EC349 Project

Wai Yan

2023-12-03

1. Introduction

In the early 2000s, the significance of online user reviews and their impact on product sales gained widespread recognition (Chevalier and Mayzlin 2006). Leveraging this information source has become instrumental for businesses in understanding consumer behaviour and enhancing profitability. Notably, the application of machine learning, particularly deep learning, has become prevalent in analysing user reviews, with a particular emphasis on the hospitality and tourism industry (Dickinger and Mazanec 2015; Alaei, Becken, and Stantic 2019). One of the common approaches is sentiment analysis which involves categorising the textual content into positive, neutral, and negative sentiments (Schmunk et al. 2013). To better predict the user reviews, sentiment, captured by the count of positive and negative words in the review text, is introduced as one of the variables, alongside other variables available in the datasets.

2. Related work

Literature in review rating prediction using Yelp datasets was well-documented, with an extensive evaluation of both machine learning and deep learning models (Elkouri 2015; Asghar 2016; A. Rafay, M. Suleman, and A. Alim 2020; S. Liu 2020; Z. Liu 2020). Surprisingly, most researchers found that logistic regression models tend to outperform other supervised learning algorithms such as random forest, support vector machine (SVM) and Naive Bayes Classifier. They yield accuracy ranged between 64-65% with some variations in training times (Elkouri 2015; S. Liu 2020; Z. Liu 2020; Asghar 2016). However, there are mixed findings in the comparison between machine learning and deep learning methods[1].

Additionally, in terms of text pre-processing, the td-idf transformer or vectorizer was used to generate weight of each word, as the length of the text may affect the count for “buzzwords” (Elkouri 2015; Asghar 2016; A. Rafay, M. Suleman, and A. Alim 2020; S. Liu 2020; Z. Liu 2020). Also, more sophisticated natural language processing models were employed to extract the true meaning of textual data (Leung, Chan, and Chung 2006; Pang, Lee, et al. 2008; Qu, Ifrim, and Weikum 2010).

3. Data setup

3.1 Data preparation

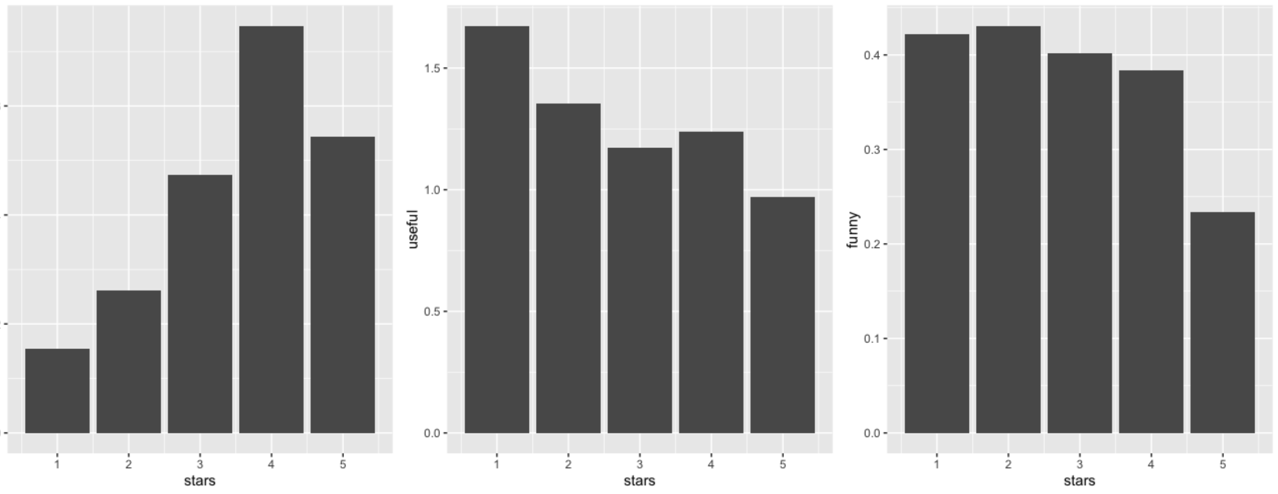
In terms of sentiment analysis, a structured pre-processing approach, as suggested in previous literature, was employed to enhance the classification of textual data (Noori 2021). Using a curated list of prevalent positive and negative words commonly employed by online users[2], the text in review data was converted into lower case, with punctuation, digits and English “Stop words[3]” removed. Here are some examples of negative and positive words:

