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)

Predicting business reviews with sentiment analysis, OLS and shrinkage methods.

Andriy Sinclair

2023-12-05

Team Data Science Process (TDSP) Methodology

The TDSP methodology [1] was chosen to guide this project due to it's simplicity and applicability to modern data science problems. Most importantly, TDSP has intuitive and easy-to-follow documentation that was referred upon consistently for the duration of this project. Other methodologies like CRISP-DM, although the most widely used, are not tailored to modern Machine Learning projects but rather to Data Mining and so were not 100% applicable to this project. In addition, the documentation is long-winded and difficult to follow. The TDSP was modified slightly to suit an academic project and the "business understanding" section was omitted.

1. Introduction

The aim of this project is to enhance the following skills:

- Proficiency in the R programming language.
- Understanding and implementing data wrangling and data cleaning procedures, to avoid the problem of "garbage in garbage out".
- Data understanding through visualisations
- Implementing, explaining and comparing the performance of various statistical learning methods
- Utilisation of sentiment analysis.

Additionally, this project was able to deliver some insights into the nature of business reviews.

2. Data Acquisitions and Understanding

- Handling duplicates, especially in reviews.
- Transforming variables for modeling.
- Checking and addressing negative values.
- Managing missing values by dropping rows.
- Removing outliers in the "state" variable.
- Converting categorical variables to factors.
- Dropping "Compliment_cool" due to multicollinearity.
- Renaming variables for clarity.
- Merging review data based on "business_id" and "user_id."
- Utilizing a 1:4 test:train split.

[1] Microsoft. (2016). Team Data Science Process (TDSP). Retrieved from https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/

Data Visualisations

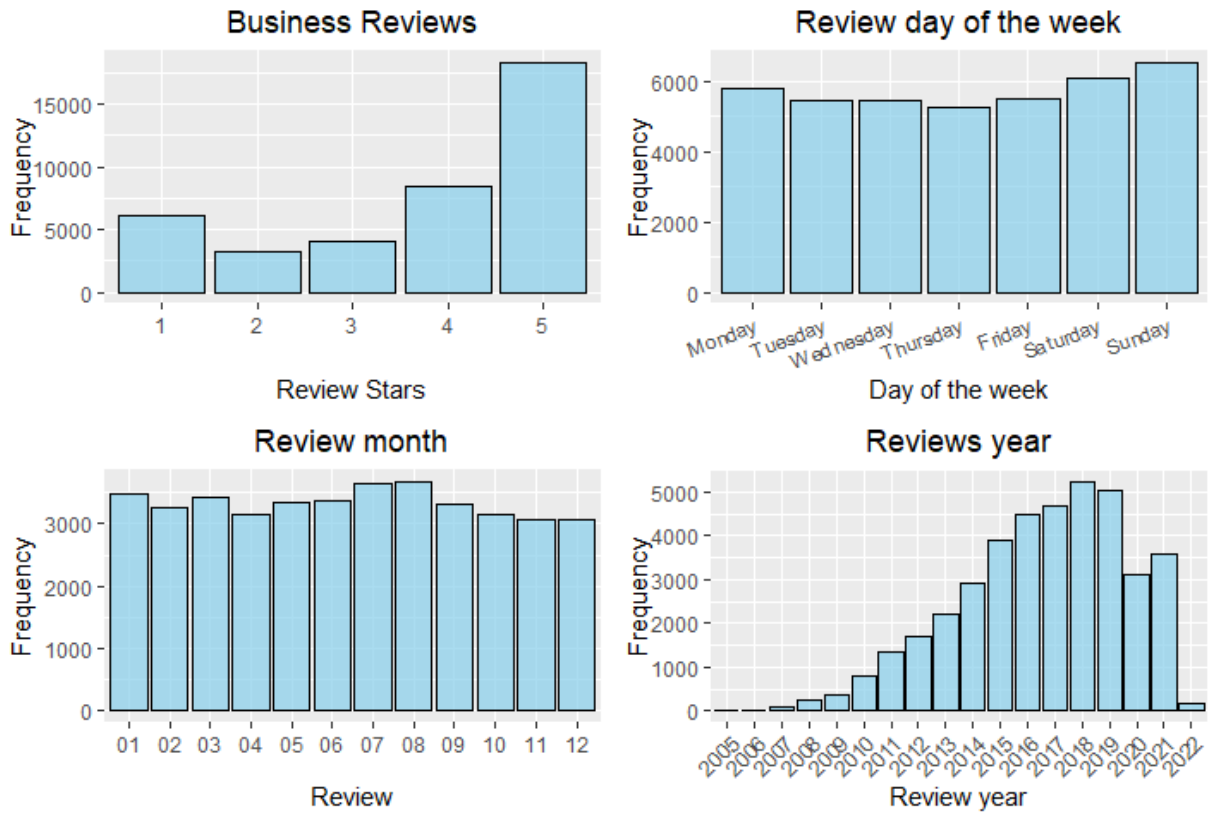

Figure 1:

