Individual Project EC349

Rowan Vinayak u2100546

2023-12-2

word count (excluding code chunks): 1341

Introduction

In this project, I predict and analyse yelp review ratings through business and user characteristics, as well as sentiment and text analysis.

I adopt a flexible approach to the **CRISP-DM** methodology. It provides a structured, comprehensive and cyclical framework, supporting the iterative nature of data science (DSPA, 2021). My flexible approach ensured that I constantly iterated through processes, allowing me to build an initial model that I could then improve from. I value the CRISP-DM methodology's emphasis on the initial *understanding* phase which helped me focus on the specific problem. Its five phases (deployment is not relevant to this assignment) are intuitive and encouraged me to critically think and ask questions throughout about my analysis.

Classification or regression problem?

Since user reviews are discrete, I employed a classification approach by transforming the outcome variable **review_star** into a factor variable with 5 levels.

I utilise confusion matrices to evaluate the random forest model's performance on the test dataset.

This report contains 4 sections:

1. Data preparation
2. Exploratory data analysis (EDA) and model selection
3. Model implementation, tuning and accuracy *(and most difficult challenge)*
4. Model evaluation

Data preparation

I merged all datasets except the tip dataset (it contained no useful information above the others).

Data Cleaning

To enhance data quality and completeness, I removed the:

- 39 attributes, each with at least 95,000 missing observations.
- hours variables - each having at least 80,000 missing observations.
- variables with high cardinality (address, postal code, latitude, longitude) since these may cause overfitting. Geographic impacts are already covered by the state variable.

I removed observations (54) with missing data.

Finally, I produced a smaller sample of 200,000 (for computational time).

I explore the impact of 9 additional variables on **review_star**:

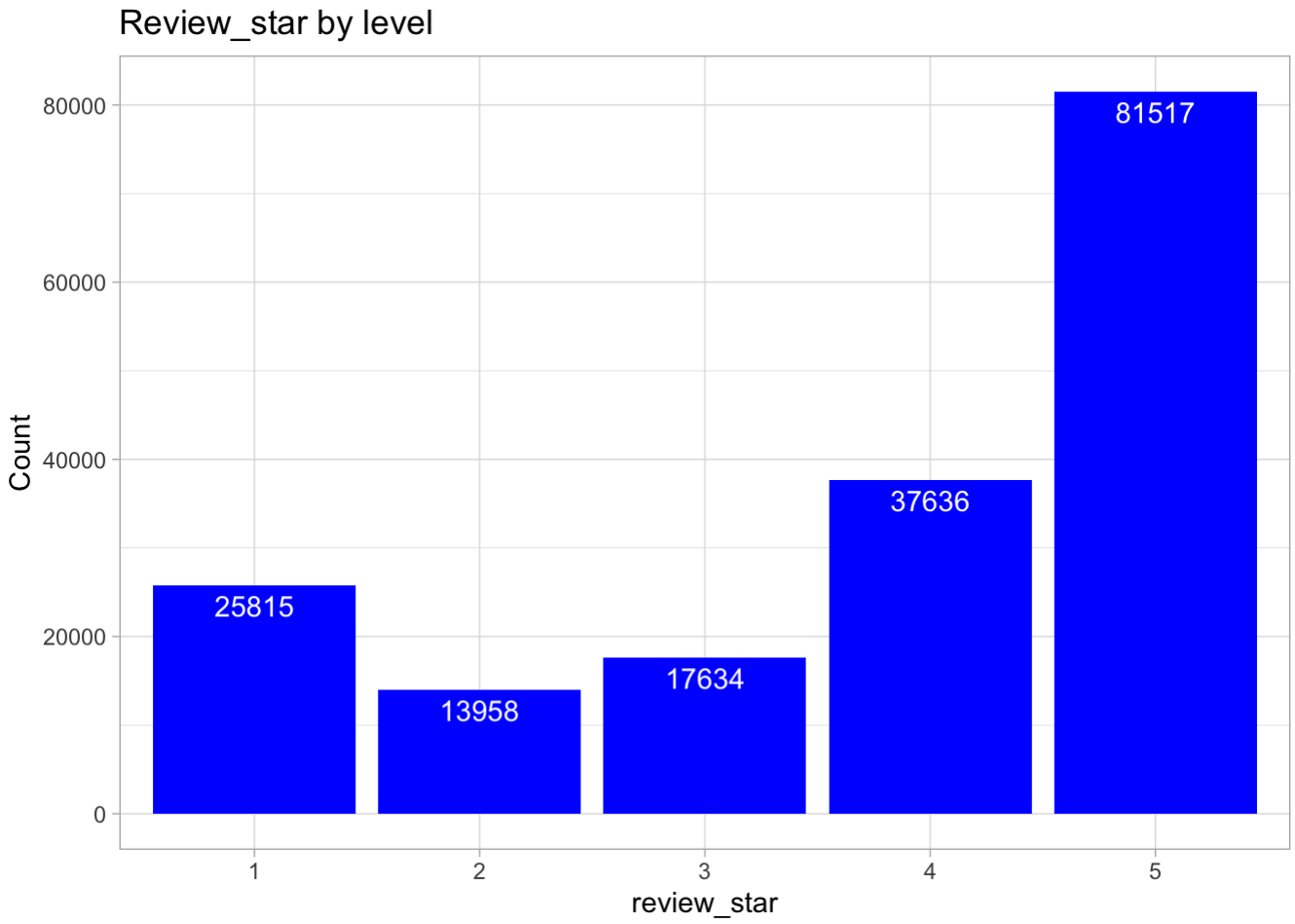
1. Length of review (words)
2. User's number of friends
3. Sentiment of review using AFINN lexicon[1] (sentiment_score)

4. Emotional sentiment of review using NRC lexicon[1]
5. Month a user has been yelping since
6. Year a user has been yelping since
7. Month of review
8. Year of review
9. Business's number of checkin dates (DateCount)

[1]I follow the text mining process outlined by Silge and Robinson (2017, ch.2).

The NRC lexicon provides data about the emotions in text, producing the variables positive, negative and 8 emotion variables (anger, sadness, fear etc.).