Positive words	Negative words
winning	rankle
exceedingly	slowed
magnanimous	pitiless
unforgettable	skeptically
faithfulness	unreasonably
sweet	stubbornly
amiabily	scandalized
agreeably	mendacious
merciful	disrupt
ethical	imbecile

[1]Mostly Long Short Term Memory (LSTM) and Bidirectional Encoder Representations from Transformers (BERT)
[2]Available online at github:
- positive-text.txt: https://gist.github.com/mkulakowski2/4289437#file-positive-words-txt-L9
- negative-text.txt: https://gist.github.com/mkulakowski2/4289441
[3]Words like ‘the’, ‘that’, ‘is’ etc, occur frequently across reviews and are not very useful.

3.2 Feature selection

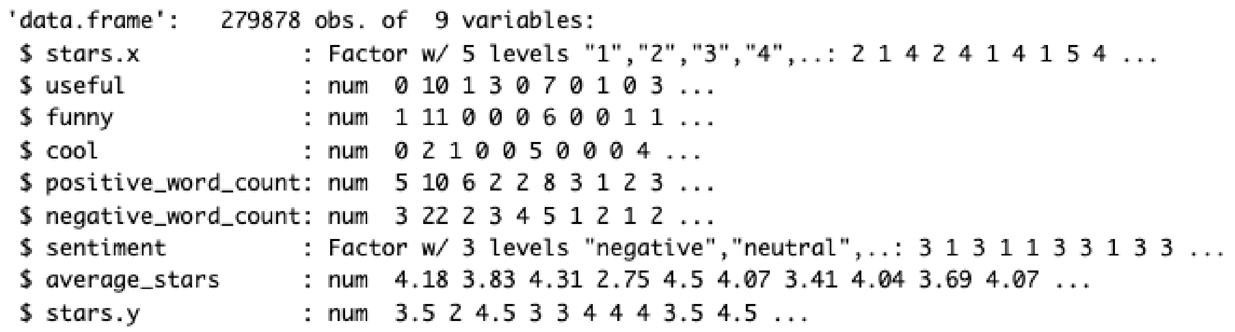
To identify relevant variables for analysis, preliminary multinomial logistic regressions were conducted, and graphs were generated to assess the associations between potential predictors and the outcome variable, which ranges from 1 to 5.The bar graphs below show the distribution of votes: useful, cool and funny.



Initially, variables such as word count of text, check-in count, sum of compliments of each user, and review count (both user and business) were considered. However, subsequent analysis revealed insignificant coefficients, leading to their exclusion from the final model.

3.3 Missing values

As the user, review and business data are merged, the missing values are omitted and the types of some variables are changed to better conduct analysis and interpret results. The training data now consists of 269878 observations. The final list of variables is as below:



```
'data.frame':   279878 obs. of  9 variables:
 $ stars.x           : Factor w/ 5 levels "1","2","3","4",..: 2 1 4 2 4 1 4 1 5 4 ...
 $ useful            : num  0 10 1 3 0 7 0 1 0 3 ...
 $ funny             : num  1 11 0 0 0 6 0 0 1 1 ...
 $ cool              : num  0 2 1 0 0 5 0 0 0 4 ...
 $ positive_word_count: num  5 10 6 2 2 8 3 1 2 3 ...
 $ negative_word_count: num  3 22 2 3 4 5 1 2 1 2 ...
 $ sentiment         : Factor w/ 3 levels "negative","neutral",..: 3 1 3 1 1 3 3 1 3 3 ...
 $ average_stars     : num  4.18 3.83 4.31 2.75 4.5 4.07 3.41 4.04 3.69 4.07 ...
 $ stars.y           : num  3.5 2 4.5 3 3 4 4 4 3.5 4.5 ...
```

where stars.x is the user rating and stars.y is the business rating.

