Yelp Restaurant Reviews Analysis and Rating Prediction

CSDA1050 – Advanced Analytics Capstone

By Steven Too Heng Kwee

ID: 304449

Introduction

Yelp is an American multinational corporation founded in 2004 which aimed at helping people locate local business based on social networking functionally and reviews. Yelp also has a star rating system that lets users easily see what the general opinion about a particular establishment is without having to read all the reviews for that particular business.

Millions of people use Yelp restaurant reviews and ratings in their food choice decision-making. Empirical data research demonstrated that an average one-star increase led to 59% increase in revenue of independent restaurants (Lucas, 2011).

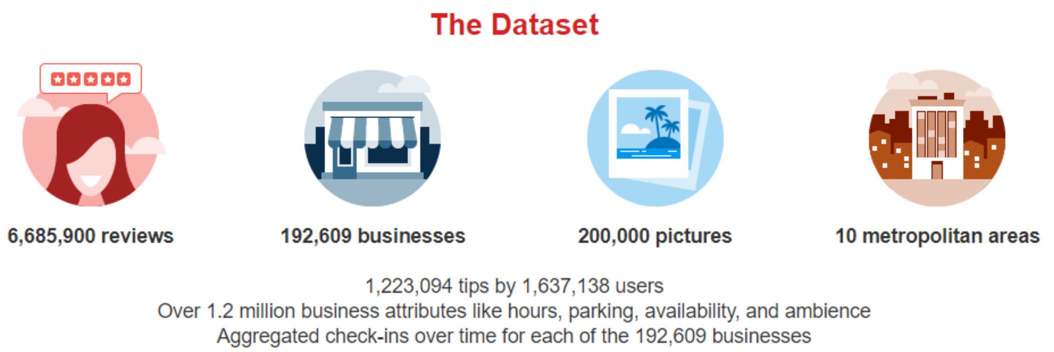
However, user rating is very subjective and biased. The same expressed opinion can be rated differently by users.

Research Question

Analysing the reviews and star rating set by reviewers. Did they provide a proper rating? Can we detect the inconsistencies and provide a more accurate rating instead?

Data and Description

The data comes from Yelp Dataset Challenge available at https://www.yelp.ca/dataset. It is a small subset of Yelp data about local businesses in 10 metropolitan areas.



We downloaded a 5.6 GB TAR file. This TAR file contained second TAR file that we extracted to get a series of JSON files: business, checkin, photos, review, tip, and user. Total real size is 8.05 GB.

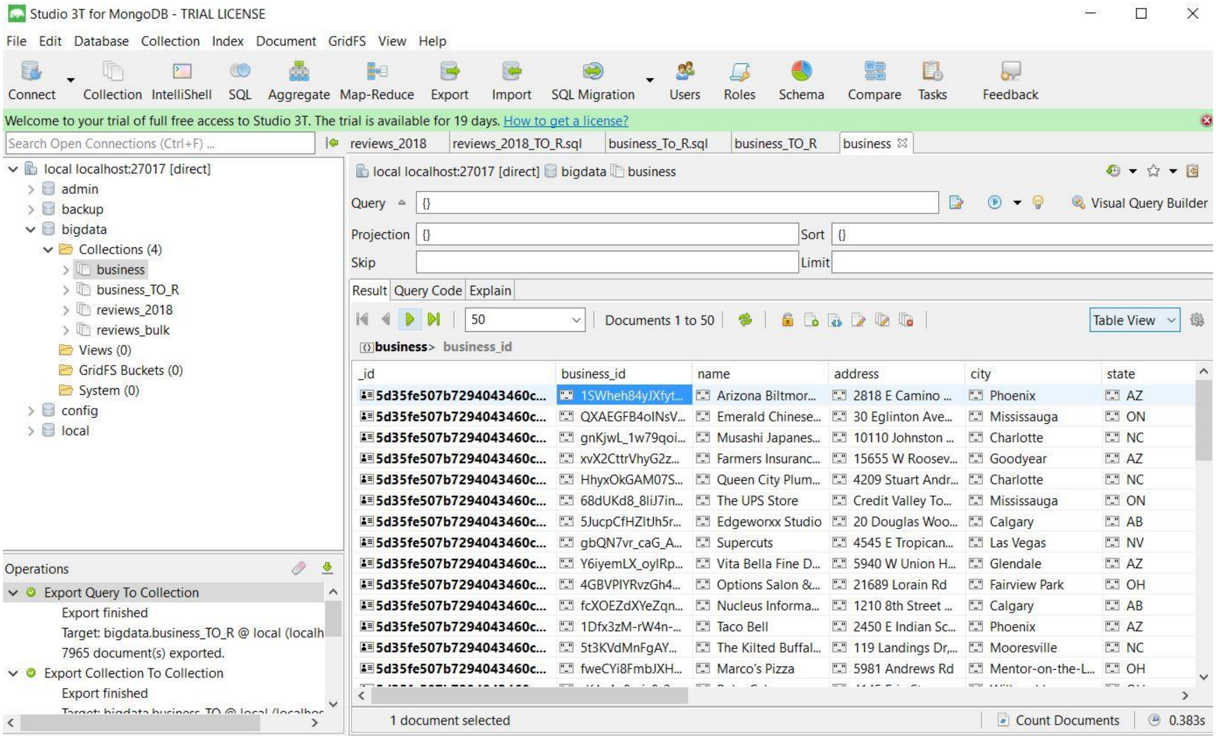
We will be focusing on the following files:

Business.json: 131 Mb. Contains business data including location data, attributes, categories and average star rating

Review.json: 4.97 Gb. Contains full review text data including the user_id that wrote the review and the business_id the review is written for.