Exploratory data analysis (EDA) and model selection



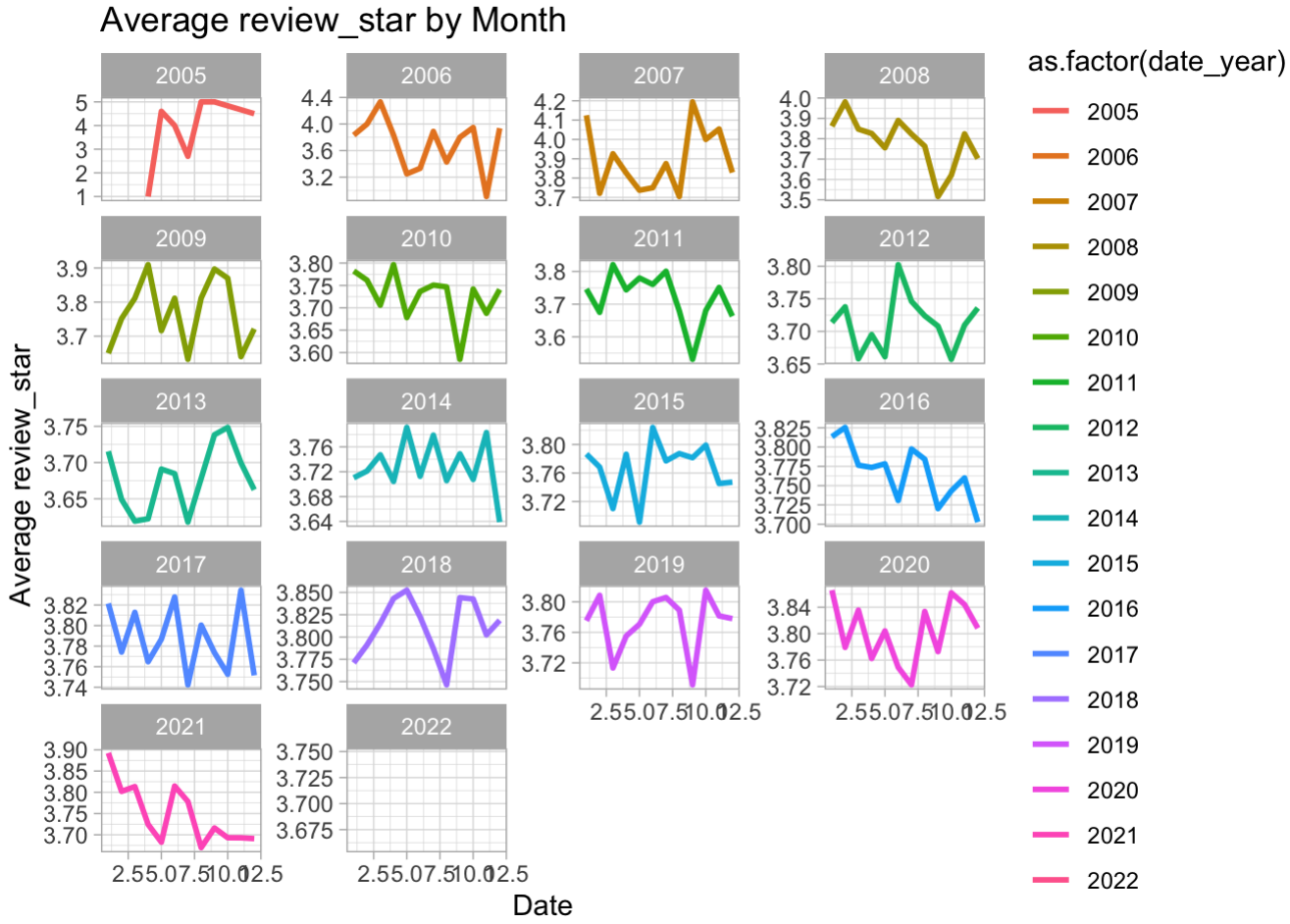
In the training dataset, **review_star** has uneven class distribution with 5 being the most common rating. A naive model predicting 5 for each review would have an accuracy of 0.4617 on the test dataset:

```
        1         2         3         4         5 
0.1462283 0.0790520 0.0998981 0.2131498 0.4616718 
```

