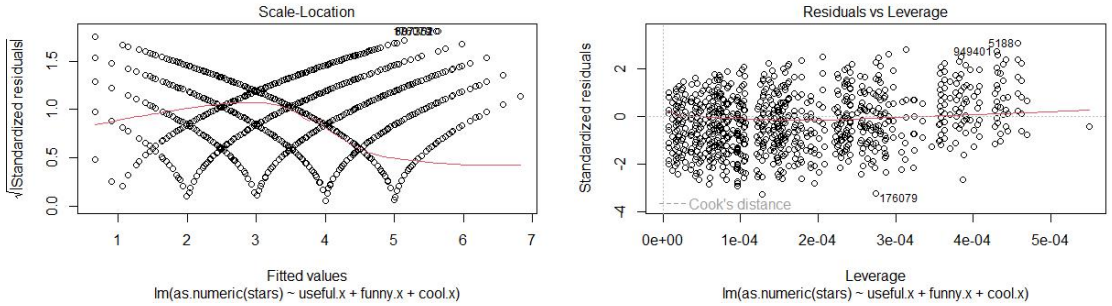
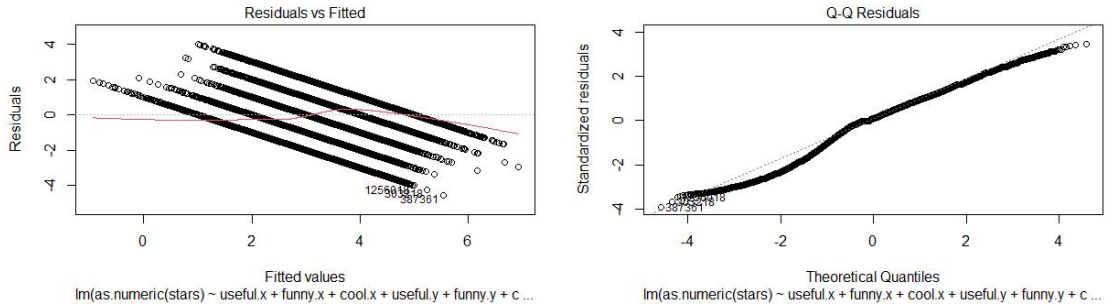


Figure 1. Plots for model (1)

In model (1), it is apparent that all predictors have a substantial impact on the rating of stars. The coefficient of R-squared is the fraction of dependent variable variance predicted by independent factors (Wright, 1921). The capability of the model (1) to account for the variations is inadequate, with an approximate R-squared of 7.25%. Among the other graphs, there exist numerous outliers. The relationship between the variables may be somewhat non-linear, as suggested by the non-linear red line. The presence of an S-shape in the Q-Q plot indicates that the data may deviate from a normal distribution.

$$Stars_{ij} = \beta_1 cool.x_{ij} + \beta_2 funny.x_{ij} + \beta_3 useful.x_{ij} + \beta_4 X_{ij} + \varepsilon_{ij} \quad (2)$$



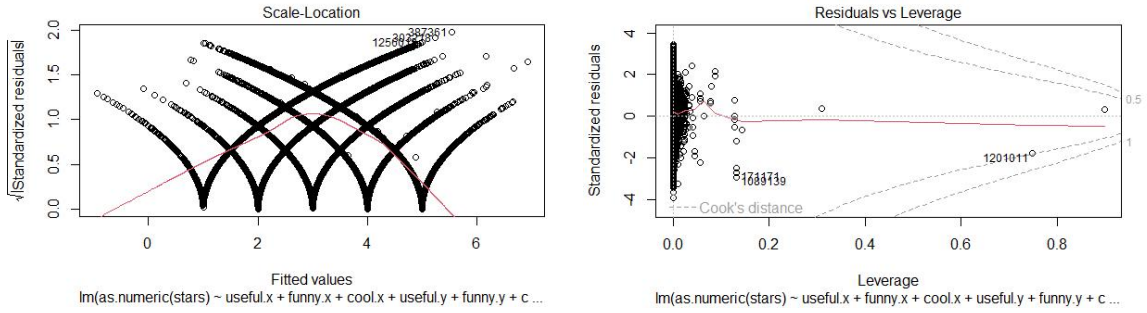


Figure 2. Plots for model (2)

Clearly, model (2) with all accountable variables, as represented by Xij, has a higher R^2 value of 37.67% than model (1). From the Q-Q plot, model (2) exhibits a normal distribution. Model (2) has relative standard error (RMSE) of 1.154363.

Ridge regression with similar variables may fit linear models better than OLS due to its reduced MSE. Through cross-validation, the optimal lambda value for the ridge regression is 0.08563297, which is close to 0 such that estimators are similar to those in the OLS regression model. Surprisingly, the ridge model does not outperform model (2), with higher RMSE.

On contrary, the RMSE for LASSO regression and OLS model (2) are nearly same when using a similar set of variables. Although the stars are on a scale from 1 to 5, every single model has an MSE greater than 1, which is a relatively high value.

Decision Trees Analysis & Evaluation

Decision tree analysis allows for more flexibility in the specification of the relationship between variables. Classification tree will be implemented since stars were declared to be categorical variables. Predicting stars to be only 1 and 5 at probability of 17% and of 83% respectively (Figure 3), the decision tree is heavily

influenced solely by average stars. The majority of each leave is roughly 50%, which suggests a poor predictability.