Data processing

Due the size of the files which makes them impossible to load directly into a Pandas dataframe. MongoDB is used as the repository and segmentation tool. PyMongo is then used to retrieve the data into Python



Please check sprint 1 for instructions. (https://github.com/stoohengkwee/CSDA-1050F18S1/tree/master/StevenToo-304449/sprint%201)

Proposed Methodology

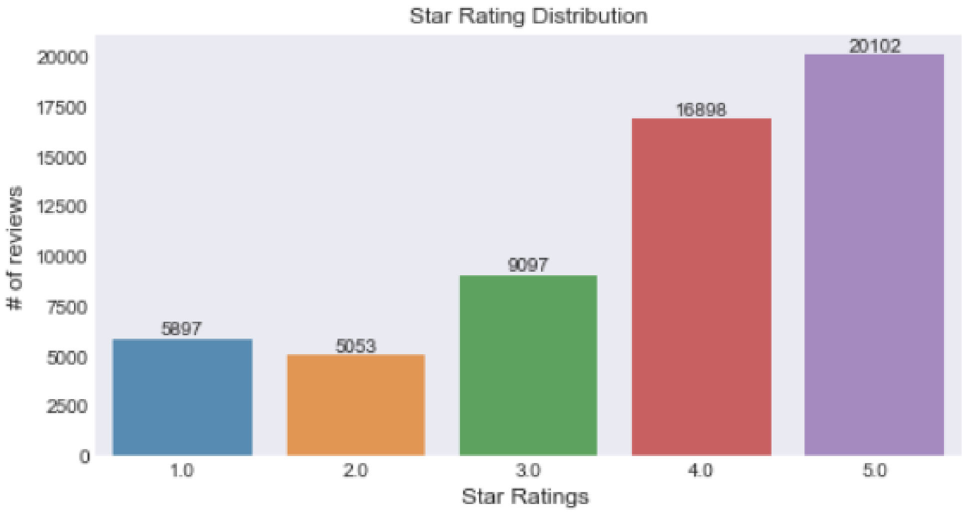
- Data segmented to 2018 reviews for restaurants in Toronto
- Python Notebook will be used for codebase and analytics
- Textual data clean that might be required: lower text, tokenize, remove punctuation and stop words, lemmatize.
- Exploration of the restaurants and reviews. Perform Sentiment Analysis to determine word/feature drivers(Word Cloud)
- For the rating analysis and prediction, we explore several machine learning methods namely Naive Bayes, Random Forest Classifier, Support Vector Machine (SVM), Decision Tree and Multilayer Perceptron Classifier to make relevant predictions.

Exploratory Data Analysis

Number of reviews extracted: 57,047

Number of businesses: 7,965

Above is the Top 10 Restaurants with the most reviews and the rating distribution set by yelp.



Observations: The business ratings set by Yelp follows a normal distribution. However, the users review ratings are mostly 4.0, 5.0 and notice the higher proportion of 1 star. Based on the reviews alone we can see the tendency by reviewers to over and under rate.

Features associated with positive user reviews: place, food, service, time, décor, furniture

Features associated with negative user reviews: food, service, place, order, wait, server, lunch

Observations: Most words are common on both positive and negative reviews. This denotes the need for context to determine the actual sentiment

The above shows that user ratings would not be possible if based on word count and single word extraction. The model predictor would have to take into account the whole set of words of each review to provide context and then trained for accurate stars.

Modelling Process

Please check for documentation and codebase in sprint 2 at https://github.com/stoohengkwee/CSDA-1050F18S1/tree/master/StevenToo-304449/sprint%202

- Classifying the dataset and splitting it into the reviews and stars

```
# CLASSIFICATION
data_classes = data[(data['stars_x']==1) | (data['stars_x']==3) | (data['stars_x']==5)]
data_classes.head()
print(data_classes.shape)

# Seperate the dataset into X and Y for prediction
x = data_classes['text']
y = data_classes['stars_x']
print(x.head())
print(y.head())
```

In order to reduce processing time and increase accuracy, the dataset was categorized to to 1, 3 and 5 stars thereby reducing the reviews to 35,096. Attempt was done with all 5 categories but proven less accurate.

- Cleaning was done in removing punctuations and stopwords
- Machine learning requires text reviews to be converted into word count representation called Vectorization. The function CountVectorizer is used here with default parameters and n-grams not set.
- The train and test dataset was split into 80/20 sets.