Relationship between review_star and key predictors

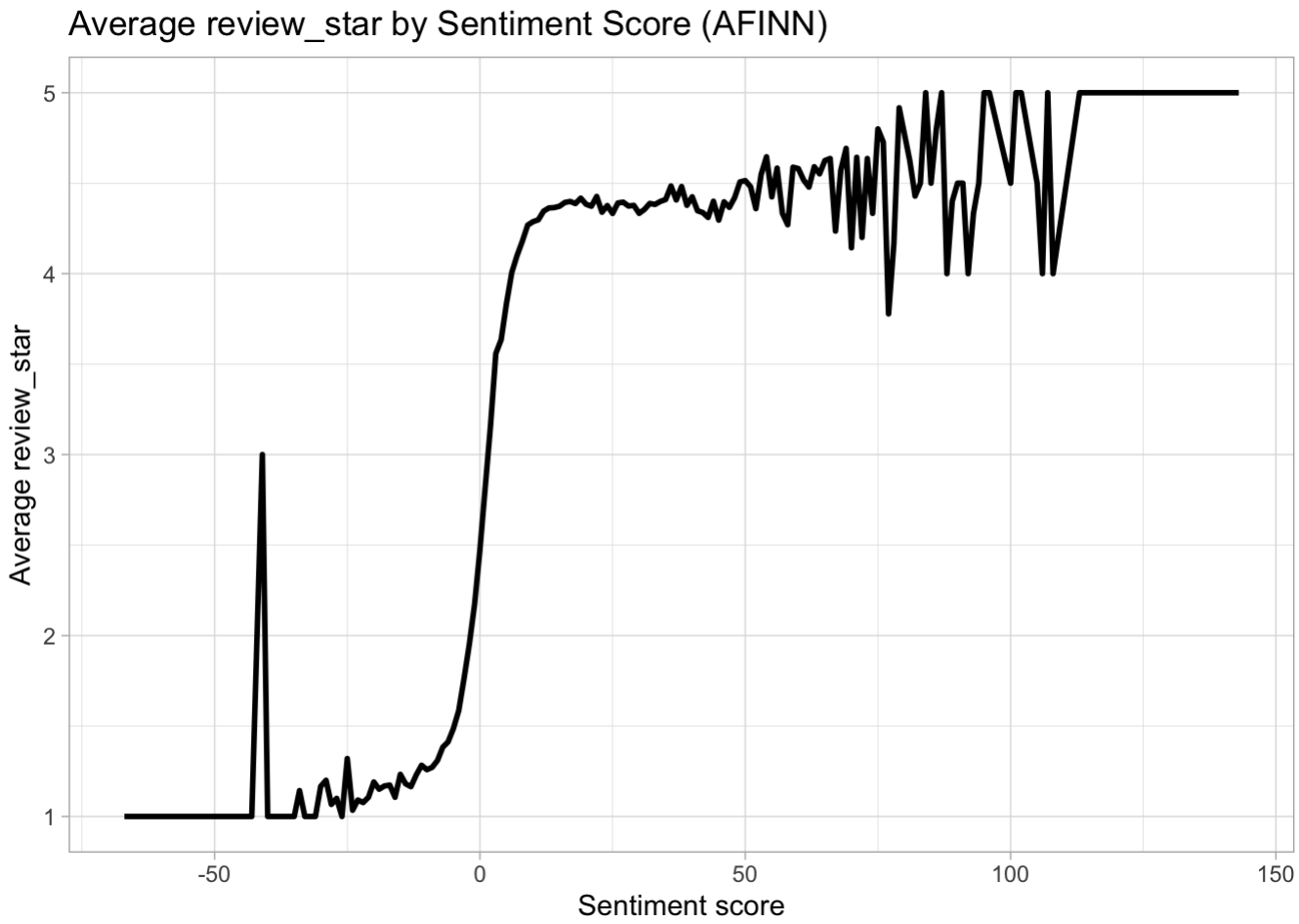
Review Date

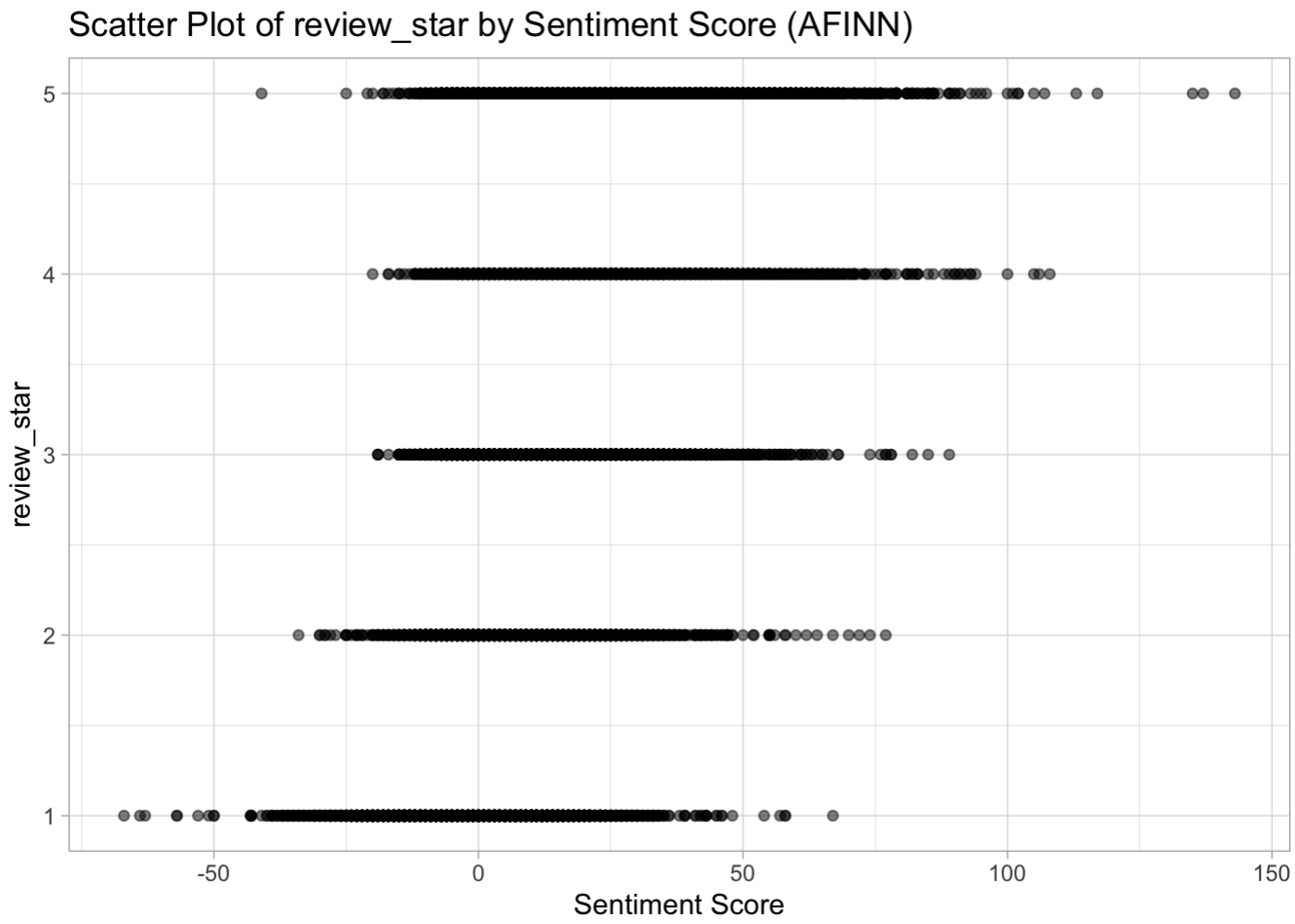
The graphs below illustrate monthly averages of **review_star** by year. Monthly fluctuations suggest some seasonality in **review_star**. For most years, the average of **review_star** dips between September and December.