4. Methodology

4.1 Identification strategy: Random Forest

Random forest provides a comprehensive prediction model to prevent overfitting due to sensitivity towards outliers and noises (Breiman 2001). Imbalanced class, characterised by huge differences in class probabilities, can impact the performance of prediction model especially decision trees (Japkowicz and Stephen 2002; Chawla 2003; Muchlinski et al. 2016). In this case where the ratings of 4 and 5 are exceptionally high compared to other ratings (as shown below), the use of random forest can be rationalised.

```r
actual_frequency <- read.csv("actual_frequency_table.csv")
kable(actual_frequency, caption = "Actual frequency")
```

Table 2: Actual frequency

Actual_Outcome	Actual_Frequency
1	1547
2	803
3	1012
4	2052
5	4586

The internal feature selection and decorrelation of random forest are also applicable in this case where predictors are found to be higher correlated[4]. This is because random forest strategically limits predictor consideration at each split, mitigating the impact of a single dominant predictor (James et al. 2021).

4.2 Application

The model is specified below:

- Outcome: stars.x (user review)
- Predictors: useful, funny, cool, positive_word_count, negative_word_count, sentiment, average_stars, stars.y (business review)
- Number of trees[5]: 100
- The number of variables at each split, m is set to be the square root of number of predictors, as suggested in most general cases
- The node size is set to be 10

4.3 Results and model evaluation

The confusion matrix and overall accuracy are computed to evaluate the performance of the model. The model managed to produce 59.66% of correct predictions. The confusion matrix presented below summarises the performance of the random forest model:

```r
confusion_matrix <- read.csv("confusion_matrix.csv",
                             header = FALSE,
                             col.names = c( "Prediction", "1 star", "2 stars", "3 stars",
                                           "4 stars", "5 stars"))
kable(confusion_matrix, caption = "Random Forest Confusion Matrix")
```

[4]Variance Inflation Factor (VIF) is used to detect multicollinearity between predictors, values as high as 27 were obtained.
[5]Initial attempt was 500 trees but this was not allowed due to limited memory capacity.

Table 3: Random Forest Confusion Matrix

Prediction	X1.star	X2.stars	X3.stars	X4.stars	X5.stars
NA	1	2	3	4	5
1	1242	355	174	85	91
2	87	108	47	42	31
3	46	58	96	71	41
4	54	113	273	473	376
5	118	169	422	1381	4047

Despite achieving a marginally higher accuracy, the model's limitations in predicting ratings between 2 and 3 persisted, with results skewed towards ratings 1 and 5. The results may imply that imbalanced class still deteriorate the performance of the model to certain extent. Additionally, the constraint on the number of trees may have influenced the model's predictive performance, especially considering the scale of the dataset.

The out-of-bag estimation error, standing at 40.54%, raises concerns about the generalisation performance of the random forest model. This substantial error suggests a potential risk of overfitting, as opposed to the theoretical explanation where random forest is better in handling outliers. It is also possible that the feature selection does not completely reflect the choice of star ratings.

4.4 Comparison with Logistic Regression

Due to the limited interpretability of the random forest, the analysis was replicated using multinomial logistic regression. In this comparison, logistic regression exhibited a slightly diminished performance compared to the random forest, achieving an overall accuracy of 58.65%. These findings align with recent application in the medical field, where no significant differences were noted between the two models(Christodoulou et al. 2019).

In the logistic regression analysis, the advantage lies in its ability to show the interaction between the outcome variable and predictors. Most predictors appeared to be able to identify underlying patterns within the review data, with some degree of ambiguity in distinguishing between 4 and 5-star ratings.

```r
multinom_coef_df <- read.csv("regression_coefficients.csv")
kable(multinom_coef_df, caption = "Logistic Regression Coefficients")
```

Table 4: Logistic Regression Coefficients

X.Intercept	useful	funny	cool	positive_word_count	negative_word_count	sentimentneutral	sentimentpositive	average_stars	stars.y
-4.577626	-0.0832350	-0.0422358	0.2498174	0.1401503	-0.1105936	0.2365958	0.4106026	0.6643919	0.5061668
-7.397184	-0.2337808	-0.1580130	0.7035952	0.2566341	-0.2964977	0.3281336	0.7015634	1.2324862	0.7499524
-10.191325	-0.3337514	-0.2964029	0.9819081	0.3353011	-0.5305205	0.4222237	1.2211666	1.6748479	1.1271462
-15.139453	-0.3306019	-0.3192753	0.9673846	0.3423598	-0.7066851	0.2542999	1.2775829	2.4241066	1.8670458