Models

(1). Multinomial Naive Bayes - We are using Multinomial Naive Bayes over Gaussian because with sparse data, Gaussian Naive Bayes assumption of a normal distribution is not met and by default is not a good fit in this present case.

```
Confusion Matrix for Multinomial Naive Bayes:
[[ 899  247   37]
 [ 153 1359  296]
 [  52  267 3710]]
Score: 85.01
Classification Report:              precision    recall  f1-score   support

        1.0       0.81      0.76      0.79      1183
        3.0       0.73      0.75      0.74      1808
        5.0       0.92      0.92      0.92      4029

avg / total       0.85      0.85      0.85      7020
```

(2). Random Forest Classifier

```
Confusion Matrix for Random Forest Classifier:
[[ 689  220  274]
 [ 199  842  767]
 [  63  323 3643]]
Score: 73.7
Classification Report:              precision    recall  f1-score   support

        1.0       0.72      0.58      0.65      1183
        3.0       0.61      0.47      0.53      1808
        5.0       0.78      0.90      0.84      4029

avg / total       0.73      0.74      0.72      7020
```

(3). Decision Tree

```
Confusion Matrix for Decision Tree:
[[ 702  288  193]
 [ 275  895  638]
 [ 172  537 3320]]
Score: 70.04
Classification Report:              precision    recall  f1-score   support

        1.0       0.61      0.59      0.60      1183
        3.0       0.52      0.50      0.51      1808
        5.0       0.80      0.82      0.81      4029

avg / total       0.70      0.70      0.70      7020
```

(4). Support Vector Machines

```
Confusion Matrix for Support Vector Machines:
[[   5    1 1177]
 [   1    1 1806]
 [   0    0 4029]]
Score: 57.48
Classification Report:              precision    recall  f1-score   support

        1.0       0.83      0.00      0.01      1183
```

```
        3.0       0.50      0.00      0.00      1808
        5.0       0.57      1.00      0.73      4029

avg / total       0.60      0.57      0.42      7020
```

(5). MULTILAYER PERCEPTRON CLASSIFIER

```
Confusion Matrix for Multilayer Perceptron Classifier:
[[ 945  182   56]
 [ 156 1306  346]
 [  40  266 3723]]
Score: 85.1
Classification Report:
             precision    recall  f1-score   support

        1.0       0.83      0.80      0.81      1183
        3.0       0.74      0.72      0.73      1808
        5.0       0.90      0.92      0.91      4029

avg / total       0.85      0.85      0.85      7020
```

Summary

Models	Precision % - 1,3,5 stars	Precision % - all stars
Multilayer Perceptron	85.1	54.72
Multinomial Naive Bayes	85.01	56.91
Random Forest Classifier	73.7	47.25
Decision Tree	70.07	43.3
Support Vector Machine	57.48	35.24

Multilayer Perceptron Classifier produced the best accuracy score. Let us use it to predict a random positive review and a random negative review!

```
# POSITIVE REVIEW
pr = data['text'][74]
print(pr)
print("Actual Rating: ",data['stars_x'][74])
pr_t = vocab.transform([pr])
print("Predicted Rating:")
mlp.predict(pr_t)[0]
```

```
Oh my god the cinnamon bun pancakes were DELICIOUS! I only ordered off SkipTheDishes but this place was really good. We got a b
asically breakfast plate with eggs and all that which was good. But the pancakes were really the star of the dish. I dream abou
t these pancakes sometimes when I have a craving for something sweet. I highly recommend them.
Actual Rating:  4.0
Predicted Rating:

4.0
```

```
# NEGATIVE REVIEW
nr = data['text'][90]
print(nr)
print("Actual Rating: ",data['stars_x'][90])
nr_t = vocab.transform([nr])
print("Predicted Rating:")
mlp.predict(nr_t)[0]
```

```
This place is a complete hit and miss depending on when you visit. I just finished throwing out a chicken wrap consisting of dr
y, inedible chicken scraps. The soup was good, as usual, but the crappy wraps ruined the entire experience.  Weekends are gener
ally a bad time to visit. If you are curious to try this place out, best time to go in terms of food quality is lunchtime durin
g weekdays.
Actual Rating:  1.0
Predicted Rating:

1.0
```

Discussion and Conclusion

- This above model has many applications not limited to reviews. I can be uses to any text that required some sort of scoring or detect unfair or erroneous ratings
- We are able to accurately the stars rating according to the reviews. However when we look at the negative sample, it appears that it should have received a higher rating.

Improvement Prospects

- Review the vectorization process. Processing time too long.
- Review the whole modelling concept. Star rating prediction trained current ratings. Research to undertake for rating based on just reviews text and sentiment alone. Also to consider normalizing the data and adding some weight element to the words.

References

https://github.com/Yelp/dataset-examples

https://www.geeksforgeeks.org/python-nlp-analysis-of-restaurant-reviews/

http://www.developintelligence.com/blog/2017/03/predicting-yelp-star-ratings-review-text-python/