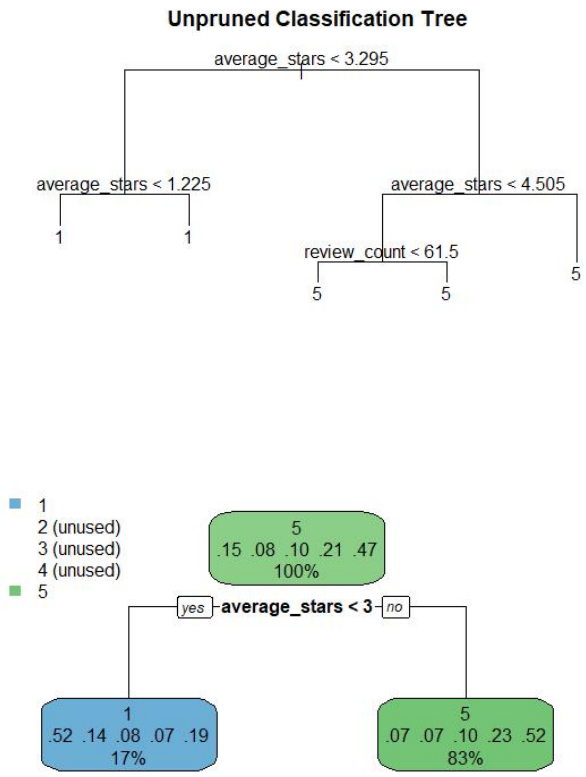


Figure 3. Unpruned Classification Trees

In an effort to avoid over-fitting issue in the tree, the minimum bucket size is restricted to 10000 and the minimum split is limited to 5. Simultaneously, the maximum depth is capped to 10. The revised classification tree, shown in figure 4, incorporates both the number of reviews and the average star rating, resulting in more adaptable forecasts. The rating of stars is predicted to be 1 at 17%, 4 at 18% and 5 at 65%.

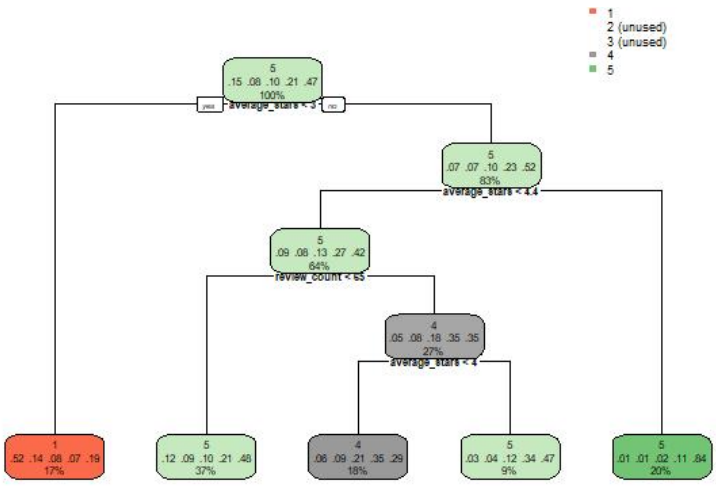


Figure 4. Restricted Classification Trees

While decision trees are conceptually uncomplicated and easy to execute and visualize, small perturbations in the data can have significant impacts on the resulting tree. Hence, there might be a considerable amount of variance and inadequate accuracy in making predictions. We will apply several techniques including bagging, boosting, and random forest to enhance the accuracy of our predictions. Bagging is an approximately unbiased model. In random forests, the concept of bootstrap aggregation is utilized in conjunction with random trees. Boosting lets trees grow in a way that gets rid of bias.

For bagging, the rate of misclassification error is relatively high, with an out-of-bag estimate of 0.4784. With 100 bootstrap replications, the bagging classification trees correctly forecast star ratings 52.16% of the time. Nevertheless, random forests do not perform well, with relatively high error rate of 0.5907672 for 200 trees.

Unlike bagging averaging distinct training samples, boosting focuses on averaging different models. Boosting exhibits a reduced error rate of 0.4612786, surpassing the performance of bagging. Although bagging and boosting algorithms outperform classification trees, the resulting models cannot be interpretative in the form of trees.

In conclusion, the OLS regression model that is appropriately specified presents comparable performance to a LASSO model. Ridge regression performs poorly in predicting star ratings. Moreover, boosting has the lowest error rate in this project. Random forests conducted unfavorably exhibiting a rather high error rate. This can be attributed to a limited quantity of trees.

Challenges

Overall, I ran into a number of issues with my project. Loading data is a huge time sink, even for the small samples. Honestly, I am just starting up with the R language, so getting to know the system and the scripts takes a long time. Even though my models have quite poor predictability, I hope models and analysis can provide some valuable insights.

Reference

Emmanuel, T., Maupong, T., Mpoeleng, D., Semong, T., Mphago, B. and Tabona, O. (2021). A Survey on Missing Data in Machine Learning. *Journal of Big Data*, 8(1). doi:https://doi.org/10.1186/s40537-021-00516-9.

Wright, S. (1921). Correlation and Causation. *Journal of Agricultural Research*, 20(7), pp.557–585.