(The table can be found in Regression Coefficient)

```r
log_confusion_matrix <- read.csv("log_confusion_matrix.csv", header = FALSE,
                                 col.names = c( "Prediction", "1 star", "2 stars",
                                               "3 stars", "4 stars", "5 stars"))
kable(log_confusion_matrix, caption = "Logistic Regression Confusion Matrix")
```

Table 5: Logistic Regression Confusion Matrix

Prediction	X1.star	X2.stars	X3.stars	X4.stars	X5.stars
NA	1	2	3	4	5
1	1298	433	192	91	104
2	34	39	31	24	12
3	20	27	53	33	23
4	66	85	208	281	253
5	129	219	528	1623	4194

Despite being slightly better in predicting polarity with ratings of 1 and 5, it tends to present bias towards dominant classes, resulting in less accurate predictions for 2 and 3-star ratings compared to random forest, justifying the final choice of random forest model.

5. Discussion and Conclusion

The data project faces challenges with prolonged training times and high memory requirements for both random trees and logistic regression, highlighting the need for simplified models that balance processing efficiency with performance. Additionally, it highlighted the need for a standardised approach to address imbalanced data as some existing suggestions like re-sampling and improved weights may lead to increased computational difficulties (Japkowicz and Stephen 2002; Shahhosseini and Hu 2020).

In conclusion, the analysis indicates that random forest provided a higher accuracy in classifying imbalanced classes compared to logistic regression, contrary to most of the previous literature. Despite this advantage, the random forest model does exhibit some misclassification errors, which may be attributed to overfitting and imbalanced class. However, its distinct benefits, including enhanced generalisation and decorrelation, underscore its suitability for this specific application. Considering overall accuracy, potential overfitting and class imbalance, random choice remains the most effective model to the best of my knowledge.

GitHub

(Word count: 1174)

References

A. Rafay, M. Suleman, and A. Alim. 2020. "2020 International Conference on Emerging Trends in Smart Technologies (ICETST)." In, 8138–43. https://doi.org/10.1109/ICETST49965.2020.9080713.

Alaei, Ali Reza, Susanne Becken, and Bela Stantic. 2019. "Sentiment Analysis in Tourism: Capitalizing on Big Data." *Journal of Travel Research* 58 (2): 175–91. https://doi.org/10.1177/0047287517747753.

Asghar, Nabiha. 2016. "Yelp Dataset Challenge: Review Rating Prediction." *arXiv.org*, May. http://0-search.proquest.com.pugwash.lib.warwick.ac.uk/working-papers/yelp-dataset-challenge-review-rating-prediction/docview/2079845554/se-2?accountid=14888.

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32. https://doi.org/10.1023/A:1010933404324.

Chawla, Nitesh V. 2003. "C4. 5 and Imbalanced Data Sets: Investigating the Effect of Sampling Method, Probabilistic Estimate, and Decision Tree Structure." In *Proceedings of the ICML*, 3:66. Citeseer.

Chevalier, Judith A., and Dina Mayzlin. 2006. "The Effect of Word of Mouth on Sales: Online Book Reviews." *Journal of Marketing Research* 43 (3): 345–54. https://doi.org/10.1509/jmkr.43.3.345.

Christodoulou, Evangelia, Jie Ma, Gary S. Collins, Ewout W. Steyerberg, Jan Y. Verbakel, and Ben Van Calster. 2019. "A Systematic Review Shows No Performance Benefit of Machine Learning over Logistic Regression for Clinical Prediction Models." *Journal of Clinical Epidemiology* 110: 12–22. https://doi.org/https://doi.org/10.1016/j.jclinepi.2019.02.004.

Dickinger, Astrid, and Josef A. Mazanec. 2015. "Significant Word Items in Hotel Guest Reviews: A Feature Extraction Approach." *Tourism Recreation Research* 40 (3): 353–63. https://doi.org/10.1080/02508281.2015.1079964.

Elkouri, Andrew. 2015. "Predicting the Sentiment Polarity and Rating of Yelp Reviews."

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2021. *An Introduction to Statistical Learning: With Applications in r*. Second. New York: Springer.

