Capstone1: Final Report

The purpose of this document is to summarize and consolidate the findings of this project.

Table of Contents

1. Defining the problem and the client

With the event of social media and online directories - such as Yelp, TripAdvisor and Zomato - online marketing is becoming an increasing important aspect of promoting a business.

In particular, reviews and ratings of a restaurant could make or break a new establishment. It is now a common practice for customers to checkout a restaurant (or establishment) using online reviews and ratings before giving it a try.

This project will predict Yelp's 5-star rating based on the review written.

The prediction could be used as part of a new feature to change the default rating from a static value (e.g. default of 3 stars or 0 stars) into a dynamic default value based on the review. This default rating would still need to be accepted by the reviewer. Alternatively, this could be used to trigger is a verification for the user to confirm the rating if the predicted rating is significantly different from the actual rating. Regardless of implementation method, the intent of the prediction is to help make review ratings more accurately reflect the reviewer's opinion.

The client for this project would be Yelp.

An easier review experience with more reflective ratings, would add value to all to their major stakeholders:
- a dynamic rating could help reduce the cognitive load on the reviewers, making reviews easier
- better rating accuracy helps new customers with their online research
- more reflective ratings reduce frustration / confusion of establishment proprietors arising from an unconsidered rating
- the platform's (e.g. Yelp) quality of reviews is improved - providing better, cleaner data

2. Collecting the data

For more details, please refer to the document titled:
`Capstone1_writeup_datawrangling` and its accompanying jupyter notebook.

The dataset was obtained as part the '[Yelp dataSet Challenge - Round 10](https://www.yelp.com.sg/dataset)' (September 1, 2017 to December 31, 2017). In particular, the `review.json` and `business.json` files from the JSON dataset.

These are currently stored in the folder labeled `01_raw_data`. In accordance with the terms and conditions of the dataset usage, the data shall not be shared in the Github repo.

The raw data in the original document comes in the format of dictionaries separated by lines:

```
{}
{}
{}
```

Review as well as business data were loaded by establishing a connection to the file using the JSON dictionary, then the data was read using a list comprehension and finally the list was converted into a DataFrame.

3. Cleaning and wrangling the data

After the dataset was loaded in, 3 functions were used: `.shape`, `.info`, `.describe`. A null check was performed, and a head of 50 was used to eyeball the data set.

The following observations were made about the `review` data:
1. There were no null values
2. Except for the data, the data types from the column were correct
3. The full data set had ~ 4.7 million observations
4. Non-restaurant businesses were also reviewed (e.g. accommodation)
5. Reviews were not all in English

For the `business` data:
1. There were only 2 null values (1 Lat, 1 Long)
2. The data types were not an issue but there were a number of columns related to the location of the business
3. The full data set had ~ 150 k observations
4. Non-restaurants were included
5. A number of locations were outside the USA (e.g. `state: ON, postal_code: M4K 1N7` refers to Toronto). Unique state values and the range of lat long values were further analyzed and revealed that a number of businesses were outside the USA.

Cleaning the dataset

To clean the review data, very little clean up was necessary. The date was converted to datetime but the stars were kept as integers vs. categories.

To clean the business data:
- non-restaurant businesses were removed. Leaving ~ 50k observations (almost 1/3 the original number of businesses although the impact on the number of reviews is less severe).
- next, restaurants outside the USA were removed using a bounding box of min-max latitude and longitudinal values. The bounding box for the US is (49.3457868 # north lat) (24.7433195 # south lat) (-124.7844079 # west long) (-66.9513812 # east long). This reduced the dataset to ~48k observations.

The restaurant data frame was merged with the review dataframe using an inner join on the `'business_id'`. For the merged data, the following observations were made:
1. There were no null values
2. Inner join reduced the total number of observations (i.e. reviews) from ~4.74 million from all businesses to ~ 2.92 million from restaurants alone to the ~2.88 from restaurants in the USA bounding box.

A bar chart to showed a bias towards positive 5 star ratings but uncovered no outliers. Data exploration will be discussed in the next section.

Pre-processing text

The pre-processing of the review text involved 5 steps. These were defined in function named `` `pre_process_review` ``:
1. Removed punctuation
2. Convert characters to lower case
3. Removal of stop words
4. Lemmatization of words
5. Conversion for review text string to a list of words (tokenization)

Note that the additional step of expanding contractions was not used, but it would come as step 1 if it were needed.

4. Exploring other potential data sets that could be used

Other potential data set that could be used to explore the current data set is: `` `user.json` `` file from the '[Yelp dataSet Challenge - Round 10](https://www.yelp.com.sg/dataset)' dataset. This could be joined onto the column in the review dataframe titled: `user_id`.
This data would be especially interesting if one were to do an analysis regarding the usefulness of a review. However, it is not expected to impact the current analysis on prediction of the star rating for a given review, since majority of reviews come from reviewers with few reviews (as seen in the next section).

5. Exploring the data

For more details, please refer to the documents titled: `` `Capstone1_datastorytelling` ``, `` `Capstone1_writeup_inferentialStatistics` `` and their accompanying Jupyter notebooks.

An initial analysis of the data helped support the initial hypothesis of the project outlined at the start of this document:
- Yelp reviews are relevant in today's context
- Review ratings could have significant impact on a restaurant
- Review text can be used as a predictor of star ratings

Hypothesis: Yelp reviews are relevant in today's context

A plot of the number of Yelp reviews over time (figure 1) suggests that Yelp continues to be relevant in 2017. It can be seen that the number of reviews grows rapidly till 2015 when they begin to a phase of slower but positive growth.

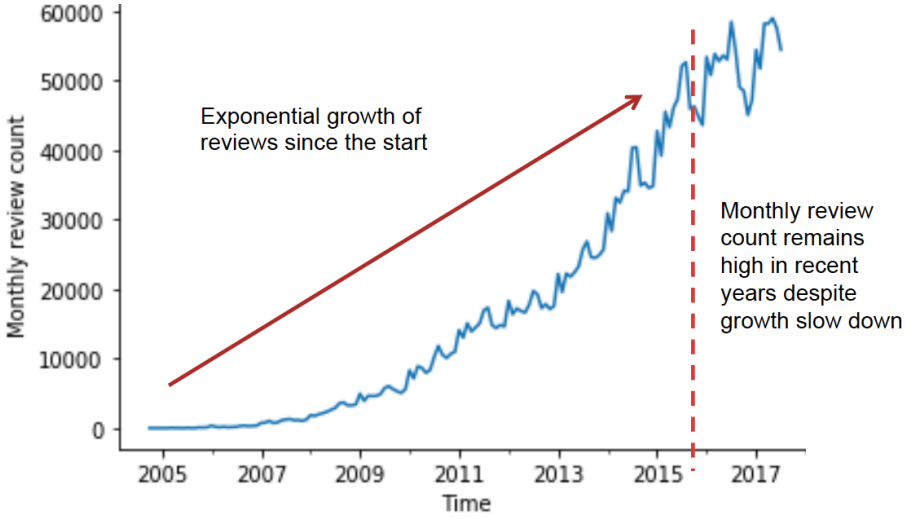

Figure 1: Monthly restaurant review counts

Hypothesis: Review ratings could have significant impact on a restaurant

It was observed that the majority of restaurants have less than 20 reviews (see figure 2). For these restaurants, a single review rating has a significant impact on the average star rating for the restaurant. This would impact the restaurant's standing within the Yelp community, which might significantly impact those restaurants.