Zhang, Z. (2015). Missing values in big data research: some basic skills. *Annals of translational medicine*, [online] 3(21), p.323. doi:https://doi.org/10.3978/j.issn.2305-5839.2015.12.11.

````
---
title: "EC349 Assignment R Markdown"
author: "Lina Tang"
date: "2023-12-02"
output:
  html_document: default
  word_document: default
  pdf_document: default
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

**Methodology**
This project will use John Rollin’s General DS Methodology: BAD_RCUP-MED/F to produce accurate predictions about the "stars" ratings since it is concise, easy to grasp and apply. Firstly, predictive analytic on "stars" on user reviews and businesses is the problem definition. Understanding and organizing the data are in the second phase. Summary and graph descriptive statistics will be investigated. Third, in order to find the best one, a range of analysis approaches will be utilized, such as regression and decision trees. Based on the results of model development and cross-validation, I will construct exhaustive analyses and evaluate the predictability of each model.

**Problem Understanding & Analytic Approach**
The objective of this research is to forecast star ratings by utilizing user and review data. Both regression and decision tree analysis will be used in order to identify the most suitable model. The analysis will specifically focus on OLS, ridge, and LASSO regression. Furthermore, with decision trees, enhancement techniques such as bagging, random forests, and boosting will be adopted. Each approach will be analyzed and examined.

```{r Load Libraries, echo=TRUE, message=FALSE, warning=FALSE, paged.print=TRUE}
library(tidyverse)
library(caret)
library(hexbin)
library(jsonlite)
library(magrittr)
library(dplyr)
library(glmnet)
library(ggplot2)
library(tree)
library(rpart)
library(rpart.plot)
```

```{r message=FALSE, warning=FALSE}
getwd()
```

```{r message=FALSE, warning=FALSE}
setwd("C:/Users/123/Desktop/EC349 Assignment")
```

```{r load data, message=FALSE, warning=FALSE}
cat("\014")
rm(list=ls())

load("C:/Users/123/Desktop/EC349 Assignment/yelp_review_small.Rda")
load("C:/Users/123/Desktop/EC349 Assignment/yelp_user_small.Rda")
```

```{r inspect data - review_data_small, message=FALSE, warning=FALSE}
head(review_data_small)
str(review_data_small)
summary(review_data_small)
dim(review_data_small)
colnames(review_data_small)
```
````

In the review sample, review_id, text and date will not be included as stars prediction factors because to their specificity and difficulty in definition.

```{r inspect data - user_data_small, message=FALSE, warning=FALSE}
head(user_data_small)
str(user_data_small)
summary(user_data_small)
dim(user_data_small)
colnames(user_data_small)
```

Meanwhile, in the user sample, name, yelping_since and elite will not be involved for the same reasons. From the brief inspection of data, the presence of extreme values implies the sample suffers from outlier problems. Initially, we will join the two samples based on the user_id and thereafter do data cleansing.

```{r merge data by "user_id", echo=TRUE, warning=FALSE, message=FALSE}
df <- review_data_small %>% left_join (user_data_small, by = "user_id")
str(df)
summary(df)
df$stars <- as.numeric(df$stars)
```

```{r  deal with missing values - delete, message=FALSE, warning=FALSE}
sapply(df, function(x) sum(is.na(x)))
df = df[complete.cases(df), ]
str(df)
summary(df)
```

While stars may appear to be a categorical variable, R Studio does not recognize factors as a response variable. All main variables will be designated as numerical values in the regression analysis but some of them will be placed as factors in the decision trees. The combined sample has a significant amount of missing data. Following Zhang's (2015) approach, the missing values are excluded from the data. While scholars (Emmanuel et al., 2021) may argue the presence of bias, it is important to note that R Studio does not handle missing values in either regression or decision tree analyses.

```{r set all main variables as numeric, message=FALSE, warning=FALSE}
df$useful.x <- as.numeric(df$useful.x)
df$funny.x <- as.numeric(df$funny.x)
df$cool.x <- as.numeric(df$cool.x)
df$funny.y <- as.numeric(df$funny.y)
df$useful.y <- as.numeric(df$useful.y)
df$fans <- as.numeric(df$fans)
df$average_stars <- as.numeric(df$average_stars)
df$compliment_cool <- as.numeric(df$compliment_cool)
df$compliment_more <- as.numeric(df$compliment_more)
df$compliment_hot <- as.numeric(df$compliment_hot)
df$compliment_profile <- as.numeric(df$compliment_profile)
df$compliment_note <- as.numeric(df$compliment_note)
df$compliment_cute <- as.numeric(df$compliment_cute)
df$compliment_list <- as.numeric(df$compliment_list)
df$compliment_funny <- as.numeric(df$compliment_funny)
df$compliment_plain <- as.numeric(df$compliment_plain)
df$compliment_writer <- as.numeric(df$compliment_writer)
df$compliment_photos <- as.numeric(df$compliment_photos)
```

```{r deal with outliers, message=FALSE, warning=FALSE}
hist(df$useful.x, breaks=10)
hist(df$funny.x, breaks=10)
hist(df$cool.x, breaks=10)
df <- subset(df, useful.x < 6 & funny.x < 6 & cool.x < 6)
```

Review data involves the up-to-date evaluation of users to the restaurant. By contrast, user data contains previous accumulative reviews of users. In general, human or mechanical mistake, respondents' reluctance to answer specific questions, study

dropout, merging unrelated data, and errors due to malfunctioning equipment are common causes of missing values and sampling errors (Emmanuel et al., 2021). The histograms demonstrated that a majority data of user.x, funny.x and cool.x are below 50. As stars are clearly categorical values ranging from 1 to 5, values for these variables are restricted to 5, to exclude outliers.

```{r split data into training and test samples, message=FALSE, warning=FALSE}
set.seed(1)
df$stars <- as.numeric(df$stars)
training.samples <- df$stars %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data <- df[training.samples, ]
test.data <- df[-training.samples, ]
```

Additionally, data is randomly split into training and test samples, with probability of 0.8 and 0.2 respectively.