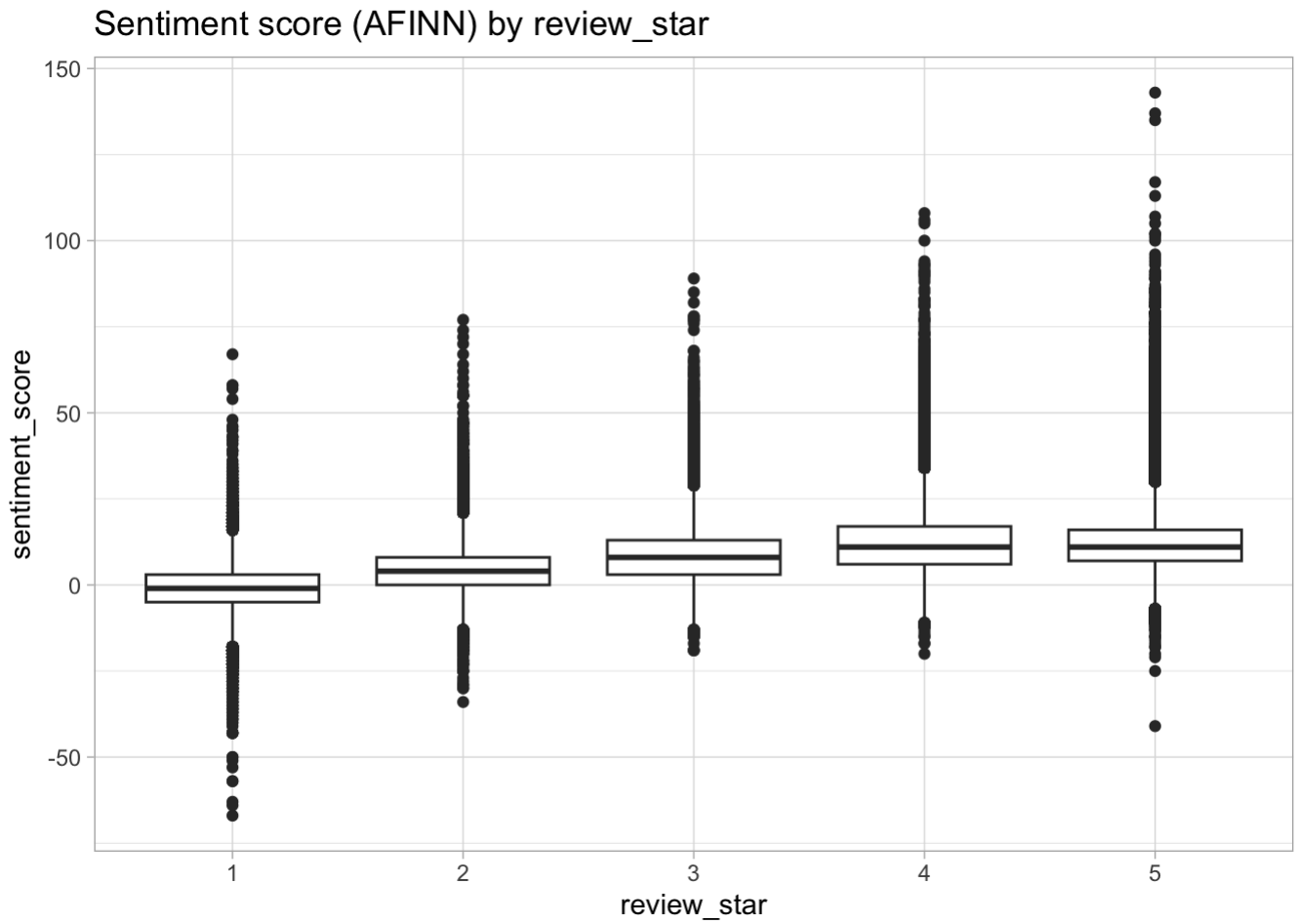


Sentiment score (AFINN)

The graphs below show a positive relationship between sentiment score and **review_star** for most observations. However, some observations with low sentiment scores have a high **review_star** and vice versa which may challenge the model. The box plot illustrates that sentiment_score does not distinguish well between 4 and 5 star reviews.


Average review_star by Sentiment Score (AFINN)
Average review_star
Sentiment score
5
4
3
2
1
-50
0
50
100
150

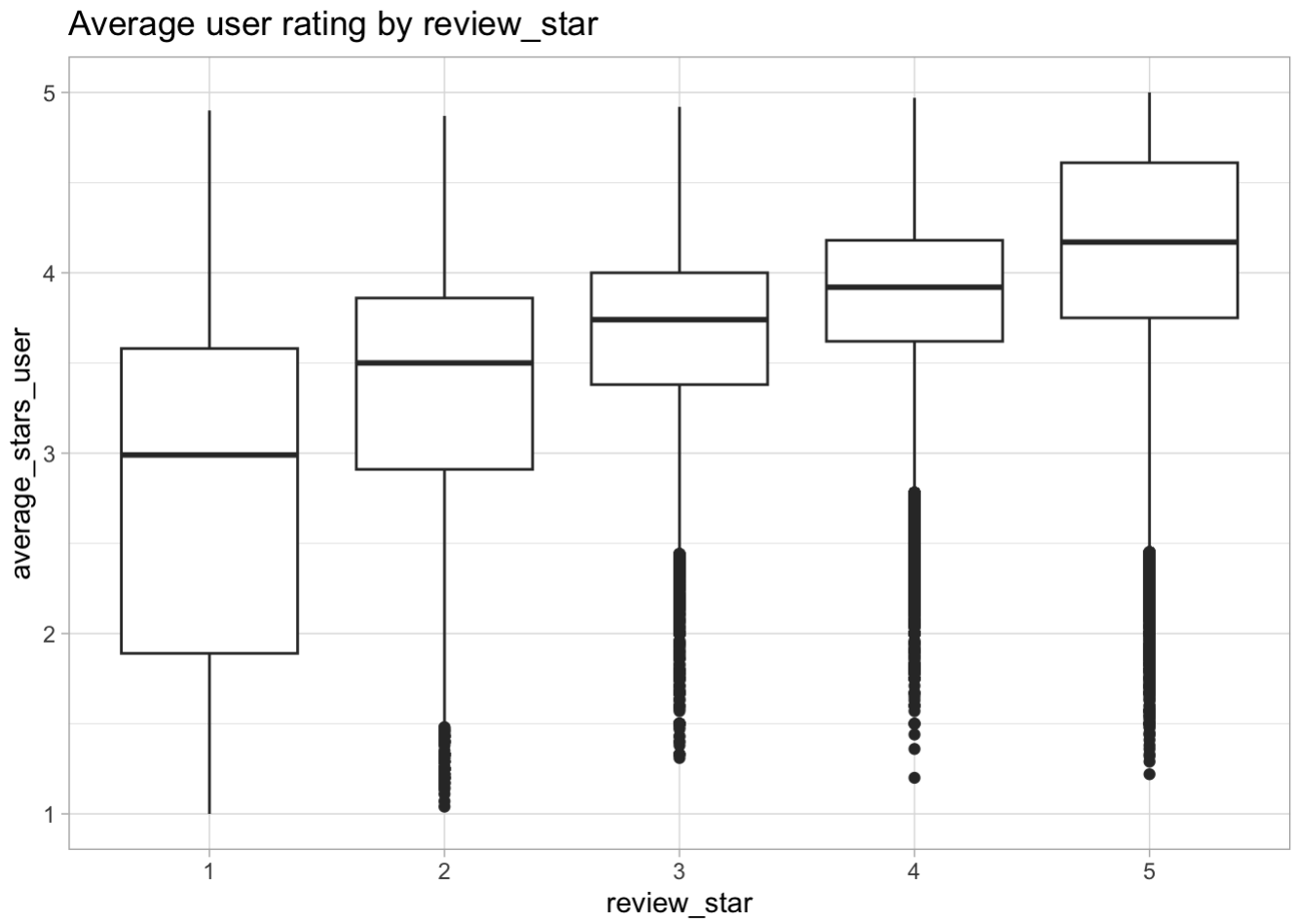

Scatter Plot of review_star by Sentiment Score (AFINN)
review_star
Sentiment Score
5
4
3
2
1
-50
0
50
100
150