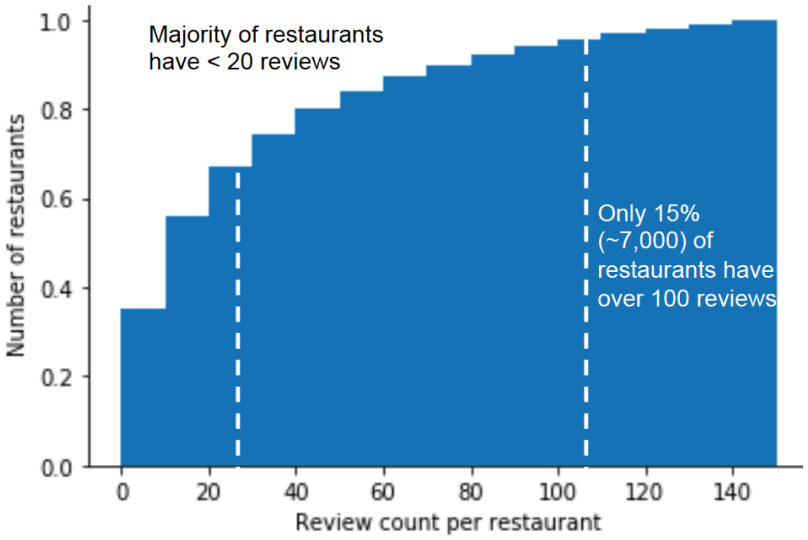

Figure 2: Cumulative distribution of review counts

Hypothesis: Review text can be used as a predictor of star ratings

The hypothesis that review text is a good predictor of star ratings is explored by reviewing other available data (e.g. user data, restaurant data, location, text length etc).

An analysis of users writing reviews showed that the top two reviewers wrote thousands of reviews and the top users all remain prolific writers (see figure 3). However, further analysis of the distribution of restaurant reviews per user (figure 4), revealed that the majority of reviews were written by first time reviewers. This implies that analyzing a particular user's behavior might be helpful for prolific writers, but there is limited impact of such an analysis on review ratings as a whole.

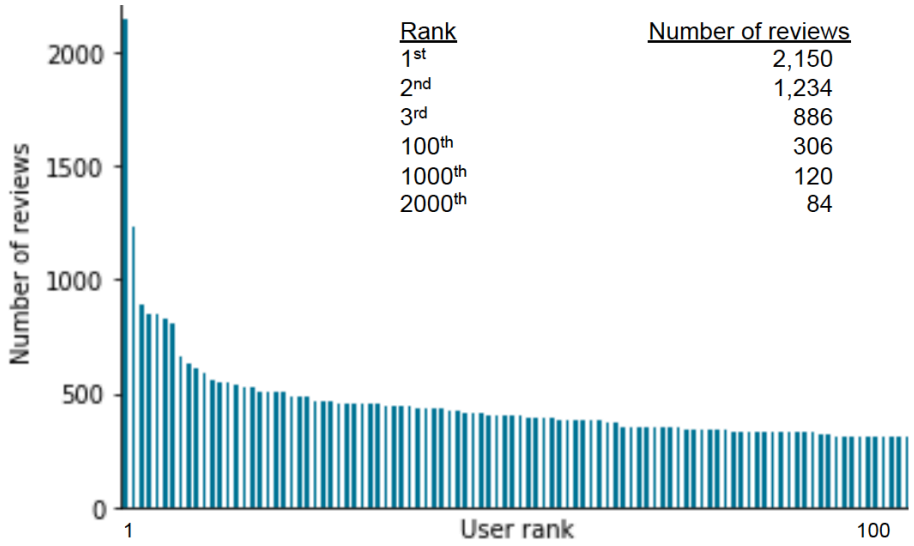

Figure 3: Number of reviews written by the top reviewers (each)

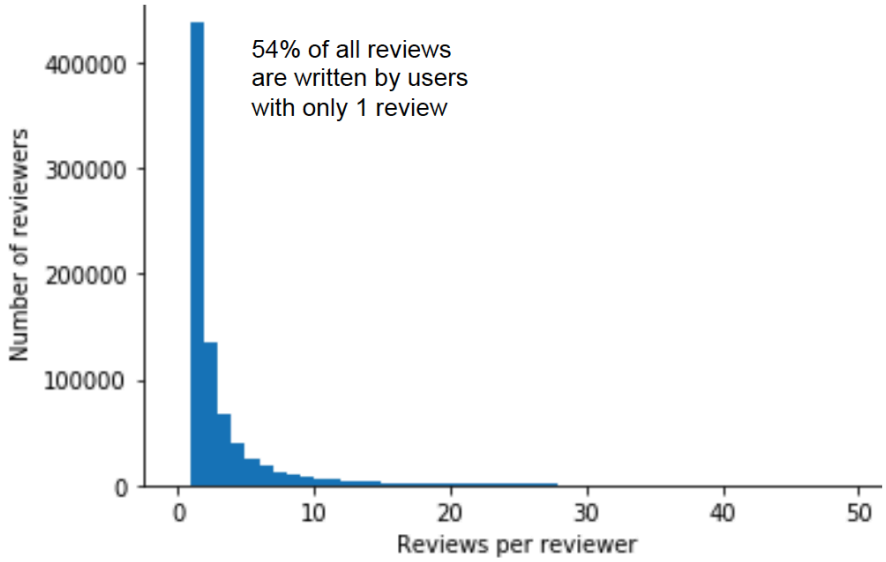

Figure 4: Distribution of restaurant reviews per user

Next, the restaurants were analyzed to understand what ratings they got and the number of reviews they received. It was found that on average, restaurants were rated fair to good (3-4 stars) (see figure 5). However, restaurants with better ratings tend to get more reviews (see figure 6).