Regression Analysis & Evaluation

```{r Build model1, message=FALSE, warning=FALSE}
model1 <- lm(as.numeric(stars) ~ useful.x + funny.x + cool.x, data = train.data)
summary(model1)
plot(model1)
```

In model (1), it is apparent that all predictors have a substantial impact on the rating of stars. The coefficient of R-squared is the fraction of dependent variable variance predicted by independent factors (Wright, 1921). The capability of the model (1) to account for the variations is inadequate, with an approximate R-squared of 7.25%. Among the other graphs, there exist numerous outliers. The relationship between the variables may be somewhat non-linear, as suggested by the non-linear red line. The presence of an S-shape in the Q-Q plot indicates that the data may deviate from a normal distribution.

```{r Prediction using model1, message=FALSE, warning=FALSE}
predictions_m1 <- predict(model1, newdata = test.data)
compare <- data.frame(actual = as.numeric(test.data$stars),
                      predicted = predictions_m1)
OLS_MSE_m1<- mean((predictions_m1 - as.numeric(test.data$stars)) ^ 2) #Note how it outperforms OLS
OLS_MSE_m1 #2.001981
sqrt(OLS_MSE_m1) #1.414914
```

```{r Model2: with more variables, message=FALSE, warning=FALSE}
model2 <- lm(as.numeric(stars) ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y + average_stars + fans + review_count + compliment_hot + compliment_more + compliment_profile + compliment_cute + compliment_list + compliment_note + compliment_plain + compliment_cool + compliment_funny + compliment_writer + compliment_photos, data = train.data)
summary(model2)
plot(model2)
```

Clearly, model (2) with all accountable variables has a higher R^2 value of 37.67% than model (1). From the Q-Q plot, model (2) exhibits a normal distribution.

```{r Prediction using model2, message=FALSE, warning=FALSE}
predictions_m2 <- model2 %>% predict(test.data)
compare <- data.frame(actual = as.numeric(test.data$stars),
                      predicted = predictions_m2)
OLS_MSE_m2<- mean((predictions_m2 - as.numeric(test.data$stars)) ^ 2) #Note how it outperforms OLS
OLS_MSE_m2 #1.332553
sqrt(OLS_MSE_m2) #1.154363
```

Model (2) has relative standard error (RMSE) of 1.154363.

```{r Model3: exclude insignificant affected variables, message=FALSE, warning=FALSE}
model3 <- lm(as.numeric(stars) ~ useful.x + funny.x + cool.x + useful.y + funny.y +
cool.y + average_stars + fans + review_count + compliment_profile + compliment_more +
compliment_list + compliment_writer + compliment_plain + compliment_note, data =
train.data)
summary(model3)
plot(model3)
```

```{r Prediction using model3, message=FALSE, warning=FALSE}
predictions_m3 <- model3 %>% predict(test.data)
compare <- data.frame(actual = as.numeric(test.data$stars),
                      predicted = predictions_m3)
OLS_MSE_m3<- mean((predictions_m3 - as.numeric(test.data$stars)) ^ 2) #Note how it
outperforms OLS
OLS_MSE_m3 #1.332569
sqrt(OLS_MSE_m3) #1.15437
```

The RMSE of OLS linear regression model 3 is 1.15437, which is exactly same as model 2. Therefore, we will focus on model 2 in the later analysis.

```{r warning=FALSE, message=FALSE, include=FALSE}
library(sandwich)
library(lmtest)
coeftest(model3, vcov = vcovHC(model3, type="HC3"))
```

Ridge regression with similar variables may fit linear models better than OLS due to its reduced MSE.

```{r Ridge on Train Dataset, warning=FALSE, message=FALSE}
library(glmnet)
library(ggplot2)

set.seed(1)
train_y <- train.data$stars
train_x <- data.matrix(train.data[, c('useful.x', 'funny.x', 'cool.x', 'useful.y',
'funny.y','cool.y', 'average_stars', 'compliment_hot', 'review_count',
'compliment_more', 'compliment_profile', 'compliment_cute', 'compliment_list',
'compliment_note', 'compliment_plain', 'compliment_cool', 'compliment_funny',
'compliment_writer', 'compliment_photos')])

test_y <- test.data$stars
test_x <- data.matrix(test.data[, c('useful.x', 'funny.x', 'cool.x', 'useful.y',
'funny.y','cool.y', 'average_stars', 'compliment_hot', 'review_count',
'compliment_more', 'compliment_profile', 'compliment_cute', 'compliment_list',
'compliment_note', 'compliment_plain', 'compliment_cool', 'compliment_funny',
'compliment_writer', 'compliment_photos')])

ridge.mod <- glmnet(as.matrix(train_x), as.matrix(train_y), alpha = 0, lambda = 3,
thresh = 1e-12)
summary(ridge.mod)
```

```{r message=FALSE, warning=FALSE}
#perform k-fold cross-validation to find optimal lambda value
cv_model <- cv.glmnet(as.matrix(train_x), as.matrix(train_y), alpha = 0, nfolds = 3)

#find optimal lambda value that minimizes test MSE
lambda_ridge_cv <- cv_model$lambda.min
lambda_ridge_cv #0.08563297

#find optimal lambda value that minimizes test MSE
plot(cv_model)
```