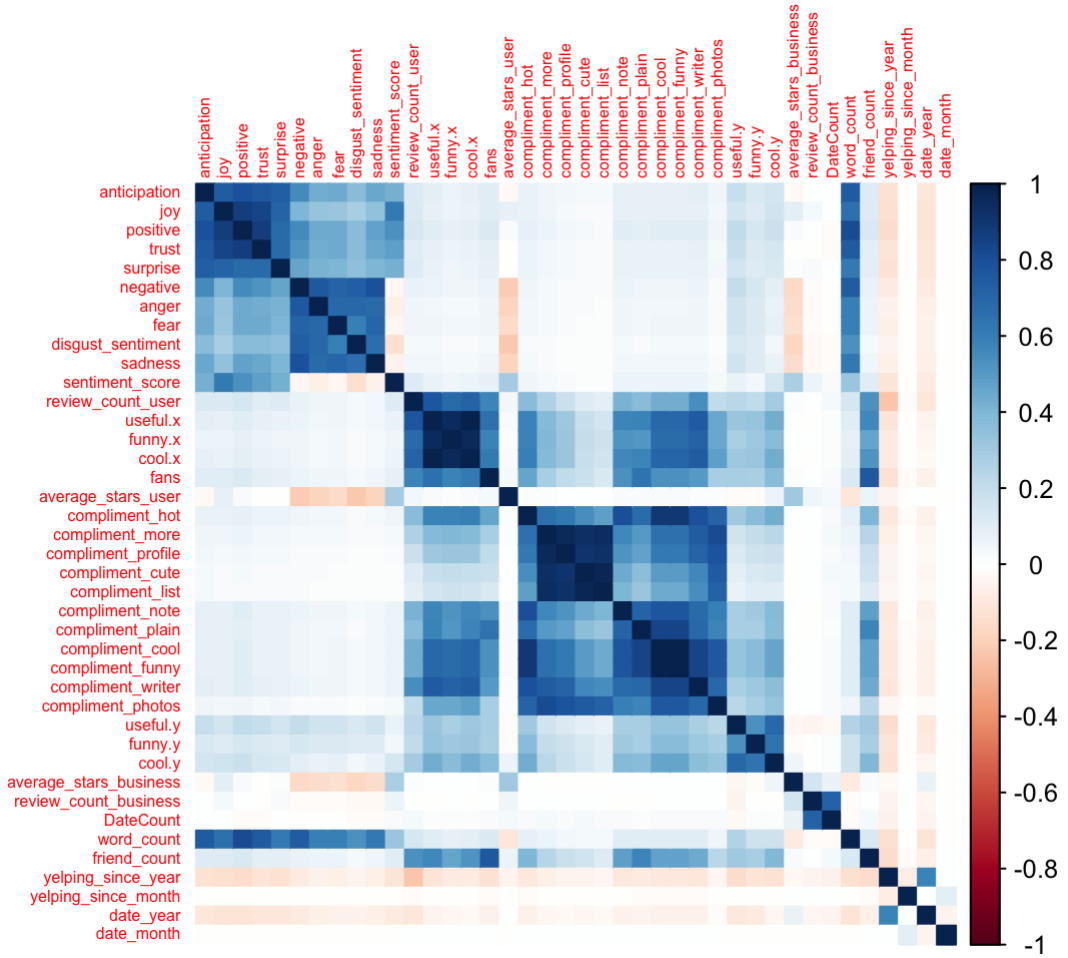
Many of the R^2s from a linear regression of **review_star** on predictors are low, highlighting the highly non-linear relationship between **review_star** and several predictors.

```
R² for sentiment_score : 0.2069115
R² for negative : 0.1327052
R² for positive : 1.617625e-06
R² for joy : 0.00904919
R² for anger : 0.09341091
R² for trust : 0.001156775
R² for sadness : 0.0967133
R² for disgust_sentiment : 0.1438181
R² for anticipation : 0.005373931
R² for fear : 0.06604338
R² for surprise : 0.002176667
R² for average_stars_user : 0.3327123
R² for word_count : 0.04435323
R² for DateCount : 0.001255075
R² for friend_count : 0.001479097
```

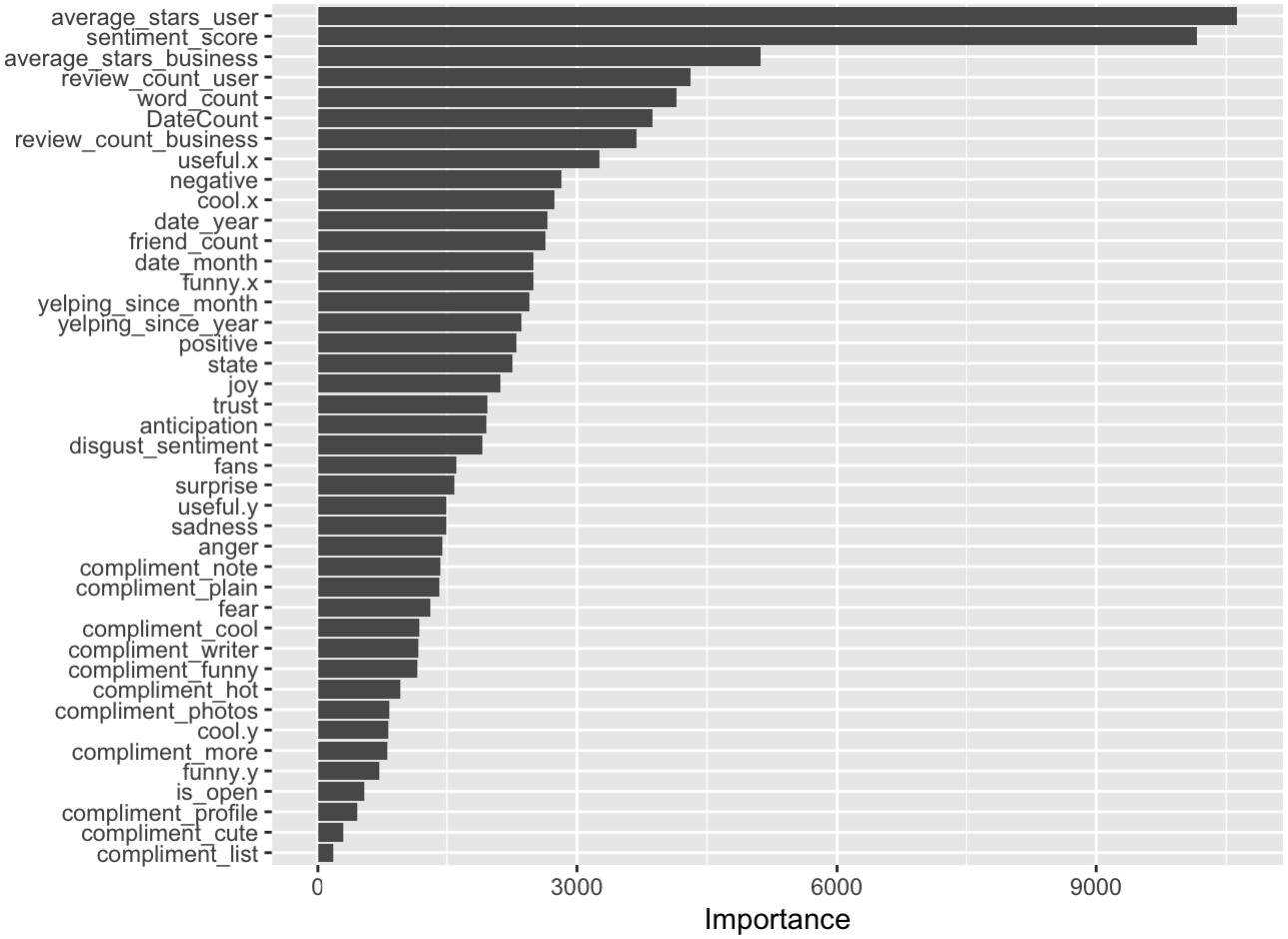
The graph below shows the strong positive relationship between a user's average stars and **review_star**.