Japkowicz, Nathalie, and Shaju Stephen. 2002. "The Class Imbalance Problem: A Systematic Study." *Intelligent Data Analysis* 6 (5): 429–49. https://doi.org/10.3233/IDA-2002-6504.

Leung, Cane WK, Stephen CF Chan, and Fu-lai Chung. 2006. "Integrating Collaborative Filtering and Sentiment Analysis: A Rating Inference Approach." In *Proceedings of the ECAI 2006 Workshop on Recommender Systems*, 62–66.

Liu, Siqi. 2020. "Sentiment Analysis of Yelp Reviews: A Comparison of Techniques and Models."

Liu, Zefang. 2020. "Yelp Review Rating Prediction: Machine Learning and Deep Learning Models." *arXiv.org*, December. http://0-search.proquest.com.pugwash.lib.warwick.ac.uk/working-papers/yelp-review-rating-prediction-machine-learning/docview/2470136753/se-2?accountid=14888.

Muchlinski, David, David Siroky, Jingrui He, and Matthew Kocher. 2016. "Comparing Random Forest with Logistic Regression for Predicting Class-Imbalanced Civil War Onset Data." *Political Analysis* 24 (1): 87–103.

Noori, Behrooz. 2021. "Classification of Customer Reviews Using Machine Learning Algorithms." *Applied Artificial Intelligence* 35 (8): 567–88. https://doi.org/10.1080/08839514.2021.1922843.

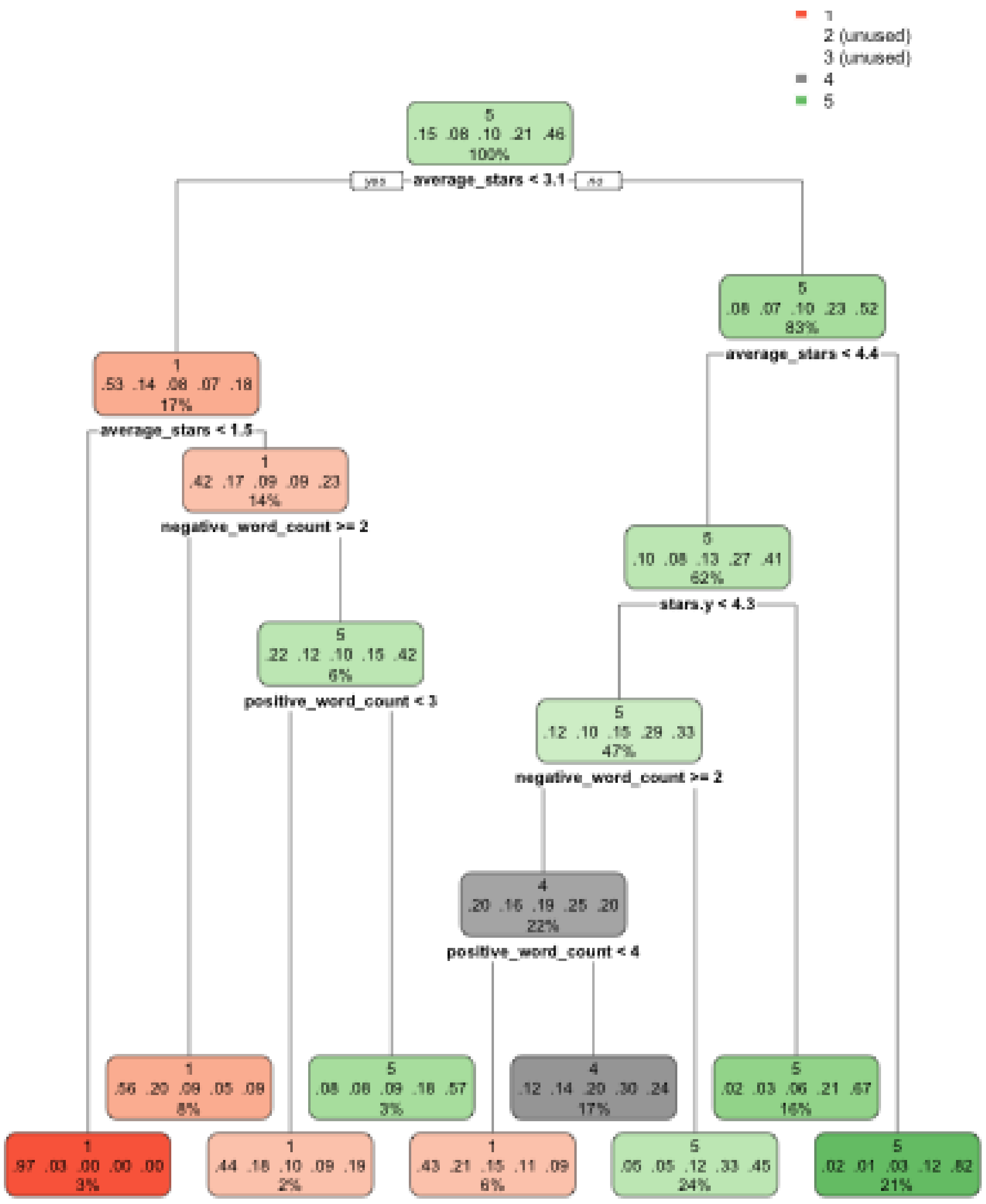
Pang, Bo, Lillian Lee, et al. 2008. "Opinion Mining and Sentiment Analysis." *Foundations and Trends® in Information Retrieval* 2 (1–2): 1–135.