From **Figure 1** we can see that 5 stars was the highest review category, with the frequency of very negative reviews at 1 star surpassing more neutral reviews of 2 or 3 stars. The majority of reviews were given on weekends, as expected. The summer months of June, July and August have the highest frequency of reviews followed by the winter months of January till March. We see a seemingly exponential increase in the number of reviews given from 2005 until 2018, then we see a drop, likely due to COVID-19.

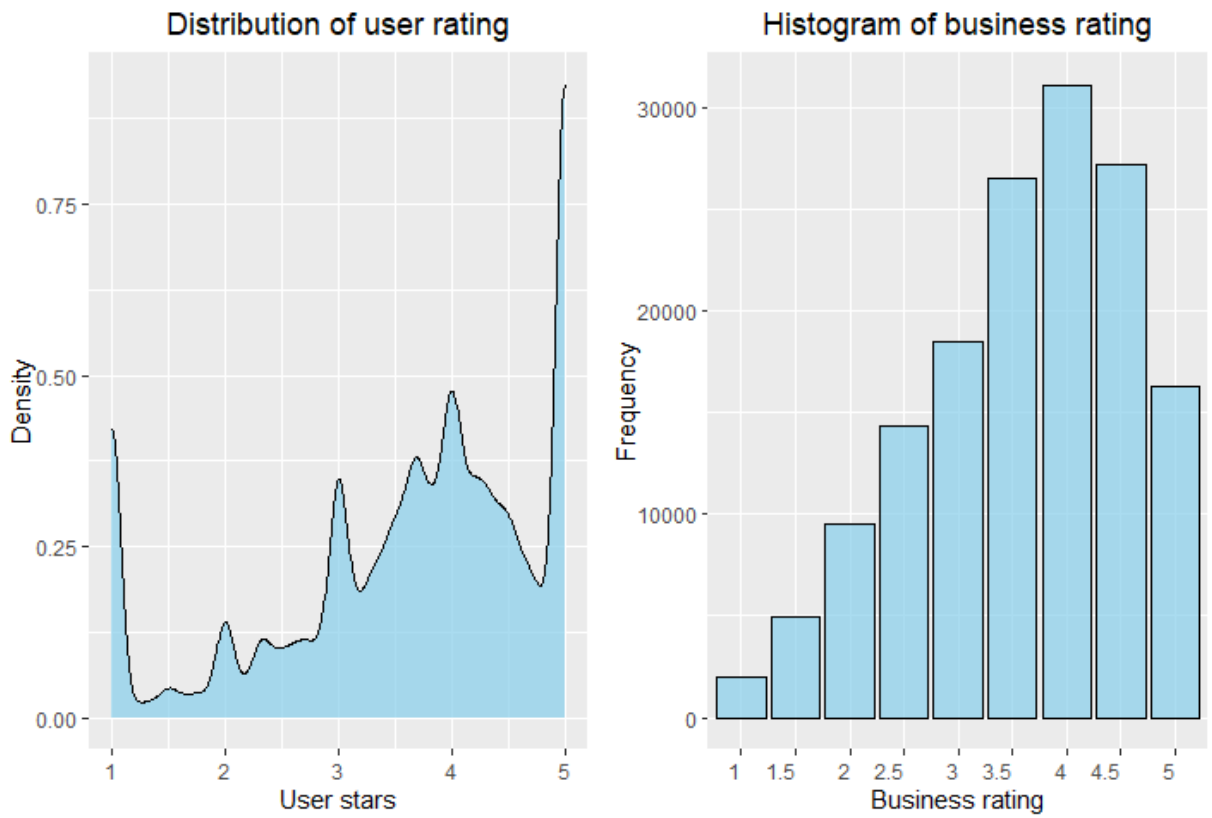

Figure 2:

Figure 2 shows that the majority of users are ranked around 4 stars, with high proportions at the extremes of 1 star and 2 star.The majority of businesses are also ranked 4 stars.

