EC349 Assignment Code

Travis Tan

2023-12-04

R Script Code

```
setwd("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/R projects/EC349-Assignment")

library(jsonlite)


library(tidyverse)
library(lubridate)
library(caret)
library(glmnet)
library(tree)
library(rpart)
library(rpart.plot)
library(ipred)
library(randomForest)
library(adabag)


#Convert data into R_data file for more efficient storage and ram usage.
user_data <- stream_in(file("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/Assignment 1/As
save(user_data, file = "user_data.Rdata")
rm(user_data)
load("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/R projects/EC349-Assignment/user_data.

summary(user_data$yelping_since)
# date in character, convert into dates
user_data <- user_data %>%
  mutate(yelping_since=ymd_hms(yelping_since))
summary(user_data$yelping_since)
# Convert None to NA in friends otherwise difficult to calculate friends:
user_data <- user_data %>%
  mutate(friends = if_else(friends == "None", NA, friends))
# Count how many friends a user has:
user_data <- user_data %>%
  mutate(friend_count = 1 + str_count(friends, ","))
# Convert NA in friend count into 0
user_data <- user_data %>%
  mutate(friend_count = if_else(is.na(friend_count), 0, friend_count))
# Need to convert 20,20 in yelping_since into 2020 in elite
```

```r
user_data <- user_data %>%
  mutate(elite = str_replace_all(elite, "20,20", "2020"))
# Need to convert empty strings into NA's in elite to calculate elite years, otherwise because of my 1
user_data <- user_data %>%
  mutate(across(elite, ~na_if(., "")))
# Count number of years user was elite for
user_data <- user_data %>%
  mutate(elite_year_count = 1 + str_count(elite, ","))
# Convert NA's into 0 for Elite years
user_data <- user_data %>%
  mutate(elite_year_count = replace_na(elite_year_count, 0))
#Could consider making a variable for each elite year and checking whether that improves performance, j
# Total compliments for every user
user_data <- user_data %>%
  mutate(total_compliments = pmap_dbl(select(., 12:22), ~sum(c(...), na.rm = TRUE)))
#Find account age, too crude? Might miss days!
#yelping_start <- year(user_data_cleaned$yelping_since)
#user_data <- user_data %>%
#  mutate(account_age = 2023 - yelping_start)
user_data <- user_data %>%
  mutate(
    yelping_since_date = as.Date(yelping_since),
    years_on_yelp = as.numeric(difftime(ymd("2023-01-01"), yelping_since_date, units = "days")) / 365.25
  )
# Renaming user variables so they can be joined well with other data sets:
colnames(user_data)[3] <- "user_review_count"
colnames(user_data)[5] <- "total_useful"
colnames(user_data)[6] <- "total_funny"
colnames(user_data)[7] <- "total_cool"
colnames(user_data)[10] <- "total_fans"
colnames(user_data)[11] <- "user_average_stars"

# # Base R, was very annoying to do, should have done dplyr like below
# user_data_cleaned <- user_data[, c("user_id", "review_count", "yelping_since", "useful", "funny", "co
#                                    "elite_year_count", "friend_count", "total_compliments")]

# Definitely do this next time!
user_data_cleaned <- user_data %>%
  select(user_id, user_review_count, years_on_yelp, total_useful, total_funny, total_cool, user_average_
         total_fans, compliment_hot, compliment_more, compliment_profile, compliment_cute, compliment_li
         compliment_cool, compliment_funny)
user_data_cleaned$yelping_since <- NULL

#Rda files less ram intensive and more efficient storage!
save(user_data_cleaned, file = "user_data_cleaned.Rdata")
load("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/R projects/EC349-Assignment/user_data_

#Check for outliers with review and

business_data <- stream_in(file("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/Assignment
save(business_data, file = "business_data.Rdata")
load("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/R projects/EC349-Assignment/business_da
```

```r
business_data <- business_data %>%
  mutate(city = as.factor(city))
business_data <- business_data %>%
  mutate(state = as.factor(state))
#business_data <- business_data %>%
#  mutate(categories=as.factor(categories))

# finding categories as factors looks kinda useless, might need to instead count how many categories a