Through cross-validation, the optimal lambda value for the ridge regression is 0.08563297, which is close to 0 such that estimators are similar to those in the OLS regression model.

```{r message=FALSE, warning=FALSE}
#find coefficients of best model
best_model <- glmnet(as.matrix(train_x), as.matrix(train_y), alpha = 0, lambda = lambda_ridge_cv)
coef(best_model)
```

```{r message=FALSE, warning=FALSE}
#use fitted best model to make predictions
y_predicted <- predict(ridge.mod, s = lambda_ridge_cv, newx = test_x)

#find SST and SSE
sst <- sum((test_y - mean(test_y))^2)
sse <- sum((y_predicted - test_y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq #0.2098919
```

```{r MSE for ridge model, message=FALSE, warning=FALSE}
ridge_MSE<- mean((y_predicted - test_y) ^ 2) #Note how it outperforms OLS
ridge_MSE #1.711681
sqrt(ridge_MSE) #1.308312
```

Surprisingly, the ridge model does not outperform model (2), with higher RMSE.

```{r LASSO Model, message=FALSE, warning=FALSE}
cv.out2 <- cv.glmnet(as.matrix(train_x), as.matrix(train_y), alpha = 1, nfolds = 3)
plot(cv.out2)
lambda_LASSO_cv <- cv.out2$lambda.min #cross-validation is the lambda minimising empirical MSE in training data

#Re-Estimate Ridge with lambda chosen by Cross validation
LASSO.mod<-glmnet(train_x, train_y, alpha = 1, lambda = lambda_LASSO_cv, thresh = 1e-12)
coef(LASSO.mod) #note that some parameter estimates are set to 0 --> Model selection!

#Fit on Test Data
LASSO.pred <- predict(LASSO.mod, s = lambda_LASSO_cv, newx = as.matrix(test_x))
LASSO_MSE<- mean((LASSO.pred - test_y) ^ 2) #Note how it outperforms OLS
LASSO_MSE #1.332983
sqrt(LASSO_MSE) #1.154549
```

On contrary, the RMSE for LASSO regression and OLS model (2) are nearly same when using a similar set of variables. Although the stars are on a scale from 1 to 5, every single model has an MSE greater than 1, which is a relatively high value.

Decision Trees Analysis & Evaluation
Decision tree analysis allows for more flexibility in the specification of the relationship between variables.

```{r Decision Tree, message=FALSE, warning=FALSE}
#What happens when you include all variables?
mod1 <- glm(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y + average_stars + fans + review_count + compliment_hot + compliment_more + compliment_profile + compliment_cute + compliment_list + compliment_note + compliment_plain + compliment_cool + compliment_funny + compliment_writer + compliment_photos, data = train.data)

#Review the results
coef(mod1)
summary(mod1)
```

```
#Predicted Values for Test Data based on the model estimates
mod1_predict<-predict(mod1, newdata = test.data)
```

```{r message=FALSE, warning=FALSE}
#What happens when you exclude insignificant variables?
mod2 <- glm(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y +
average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data)

#Review the results
coef(mod2)
summary(mod2)
```

Classification tree will be implemented since stars were declared to be categorical variables.

```{r Classification Tree, message=FALSE, warning=FALSE}
library(tree)
set.seed(1)
train.data$stars <- as.factor(train.data$stars)
test.data$stars <- as.factor(test.data$stars)
train.data$useful.x <- as.factor(train.data$useful.x)
test.data$useful.x <- as.factor(test.data$useful.x)
train.data$funny.x <- as.factor(train.data$funny.x)
test.data$funny.x <- as.factor(test.data$funny.x)
train.data$cool.x <- as.factor(train.data$cool.x)
test.data$cool.x <- as.factor(test.data$cool.x)

#With tree library
tree1<-tree(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y +
average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data)

#partition graph
partition.tree(tree1)
points(train.data[, c("useful.x", "funny.x", "cool.x", "useful.y", "funny.y",
"cool.y", "average_stars", "fans", "review_count", "compliment_more",
"compliment_profile", "compliment_list", "compliment_note", "compliment_plain",
"compliment_writer")], cex=.4)

plot(tree1)
text(tree1, pretty = 0)
title(main = "Unpruned Classification Tree")
```

```{r Classification Tree without restrictions, message=FALSE, warning=FALSE}
library(rpart)
rpart_tree<-rpart(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y +
average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data)
rpart.plot(rpart_tree)
```

Predicting stars to be only 1 and 5 at probability of 17% and of 83% respectively, the decision tree is heavily influenced solely by average stars. The majority of each leave is roughly 50%, which suggests a poor predictability.

```{r Classification Tree with restrictions, message=FALSE, warning=FALSE}
rpart_tree2<-rpart(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y +
average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data, control = rpart.control(maxdepth = 10, minsplit = 5, minbucket = 10000, cp
= 0))
```

```
print(rpart_tree2)
rpart.plot(rpart_tree2, cex = 0.5)

#importance
rpart_tree2$variable.importance
```

In an effort to avoid over-fitting issue in the tree, the minimum bucket size is restricted to 10000 and the minimum split is limited to 5. Simultaneously, the maximum depth is capped to 10. The revised classification tree, shown in figure 4, incorporates both the number of reviews and the average star rating, resulting in more adaptable forecasts. The rating of stars is predicted to be 1 at 17%, 4 at 18% and 5 at 65%.

