Predicting Restaurants reviews ratings on Yelp.com

Oluwayomi Oluwaniyi Amodu[1] and Sio Kei Chang[1]

[1] School of Computing, Dublin City University, Glasnevin, Dublin, Ireland
{oluwayomi .amodu2, sio.chang3}@mail.dcu.ie

1.0 Introduction

Online review systems, such as Yelp, have become an important source of information for consumers looking for information about businesses. Yelp is a platform that allows users to rate and review businesses such as restaurants, bars, and shopping centers. With over 224 million unique visitors per month, Yelp has become a significant player in the online review space. Yelp reviews provide businesses with valuable feedback on their products and services and can help them identify potential areas of improvement. Yelp users can also benefit from the reviews as they can receive personalized recommendations based on their preferences.

In this paper, we present a study on the Yelp dataset aimed at two objectives, one is to develop a predictive system to determine whether a user will rate a restaurant business above or below 3.8 stars. We chose 3.8 stars as the threshold because it is close to the average rating on Yelp, which is 3.77. Predicting whether a user will rate a business above or below 3.8 stars can be useful for businesses to identify potential issues and areas of improvement pre or post opening the business. The second one is to predict the actual user's star rating for a restaurant through a natural language processing model.

This research can be useful for businesses to identify potential issues and areas of improvement pre or post going opening. We would review the Yelp dataset and explore various machine learning techniques to develop a model that accurately predicts user ratings. Our analysis will also include an evaluation of the effectiveness of different feature selection and preprocessing techniques. Ultimately, we hope that our study will contribute to the ongoing research on machine learning and its applications to real-world problems.

1.1 Dataset Summary

Yelp Data Set
The data encompassed large files (over 8gb in total) and was zipped into a TAR file. Once we extracted the individual files from the TAR file, there were multiple JSON files to be used. For this project, we only needed a few of these JSON files, the ones that were most relevant. These files were: business.json, review.json, and user.json. These are still very big in size, over 8gb. We then tried to filter the data , taking in only restaurant and food related businesses , this has helped to further reduce the files.

The three files to be used combined to be about 7.7gb and gave us a lot of trouble in a few ways. Initially, reading in the data due to computational constraints (Google colabs 12.7gb of ram machines). Using a home-built computer, we were able to read in the files, but it took a few tries. This was completed using a function that converted the JSON string into a flat python dictionary which was saved as a ".csv" file, which could then be passed into Pandas. We also sent data into

a SQLite database chunk by chunk, this database was later used for Exploratory Data Analysis, or EDA, and modeling. Another issue was the time it took to run many different lines of code on files of this size.

- The business dataset broke down into many columns, the majority of which were business attributes.
- The user dataset broke down in fewer columns and gave us data regarding the Yelp users.
- The review dataset broke down into just 4 columns. 4

2.0 Introduction of Algorithms

As the internet has become the central platform for sharing experiences and opinions, extracting information from websites, conducting aspect-based opinion mining and text mining have become crucial. In this context, the authors proposed a system where users could add tags to reviews to correspond with positive or negative sentiment, enabling analysis of the opinion in the text review. In [1] authors' results surpassed the human-labeled options with 80% accuracy, achieved using Naive Bayes and Support Vector Machines algorithms on movie review data, by analyzing the sentiment of the whole review. The authors proposed using sentence-by-sentence level document classification during sentiment analysis, extracting only those characteristics or features where opinions were expressed by utilizing a WordNet lexical database[1]. In [3] for each trait, the authors included the comparing supposition sentence in the positive or negative categories and computed the complete count. The alternatives were stratified according to the recurrence of their appearance in the reviews. The authors provided an inclusive-based list of reviews of items sold online. Previous studies had already proposed utilizing opinion analysis and Yelp dataset reviews to predict star ratings using text reviews alone. The authors utilized various machine learning methods, including Perceptron, Naive Bayes, and Multiclass Support Vector Machines on a test of 100k client reviews from the Yelp dataset[3]. They used the Bing Liu Opinion Lexicon for feature selection and several preprocessing procedures, such as removing stop words, punctuation, and stemming. The best results (recall and precision) were achieved with Naive Bayes and feature selection with stop words removed and stemming. As neural networks-based approaches are gaining popularity due to requiring less language-specific information and engineering, many sentiment analysis applications utilize them. The system described in this paper used numerous innovative linguistic options, publicly available sentiment lexicon corpo-ra, and automatically generated extremity.

2.1 Models Introduction

We ran the Yelp Dataset through seven models (DecisionTreeClassifier, ExtraTreesClassifier, KNeighborsClassifier, Naive Bayes, LogisticRegression, RandomForestClassifier, AdaBoostClassifier), below we talk about the best performers.