# Counting how many non-empty attributes a business has, the more famous/ubiquitous business should hav
# they have dedicated staff or is more innovative in keeping up with the trend
# This means number of non-empty attributes could be seen as a measure of engagement with yelp?
# Could be problematic because no one really goes to fill out all 39 attributes including unrelated one
# Unless this also carries information about how big the place is because the more general purpose the
b_attributes <- business_data %>%
  select(attributes)
business_data <- business_data %>%
  mutate(engagement = rowSums(!is.na(b_attributes)))

# Check how businesses choose to fill the attributes according to categories?
business_engagement <- business_data %>%
  select(name, review_count, categories, engagement)

# Find category count and see how it's distributed:

# business_data <- business_data %>%
#   mutate(category_count = 1 + str_count(categories, ","))
# ggplot(business_data, aes(x = category_count)) +
#   geom_histogram(binwidth = 1, fill = "blue", color = "black") +
#   labs(title = "Histogram of Category Count", x = "Category Count", y ="Frequency")
# ggplot(business_data, aes(x= engagement, y = category_count)) +
#   geom_point() +
#   labs(title = "Engagement vs Category count", x = "Engagement count", y = "Category_count")

# Categories can be attached by algorithm for related terms so may not be useful?
# Looks pretty useless, instead categorise the businesses, bars & restaurants, financial services etc.?
# Try creating a character vector to add up all the words between commas and then use unique function t

categories <- paste(business_engagement$categories, collapse = ",")
categories <- strsplit(categories, ",")
categories <- lapply(categories, str_trim)
unique_categories <- unique(unlist(categories))
print(unique_categories)
unique_categories <- str_replace_all(unique_categories, " ", "")
unique_categories_matrix <- matrix(unique_categories, ncol = 1, byrow= TRUE)
unique_categories <- as.data.frame(unique_categories_matrix)

# There are 1300+ unique categories, how would I do this? Should I check which categories are most enga
# unique_categories_list <- list(unique_categories)
# named_unique_categories <- setNames(as.list(unique_categories), unique_categories)
# unique_categories_df <- data.frame(named_unique_categories)
```

```r
# Clearly too many variables, R cannot handle it and too many parameters anyways, try another way
# Give up, go to Yelp category list, just take main categories form Yelp, can analyse further if I have
#
# Add fashion and restaurants and bars for special cases because they have great engagement


business_data <- business_data %>%
  mutate(Active_Life = if_else(str_detect(categories, "Active Life"), 1, 0),
         Arts_Entertainment = if_else(str_detect(categories, "Arts & Entertainment"), 1, 0),
         Automotive = if_else(str_detect(categories, "Automotive"), 1, 0),
         Beauty_Spas = if_else(str_detect(categories, "Beauty & Spas"), 1, 0),
         Education = if_else(str_detect(categories, "Education"), 1, 0),
         Event_Planning_Services = if_else(str_detect(categories, "Event Planning & Services"), 1, 0),
         Financial_Services = if_else(str_detect(categories, "Financial Services"), 1, 0),
         Food = if_else(str_detect(categories, "Food"), 1, 0),
         Health_Medical = if_else(str_detect(categories, "Health & Medical"), 1, 0),
         Home_Services = if_else(str_detect(categories, "Home Services"), 1, 0),
         Hotels_Travel = if_else(str_detect(categories, "Hotels & Travel"), 1, 0),
         Local_Flavor = if_else(str_detect(categories, "Local Flavor"), 1, 0),
         Local_Services = if_else(str_detect(categories, "Local Services"), 1, 0),
         Mass_Media = if_else(str_detect(categories, "Mass Media"), 1, 0),
         Nightlife = if_else(str_detect(categories, "Nightlife"), 1, 0),
         Pets = if_else(str_detect(categories, "Pets"), 1, 0),
         Professional_Services = if_else(str_detect(categories, "Professional Services"), 1, 0),
         Public_Services_Government = if_else(str_detect(categories, "Public Services & Government"), 1
         Real_Estate = if_else(str_detect(categories, "Real Estate"), 1, 0),
         Religious_Organizations = if_else(str_detect(categories, "Religious Organizations"), 1, 0),
         Restaurants = if_else(str_detect(categories, "Restaurants"), 1, 0),
         Shopping = if_else(str_detect(categories, "Shopping"), 1, 0)
  )

# Further refine into this:

business_data <- business_data %>%
  mutate(Food = if_else(str_detect(categories, "Food"), 1, 0),
         Nightlife = if_else(str_detect(categories, "Nightlife"), 1, 0),
         Shopping = if_else(str_detect(categories, "Shopping"), 1, 0),
         Bars = if_else(str_detect(categories, "Bars"), 1, 0),
