Predict Rating Scores from Review Comments on Boston Airbnb Data

Jingyi Li Jingxian Chen Zheng Jing Shichao Zhong

I. Introduction

A. Background

Airbnb is a well-known platform for hotel booking internationally. It builds a good connection between guests and hosts. This kind of comfortable, convenient, and cheap homestay has gradually become the first choice for tourists from all over the world. And Airbnb has gradually become the first platform for people to find a homestay.

As users, people will first look at the rating scores of homestays, which can reflect the quality of facilities and residential satisfaction of the homestay. In general, user reviews are strongly correlated with homestay ratings, with users giving a good review accompanied by a high score, and vice versa. Therefore, we considered whether the rating scores of the homestays could be inferred by analyzing customer review comments. Also, machine learning and text analysis are the hot topics in the field of data research in recent years, so it could be a good choice to predict rating scores from review comments by machine learning methods.

B. Goal

In this report, we are interested in predicting rating scores of the homestays from review comments in Boston, MA. And we will In order to simplify the prediction results, we focus on classifying the rating scores into two levels: high level if rating scores ≥ 99; low level if rating scores < 99.

Since most homestays have high scores, we would like to pay more attention to those homestays with higher scores which means they will have better services and facilities.

We will do text analysis first, and then reduce the word features by PCA method. Finally, we focus on different classification methods and compare the results of these by F1 score.

II. Data collection

We get our data from kaggle, there are three csv files on this website–listings, reviews, calendars. Our dataset is part of these three csv files and it's based on the following two Boston Airbnb datasets: Dataset 1–Listings, which includes full descriptions(pictures, introductions, neighbors and house rules) and average review scores. In these columns, we care about the average review scores and change it into categorical type. Dataset 2 – Reviews, which includes id for each reviewer and detailed comments. We only get two columns- id and comments. Then we merge these two datasets on listing_id. Finally, we could get a dataset with three columns, which are id, comments, and review score rating. (https://www.kaggle.com/airbnb/boston)

III. Data Pre-processing

- **Keep all comments per review id**
  Since each id is representing one hotel, each comment is representing one customer' s feedback, we will have a one to many mapping correlation between each id and comment. And here we decided to keep all the comments sentences to analyze in order to make the best use of our data.
- **Expand contractions**
  Since we need to build up features based on the words appearing in the comments, we need to expand contractions in the sentence to make sense of each individual word. For example, we need to expand 'doesn't' into 'does not', 'student's' into 'student'.
- **Convert all to lowercase**
  Also, whether a word is lowercase or uppercase doesn't affect anything, we need to standardize all the words into lowercase.
- **Remove punctuations**
  Also, we don't need punctuations to build up our features, so we need to remove all the punctuations from the text except for some meaningful emoji signs like ':)'. When a comment contains an emoji sign, it will probably be a good comment, which might be an important feature for classification. So, we decided to retain them.
- **Remove stop words**
  We know that in a complete sentence, not all the words contained are useful. So, we need to filter out the useless words like 'I', 'are', 'do', etc. After that, each comment would only contain useful words without punctuations.
- **Tokenization**
  We use the method word_tokenize() to split a sentence into words. The output of word tokenization can be

converted to Data Frame for better text understanding in machine learning applications. It can also be provided as input for further text cleaning steps such as punctuation removal, numeric character removal or stemming. Machine learning models need numeric data to be trained and make a prediction. Word tokenization becomes a crucial part of the text (string) to numeric data conversion.

- **Lemmatization**
  Lemmatization is the algorithmic process of finding the lemma of a word depending on their meaning. Lemmatization usually refers to the morphological analysis of words, which aims to remove inflectional endings. It helps in returning the base or dictionary form of a word, which is known as the lemma. The NLTK Lemmatization method is based on WorldNet's built-in morph function. Text preprocessing includes both stemming as well as lemmatization. Many people find the two terms confusing. Some treat these as the same, but there is a difference between these both. Lemmatization is preferred over the former because of the below reason.

Example:

BEFORE CLEANING:

```
Terry's Hotel Alterntv in Boston was a
perfect place to stay for myself and my
partner. We mixed our trip with business
and pleasure and found the room perfectly
appointed for our needs and affordable. A
great stay!e
```

AFTER CLEANING:

```
"terry" "hotel" "alterntv" "boston"
"perfect" "place" "stay" "partner" "mix"
"trip" "business" "pleasure" "find" "room"
"perfectly" "appoint" "need" "affordable"
"great"
```

IV. DATA VISUALIZATION

A. *Barplot*

We have more than 2000 word features. Since there is not enough space to display the bar plots of all the word features, we only include 5 features here for illustration, and they are 'abnormal', 'abound', 'abruptly', 'absence', 'absurd' shown in Figure1. At first glance, the rating scores are not separated well by the features. Relatively speaking, 'absence' and 'abruptly' seems more useful than other words. The reviews that contain words 'absence' or 'abruptly' have lower rating scores in general. We also observe that the word 'absurd' only appears once in all reviews, so it should be removed.