1. Logistic Regression

   Conditional probability function P(s|r) for logistic regression modeling, where s is a member of the set of tags 1, 2, 3, and 4, and r is the review's eigenvector. For each new review, a new eigenvector r is provided. The probabilistic function then calculates all of the s values, and the s value with the highest probability is produced as the review's final class tag (star rating).The family parameter is automatically set to binomial for logistic regression, whereas the link parameter is set to logit. To give a straightforward logistic regression, this operator can only be used with the most important parameters changed. Implementations of logistic regression can handle nominal and numerical characteristic characteristics as well as binomial (or 2-class polynomial) tags for training data.

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/s}}$$

2. Naive Bayes

   The Naive Bayes classifier allows the naive Bayes hypothesis model to combine the probability P(r, s) for each eigenvector r and star s. This model is based on the assumption that each pair's conditional independence results in certain classes. The joint probability function then calculates all possible values of s, and the probability corresponding to each one is output as the class tag with the highest output at the conclusion of the review, represented by a new feature vector, r.A classifier with high bias and little variance is naive bayes. It can create a good model with ease of use and cheap computational cost even with a tiny data set. Text categorization, including spam detection, sentiment analysis, and recommendation systems are examples of typical use applications.

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

   We picked Naive Bayes as the main model in our natural language processing section because it is suitable for text classification due to its assumption of independence and high performance in solving multi-class problems. Naive Bayes models can accomplish high levels of accuracy while estimating the class-conditional marginal densities of data.

   In the context of Yelp's large dataset, Naive Bayes model is suitable for predicting star rating of businesses because it can quickly learn to use high dimensional features with limited training data compared to more sophisticated methods[2].

3. RandomForestClassifier

An ensemble learning algorithm for classification tasks is called RandomForestClassifier. To produce a final categorization, it generates numerous decision trees and combines their predictions. A random subset of the characteristics and data samples are used to build each decision tree in order to lessen overfitting and boost the model's generalization capabilities. Also, the method chooses at random a subset of attributes to take into account for each split in the tree, reducing the correlation between the trees and enhancing the model's accuracy. A flexible and effective technique, RandomForestClassifier excels with both high- and low-dimensional datasets. Compared to single decision tree methods, it can manage noisy and missing data and is less prone to overfitting. The number of trees, the maximum depth of each tree, and the amount of characteristics to take into account for each split are some of the hyperparameters that can be adjusted to improve performance.

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

4. AdaBoostClassifier

AdaBoostClassifier is an ensemble learning approach that combines several weak classifiers to produce a strong classifier and is used for classification tasks. The technique gives larger weights to the misclassified data points in order to concentrate on the challenging examples. Each weak classifier is built using a different subset of the training data. The approach trains a new weak classifier on the updated dataset and modifies the weights of the training samples based on their classification error in each iteration. The predictions of all the weak classifiers are combined, with the accuracy of each classifier being weighted in the final classification. AdaBoostClassifier is an ensemble learning algorithm that incorporates various learning techniques for classification tasks. AdaBoostClassifier is a versatile and strong algorithm that excels even with noisy data and can handle complex datasets. The number of weak classifiers, learning rate, and base estimator used to train the weak classifiers are some of the hyperparameters of the algorithm that can be adjusted to improve performance.

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

5. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric machine learning technique that may be used for classification as well as regression issues. The fundamental principle underlying KNN is to forecast the class or value of a data point by examining the k-nearest neighbors in the training set and giving the majority class or average value of those neighbors as the prediction.

3.0 The specific challenges during project completion and solutions:

3.1 Processing large datasets

Due to the large dataset of the data , the initial data setup and cleaning has been cumbersome and slow. At one point the google colab did not allow us to run the file as we are over the limit on the RAM usage.

There were a few solutions we performed to counter this problem. The first one was creating a sqlite database and preprocessing the data in chunks to store the data on the tables on one file, later we call on the database and perform data analysis on another file. This has somewhat speeded up the output execution process.