         Fashion = if_else(str_detect(categories, "Fashion"), 1, 0),
         Restaurants = if_else(str_detect(categories, "Restaurants"), 1, 0),
         Grocery = if_else(str_detect(categories, "Grocery"), 1, 0)
  )



# Also need to figure out what to do with opening hours.Some businesses don't bother filling out the ti
# Either exclude NA observations with opening times or match opening days to the other most similar bus
# business_hours <- business_data %>%
#   select(name, review_count, city, state, postal_code, is_open, categories, hours)

# Trying a crude method of isolating observations with all NA's for opening hours
b_hours <- business_data %>%
  select(hours)
```

```r
business_data <- business_data %>%
  mutate(days_open = rowSums(!is.na(b_hours)))
hours_NA <- business_data %>%
  filter(days_open == 0)

# There are 23k observations with all NA's not feasible to match opening days to other most similar bus
# Perhaps try certain conditions and have them match! Very problematic, even if I want to match a hot d
# My categories are much too vague for me to try to match to specifically similar businesses.
# open_less_than_3 <- business_hours %>%
#   filter(days_open > 0 & days_open < 3)

# These are the problematic businesses, even closed businesses do list opening hours, there is just bad
# Going online to check, one can easily check with google there is data collection problem here as some
# open_3 <- business_hours %>%
#   filter(days_open == 3) # More often than not, 3 days open is set manually and the other days are cl

# open_2 <- business_hours %>%
#   filter(days_open == 2) # Doing some google searches, it would seem these opening hours are also set

# Owners tend to change them back to more normal 4/5-day-weeks, what to do in this case?
# Could they be outliers in this case? Should I drop them?
# Don't drop them, even if there are some outliers, dropping them when there are also business owners w
# Try both ways and see what happens? Keep it first and fit a regression tree with it, for now we use d
# Just drop days_0 observations when doing bagging.

# Figure out business attributes now:
# business_attributes <- business_data %>%
#   select(name, categories, state, postal_code, attributes)
# unique_smoking <- unique(business_attributes$attributes$Smoking)
# print(unique_smoking)


# ggplot(business_data, aes(x = engagement)) +
#   geom_histogram(binwidth = 1, fill = "red", colour = "green") +
#   labs(title = "Histogram of Engagement Count", x = "Engagement count", y = "Frequency")
# Inspect categories:
# business_categories <- business_data %>%
#   select(name, categories, engagement)

# Inspect attributes:
# business_attributes <- business_data %>%
#   select(name, categories,attributes)
# Might need to go in and manually fish out useful attributes later on!

# Since I've chosen to refine the categories chosen, I should prove my claims with a graph and data, fi
category_dummy_indices <- 16:40
means_engagement <- sapply(category_dummy_indices, function(i) mean(business_data$engagement[business_da
engagement_means <- data.frame(category_dummy = names(business_data)[category_dummy_indices], mean = mea

engagment_by_category_plot <- ggplot(engagement_means, aes(x = category_dummy, y = mean)) +
  geom_bar(stat = "identity", width =0.6) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```r
  labs(title = "Mean of Engagement by Business Category", x = "Business Category", y = "Mean")

ggsave("Engagement by category.png", plot = engagment_by_category_plot, width = 10, height = 8)


# Means businesses' average star by category
# category_dummy_indices <- 16:40
# means_biz_avgstars <- sapply(category_dummy_indices, function(i) mean(business_data$business_stars[bu
# avgstars_byCAT <- data.frame(category_dummy = names(business_data)[category_dummy_indices], mean = me
#
# ggplot(avgstars_byCAT, aes(x = category_dummy, y = mean)) +
#   geom_bar(stat = "identity", width =0.4) +
#   theme_minimal() +
#   theme(axis.text.x = element_text(angle = 45, hjust = 1))
#   labs(title = "Average Stars by Business Category", x = "Business Category", y = "Mean")
# plots



# Renaming business variables so no confusion when joining
colnames(business_data)[9] <- "business_stars"
colnames(business_data)[10] <- "business_review_count"

# business_categories <- business_data %>%
#   select(name, categories, Arts_Entertainment, Automotive, Beauty_Spas, Education, Event_Planning_Ser
#          Local_Services, Mass_Media, Nightlife, Pets, Professional_Services, Public_Services_Governme
#          Restaurants, Shopping)
# Looks like some businesses didn't fill in categories, best to exclude them since they're relatively r
# be outliers
no_categories <- business_data %>%
  filter(is.na(categories)) # Only 103 observations, safe to drop!