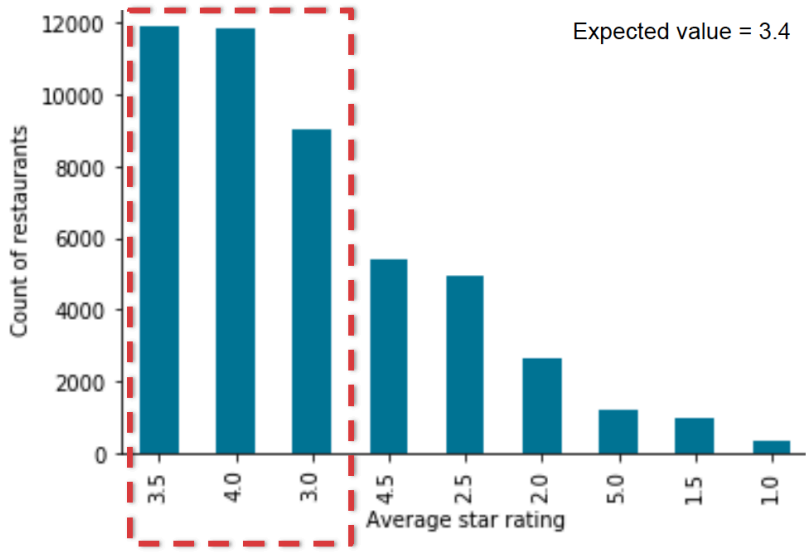

Figure 5: Distribution of average star ratings for restaurants

Figure 6: Distribution of reviews by restaurant's average star rating

If we were to use the expected value of star ratings based on the average star ratings received by restaurants, this might result in a popular restaurants (who get higher ratings) receiving a lower rating. Assigning a value based on the most common review rating or on the expected value for a review rating might result in higher than justified rating. The analysis of the distribution of reviews by star ratings (figure 7) provides the mean and mode values for review ratings.

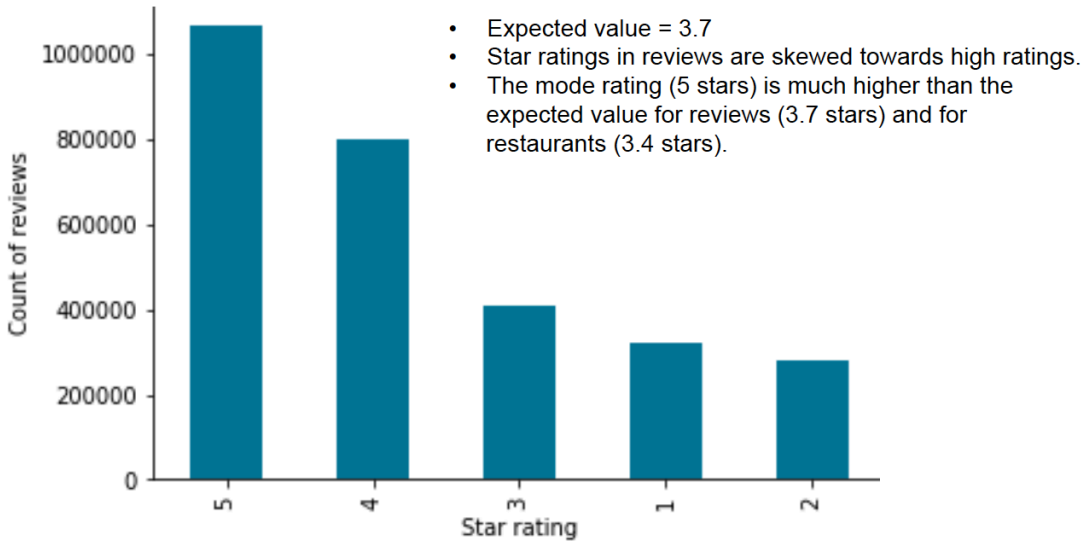

Figure 7: Distribution reviews by star ratings

An analysis of the review text was conducted to explore the link between review text and ratings. First, a simple analysis of the distribution of review lengths by stars (figure 8) indicated that the length of a review appears to be a poor proxy indicator of star ratings. Therefore, the content (and not just the length) of the text would need to be analyzed.

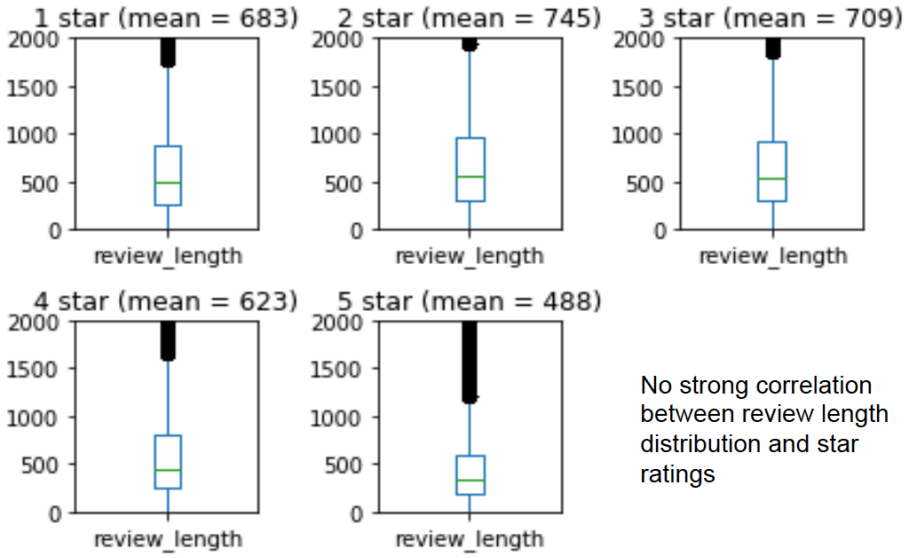

Figure 8: Distribution of number of characters in a review by stars

In addition, inferential statistics was used to validate the decision to drop all other columns other than the review text and review stars. A correlation matrix (figure 9) was generated between the numerical data fields.

	cool	funny	stars_review	useful	is_open
cool	1.000000	0.846138	0.042902	0.854624	-0.015011
funny	0.846138	1.000000	-0.041543	0.823759	-0.020031
stars_review	0.042902	-0.041543	1.000000	-0.040162	0.051385
useful	0.854624	0.823759	-0.040162	1.000000	-0.028944
is_open	-0.015011	-0.020031	0.051385	-0.028944	1.000000
latitude	-0.029170	-0.037964	-0.031977	-0.026753	0.011822
longitude	-0.047311	-0.051008	-0.033299	-0.035423	0.015738
review_count	0.022783	0.022769	0.066223	0.007781	0.084150
stars_business	0.043261	-0.000225	0.413689	0.012131	0.115686

	latitude	longitude	review_count	stars_business
cool	-0.029170	-0.047311	0.022783	0.043261
funny	-0.037964	-0.051008	0.022769	-0.000225
stars_review	-0.031977	-0.033299	0.066223	0.413689
useful	-0.026753	-0.035423	0.007781	0.012131
is_open	0.011822	0.015738	0.084150	0.115686
latitude	1.000000	0.810437	-0.159074	-0.073632
longitude	0.810437	1.000000	-0.291484	-0.078719
review_count	-0.159074	-0.291484	1.000000	0.157608
stars_business	-0.073632	-0.078719	0.157608	1.000000

Figure 9: Correlation matrix of numerical columns in the dataset

The target of our current analysis, `'stars_review'`, did not have notable correlations to any numerical data field, other than the `'stars_business'` (the average of `'stars_review'` for a restaurant). Since this field is an aggregated function of the review rating (stars_review), it shan't be used to predict the star rating for the current review as such a practice would promote the enforcement of existing opinions and negatively impacts independent judgment.