Models

Overview

Two specifications were carried out for this project:

- **Specification 1**: Regular model including all variables outlined in *Appendix 1b*
- **Specification 2**: All Variables included in **specification 1** and a sentiment score.

Sentiment analysis

A lexicon-based sentiment analysis method was used for this project. This involves assigning a sentiment score to each word based on a pre-existing sentiment dictionary. The "AFINN" dictionary[2] was used in this project due to words being ranked between -5 to +5 rather than taking three values: -1 (negative), 0 (neutral) and 1 (positive). Additionally, a sentiment score scaled by the occurence of non-zero words was used to account for varying review length. Although results are not included, this performed better than the regular sentiment score.

The AFINN dictionary requires neither stemming, as all word forms are accounted for, nor the removal of stop words, as these would be ignored. Stop words were removed purely for the visualisation of **Figure 1**.

[2]F. Å. Nielsen, "AFINN," 2011, Retrieved from (http://www2.compute.dtu.dk/pubdb/pubs/6010-full.html).

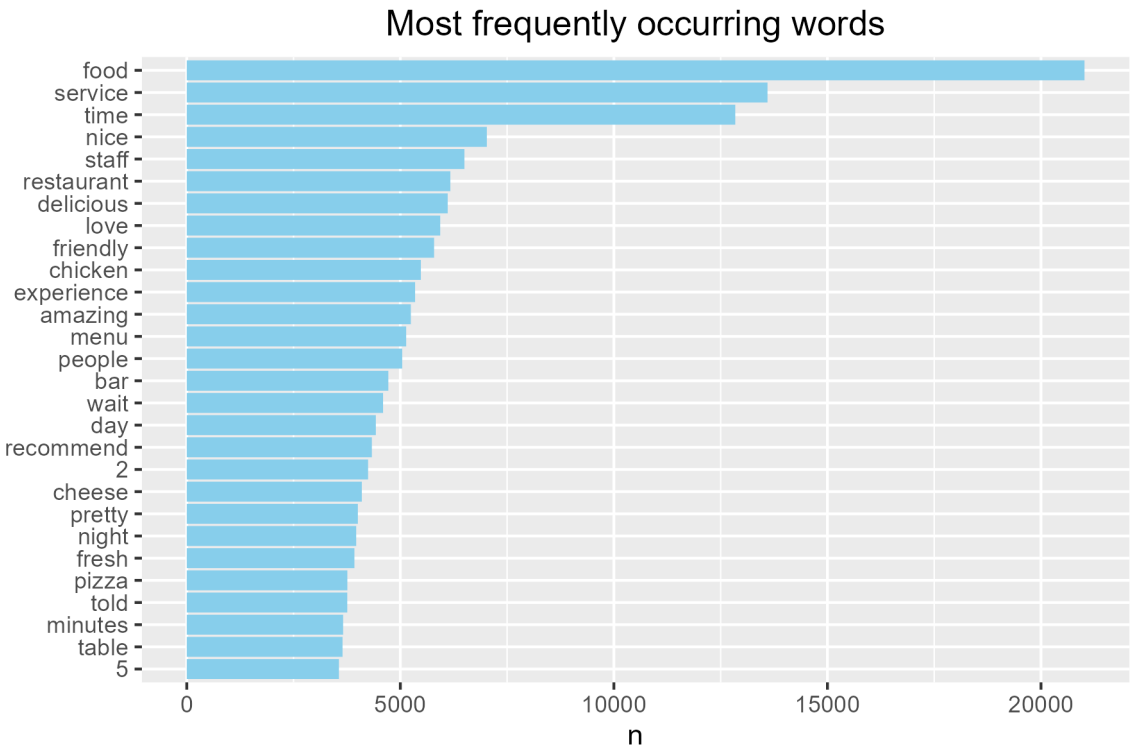

Figure 3:

Sentiment analysis gives us further insight into the nature of reviews.**Figure 1** shows that the majority of reviews are positive, which aligns with **Table 1** and **Table 2**. Moreover, it appears that most businesses reviewed are food businesses.