Another method we used was enabling GPU on our collab notebook using the code as shown below.

```
[ ] import tensorflow as tf

    device_list = tf.test.gpu_device_name()
    if device_list != '/device:GPU:0':
      raise SystemError('GPU device not found')
    print('Found GPU at: {}'. format(device_list))

    Found GPU at: /device:GPU:0
```

fig1.0 above shows the code on enabling GPU

One method we used to process such a large dataset was by parsing the data and loading the data in chunks as shown below.

```
%%time
#Here we load data from reviews

#size of review.json overpowers RAM hence we'll process data in chunks and store it to table
for data in load_rows_gen(REVIEW_PATH):
    #transformations
    data.user_id = data.user_id.apply(lambda key : userid_to_idx[key] if key in userid_to_idx else np.nan)
    data.business_id = data.business_id.apply(lambda key : businessid_to_idx[key] if key in businessid_to_idx else np.nan)
    data.date = pd.to_datetime(data.date)
    data.dropna(inplace=True)
    #sending chunk to sql
    data.to_sql('reviews', db, if_exists='append', index=False)
#del data

Loaded chunk of size: 1000000 , Time = 12.62 secs.
Loaded chunk of size: 1000000 , Time = 11.48 secs.
Loaded chunk of size: 1000000 , Time = 13.19 secs.
Loaded chunk of size: 1000000 , Time = 11.87 secs.
Loaded chunk of size: 1000000 , Time = 13.95 secs.
Loaded chunk of size: 1000000 , Time = 12.66 secs.
Loaded chunk of size: 990280 , Time = 14.73 secs.
CPU times: user 8min 30s, sys: 5min 33s, total: 14min 3s
Wall time: 1h 24min 50s
```

fig 2.0 above illustrates the code for loading the data in chunks

We combined the data from the reviews, business and user datasets and used only the 50000 of the data and deleted the rest before moving onto the modeling phase.

3.2 Cleaning and preprocess the data

As the Yelp dataset is relatively large ,noisy and spans over a variety of businesses ,it was difficult to perform analysis on the data at the beginning. As such we have gone to great lengths into cleaning the data to prepare it for our analysis. Tasks that we have done were such as

- extracting out only businesses that are restaurants or serve food
- dropped the data if the business id or user id is null
- added new feature or columns to the tables when it is needed for analysis

3.3 Class imbalance

As there is a class imbalance in the dataset , we used oversampling to counter this phenomenon. A method such as Synthetic Minority Over-sampling Technique ,or SMOTE, was used before we process the data with a natural language processing model.

3.4 Too many features

There are too many features in the dataset and some of them have little relevance to our target data as such feeding all these features into the machine learning models would reduce the accuracy of the models. We decided to implement feature selection to pick out the 10 most important features using an extra tree classifier.

3.5 Finding the optimal n_estimators and learning rate for Adaboost

While we used cross validation to find the learning rate for Adaboost, it was rather hard to get the right number for the n_estimator that can give a good accuracy for the test without falling into the trap of overfitting.

4.0 Introduction of evaluation metrics

Precision: Precision is the fraction of true positives (correctly predicted positive instances) among all positive predictions (both true positives and false positives). It measures the accuracy of positive predictions and is defined as:
Precision = true positives / (true positives + false positives)

Recall (Sensitivity) : Recall is the fraction of true positives (correctly predicted positive instances) among all actual positive instances in the dataset. It measures the completeness of positive predictions and is defined as:
Recall = true positives / (true positives + false negatives)

F1-score: F1-score is the harmonic mean of precision and recall, and it provides a single score that balances precision and recall. It is defined as:
F1-score = 2 * (precision * recall) / (precision + recall)

Support: Support is the number of instances in each class. It is used to calculate precision, recall, and F1-score.

Confusion matrix: A confusion matrix is a table that summarizes a binary classification model's performance. In a matrix format, it displays the number of true positives, true negatives, false positives, and false negatives. Precision, recall, and F1-score are calculated using the confusion matrix. For example, we would use TP and FP to determine accuracy, and TP and FN to calculate recall.

AUC: AUC is a metric used to assess a machine learning model's performance in a binary classification job. It is a measure of the area under this curve that shows the model's ability to discriminate between positive and negative cases. The AUC score runs from 0.0 to 1.0, with 0.5 being a random classifier and 1.0 representing a perfect classifier. A higher AUC shows that the model is more capable of differentiating between positive and negative samples.

5.0 Analysis on evaluation results

According to several measurements including accuracy, recall, and f1-score, we compare the performance of seven different models on the same dataset (DecisionTreeClassifier, ExtraTreesClassifier, KNeighborsClassifier, Naive Bayes, LogisticRegression, RandomForestClassifier, AdaBoostClassifier).