Similarly, to encourage independent judement of reviews to rate a business, we will remove `'name'`, `'business_id'` as assignment of the rating based on the business ID or name would reinforce the bias. Since `'review_id'` is an idex, we'll remove that too.

Therefore, we can conclude that for our analysis of `'stars_review'`, we can drop the columns: `'cool'`, `'funny'`, `'useful'`, `'is_open'`, `'latitude'`, `'longitude'`, `'review_count'`, `'stars_business'`, `'name'`, `'business_id'`, `'review_id'`

To validate if location significantly affected star ratings, the distribution of review ratings were plotted by state (figure 10).

The distribution of stars awarded by reviews are consistent with the exception of WA and SC. We see that WA has a very low number of reviews do not expect it to impact the analysis significantly.

SC has a greater proportion of low star ratings but not a very high number of reviews. We suspect this will not significantly impact the analysis either.
Since there is little correlation between `'stars_review'`, `'latitude'` and `'longitude'` as well as `'state'`, we will infer that location associated columns will not help us to determine the 'stars_review'.

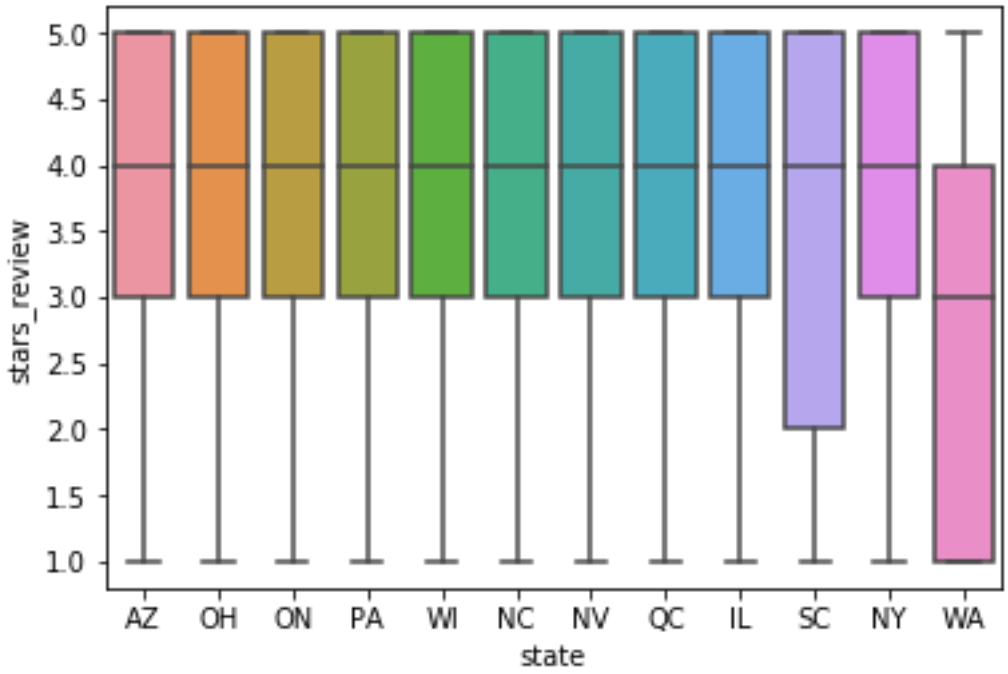

Figure 10: Distribution of review ratings by state

The remaining columns: `'user_id'`, `'attributes'`, `'categories'`, `'hours'`

- `'user_id'` : in the previous analysis we noted that majority of reviews have no previews reviews and should not be used as the basis of a predictor
- `'attributes'`, `'categories'`, `'hours'`: are descriptors of the business and not the review

After ruling out other columns in the data set as suitable predictors for the review star rating, sn initial look at the unigrams, bigrams and trigrams through word clouds (figure 11) suggests that the review text content might be an appropriate way of predicting the star ratings.

Figure 11: N-gram word clouds

Basic analysis of top words used - exploring the top 10 uni-grams based on star ratings (figure 12) and the rank of the word in terms of frequency of use (figure 13) - suggest differences based on star ratings.

TOP WORDS IN REVIEWS GROUPED BY STAR RATINGS

Rank of word in terms of frequency of use (in text by star rating):

1	time	good	good	good	great
.	like	like	like	great	good
.	one	time	time	like	time
.	back	one	would	time	best
5	get	get	one	one	one
.	would	would	get	really	like
.	go	back	really	get	delicious
.	good	really	great	go	go
	even	go	go	would	back
10	never	table	back	back	love

Figure 12: Top 10 unigrams by star rating

TOP WORDS IN REVIEWS GROUPED BY STAR RATINGS

Rank of word in terms of frequency of use (in text by star rating):

best	153	76	56	31	4
great	60	16	8	2	1
good	8	1	1	1	2
ok	118	50	35	251	540
bad	22	30	48	148	221
never	10	48	110	97	55

Top 10

Figure 13: Rank of words in terms of frequency of use (in text by star rating)

Since the exploratory data analysis section suggested that particular words such as 'bad' and 'best' might be good indicators of the star rating of the review. The hypothesis that frequency of occurrence for the two words are statistically significant were tested using permutation tests with a null hypothesis of identical distributions.

The Central Limit Theorem (CLT) is applicable to the difference in the rate of appearance of frequently used words such as 'bad' or 'best' since reviews are assumed to be independent events. In addition, the difference in frequency of

occurrences is expected to be normally distributed.

Using the test statistic - Frequency (rate) of word (bad / best) appearing – the following null hypotheses were tested:
1. The frequency of appearance of the word 'bad' is identical for 1 star rated reviews and the other reviews
2. The frequency of appearance of the word 'bad' is identical for 5 star rated reviews and the other reviews
3. The frequency of appearance of the word 'best' is identical for 1 star rated reviews and the other reviews
4. The frequency of appearance of the word 'best' is identical for 5 star rated reviews and the other reviews

In all 4 cases, the p-value for the hypotheses tests suggested that the Null Hypotheses may be rejected. The alternative hypothesis being that the words 'bad' and 'best' are statistically significant to the star rating that the review receives.

Following the previous findings and what was learnt, a new file was created for feature engineering. This working was done in a Jupyter notebook called `milestone_1.ipynb`.

This file repeats much of the work found in the previous notebooks and leverages the findings made so far. In particular:
- Inferential statistics analysis supports the notion that all columns other than the preprocessed text may be removed.
- Exploratory data analysis suggested the review text might be analyzed as a uni-gram, a bi-gram or a tri-gram. The current pre-processing in the data wrangling stage removes the stop words.

In addition to dropping all columns other than the pre-processed text and the star ratings, the following pre-processing steps will be used:
1. Removed punctuation
2. Convert characters to lower case
3. Lemmatization of words

The steps:
1. Removal of stop words
2. Conversion for review text string to a list of words (tokenization)

Will be handled by the CountVectorizer and the removal of n-grams solely comprising of stop words.