Table 1: Mean Squared Error (MSE) scores

	regular spec: test sample	regular spec: training sample	Sentiment analysis: test sample	Sentiment analysis: training sample
Random choice benchmark	4.744900	4.744900	4.7449000	4.7449000
OLS	1.224744	1.173122	0.9086261	0.9035816
Ridge	1.209762	1.179066	0.9120290	0.9068658
Lasso	1.176714	1.185021	0.9083224	0.9051351

While reviews are discrete outcome variables ranging from 0 to 5, treating this prediction as a classification problem might not capture the nuanced relationships between values. For instance, predicting 4 when the actual value is 5 and predicting 1 when the actual value is 5 both result in the same classification score of 0. However, treating it as a regression problem with Mean Squared Error (MSE) as a performance measure allows the model to receive credit for distinguishing a good reviews from bad, even if the exact value was not predicted correctly.

MSE is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

MSE takes into account the average squared deviation of the values predicted by the model from the actual values.

The MSE of a random choice model was taken for benchmarking. Every iteration generated a random number between 1 and 5 and compared that to the actual value. This resulted in a classification accuracy of 0.2, as expected, and an MSE of 4.7.

All models performed better than random choice. However,from **Table 1** it is evident that including a sentiment analysis score resulted in improved performance across all prediction models.

The prediction models chosen are as follows:

OLS

OLS (Ordinary Least Squares) was a chosen as a widely-used and well-known prediction method to be used as a benchmark. Additionally, the interpretability offered by OLS surpasses more intricate methods

From **Table 2** we can see that already highly rated businesses are likely to recieve further positive reviews and highly rated users give higher reviews. In addition, including the sentiment score results in a better fit to the data as shown by each regression's respective R^2, defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

R^2 represents the proportion of the variance in the dependent variable that is explained by the independent variables in a regression model, and as a rule of thumb including a sentiment score explains ~12% more variation in the data.

Shrinkage methods

Shrinkage methods were chosen to explore model performance over and above OLS.

The *ridge* estimator finds the $\beta_0, \beta_1, ..., \beta_k$ that minimises:

$$\begin{aligned} &= \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \sum_{k=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{k}\beta_j^2 \\ &= RSS + \lambda \sum_{j=1}^{k}\beta_j^2 \end{aligned}$$

The *lasso* estimator finds the $\beta_0, \beta_1, ..., \beta_k$ that minimises:

$$\begin{aligned} &= \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \sum_{k=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{k}|\beta_j| \\ &= RSS + \lambda \sum_{j=1}^{k}|\beta_j| \end{aligned}$$

Both Ridge and Lasso introduce a shrinkage penalty. The shape of Lasso's shrinkage penalty: $\lambda \sum_{j=1}^{k}|\beta_j|$ is a diamond meaning that the solution space where the penalty term is minimized often intersects with the axes at the coordinate axes. Ridge's penalty term: $\lambda \sum_{j=1}^{k}\beta_j^2$ is a circular contour thus the solution never reaches 0. Both methods introduce bias into the coefficient estimates so that the method does not fit the outliers of the data excessively. The magnitude of λ controls the strength of the shrinkage. When $\lambda \to \infty$ we have maximum bias and when $\lambda \to 0$ we have maximum variance.10-fold cross-validation was used to

Table 2: OLS model significant coefficients

	Specification 1	Specification 2
review_useful	−0.088***	−0.055***
	(0.003)	(0.003)
review_cool	0.115***	0.067***
	(0.004)	(0.003)
review_month10	−0.069*	−0.065*
	(0.031)	(0.027)
review_month12	−0.040	−0.056*
	(0.032)	(0.027)
review_daySaturday	−0.023	−0.063**
	(0.023)	(0.021)
review_daySunday	−0.046*	−0.050*
	(0.023)	(0.020)
user_useful	0.000**	0.000*
	(0.000)	(0.000)
user_cool	0.000***	0.000*
	(0.000)	(0.000)
average_stars	0.807***	0.580***
	(0.008)	(0.007)
business_stars	0.640***	0.453***
	(0.009)	(0.008)
business_review_count	0.000+	0.000***
	(0.000)	(0.000)
is_open	0.070***	
	(0.018)	
is_open1		0.091***
		(0.016)
ss		2.318***
		(0.025)
Num.Obs.	30 000	30 000
R2	0.463	0.585
R2 Adj.	0.462	0.584
AIC	90 074.4	82 244.6
BIC	90 689.2	82 867.8
Log.Lik.	−44 963.182	−41 047.322
F	358.120	578.119
RMSE	1.08	0.95