While decision trees are conceptually uncomplicated and easy to execute and visualize, small perturbations in the data can have significant impacts on the resulting tree. Hence, there might be a considerable amount of variance and inadequate accuracy in making predictions. We will apply several techniques including bagging, boosting, and random forest to enhance the accuracy of our predictions. Bagging is an approximately unbiased model. In random forests, the concept of bootstrap aggregation is utilized in conjunction with random trees. Boosting lets trees grow in a way that gets rid of bias.

```{r Bagging, message=FALSE, warning=FALSE}
library(ipred)
set.seed(1)

#fit the bagged model
bag <- bagging(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y +
average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data, nbagg = 100,
  coob = TRUE, control = rpart.control(minsplit = 3, cp = 0.1)
)

#display fitted bagged model
bag
```

For bagging, the rate of misclassification error is relatively high, with an out-of-bag estimate of 0.4784. With 100 bootstrap replications, the bagging classification trees correctly forecast star ratings 52.16% of the time.

```{r Random Forest, message=FALSE, warning=FALSE}
set.seed(1)
library(randomForest) #randomforest
model_RF <- randomForest(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y +
cool.y + average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data, ntree=200)
pred_RF_test = predict(model_RF, test.data)
mean(model_RF[["err.rate"]]) #0.5907672 for 200 trees.
```

Nevertheless, random forests do not perform well, with relatively high error rate of 0.5907672 for 200 trees.

```{r Boosting, message=FALSE, warning=FALSE}
library(adabag) #AdaBoost
# train a model using our training data
model_adaboost <- boosting(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y +
cool.y + average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data, boos=TRUE, mfinal=50)
summary(model_adaboost)

# Load data for Regression (this one is stored on R)
#use model to make predictions on test data
pred_ada_test = predict(model_adaboost, test.data)

# Returns the prediction values of test data along with the confusion matrix
```

```
pred_ada_test
```

Unlike bagging averaging distinct training samples, boosting focuses on averaging different models. Boosting exhibits a reduced error rate of 0.4612786, surpassing the performance of bagging. Although bagging and boosting algorithms outperform classification trees, the resulting models cannot be interpretative in the form of trees.

In conclusion, the OLS regression model that is appropriately specified presents comparable performance to a LASSO model. Ridge regression performs poorly in predicting star ratings. Moreover, boosting has the lowest error rate in this project. Random forests conducted unfavorably exhibiting a rather high error rate. This can be attributed to a limited quantity of trees.

Challenges
Overall, I ran into a number of issues with my project. Loading data is a huge time sink, even for the small samples. Honestly, I am just starting up with the R language, so getting to know the system and the scripts takes a long time. Even though my models have quite poor predictability, I hope models and analysis can provide some valuable insights.