The correlation matrix below shows many predictors are highly correlated:



Below is an impurity-based variable importance graph (Boehmke and Greenwell, 2020, ch.11). The feature importance is determined by calculating the average reduction in the loss function (in our case the **Gini index**) due to a certain feature across all trees. average_stars_user and sentment_score are the most important predictors.



Model selection

My EDA above supports the random forest model choice for 5 main reasons:

1. The correlation matrix shows that many **predictors are correlated** with each other. For example, useful.x and cool.x have .99 correlation. Random forests reduce the impact of correlated predictors by randomly selecting a subset of features at each split (Hastie et al., 2009). Random forests therefore decrease the correlation between trees without increasing variance too much. Hastie et al. (2009, p.588) show that with Bagging, the variance of the average prediction is $\varrho\sigma^2 + (1 - \varrho)B\sigma^2$. While bagging does not reduce $\varrho\sigma^2$, random forests randomly select a subset of features at each split, reducing the correlation across predictions ($\hat{Y}$) and so reducing $\varrho\sigma^2$. Hence, with correlated predictors random forest is effective.
2. The impurity graph shows that **multiple variables are important** in predicting review_star. Random forest is most effective when there are multiple good predictors.
3. Random forest is well suited to **classification** problems with uneven distribution of classes.
4. Random forest is effective at dealing with numerical and **categorical datatypes**.
5. Random forest handles **non-linear relationships** well.

Additionally, later I create 200 word variables that are moderately correlated to each other, reinforcing 1.

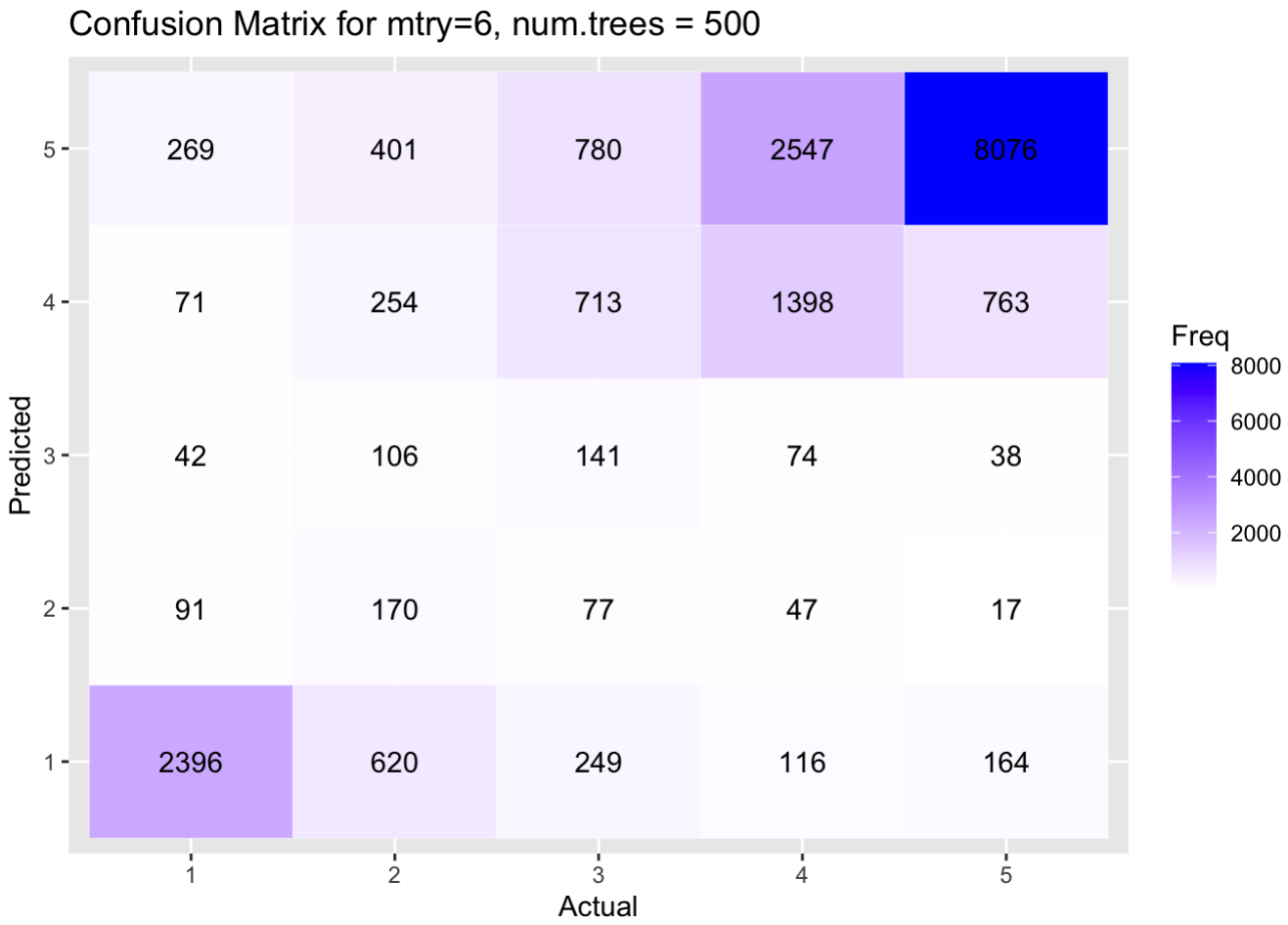
Model implementation, tuning and accuracy *(and most difficult challenge)*

The most difficult challenge in carrying out this project was the computational burden of random forest. The **randomForest** package caused R to crash (due to the number of observations and predictors). To overcome this, I took inspiration from Wright's and Ziegler's (2017) use of the **ranger** package. This package is a fast implementation of random forests for high dimensional data.