+ $p < 0.1$, * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Table 3: Ridge coefficients

	Specification 1	Specification 2
review_useful	−0.062	−0.037
review_cool	0.080	0.046
review_month10	−0.045	−0.032
review_month12	−0.021	−0.026
review_daySaturday	−0.017	−0.052
review_daySunday	−0.038	−0.041
user_useful	0.000	0.000
user_cool	0.000	0.000
average_stars	0.763	0.559
business_stars	0.617	0.445
business_review_count	0.000	0.000
is_open	0.060	
is_open1		0.080
ss		2.229
Num.Obs.	30 000	30 000
nulldev	65516	65335.28
npasses	120	121

select the optimum λ term. This attempts different λ on different subsets (or folds) of the data and selects the best performing.

The main conclusions from running lasso and ridge are summarized below:

- Both lasso and ridge exhibit less coefficient shrinkage for Specification 2.
- Lasso drops non-significant variables in Specification 1, maintaining more variables in Specification 2.

This aligns with expectations, as Specification 2 increases predictability, leading to less overall reduction in coefficients for both shrinkage methods.

Interestingly, shrinkage methods do not show improvements in predictability over OLS, as indicated in **Table 1**. The training and test MSEs are nearly identical across specifications and models, suggesting little overfitting. In such cases, shrinkage methods would not enhance performance.

Evaluation

One strength of this approach to the task is transforming if required and utilising all variables in the user and review data sets and allowing for the models to self-correct and not utilise those with low predictive power. An improvement would be to additionally utilise variables from the business data set. Specifically, a price range variable for all categories of business are likely to have predictive power over and above the current specification. Additionally, there is a "categories" variable in the business data set. However, it takes a wide variety of inputs and results in +1000 unique combinations and so would be difficult to model. More concise definitions of categories would be an excellent addition to this data set.

It is evident that adding sentiment analysis makes a large impact on predictiability. This could be further improved by using a custom lexicon dictionary that gives words scores from 1 to 5 depending on what review rank they appear most in. Additionally, including bi-grams and tri-grams could be experimented with to provide a more accurate sentiment score

On a personal level, the biggest challenge faced with this project was the lack of computing resources. Thus I had to limit my data set to 40,000 observations and I was not able to run more complex models like Random

Table 4: Lasso coefficients

	Specification 1	Specification 2
(Intercept)	−1.533	−0.793
review_useful	−0.050	−0.046
review_funny	−0.026	−0.002
review_cool	0.064	0.056
review_year2021	−0.047	−0.074
average_stars	0.794	0.577
business_stars	0.623	0.448
review_month01		0.011
review_month02		−0.006
review_month04		−0.001
review_month05		0.015
review_month06		0.014
review_month08		0.010
review_month10		−0.018
review_month12		−0.010
review_year2006		0.131
review_year2007		0.137
review_year2008		0.124
review_year2010		−0.005
review_year2011		−0.015
review_year2012		0.018
review_year2014		0.015
review_year2018		0.011
review_year2020		−0.002
review_dayMonday		−0.010
review_daySaturday		−0.049
review_daySunday		−0.038
review_dayTuesday		0.001
review_dayWednesday		0.007
compliment_plain		0.000
compliment_writer		0.000
stateAZ		0.009
stateCA		−0.034
stateFL		−0.008
stateMO		0.003
stateNV		0.029
statePA		0.019
stateTN		−0.023
business_review_count		0.000
is_open1		0.070
ss		2.316
Num.Obs.	30 000	30 000
nulldev	65516	65335.28
npasses	30	75

Forest and Adaboost. If I was able to obtain more computational resources I would like to utilise the full user data along with the full review data set to obtain the greatest possible observations and compare how model performance of that data set compares to the one I was able to use.

To make up for using simpler models I decided to implement sentiment analysis to aid in predictability. It was interesting to see by how much model performance improved by implementing this.