Qu, Lizhen, Georgiana Ifrim, and Gerhard Weikum. 2010. "The Bag-of-Opinions Method for Review Rating Prediction from Sparse Text Patterns." In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, edited by Chu-Ren Huang and Dan Jurafsky, 913–21. Beijing, China: Coling 2010 Organizing Committee. https://aclanthology.org/C10-1103.

Schmunk, Sergej, Wolfram Höpken, Matthias Fuchs, and Maria Lexhagen. 2013. "Sentiment Analysis: Extracting Decision-Relevant Knowledge from UGC." In, edited by Zheng Xiang and Iis Tussyadiah, 253265. Cham: Springer International Publishing.

Shahhosseini, Mohsen, and Guiping Hu. 2020. "Improved Weighted Random Forest for Classification Problems." *arXiv.org*, September. https://doi.org/10.1007/978-3-030-66501-2_4.

Appendix



Decision tree was also plotted but ratings of 2 and 3 were completed neglected in this model

R script

```
#Clear
cat("\014")
rm(list=ls())

#Set Directory as appropriate
setwd("/Users/waiyan_1020/Desktop/EC349 Project")
```

```r
#Pre-Processing Yelp Academic Data for the Assignment
library(jsonlite)

#Load Different Data
business_data <- stream_in(file("/Users/waiyan_1020/Desktop
                                /EC349  Supplementary Material/yelp_academic_dataset_business.json"))
checkin_data  <- stream_in(file("/Users/waiyan_1020/Desktop
                                /EC349  Supplementary Material/yelp_academic_dataset_checkin.json"))
tip_data  <- stream_in(file("/Users/waiyan_1020/Desktop
                            /EC349  Supplementary Material/yelp_academic_dataset_tip.json"))

#Load smaller user and review data
load("/Users/waiyan_1020/Desktop/EC349  Supplementary Material/yelp_review_small.Rda")
load("/Users/waiyan_1020/Desktop/EC349  Supplementary Material/yelp_user_small.Rda")

# Sentiment Analysis
# Load word list
positive_words <- read.table("positive-words.txt", header = FALSE, stringsAsFactors = FALSE)
negative_words <- read.table("negative-words.txt", header = FALSE, stringsAsFactors = FALSE)

# Text pre-processing
library (tm)
review_data_small$text <- tolower(review_data_small$text)
review_data_small$text <- gsub("[[:punct:]]", "", review_data_small$text)
review_data_small$text <- gsub("[[:digit:]]", "", review_data_small$text)
review_data_small$text <- removeWords(review_data_small$text, stopwords("english"))

# Function to count positive or negative words in a text
count_positive_words <- function(text, positive_words) {
  words <- unlist(strsplit(text, " "))
  count <- sum(words %in% positive_words)
  return(count)
}
count_negative_words <- function(text, negative_words) {
  words <- unlist(strsplit(text, " "))
  count <- sum(words %in% negative_words)
  return(count)
}

# Add columns for positive and negative word count in review data **takes forever
review_data_small$positive_word_count <- sapply(review_data_small$text,
                                                count_positive_words, positive_words =
                                                  positive_words$V1)
review_data_small$negative_word_count <- sapply(review_data_small$text,
                                                count_negative_words, negative_words =
                                                  negative_words$V1)

# Add a column indicating sentiment based on the count
review_data_small$sentiment<- ifelse(review_data_small$positive_word_count
                                     >review_data_small$negative_word_count, "positive",
                              ifelse(review_data_small$positive_word_count