In view of this, a new CSV export, `review_text_stars.csv` was prepared.

6. Analyzing the data set using machine learning

For the final analysis was done in 8 stages. An overview showing these sub-stages are shown in figure 14 below.

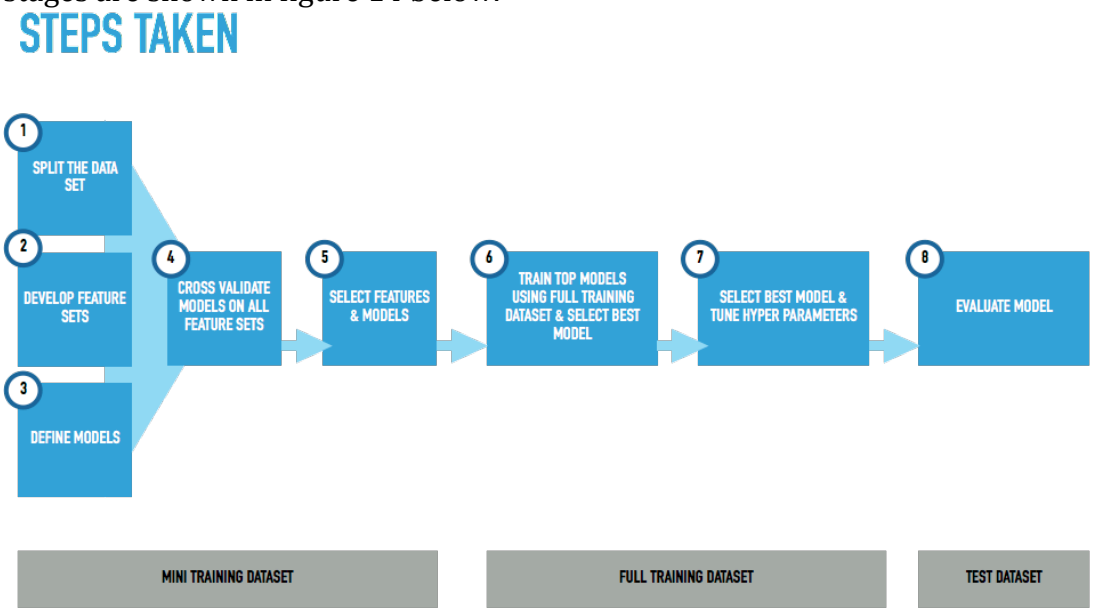

Figure 14: Steps taken in the final analysis

Step 1: Split the data set

In addition to the usual train-test data split, 2 additional mini-datasets were made from the train dataset.

The subsets of the full training dataset were made for the purposes of selecting suitable feature sets and models in a manageable period of time, due to the size of the full dataset.

2 datasets of varying sizes were used so that the performance by different models and feature engineering techniques could be observed. This was to identify significant performance issues due to different sample size requirements of machine learning models or feature sets.

The analysis was done in Jupyter notebooks `` `feature_engineering-mini-020k.ipynb` ``, `` `feature_engineering-mini_100k.ipynb` ``, `` `final_analysis.ipynb` ``.

The number of observations for each data split can be found in figure 15 below.

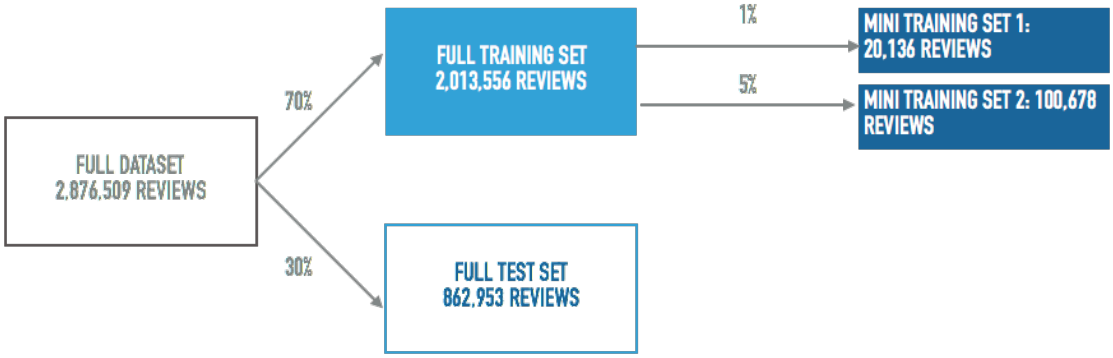

Figure 15: Data splits

Step 2: Develop feature sets

3 broad models of feature sets were used:

1. Bag-of-words
2. Term Frequency – Inverse Document Frequency (TFIDF)
3. Word to Vector

For each of these feature sets, 4 variants were created through additional feature engineering steps. In total, this produced 12 feature sets. These are explained in figure 16 below.

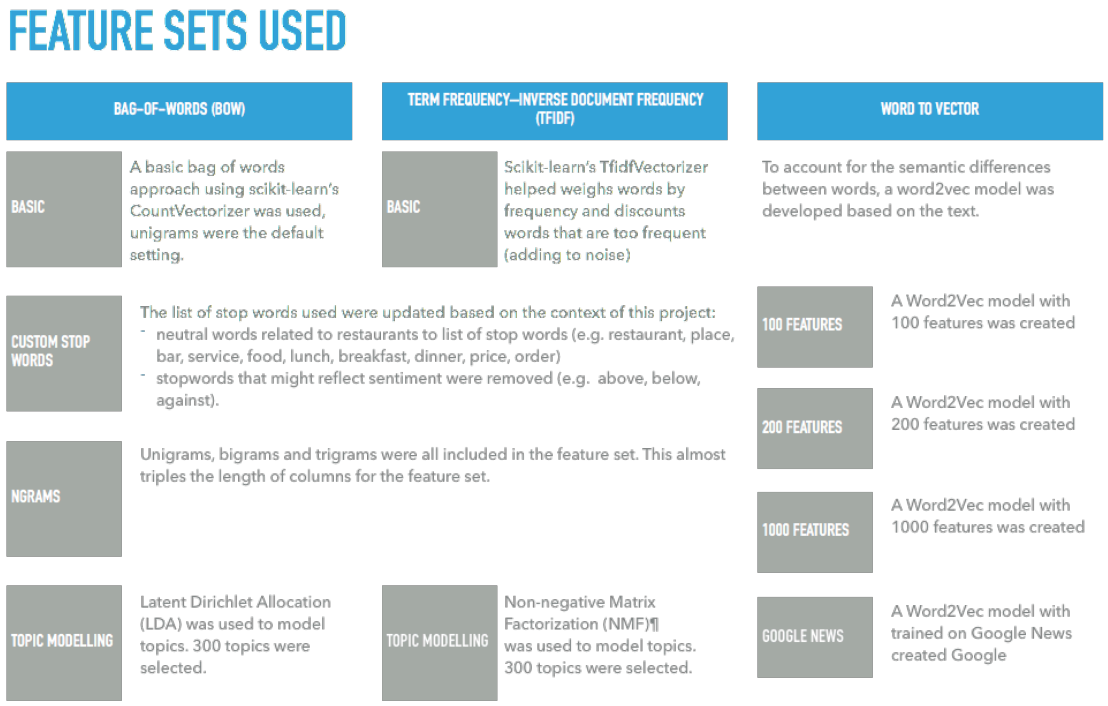

Figure 16: Feature sets used

Step 3: Define models

A total of 7 machine learning models were used for the multi-class classification problem.