Reference
Emmanuel, T., Maupong, T., Mpoeleng, D., Semong, T., Mphago, B. and Tabona, O. (2021). A Survey on Missing Data in Machine Learning. Journal of Big Data, 8(1). doi:https://doi.org/10.1186/s40537-021-00516-9.
Wright, S. (1921). Correlation and Causation. Journal of Agricultural Research, 20(7), pp.557–585.
Zhang, Z. (2015). Missing values in big data research: some basic skills. Annals of translational medicine, [online] 3(21), p.323. doi:https://doi.org/10.3978/j.issn.2305-5839.2015.12.11.

```
### EC349 Project R Script
### Lina Tang u2144108
# load library [already connect with Github]
library(tidyverse)
library(caret)
library(hexbin)
library(jsonlite)
library(magrittr)
library(dplyr)
library(glmnet)
library(ggplot2)
library(tree)
library(rpart)
library(rpart.plot)
install.packages("knitr")
library(knitr)
install.packages("tinytex")
tinytex::install_tinytex()
getwd()
setwd("C:/Users/123/Desktop/EC349 Assignment")

cat("\014")
rm(list=ls())

# load data
load("C:/Users/123/Desktop/EC349 Assignment/yelp_review_small.Rda")
load("C:/Users/123/Desktop/EC349 Assignment/yelp_user_small.Rda")

# inspect data - review_data_small
head(review_data_small)
str(review_data_small)
summary(review_data_small)
dim(review_data_small)
colnames(review_data_small)

# inspect data - user_data_small
head(user_data_small)
str(user_data_small)
summary(user_data_small)
dim(user_data_small)
colnames(user_data_small)

# merge data by "user_id"
df <- review_data_small %>% left_join (user_data_small, by = "user_id")
str(df)
summary(df)
df$stars <- as.numeric(df$stars)

# deal with missing values - delete
sapply(df, function(x) sum(is.na(x)))
df = df[complete.cases(df), ]
str(df)
summary(df)

# set all main variables as numeric
df$useful.x <- as.numeric(df$useful.x)
df$funny.x <- as.numeric(df$funny.x)
df$cool.x <- as.numeric(df$cool.x)
df$funny.y <- as.numeric(df$funny.y)
df$useful.y <- as.numeric(df$useful.y)
df$fans <- as.numeric(df$fans)
df$average_stars <- as.numeric(df$average_stars)
df$compliment_cool <- as.numeric(df$compliment_cool)
df$compliment_more <- as.numeric(df$compliment_more)
df$compliment_hot <- as.numeric(df$compliment_hot)
df$compliment_profile <- as.numeric(df$compliment_profile)
df$compliment_note <- as.numeric(df$compliment_note)
df$compliment_cute <- as.numeric(df$compliment_cute)
df$compliment_list <- as.numeric(df$compliment_list)
```

```
df$compliment_funny <- as.numeric(df$compliment_funny)
df$compliment_plain <- as.numeric(df$compliment_plain)
df$compliment_writer <- as.numeric(df$compliment_writer)
df$compliment_photos <- as.numeric(df$compliment_photos)

# deal with outliers
hist(df$useful.x, breaks=10)
hist(df$funny.x, breaks=10)
hist(df$cool.x, breaks=10)
df <- subset(df, useful.x < 6 & funny.x < 6 & cool.x < 6)

# split data into training and test samples
set.seed(1)
df$stars <- as.numeric(df$stars)
training.samples <- df$stars %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data <- df[training.samples, ]
test.data <- df[-training.samples, ]



###### OLS Regression Analysis ######
# Build model 1
model1 <- lm(as.numeric(stars) ~ useful.x + funny.x + cool.x, data = train.data)
summary(model1)
plot(model1)

### . to avoid codes being 'eaten'
### .
### .
### .
# Prediction using model 1
predictions_m1 <- predict(model1, newdata = test.data)
compare <- data.frame(actual = as.numeric(test.data$stars), predicted = predictions_m1)
OLS_MSE_m1<- mean((predictions_m1 - as.numeric(test.data$stars)) ^ 2) #Note how it
outperforms OLS
OLS_MSE_m1 #2.001981
sqrt(OLS_MSE_m1) #1.414914

# Model 2: with more variables
model2 <- lm(as.numeric(stars) ~ useful.x + funny.x + cool.x + useful.y + funny.y +
cool.y + average_stars + fans + review_count + compliment_hot + compliment_more +
compliment_profile + compliment_cute + compliment_list + compliment_note +
compliment_plain + compliment_cool + compliment_funny + compliment_writer +
compliment_photos, data = train.data)
summary(model2)
plot(model2)

### . to avoid codes being 'eaten'
### .
### .
### .
# Prediction using model 2
predictions_m2 <- model2 %>% predict(test.data)
compare <- data.frame(actual = as.numeric(test.data$stars),
                      predicted = predictions_m2)
OLS_MSE_m2<- mean((predictions_m2 - as.numeric(test.data$stars)) ^ 2) #Note how it
outperforms OLS
OLS_MSE_m2 #1.332553
sqrt(OLS_MSE_m2) #1.154363

# Model 3: exclude insignificant affected variables
model3 <- lm(as.numeric(stars) ~ useful.x + funny.x + cool.x + useful.y + funny.y +
cool.y + average_stars + fans + review_count + compliment_profile + compliment_more +
compliment_list + compliment_writer + compliment_plain + compliment_note, data =
train.data)
summary(model3)
plot(model3)
```

```
### . to avoid codes being 'eaten'
### .
### .
### .
# Prediction using model 3
predictions_m3 <- model3 %>% predict(test.data)
compare <- data.frame(actual = as.numeric(test.data$stars),
                      predicted = predictions_m3)
OLS_MSE_m3<- mean((predictions_m3 - as.numeric(test.data$stars)) ^ 2) #Note how it
outperforms OLS
OLS_MSE_m3 #1.332569
sqrt(OLS_MSE_m3) #1.15437

# For Robust Standard Errors
library(sandwich)
library(lmtest)
coeftest(model3, vcov = vcovHC(model3, type="HC3"))



###### Ridge Regression ######
# Ridge on Train Dataset
library(glmnet)
library(ggplot2)

set.seed(1)
train_y <- train.data$stars
train_x <- data.matrix(train.data[, c('useful.x', 'funny.x', 'cool.x', 'useful.y',
'funny.y','cool.y', 'average_stars', 'compliment_hot', 'review_count',
'compliment_more', 'compliment_profile', 'compliment_cute', 'compliment_list',
'compliment_note', 'compliment_plain', 'compliment_cool', 'compliment_funny',
'compliment_writer', 'compliment_photos')])
test_y <- test.data$stars
test_x <- data.matrix(test.data[, c('useful.x', 'funny.x', 'cool.x', 'useful.y',