Additionally performing cross-validation on all models and averaging the MSE would provide a more robust measure. Nevertheless, the models were ran using different seeds and similar results were obtained

Bibliography

Microsoft. (2016). Team Data Science Process (TDSP). Retrieved 12 5, 2023, from https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/

Yelp. (2022). Yelp Dataset. Retrieved from https://www.yelp.com/dataset

F. Å. Nielsen, "AFINN," 2011, Retrieved from (http://www2.compute.dtu.dk/pubdb/pubs/6010-full.html).

Appendix

All code is contained within: * "Spec1.R" * "Spec2_sentiment_analysis.R" * "Visualisations.R" and can be accessed through this repository: *https://github.com/andriysinclair/PredictingBusinessReviews*

Appendix 1: Datasets used

Business data set

- **business_id**: A string representing a 22-character unique business ID.
- **name**: A string representing the business's name.
- **address**: A string representing the full address of the business.
- **city**: A string representing the city where the business is located.
- **state**: A string representing the 2-character state code, if applicable.
- **postal_code**: A string representing the postal code of the business.
- **latitude**: A float representing the latitude of the business location.
- **longitude**: A float representing the longitude of the business location.
- **stars**: A float representing the star rating of the business, rounded to half-stars.
- **review_count**: An integer indicating the number of reviews for the business.
- **is_open**: An integer, 0 or 1, indicating whether the business is closed or open, respectively.
- **attributes**: An object representing business attributes, where some attribute values might be objects.
  - **RestaurantsTakeOut**: A boolean indicating whether the business offers takeout.
  - **BusinessParking**: An object with boolean values indicating parking options (e.g., garage, street, valet).
- **categories**: An array of strings representing business categories.
- **hours**: An object representing the hours of operation for each day of the week, using a 24hr clock.

User data set

- **user_id**: A string representing a unique 22-character user ID, mapping to the user in user.json.
- **name**: A string representing the user's first name.
- **review_count**: An integer indicating the number of reviews the user has written.
- **yelping_since**: A string indicating when the user joined Yelp, formatted as YYYY-MM-DD.

- **friends**: An array of strings representing user IDs of the user's friends.
- **useful**: An integer indicating the number of useful votes sent by the user.
- **funny**: An integer indicating the number of funny votes sent by the user.
- **cool**: An integer indicating the number of cool votes sent by the user.
- **fans**: An integer indicating the number of fans the user has.
- **elite**: An array of integers representing the years the user was elite.
- **average_stars**: A float indicating the average rating of all reviews given by the user.
- **compliment_hot**: An integer indicating the number of hot compliments received by the user.
- **compliment_more**: An integer indicating the number of more compliments received by the user.
- **compliment_profile**: An integer indicating the number of profile compliments received by the user.
- **compliment_cute**: An integer indicating the number of cute compliments received by the user.
- **compliment_list**: An integer indicating the number of list compliments received by the user.
- **compliment_note**: An integer indicating the number of note compliments received by the user.
- **compliment_plain**: An integer indicating the number of plain compliments received by the user.
- **compliment_cool**: An integer indicating the number of cool compliments received by the user.
- **compliment_funny**: An integer indicating the number of funny compliments received by the user.
- **compliment_writer**: An integer indicating the number of writer compliments received by the user.
- **compliment_photos**: An integer indicating the number of photo compliments received by the user.

Review dataset

- **review_id**: A string representing a unique 22-character review ID.
- **user_id**: A string representing a unique 22-character user ID, mapping to the user in user.json.
- **business_id**: A string representing a unique 22-character business ID, mapping to business in business.json.
- **stars**: An integer indicating the star rating of the review.
- **date**: A string representing the date of the review formatted as YYYY-MM-DD.
- **text**: A string representing the text content of the review.
- **useful**: An integer indicating the number of useful votes received for the review.
- **funny**: An integer indicating the number of funny votes received for the review.
- **cool**: An integer indicating the number of cool votes received for the review.

Appendix 1b: Variables used in predictive models

stars_review was the outcome variable

Predictors:

Business data set

- **name**
- **state**
- **stars_business**
- **review_count**
- **is_open**

User data set

- **user_id**
- **review_count**
- **review_month**
- **review_year**
- **review_day**

- **friends**
- **useful_user**
- **funny_user**
- **cool_user**
- **fans**
- **elite**
- **average_stars**
- **compliment_hot**
- **compliment_more**
- **compliment_profile**
- **compliment_cute**
- **compliment_list**
- **compliment_note**
- **compliment_plain**
- **compliment_funny**
- **compliment_writer**
- **compliment_photos**

Review data set

- **stars_review**
- **date**
- **text**
- **useful_review**
- **funny_review**
- **cool_review**