1. Naïve-Bayes
2. Logistic Regression
3. Support Vector Machines (SVM)
4. Random Forest
5. Stochastic Gradient Descent (SGD)
6. Extreme Gradient Boosting (XGB)
7. Deep Neural Nets (DNN)

Each of these models were trained using a number of hyper parameters to ensure that the model was not severely underperforming due to the lack of tuning. The hyper parameters chosen for the cross validation can be found in figure 17 below.

Figure 17: Machine learning models used

Step 4: Cross validate models on all feature sets

Models were trained on the 20k and 100k observations mini-datasets for each of the different features sets. The results are shown in the figures 18 and 19 below.

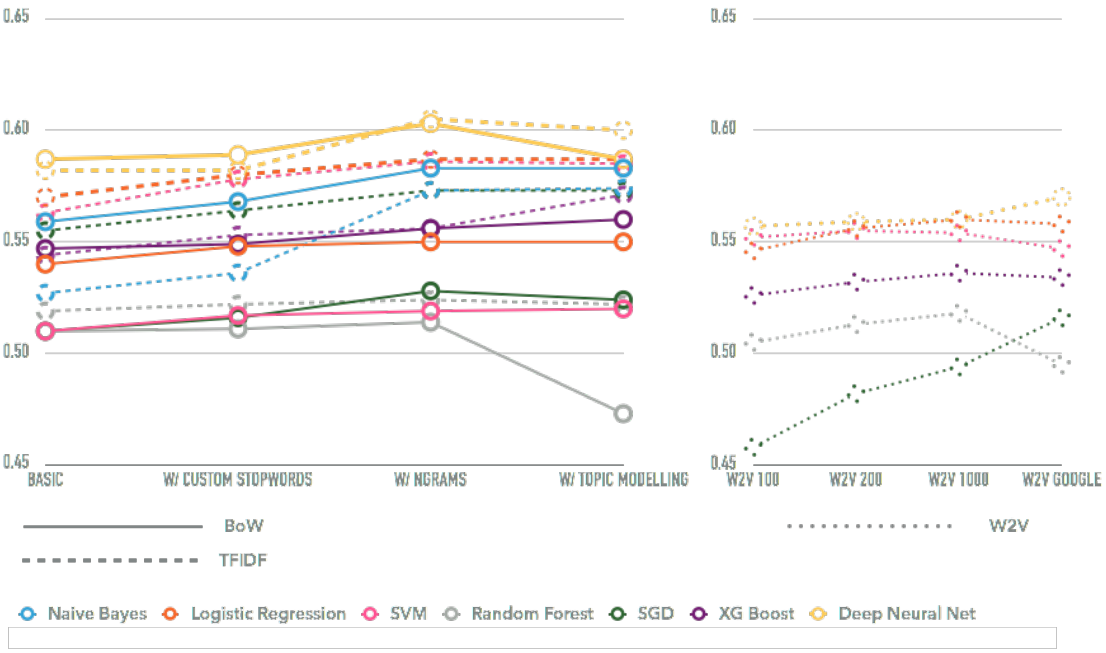

Figure 17: Cross validation accuracy of models and features (20k mini dataset)

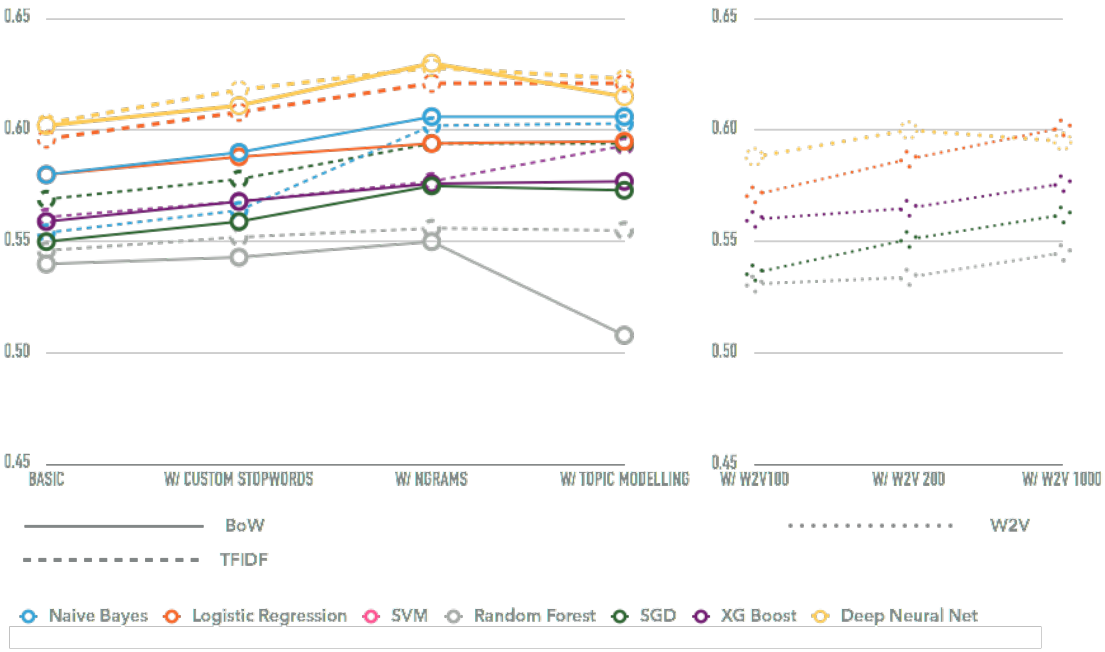

Figure 18: Cross validation accuracy of models and features (100k mini dataset)

The tabular results are shown in the appendix.