Based on the provided classification reports, the models with the highest accuracy for this task are:

- AdaBoostClassifier (accuracy: 0.73)
- RandomForestClassifier (accuracy: 0.69)
- LogisticRegression (accuracy: 0.69)
- Naive Bayes (accuracy: 0.691)
- ExtraTreesClassifier (accuracy: 0.68)
- KNN Classifier (accuracy: 0.60)

Due to its capacity to convert weak learners (basic models that perform just marginally better than random guessing) into a strong learner that can generalize well to new data, AdaBoost is typically regarded as a powerful machine learning technique. This ability was used by the AdaBoostClassifier to better capture the intricate connections between the input characteristics and the target variable. Furthermore, it's possible that the other models tried weren't as appropriate for the details of the data.

The AdaBoostClassifier has the highest accuracy among these models and in comparison to other models , it performs the best in terms of precision, recall, and f1-score. However ,this does not mean we think the figures are good. The model has an accuracy up to as high as 73% . We tested different n_estimator to try to improve the performance. However , as shown from the classification report below, the recall and the f1-score for “0” are low, this indicates that the model could be missing a large number of positive examples, meaning that it is not sensitive enough to the positive class.

```
Accuracy: 0.7293333333333333
AdaBoostClassifier   Classification Report

              precision    recall  f1-score   support

           0       0.63      0.31      0.41      4636
           1       0.75      0.92      0.82     10364

    accuracy                           0.73     15000
   macro avg       0.69      0.61      0.62     15000
weighted avg       0.71      0.73      0.70     15000
```

fig 3.0 above shows the classification report for AdaBoost.

We also looked at KNN Classifier more deeply with the AUC score after having generated the classification report and looked at its f1-score. The AUC score can give us a further indication to see if there is any overfitting on the model and also to inspect the performance of the model.