'funny.y','cool.y', 'average_stars', 'compliment_hot', 'review_count',
'compliment_more', 'compliment_profile', 'compliment_cute', 'compliment_list',
'compliment_note', 'compliment_plain', 'compliment_cool', 'compliment_funny',
'compliment_writer', 'compliment_photos')])

ridge.mod <- glmnet(as.matrix(train_x), as.matrix(train_y), alpha = 0, lambda = 3,
thresh = 1e-12)
summary(ridge.mod)

# perform k-fold cross-validation to find optimal lambda value
cv_model <- cv.glmnet(as.matrix(train_x), as.matrix(train_y), alpha = 0, nfolds = 3)

# find optimal lambda value that minimizes test MSE
lambda_ridge_cv <- cv_model$lambda.min
lambda_ridge_cv #0.08563297

# find optimal lambda value that minimizes test MSE
plot(cv_model)

# find coefficients of best model
best_model <- glmnet(as.matrix(train_x), as.matrix(train_y), alpha = 0, lambda =
lambda_ridge_cv)
coef(best_model)

# use fitted best model to make predictions
y_predicted <- predict(ridge.mod, s = lambda_ridge_cv, newx = test_x)

# find SST and SSE
sst <- sum((test_y - mean(test_y))^2)
sse <- sum((y_predicted - test_y)^2)

# find R-Squared
rsq <- 1 - sse/sst
rsq #0.2098919
```

```
# calculateing RMSE for ridge model
ridge_MSE<- mean((y_predicted - test_y) ^ 2) #Note how it outperforms OLS
ridge_MSE #1.711681
sqrt(ridge_MSE) #1.308312



###### LASSO Analysis ######
cv.out2 <- cv.glmnet(as.matrix(train_x), as.matrix(train_y), alpha = 1, nfolds = 3)
plot(cv.out2)
lambda_LASSO_cv <- cv.out2$lambda.min #cross-validation is the lambda minimising
empirical MSE in training data

#Re-Estimate Ridge with lambda chosen by Cross validation
LASSO.mod<-glmnet(train_x, train_y, alpha = 1, lambda = lambda_LASSO_cv, thresh = 1e-
12)
coef(LASSO.mod) #note that some parameter estimates are set to 0 --> Model selection!

#Fit on Test Data
LASSO.pred <- predict(LASSO.mod, s = lambda_LASSO_cv, newx = as.matrix(test_x))
LASSO_MSE<- mean((LASSO.pred - test_y) ^ 2) #Note how it outperforms OLS
LASSO_MSE #1.332983
sqrt(LASSO_MSE) #1.154549



###### Decision Tree ######
#What happens when you include all variables?
mod1 <- glm(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y +
average_stars + fans + review_count + compliment_hot + compliment_more +
compliment_profile + compliment_cute + compliment_list + compliment_note +
compliment_plain + compliment_cool + compliment_funny + compliment_writer +
compliment_photos, data = train.data)

#Review the results
coef(mod1)
summary(mod1)

#Predicted Values for Test Data based on the model estimates
mod1_predict<-predict(mod1, newdata = test.data)

#What happens when you exclude insignificant variables?
mod2 <- glm(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y +
average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data)

#Review the results
coef(mod2)
summary(mod2)

# Classification Tree
library(tree)
set.seed(1)
train.data$stars <- as.factor(train.data$stars)
test.data$stars <- as.factor(test.data$stars)
train.data$useful.x <- as.factor(train.data$useful.x)
test.data$useful.x <- as.factor(test.data$useful.x)
train.data$funny.x <- as.factor(train.data$funny.x)
test.data$funny.x <- as.factor(test.data$funny.x)
train.data$cool.x <- as.factor(train.data$cool.x)
test.data$cool.x <- as.factor(test.data$cool.x)

#With tree library
tree1<-tree(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y +
average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data)
```

```
#partition graph
partition.tree(tree1)
points(train.data[, c("useful.x", "funny.x", "cool.x", "useful.y", "funny.y",
"cool.y", "average_stars", "fans", "review_count", "compliment_more",
"compliment_profile", "compliment_list", "compliment_note", "compliment_plain",
"compliment_writer")], cex=.4)

plot(tree1)
text(tree1, pretty = 0)
title(main = "Unpruned Classification Tree")

# Classification Tree without restrictions
library(rpart)
rpart_tree<-rpart(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y +
average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data)
rpart.plot(rpart_tree)

# Classification Tree with restrictions
rpart_tree2<-rpart(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y +
average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data, control = rpart.control(maxdepth = 10, minsplit = 5, minbucket = 10000, cp
= 0))
print(rpart_tree2)
rpart.plot(rpart_tree2, cex = 0.5)

#importance
rpart_tree2$variable.importance



###### Bagging ######
library(ipred)
set.seed(1)

#fit the bagged model
bag <- bagging(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y + cool.y +
average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data, nbagg = 100,
               coob = TRUE, control = rpart.control(minsplit = 3, cp = 0.1)
)

#display fitted bagged model
bag



###### Random Forests ######
library(randomForest) #randomforest
set.seed(1)
model_RF<-randomForest(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y +
cool.y + average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data, ntree=200)
pred_RF_test = predict(model_RF, test.data)
mean(model_RF[["err.rate"]]) #0.5907672 for 200 trees.


###### Boosting ######
library(adabag) #AdaBoost

# train a model using our training data
model_adaboost <- boosting(stars ~ useful.x + funny.x + cool.x + useful.y + funny.y +
cool.y + average_stars + fans + review_count + compliment_more + compliment_profile +
compliment_list + compliment_note + compliment_plain + compliment_writer, data =
train.data, boos=TRUE, mfinal=50)
```

```
summary(model_adaboost)

# Load data for Regression (this one is stored on R)
#use model to make predictions on test data
pred_ada_test = predict(model_adaboost, test.data)

# Returns the prediction values of test data along with the confusion matrix
pred_ada_test
```