It should be noted that:

1. Naïve-Bayes cannot be trained on word to vec feature sets due to the negative values in the feature set
2. the Google News feature set was dropped from the 100k analysis as the results in the 20k analysis were not promising
3. SVM model was not trained on the 100k dataset as it took over 24 hours for each feature set indicating that this model would not scale well for the full dataset

Step 5: Feature and model selection

The result of the model performance is as follows:
Results for 20k dataset:

1. DNN (TFIDF ngrams)
2. DNN (Count ngrams)
3. logistic regression (TFIDF ngrams)
4. SVM (TFIDF ngrams)
5. Naive-Bayes (Count ngrams)

Results for 100k dataset:

6. DNN (Count ngrams)
7. DNN (TFIDF ngrams)
8. logistic regression (TFIDF ngrams)
9. Naive-Bayes (Count ngrams)
10. Naive-Bayes (TFIDF ngrams)

A generalized summary of the result findings about the feature sets is as follows:

- Custom stop words showed a small improvement
- Unigram, bigram and trigram feature sets provided better performance for all models
- Topic modelling provided little to no improvement for all models (except XG boost's TFIDF)
- Word 2 Vector feature sets showed poorer performance vs ngrams feature sets for all models (except 100k logistic regression)

The impact of varying the sample size is as follows:

- Without exception, all models trained on the 100k dataset showed noticeable improvement
- The performance of the different models trained on the different feature sets did not show significant variance relative to each other when comparing the 20k and 100k datasets. (Remember, this is used as a proxy to indicate issues due to sample size requirements.)
- The SVM model trained exceptionally slowly on the 100k dataset and had to be abandoned due to its inability to scale.

In conclusion, 2 models (DNN and Logistic Regression) were selected for the next stage of training the full dataset on. The feature sets selected were the Bag-of-Words feature set as well as the TF-IDF feature set, both to be trained using customized stopwords as well as unigrams, bigrams and trigrams. However, topic modeling will not be used, as it did not provide a significant improvement to accuracy.

Step 6: Train top models using full dataset & select best model

As mentioned in the previous step, TF-IDF and Bag-of-words feature sets were used to train 2 models a Deep-Neural-Network and Logistic Regression.

The analysis was done in Jupyter notebook `` `final_analysis.ipynb` ``.

The DNN models trained on both feature sets were trained using 3 different architectures shown in table 1 below:

Table 1: DNN architecture for initial training

	Layers	Nodes per layer	Dropout	Activation
1	2	512	0.3	relu
2	2	256	0.3	relu
3	3	512	0.3	relu

The Logistic Regression model was trained using 10 different combinations of hyper parameters:

- C: 0.0001, 0.01, 1, 100, 10000
- Penalty: l1, l2

The result shown in figure 19 below is that DNN model with 2 layers, 256 (relu) nodes per layer performed the best when trained on the TF-IDF feature set.

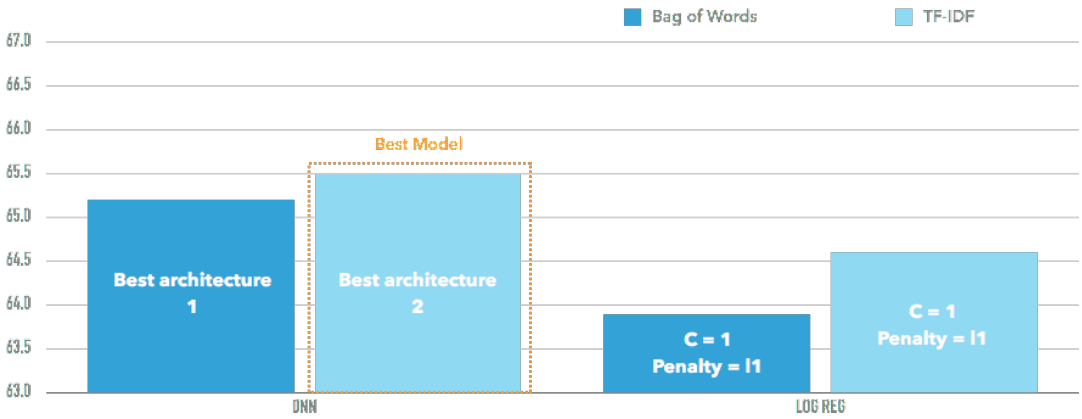

Figure 19: Training accuracy for DNN and Log-Reg trained on full dataset

The DNN model trained on TF-IDF will be tuned in the next stage to optimize the performance of the model.

Step 7: Tune hyper parameters

The DNN model trained on TF-IDF (with custom stop words, unigrams, bigrams and trigrams) was tuned using a variety of DNN architectures as shown in table 2:

Table 2: DNN architecture for tuning

	Layers	Nodes per layer	Dropout	Activation	Optimizer
1	2	512	0.3	relu	adam
2	2	256	0.3	relu	adam
3	3	512	0.3	relu	adam
4	2	256	0.3	tanh	adam
5	2	256	0.3	sigmoid	adam
6	2	256	0.2	relu	adam
7	2	128	0.3	relu	adam
8	2	128	0.3	sigmoid	adam
9	2	128	0.4	sigmoid	adam
10	2	256	0.4	sigmoid	adam
11	2	256	0.4	sigmoid	sgd

The training accuracy is shown in figure 20 below.

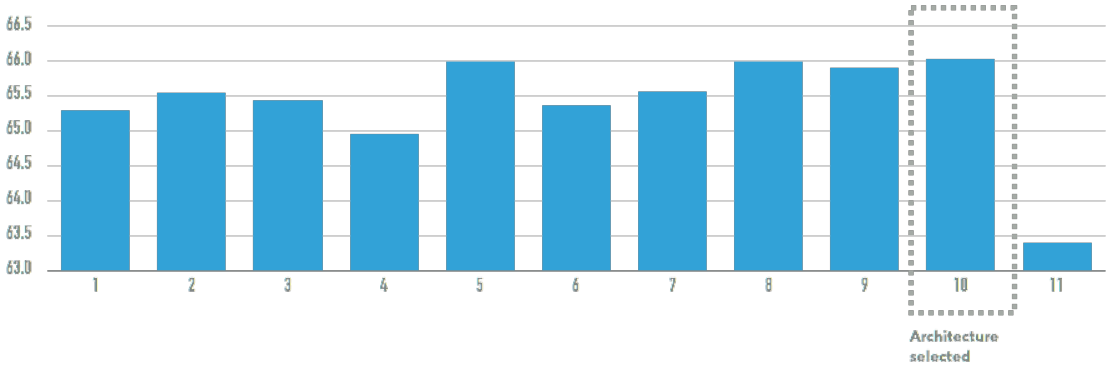

Figure 20: Training accuracy for DNN trained on different architectures

The best performing DNN model had the architecture shown in figure 21 below.

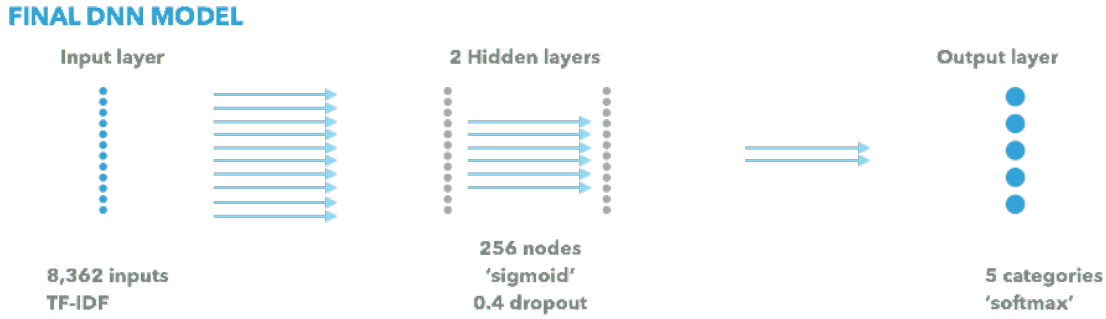

Figure 21: Final DNN model

Step 8: Evaluate model

Before evaluating the model a base case accuracy should be proposed. If one assumes than a random rating is assigned, the base case accuracy is approximately 20% for 5 classes. However, given that the most frequently occurring review is a 5 star review, if a default value of 5 stars is assigned, the base case accuracy would increase to 37.0%.