An untuned ranger with 500 trees and mtry = 6 has an accuracy of 0.6208461 on the test dataset:

```
# Set seed for reproducibility
set.seed(1)
# Use the ranger package to run a random forest of review_star on all the predictors,
except ones which would cause over fitting
yelp_rf <- ranger(
  review_star ~ .-user_id -business_id -friends -text -review_id - name.x -name.y -el
ite -yelping_since -categories -CheckInDates -city -date,
  data = train_data,
  respect.unordered.factors = "order",
  verbose = FALSE
)

# Run the model on the test data
test_predictions <- predict(yelp_rf, test_data)

# Store predictions as factor
predicted_classes <- as.factor(test_predictions$predictions)

# Confusion matrix to test accuracy of model on test data.
confusionMatrix(predicted_classes, test_data$review_star)$overall["Accuracy"]
```

```
 Accuracy
0.6208461
```




```
How often the prediction is within 1 of the actual rating: 0.875229357798165
```

Corpus text analysis

A corpus is a structured set of documents for pre-processing text data (Welbers et al. 2017). The code is extensive, so I give a summary of my method:

I converted the text data to a corpus, cleaned the corpus (e.g. converted words to lowercase), stemmed the corpus and converted the corpus to a Document Term Matrix showing the frequencies of each word. After removing words that appear in less than 0.25% of all documents, I converted the matrix to a data frame for analysis and combined this with the training dataset. However, there were too many word variables (~ 2,000) which was a burden on computational power. Therefore, I kept only the top 200 words most correlated with **review_star** in the training data. I also added these word variables to the test data.

The top 10 most correlated words have correlations of:

```
       Value
1  0.5036775
2  0.5036775
3  0.4443758
4  0.4443758
5  0.3660386
6  0.3660386
7  0.3587262
8  0.3587262
9  0.3567468
10 0.3567468
```

Therefore, random forest is an appropriate model (predictors are correlated - see section 2)

An untuned ranger (num.trees = 500, mtry = 15) after corpus text analysis has an accuracy of:

```
 Accuracy
0.6327727
```

```
How often the prediction is within 1 of the actual rating: 0.885423037716616
```

HyperTuning

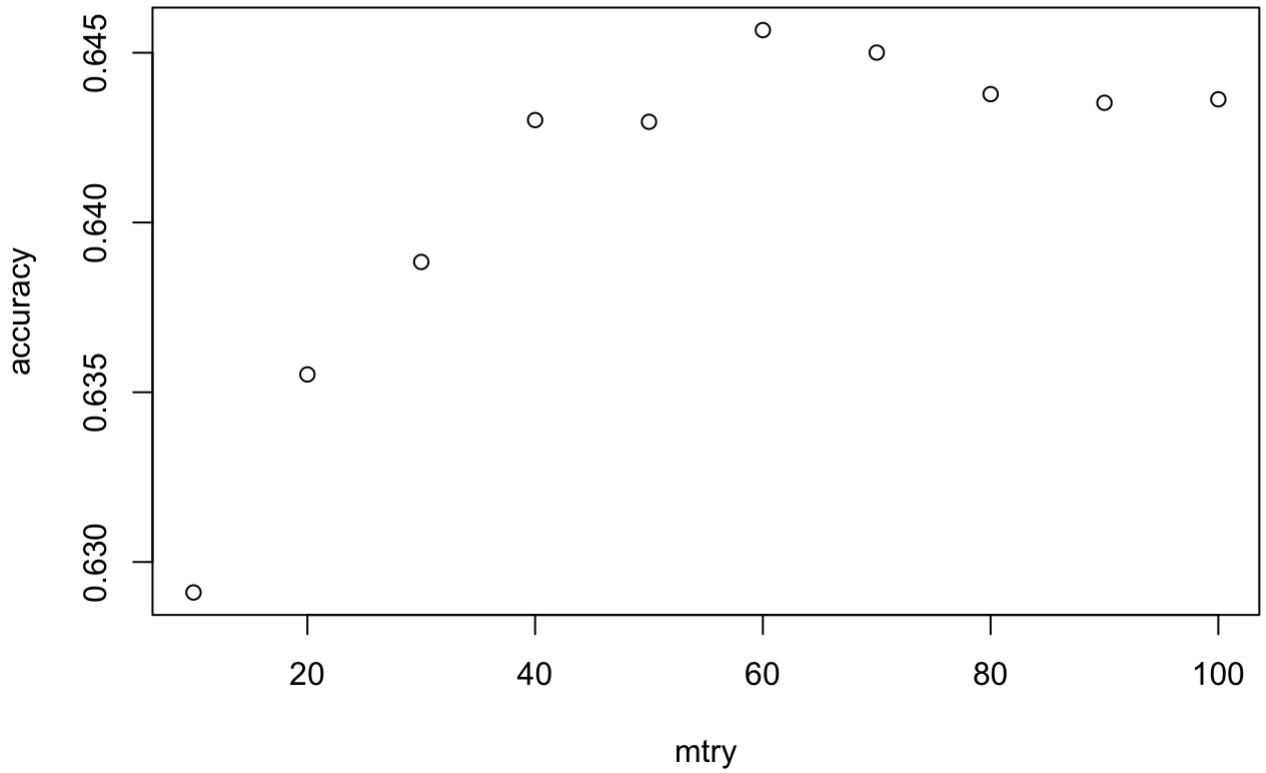
For random forest classification, the default mtry (number of variables randomly sampled at each decision tree split) is

$$\sqrt{\text{number of features (k)}}$$

Among popular machine learning algorithms, random forests display the least variability in prediction accuracy when tuned (Probst et al., 2018). However, a higher mtry increases the likelihood that decision trees choose important features in many splits (Ellis, 2022). Since some of the predictors have lower predictive power, a higher (than default) mtry produces higher accuracy. The optimal mtry is 60 with an accuracy of 0.6456677.

(n.b. increasing num.trees from 500 had no effect on accuracy).

```
             mtry  accuracy
10.Accuracy    10 0.6291030
20.Accuracy    20 0.6355250
30.Accuracy    30 0.6388379
40.Accuracy    40 0.6430173
50.Accuracy    50 0.6429664
60.Accuracy    60 0.6456677
70.Accuracy    70 0.6450051
80.Accuracy    80 0.6437819
90.Accuracy    90 0.6435270
100.Accuracy  100 0.6436290
```