business_data <- business_data %>%
  filter(!is.na(categories))


checkin_data  <- stream_in(file("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/Assignment

# Check which businesses have checkins
checkin_data <- checkin_data %>%
  mutate(checkin_count = 1 + str_count(date, ","))
checkin_data$date <- NULL
business_data <- left_join(business_data, checkin_data, by = "business_id")

sum(is.na(business_data$checkin_count))
# 18359 have NA's, that means no one checked in? Need to verify whether there are 0's in the dataset!Ca
summary(business_data$checkin_count) # Min = 1, so NA means 0 checkin's
business_data <- business_data %>%
  mutate(checkin_count = if_else(is.na(checkin_count), 0, checkin_count))

# business_data_cleaned <- business_data %>%
#   select(business_id, city, state, latitude, longitude, business_stars, business_review_count, is_ope
#          Automotive, Beauty_Spas, Education, Event_Planning_Services, Financial_Services, Food, Healt
```

```r
#          Local_Flavor, Local_Services, Mass_Media, Nightlife, Pets, Professional_Services, Public_Ser
#          Religious_Organizations, Restaurants, Shopping)

# Category refinement
business_data_prepped <- business_data %>%
  select(business_id, state, latitude, longitude, business_stars, business_review_count, engagement,
         Arts_Entertainment, Bars, Event_Planning_Services, Food, Grocery, Local_Flavor, Nightlife, Res


save(business_data_prepped, file = "business_data_prepped.Rdata")
load("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/R projects/EC349-Assignment/business_da


# review_data <- stream_in(file("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/Assignment
# save(review_data, file = "review_data.Rdata")
# rm(review_data)
# load("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/R projects/EC349-Assignment/review_d
# 6990247 obs
# 7 million observations too big, can't run random forest!
# days_0 <- review_data %>%
#   filter(days_open == 0) # Nearly 400k reviews with business with no opening hours, could drop days_o
# review_data <- review_data %>%
#   filter(days_open > 0) # I don't believe days open truly affects how well a user may rate a business


# Originally worked with big data, RF couldn't run
load("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/Assignment 1/Assignment/Small Datasets/
review_data_small$review_id <- NULL
review_data_small$text <- NULL
save(review_data_small, file = "review_data_small_cleaned.Rdata")
load("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/R projects/EC349-Assignment/review_data

# Converting review dates into review age
review_data_small <- review_data_small %>%
  mutate(date = ymd_hms(date))
review_data_small <- review_data_small %>%
  mutate(
    review_since_date = as.Date(date),
    review_age = as.numeric(difftime(ymd("2023-01-01"), review_since_date, units = "days")) / 365.25, #
  )
review_data_small$date <- NULL
review_data_small$review_since_date <- NULL

tip_data <- stream_in(file("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/Assignment 1/Assi