                                     <review_data_small$negative_word_count, "negative", "neutral"))
```

```r
#Select average stars in user data as variable
library(tidyverse)
user_data_small2 <- user_data_small %>%
  select(user_id, average_stars)

#Merge review and user data by user id
user_review_data <- left_join(review_data_small, user_data_small2, by = "user_id")

#Select stars in business data as variables
business_data2 <- business_data %>%
  select(business_id, stars)

#Merge business and user data
user_business_data <- left_join(user_review_data, business_data2, by = "business_id")

#Remove unused variables
user_business_data <- user_business_data %>%
  select(stars.x, useful, funny, cool, positive_word_count,
         negative_word_count, sentiment, average_stars, stars.y)

#Change variables from int to numeric; stars to factor
str(user_business_data)
user_business_data$useful <- as.numeric(user_business_data$useful)
user_business_data$funny <- as.numeric(user_business_data$funny)
user_business_data$cool<- as.numeric(user_business_data$cool)
user_business_data$stars.x <- as.factor(user_business_data$stars.x)
user_business_data$positive_word_count <- as.numeric(user_business_data$positive_word_count)
user_business_data$negative_word_count <- as.numeric(user_business_data$negative_word_count)
user_business_data$sentiment <- as.factor(user_business_data$sentiment)

#Remove missing values
sum(is.na(user_business_data))
user_business_data <- na.omit(user_business_data)

#Set seed for reproducibility
set.seed(1)

#Split the data into test and training
test_obs <- sample(1:nrow(user_business_data), 10000)
review_test <- user_business_data[test_obs, ]
review_train <- user_business_data[-test_obs, ]

#Generate frequency table for actual user review
test_actual <- table (review_test$stars.x)
actual_df <- as.data.frame(test_actual)
colnames(actual_df) <- c("Actual_Outcome", "Actual_Frequency")
write.csv(actual_df, "actual_frequency_table.csv", row.names = FALSE)

#Random forests- 100 trees**takes even longer
library(randomForest)
rf_model <- randomForest(stars.x ~., data = review_train, ntree=100,
                         mtry = sqrt(ncol(review_train)-1), nodesize = 10)
```

```r
# Print the model summary
print(rf_model)

#Fit the model
rf_pred<-predict(rf_model, newdata = review_test)
summary (rf_pred)

#Evaluate the accuracy/error rate
library(caret)
rf_conf_matrix <- confusionMatrix(rf_pred, review_test$stars.x)
rf_accuracy <- rf_conf_matrix$overall["Accuracy"]
print(rf_conf_matrix)
print(paste("Accuracy:", round(rf_accuracy, 4)))
write.table(rf_conf_matrix$table, "confusion_matrix.csv", sep = ",", col.names = NA,
            qmethod = "double")

#Compare with logistic regression again (all predictors)
library(nnet)
multinom_model <- multinom(stars.x ~., data = review_train)
summary (multinom_model)
multinom_coef_df <- as.data.frame(coef(summary(multinom_model)))
write.csv(multinom_coef_df, file = "regression_coefficients.csv", row.names = FALSE)

# Make predictions on the test set
multinom_prediction <- predict(multinom_model, newdata = review_test)

#Evaluate the accuracy/error rate
library(caret)
log_conf_matrix <- confusionMatrix(multinom_prediction, review_test$stars.x)
log_accuracy <- log_conf_matrix$overall["Accuracy"]
print(log_conf_matrix)
print(paste("Accuracy:", round(log_accuracy, 4)))
write.table(rf_conf_matrix$table, "log_confusion_matrix.csv", sep = ",", col.names = NA,
            qmethod = "double")
```