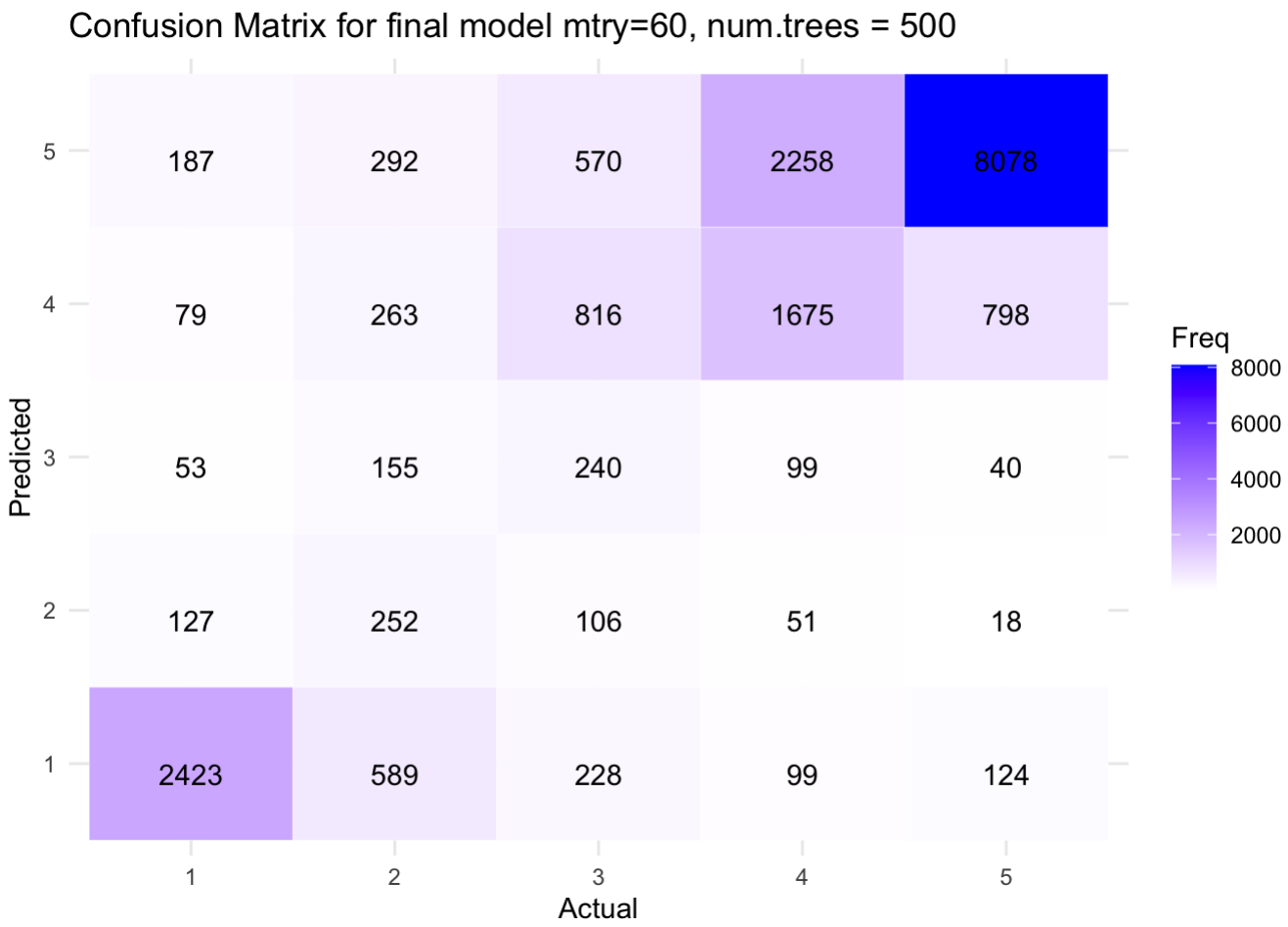


The final tuned model

```
set.seed(1)
yelp_rf3 <- ranger(
  review_star ~ .-user_id -business_id -friends -text -review_id - name.x -name.y -el
ite -yelping_since -categories -CheckInDates -city-date,
  data = train_data_corpus,
  respect.unordered.factors = "order",
  mtry=60,
  min.node.size = 1,
  verbose = FALSE
)

test_predictions3 <- predict(yelp_rf3, test_data_corpus)

predicted_classes3 <- as.factor(test_predictions3$predictions)

confusionMatrix(predicted_classes3, test_data_corpus$review_star)$overall["Accuracy"]
```

```
 Accuracy
0.6456677
```

```
How often the prediction is within 1 of the actual rating: 0.897859327217125
```



Model evaluation

Although the model's accuracy on the test dataset improved with sentiment analysis, sentiment scores were sometimes largely positive when the actual review was 1 star, perhaps failing to pick up sarcasm in text. Therefore, the model could benefit from similar approaches to Bharti et al. (2016) who use sarcasm sentiment detection in tweets.

The corpus text analysis, although increasing model accuracy, could have been better implemented and refined. The word cloud shows that 'get' and 'one' are common in both 1 and 5 star reviews, potentially impacting the model's predictive power. The removal of more 'stopwords' could improve this.

Word Cloud for review_star = 1

Word Cloud for review_star = 5

Additionally, the model could benefit from cross-validation for robustness and to further prevent overfitting.

Overall, new variables such as the year/ month of a review and yelping since, word count and friend count strengthened the model. The final model achieved an accuracy of 64.57% on the test dataset and correctly predicted within 1 star 89.79% of the time. There is potential for improved accuracy with more refined text and sentiment analysis.

Bibliography

- Bharti, S.K., Vachha, B., Pradhan, R.K., Babu, K.S. and Jena, S.K. (2016). Sarcastic sentiment detection in tweets streamed in real time: a big data approach. Digital Communications and Networks, [online] 2(3), pp.108–121. doi:https://doi.org/10.1016/j.dcan.2016.06.002 (doi:https://doi.org/10.1016/j.dcan.2016.06.002).
- Boehmke, B. and Greenwell B. (2020). Chapter 11 Random Forests | Hands-on Machine Learning with R. [online] Github.io. Available at: https://bradleyboehmke.github.io/HOML/random-forest.html (https://bradleyboehmke.github.io/HOML/random-forest.html)
- DSPA. 2021. Evaluating CRISP-DM For Data Science. Data Science Process Alliance
- Ellis, C. (2022). Mtry in random forests. [online] Crunching the Data. Available at: https://crunchingthedata.com/mtry-in-random-forests/ (https://crunchingthedata.com/mtry-in-random-forests/)
- Hastie, T., Tibshirani, R. and Friedman, J. (2009). The elements of statistical learning, second edition : data mining, inference, and prediction. 2nd ed. New York: Springer
- Probst, P., Boulesteix, A.-L. and Bischl, B. (2019). Tunability: Importance of Hyperparameters of Machine Learning Algorithms. Journal of Machine Learning Research, [online] 20, pp.1–32.
- Silge, J. and Robinson, D. (2017). Text Mining with R. ‘O’Reilly Media, Inc.’
- Wright, M.N. and Ziegler, A. (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. Journal of Statistical Software, 77(1). doi:https://doi.org/10.18637/jss.v077.i01 (doi:https://doi.org/10.18637/jss.v077.i01)