# 908915 obs for tip data, trying to find how many tips each user and each business gave!
# To calculate total number of tips a user gave out, make user_id in tip data factor and find how many
tip_data <- tip_data %>%
  mutate(as.factor(user_id),
         as.factor(business_id))
# user_tip_count computation
user_level_counts <- table(tip_data$`as.factor(user_id)`)
user_tip_count <- as.data.frame(user_level_counts)
```

```r
colnames(user_tip_count)[1] <- "user_id"
colnames(user_tip_count)[2] <- "user_tip_count"
user_tip_count <- user_tip_count %>%
  mutate(user_id = as.character(user_id)) # Convert from factor back into character for joining later

# business_tip_count computation
business_level_counts <- table(tip_data$`as.factor(business_id)`)
business_tip_count <- as.data.frame(business_level_counts)
colnames(business_tip_count)[1] <- "business_id"
colnames(business_tip_count)[2] <- "business_tip_count"
business_tip_count <- business_tip_count %>%
  mutate(business_id = as.character(business_id))

# Tip count files
save(user_tip_count, file = "user_tip_count.Rdata")
save(business_tip_count, file = "business_tip_count.Rdata")

# Joining review and tip data
review_data_small <- left_join(review_data_small, user_tip_count, by = "user_id")
review_data_small <- left_join(review_data_small, business_tip_count, by = "business_id")
# Verifying
sum(is.na(review_data_small$user_tip_count)) # Previously 4184814 reviews in big data, 837536 in small,
review_data_small <- review_data_small %>%
  mutate(user_tip_count = if_else(is.na(user_tip_count), 0, user_tip_count))
sum(is.na(review_data_small$business_tip_count)) # Previously 492433 reviews in big data, 98305 in smal
review_data_small <- review_data_small %>%
  mutate(business_tip_count = if_else(is.na(business_tip_count), 0, business_tip_count))

# Load prepped and cleaned user and biz data
load("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/R projects/EC349-Assignment/user_data_
load("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/R projects/EC349-Assignment/business_da


# inner_join review data to user and businesses so we don't generate empty reviews by users or business
review_data_small_prepped <- inner_join(review_data_small, user_data_cleaned, by = "user_id")
review_data_small_prepped <- inner_join(review_data_small_prepped, business_data_prepped, by = "business
# Verify for no NA values in any columns
na_columns <- sapply(review_data_small_prepped, function(x) any(is.na(x)))
print(na_columns)
review_data_small_prepped$user_id <- NULL
review_data_small_prepped$business_id <- NULL
# review_data_small_prepped$total_compliments <- NULL # Old variable from summing up all compliments


save(review_data_small_prepped, file = "review_data_small_prepped.Rdata")

# This is where I would clear memory for linear regression, reload libraries and only load this file so
rm(list = ls())
load("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/R projects/EC349-Assignment/review_data

library(jsonlite)
library(tidyverse)
library(lubridate)
```

```r
library(caret)
library(glmnet)
library(tree)
library(rpart)
library(rpart.plot)
library(ipred)
library(randomForest)
library(adabag)

review_data_small_prepped$state <- NULL


# SCALING for LASSO
review_data_small_prepped_scaled <- review_data_small_prepped
review_data_small_prepped_scaled$stars <- NULL
review_data_small_prepped_scaled <- scale(review_data_small_prepped_scaled)
review_data_small_prepped_scaled <- as.data.frame(review_data_small_prepped_scaled)
review_data_small_prepped_scaled$stars <- review_data_small_prepped$stars

set.seed(1)
# Create a separate partition within the data frame of the original data set to be called upon to creat
# The separate partition can be treated like a separate list and so can be called with a name in the ro
scaled_review_partition <- createDataPartition(y = review_data_small_prepped_scaled$stars, 1, p = 0.75)
# The partition is separate in the data frame of user and thus needs to be called when referring to the
scaled_training_set <- review_data_small_prepped_scaled[scaled_review_partition[[1]], ]
scaled_test_set <- review_data_small_prepped_scaled[-scaled_review_partition[[1]], ]

vars_excluded <- c("stars")
x <- as.matrix(scaled_training_set[,setdiff(names(scaled_training_set), vars_excluded)])
y <- scaled_training_set$stars

grid <- 10^seq(0,-15, length=150)
cv.scaled.lasso <- cv.glmnet(x, y, alpha=1, lambda = grid, intercept = FALSE)
print(cv.scaled.lasso)
plot(cv.scaled.lasso)
print(cv.scaled.lasso$lambda.min) # minLambda =2.527462e-15
lambda.lasso <- cv.scaled.lasso$lambda.min
#dcGmatrix needs to be converted into proper matrix before conversion into dataframe to be viewed
#matrix is stored as a value and cannot be viewed and saved as data directly to do that you need to con
coef.scaled.lasso <- as.matrix(coef(cv.scaled.lasso, s= lambda.lasso))
coef.scaled.lasso.df <- as.data.frame(coef.scaled.lasso)
colnames(coef.scaled.lasso.df)[1] <- "Scaled LASSO coefficients"
save(coef.scaled.lasso.df, file="scaled LASSO coefficients.Rdata")


# Ridge attempt
grid <- 10^seq(0,-15, length=150)
cv.scaled.ridge <- cv.glmnet(x, y, alpha=0, lambda = grid, intercept = FALSE)
print(cv.scaled.ridge)
plot(cv.scaled.ridge)
print(cv.scaled.ridge$lambda.min) # minLambda =2.527462e-15
lambda.lasso <- cv.scaled.ridge$lambda.min
#dcGmatrix needs to be converted into proper matrix before conversion into dataframe to be viewed
```

```r
#matrix is stored as a value and cannot be viewed and saved as data directly to do that you need to con
coef.scaled.ridge <- as.matrix(coef(cv.scaled.ridge, s= lambda.lasso))
coef.scaled.ridge.df <- as.data.frame(coef.scaled.ridge)
colnames(coef.scaled.ridge)[1] <- "Scaled Ridge coefficients"
save(coef.scaled.ridge.df, file="scaled Ridge coefficients.Rdata")