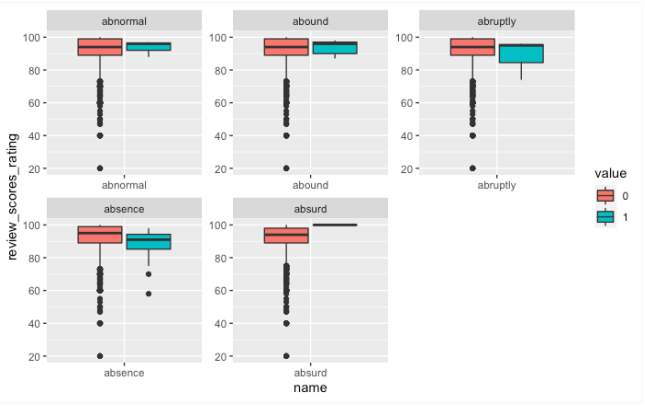

Fig. 1. Barplot of word features

B. *Heatmap*

We created a heatmap to see the correlation between word features. Since there are more than 2000 word features, we only showed a few of them. Although most correlations are zero, there does exist some positive ones. For example, the words 'delightful' and 'smooth' are slightly positively correlated. Words 'Complain' and 'odor' are also slightly positively correlated, which makes sense because 'odor' usually implies bad smell which can further cause complaint. There also exists a slight positive correlation between 'delightful' and 'ease', which makes sense because they are both positive. However, generally speaking, the correlations are so low that we can ignore them and assume all the words are independent of each other. Therefore, we may expect the Naive Bayes model to outperform others because of its assumption on feature independence.

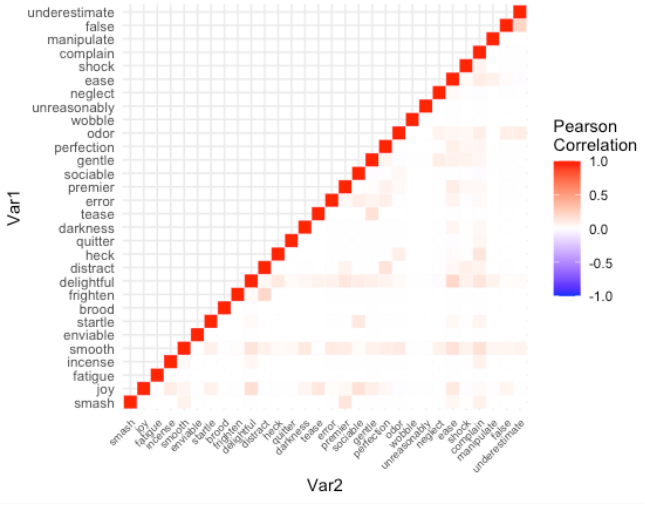

Fig. 2. Heatmap for correlation

C. *Word Count*

We also count the occurrences of each word in reviews. We displayed them in Word Cloud, in which larger words imply higher frequency and vice versa. We also displayed the top ten words in a table. For example, the top three words are ‘stay’, ‘great’, ‘good’, which appear over 1000 times in all reviews. This implies that most reviews are positive, which explains why most reviews have very high rating scores. In contrast, the words that appear at low frequency have a small size. The ones that only appear once or twice cannot be found in the word cloud.

Fig. 3. Word cloud plot

Word count in reviews

word	num_words
stay	1553
great	1198
good	1078
place	1055
host	1053
boston	954
clean	954
apartment	906
location	862
nice	697
room	643

Fig. 4. Word count table of top 10

D. *Sentiment analysis - binary*

We found a R package that contains sentiment words of two classes ‘positive’ and ‘negative’. We only kept the sentiment words in our design matrix. As shown in Figure 7, we have about 400 positive words and 300 negative words. A few word examples along with their classes are displayed .

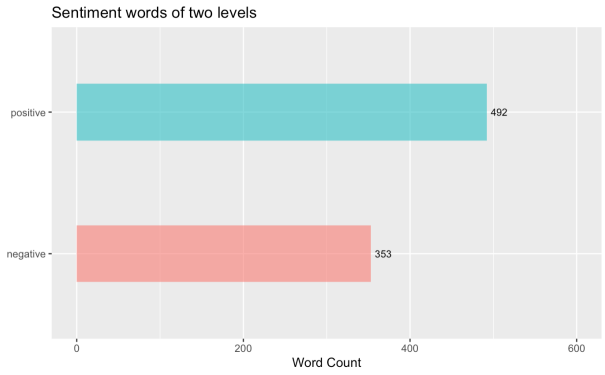

Fig. 5. Count of sentiment words

word	sentiment
clean	positive
good	positive
work	positive
welcome	positive
ignore	negative
issue	negative
easy	positive

Fig. 6. Examples of sentiment words

V. METHOD

One listing id may have multiple review comments. Using only one or a subset of the comments may not be representative enough and will lead to biased results. Therefore, we used all the comments of each listing id to predict its rating score.