In addition, the level of accuracy should take into the context of a highly subjective rating system by humans. Sentiment analysis research shows *human raters agree with a rating 79% of the time*[1].

The DNN model was evaluated using the test set data that had been set-aside since the start of the analysis. The overall weighted average accuracy of the model is 66.2%. The confusion matrix is shown in the table 3 below:

Table 3: Confusion matrix of final DNN model

	1	2	3	4	5
1	71, 488	18, 530	2, 942	1, 178	1, 382
2	17, 547	41, 503	19, 881	3, 957	1, 401
3	3, 783	16, 802	58, 523	37, 859	5, 749
4	1, 130	1, 994	19, 139	135, 826	82, 350
5	969	581	2, 129	52, 600	263, 710

For the confusion matrix, rows correspond to actual ratings and the columns correspond to predicted ratings.

The accuracy, precision, recall is as described by class in table 4.

Table 4: Class specific results

	1	2	3	4	5
Accuracy	66.2 %	66.2 %	66.2 %	66.2 %	66.2 %
Precision	75.3 %	52.3 %	57.0 %	58.7 %	74.4 %
Recall	74.8 %	49.2 %	47.7 %	56.5 %	82.4 %

As we may observe from the table above, the model is good at distinguishing 1 and 5 star reviews, but it struggles to accurately classify ratings in between.

Whilst accuracy, precision and recall are typical metrics for multi-class classification problems, our particular use case of star ratings have numerical implications to the classes we assign. Namely, a misclassification of a 4-star rating for a 5 star review is not as significant as a misclassification of a 1-star rating to a 5 star review. The average star prediction per class is given in table 5 below.

Table 5: Average rating predicted for a class of reviews

	1	2	3	4	5
Average predicted rating (stars)	1.35	2.17	3.20	4.23	4.80

On average the predicted star rating was 3.78 stars / review (vs. the actual overall average rating of 3.70 stars/ review).

In general, the averages ratings are slightly higher for most classes (except for 5 star reviews) and the overall average predicted rating is quite close to the actual.

To further reduce the distance of the average star rating, a customized loss function weighted by the squared difference between the actual and predicted star rating (similar to R-squared error) could be created. This approach could result in lower classification accuracy, but the distance of the misclassification would reduce.

7. Conclusions and business implications

The model produced achieved an overall accuracy of about 66.2%. Whilst this level of accuracy would imply that it is not ready to replace human rating, the use case of providing a dynamic default rating based on the review is still feasible as it provides an improvement over the base case accuracy of 37.0% (by having a default rating of 5 stars).

In addition, one could speculate that if the variation in classification can be partially attributed to the subjectivity of users doing the ratings, this dynamic default rating may provide an added benefit of calibrating ratings by subconsciously influencing users. However, this is pure speculation and further research (e.g. AB testing) is necessary to validate this hypothesis.

It should also be noted that when creating topic models in the feature engineering stage, the topics trained on the entire data set were fairly generic. However, using a sematic topic modeling approach as suggested by the following [paper] (https://www.yelp.com.sg/html/pdf/YelpDatasetChallengeWinner_PersonalizingRatings.pdf) may improve the accuracy of the model (over the traditional LDA approach used), as well as form the basis of a more robust product. In particular, by identifying the topics mentioned in the review and combining user data, it may be possible to identify preferences of individual users (e.g. better service, better value) and create a more personalized product through:

1. more accurate rating predictions
2. personalized search rankings
3. personalized recommendations

This basic model therefore could provide the basis for a dynamic rating system (subject to change by the reviewer) and provide helpful inputs for a more personalized user experience on Yelp.

Appendix

CROSS VALIDATION ACCURACY OF FEATURE SETS

			Basic	w/ custom stopwords	w/ ngrams	w/ topic modelling
20K DATASET	BAG OF WORDS	Naive Bayes	0.559	0.568	0.583	0.583
		Logistic Regression	0.540	0.548	0.550	0.550
		SVM	0.510	0.517	0.519	0.520
		Random Forest	0.510	0.511	0.514	0.473
		SGD	0.510	0.516	0.528	0.524
		XG Boost	0.547	0.549	0.556	0.560
		Deep Neural Net	0.587	0.589	0.603	0.587
	TFDIF	Naive Bayes	0.527	0.536	0.573	0.574
		Logistic Regression	0.570	0.580	0.587	0.587
		SVM	0.563	0.578	0.586	0.585
		Random Forest	0.519	0.522	0.524	0.522
		SGD	0.555	0.564	0.573	0.573
		XG Boost	0.544	0.553	0.556	0.571
		Deep Neural Net	0.582	0.582	0.605	0.600
100K DATASET	BAG OF WORDS	Naive Bayes	0.580	0.590	0.606	0.606
		Logistic Regression	0.580	0.588	0.594	0.595
		SVM				
		Random Forest	0.540	0.543	0.550	0.508
		SGD	0.550	0.559	0.575	0.573
		XG Boost	0.559	0.568	0.576	0.577
		Deep Neural Net	0.602	0.611	0.630	0.615
	TFDIF	Naive Bayes	0.554	0.564	0.602	0.603
		Logistic Regression	0.596	0.608	0.621	0.621
		SVM				
		Random Forest	0.546	0.552	0.556	0.555
		SGD	0.569	0.578	0.594	0.594
		XG Boost	0.561	0.568	0.577	0.593
		Deep Neural Net	0.603	0.618	0.628	0.623

CROSS VALIDATION ACCURACY OF WORD 2 VECTOR FEATURES

		100 features	200 features	1000 features	GoogleNews
20K DATASET	Logistic Regression	0.546	0.556	0.560	0.558
	SVM	0.552	0.555	0.554	0.547
	Random Forest	0.505	0.513	0.518	0.495
	SGD	0.458	0.482	0.494	0.516
	XG Boost	0.526	0.532	0.536	0.534
	Deep Neural Net	0.557	0.559	0.560	0.570
100K DATASET	Logistic Regression	0.571	0.587	0.601	
	SVM				
	Random Forest	0.531	0.534	0.545	
	SGD	0.536	0.551	0.562	
	XG Boost	0.560	0.565	0.576	
	Deep Neural Net	0.588	0.600	0.595	

References

[1] (https://www.yelp.com.sg/html/pdf/YelpDatasetChallengeWinner_PersonalizingRatings.pdf)