# Re-clear memory! Reload libraries and only load this file so environment is cleaner!
rm(list = ls())
load("C:/Users/Travis Tan/OneDrive - University of Warwick/EC349/R projects/EC349-Assignment/review_data

library(jsonlite)
library(tidyverse)
library(lubridate)
library(caret)
library(glmnet)
library(tree)
library(rpart)
library(rpart.plot)
library(ipred)
library(randomForest)
library(adabag)

review_data_small_prepped <- review_data_small_prepped %>%
  mutate(stars=as.factor(stars))

set.seed(1)
# Create a separate partition within the data frame of the original data set to be called upon to creat
# The separate partition can be treated like a separate list and so can be called with a name in the ro
review_partition <- createDataPartition(y = review_data_small_prepped$stars, 1, p = 0.75)
# The partition is separate in the data frame of user and thus needs to be called when referring to the
review_training_set <- review_data_small_prepped[review_partition[[1]], ]
review_test_set <- review_data_small_prepped[-review_partition[[1]], ]


model_RF<-randomForest(stars~.,data=review_training_set, ntree=100)
mean(model_RF[["err.rate"]]) # 0.5589551 mean oob error
pred_RF_test = predict(model_RF, review_test_set)
actual_values <- review_test_set$stars
test_set_error_rate <- mean(pred_RF_test != actual_values) # 0.4033255 test set error rate

model_RF_200<-randomForest(stars~.,data=review_training_set, ntree=200)
mean(model_RF_200[["err.rate"]]) # 0.5526197 mean oob error
pred_RF_test_200 = predict(model_RF_200, review_test_set)
actual_values <- review_test_set$stars
test_set_error_rate_200 <- mean(pred_RF_test_200 != actual_values) # 0.4007445 test set error rate

model_adaboost <- boosting(stars~ ., data=review_training_set, boos=TRUE, mfinal=50)
actual_values <- review_test_set$stars
pred_test_ada50 <- predict(model_adaboost, review_test_set)
```

```r
test_error_rate_ada50 <- mean(pred_test_ada50$class != actual_values)
print(test_error_rate_ada50) # test_error_rate_ada50= 0.4530255

model_adaboost_stump100 <- boosting(stars~ ., data=review_training_set, boos=TRUE, mfinal=100)
actual_values <- review_test_set$stars
pred_test_ada100 <- predict(model_adaboost_stump100, review_test_set)
test_error_rate_ada100 <- mean(pred_test_ada100$class != actual_values)
print(test_error_rate_ada100) # test_error_rate_ada100= 0.4529025 diminishing returns to increasing num

bag <- bagging(stars~., data=review_training_set, nbagg = 50,
               coob = TRUE, control = rpart.control(minsplit = 2, cp = 0.1)
)
actual_values <- review_test_set$stars
pred_test_bag <- predict(bag, review_test_set)
test_set_error_rate_cp.1 <- mean(pred_test_bag$class != actual_values)
print(test_set_error_rate_cp.1) # 0.4803091 = test_set_error_rate_cp.1
print(pred_test_bag$error) # 0.4803091 mean error rate
```