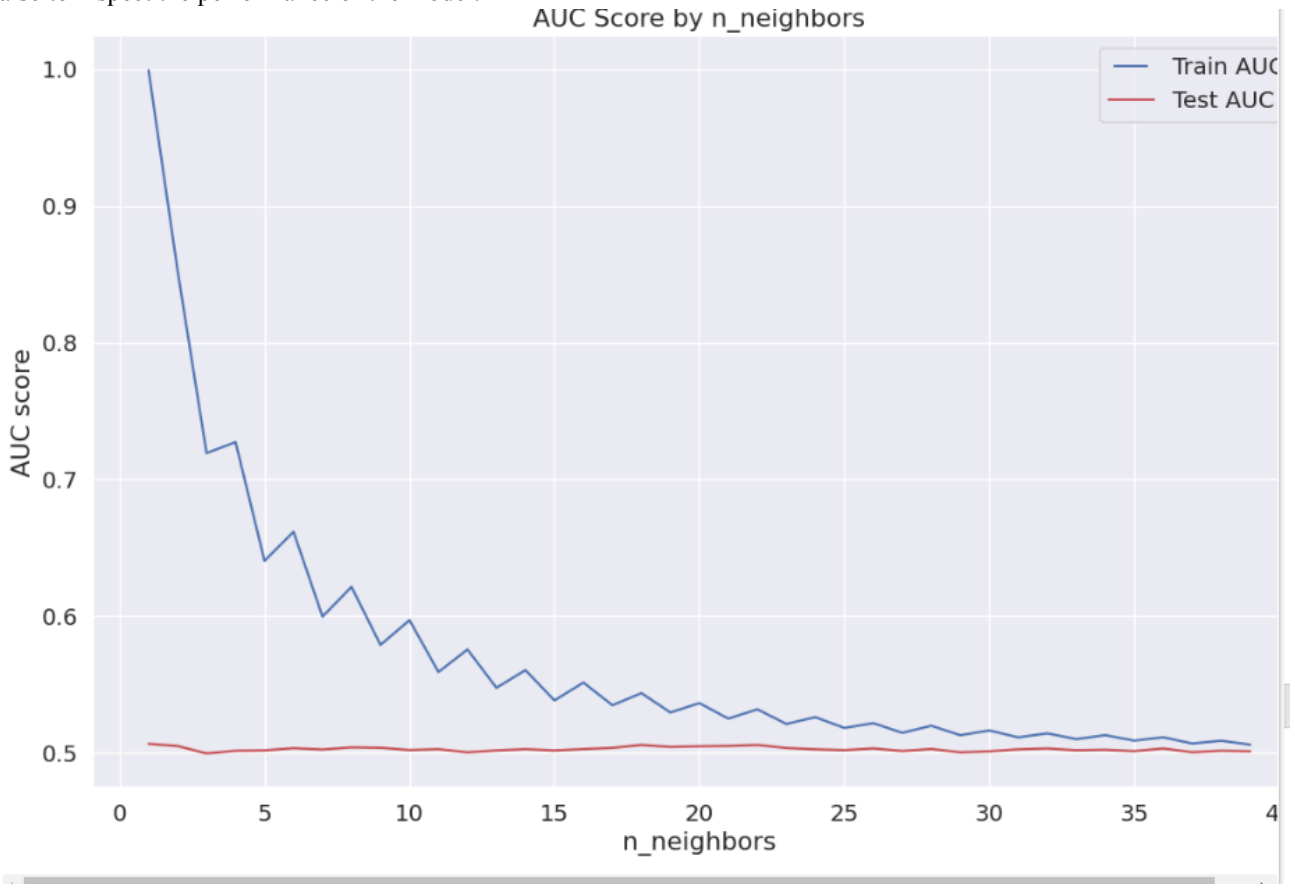

fig 4.0 above indicates the AUC score diagram for the train and test set of KNN

As shown on the diagram fig4.0 above the gap between the train and test data is huge at the beginning ,the first few n_neighbors , indicating that the model is overfitting at the small values of n_neighbors. The AUC score of the "test"AUC is around 0.5 and this means the model's performance is no better than random guessing. Overall the findings are not that great, so we decided to try a natural language processing model to predict the actual star rating.

In the second part of the project , we conducted natural language processing using the reviews text. We picked naive bayes model as it is suitable for predicting star rating of businesses because it can quickly learn to use high dimensional features with limited training data compared to more sophisticated methods[2].

The result was as follows:

Classification Report for Naive Bayes

	precision	recall	f1-score	support
1.0	0.65	0.74	0.69	27393
2.0	0.36	0.29	0.33	17838
3.0	0.42	0.38	0.40	23163
4.0	0.50	0.47	0.49	47999
5.0	0.79	0.83	0.81	98278
accuracy			0.64	214671
macro avg	0.54	0.54	0.54	214671
weighted avg	0.63	0.64	0.64	214671

Fig 5.0 shows the classification report for Naive Bayes -(Natural Language Processing Model)

It should be noted that the results (F1-score etc) for 2, 3 and 4 star ratings are sub-optimal, indicating that the model is not very good at forecasting these ratings. The accuracy is shown below.

```
Accuracy Score of Naive Bayes Model
0.6435103018106777
```

review_id	user_id	business_id	stars	useful	funny	cool	text	date	word_count	char_count	avg_word	Predicted
(R5xCJKJ_A	41270.0	57172.0	4.0	1	3	1	I was just introduced to this longstanding, lo...	2013-11-08 18:21:14	388	1955	4.284153	4.0

The fig 6.0 above shows the table for the star rating prediction output

As indicated in the table above, the predicted rating for this review is 4 which is identical to the actual review rating.

6.0 Conclusion

In conclusion, we found that AdaBoost Classifier is a better model in comparison to other models such as random forest or logistic regression when predicting if a user would give a star rating of over the average rating or not. But the model's performance fell short of our expectations.
We tried to predict the actual star rating with natural language processing model but the result is rather less satisfactory
Throughout the course of this project we learned that a large dataset would require more work to set up even before any preprocess or cleaning takes place. Furthermore, the idea that the more data the better may not always be true , the quality of the data and integrity of the data is also important.

The size of the data, disparity of the star rating distribution and the sparsity of the data have all added difficulties in this project. It is possible that there may be a chance that some of the ratings on yelp are fake and have skewed our result, as such further investigation and research is needed. A fake review detection could be carried out to filter the data before moving onto the next step.

Future research on this subject could take several different directions:
Future studies can also concentrate on better comprehending the elements that influence consumer reviews of companies. It may be possible to learn more about the elements of a company that customers value most by reading through user evaluations and looking for recurring themes or subjects. They would be used to enhance the performance of the classification models and provide guidance when choosing the features to include in them.
As alternate models for comparison, gradient boosting methods like XGBoost or LightGBM may be investigated. This leads to the conclusion that investigating deep learning models like recurrent neural networks or convolutional neural networks may bring additional performance benefits.

7.0 References

[1] Pang, B., Lee, L. and Vaithyanathan, S. (n.d.). Thumbs up? Sentiment Classification using Machine Learning Techniques. [online] Available at: https://www.cs.cornell.edu/home/llee/papers/sentiment.pdf.
[2] Text classification using Naive Bayes classifier. https://iq.opengenus.org/text-classification-naive-bayes/ Accessed 4/8/2023.
[3] Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04. [online] doi:https://doi.org/10.1145/1014052.1014073.
[4] Kalchbrenner, N., Grefenstette, E. and Blunsom, P. (n.d.). A Convolutional Neural Network for Modelling Sentences. [online] Available at: https://arxiv.org/pdf/1404.2188.pdf).