A. *Dimensionality reduction*

We only used the sentiment words as our predictors, and created three candidate design matrices for classification, which are raw word count matrix, word importance matrix with TF-IDF vlaues, and one-hot encoding matrix. We applied PCA on all three matrices to reduce the dimensionality by capturing 90% of the variance. Before doing PCA, we split

our data into 75% training and 25% testing, and then applied PCA on the training set only, to prevent the model seeing any information from test data. We then used the same loading matrix to transform test data and used it for model evaluation.

B. *Clustering*

Besides PCA, we also tried nonlinear dimensionality reduction techniques such as mds. However, the 2d mds plot does not show any well-separated clusters. So we tried three clustering methods, Kmeans, hierarchical-clustering and DBSCAN, and hoped to see a clear separation of two groups by rating score levels. Unfortunately, none of the results make much sense.

Fig. 7. K-means clustering results with K=2

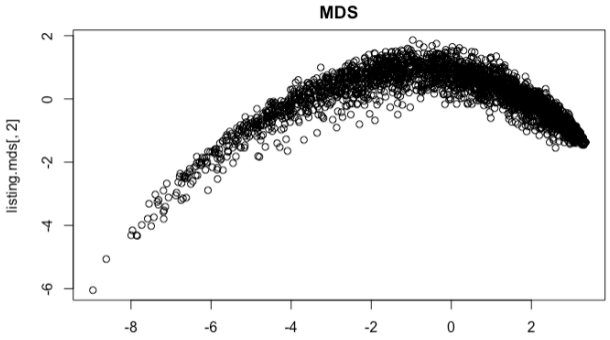

Fig. 8. MDS

C. *Classification*

We ran classification models on three different matrices respectively as well as their PC versions. We found that running models on the PC version of the averaged count matrix provides the best result, in terms of the test error, precision, recall and f1 score. More detailed model performance and comparison is provided in the next section. In short, we tried 8 machine learning models, which are naive bayes, LDA, QDA, logistic regression, support vector machine, random forest, xgboost and neural network (mlp). We evaluated and compared the models in terms of the test error, precision, recall and f1 score. We used the dummy classifier as our baseline model, which has 75% accuracy due to the unbalanced data.

VI. Results

A. *LDA, QDA, NB*

Naive bayes is known for its good performance on high dimensional text data, with an assumption that all features are independent from each other. As is shown by the heatmap previously, there exists little correlation among our word features, so we expect NB to outperform LDA and QDA. However, it turns out that QDA has the best performance in terms of test error and LDA has the best performance in terms of F1 score.

B. *Logistic regression*

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable. Here we use “high” and “low” levels as the response, which is encoded as “1” and “0”. The logistic regression model itself simply models probability of output in terms of input and does not perform statistical classification (it is not a classifier), though it can be used to make a classifier. So for our problem, we define the output with the probability larger than 0.5 as “low”, and those with the probability lower than 0.5 as “high”.

C. *Support vector machine(SVM)*

SVM is a supervised learning model and related learning algorithm for analyzing data in classification and regression analysis. Given a set of training data, each training data is marked as belonging to one or the other of the two categories, and the SVM training algorithm establishes a model that assigns new data to one of the two categories, making it a binary linear classifier. By using kernel function, SVM can be changed into a nonlinear classifier, which is suitable for our data set better.

In our problem, we use method tune() to adjust the kernel function type and its parameters. By 5-fold cross-validation, we choose the best parameters with the smallest cv-error. It turns out that the best model has the 3rd degree polynomial kernel with other parameters: cost=1, gamma=0.5. The result shows that this model is over-fitting since the training error is almost zero but test error is pretty high.

D. *Neural Network*

A multilayer perceptron (MLP) is a class of feedforward artificial neural networks (ANN). The term MLP is used ambiguously, sometimes loosely to refer to any feedforward ANN, sometimes strictly to refer to networks composed of

multiple layers of perceptrons (with threshold activation); An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

For our problem, we have chosen to use an MLP neural network with two hidden layers both using "relu" as activation function. Also we used early stopping and dropout techniques to avoid overfitting. Finally, we ended up with 0.86 F1 score value on testset data.

E. Random Forest(RF)

Random Forest is widely used in binary classification problems because of its good performance. We ran RF models in R using the package called "randomforest". After tuning the hyperparameters, we found that the best rf model has 1000 decision trees and 100 as the maximum number of features of each sample. We also performed bootstrap to resample a subset of features for each decision tree. After tuning these parameters, we achieved a testing error of 0.2 and a F1 score of 0.875. Besides, we also include a feature importance plot below.

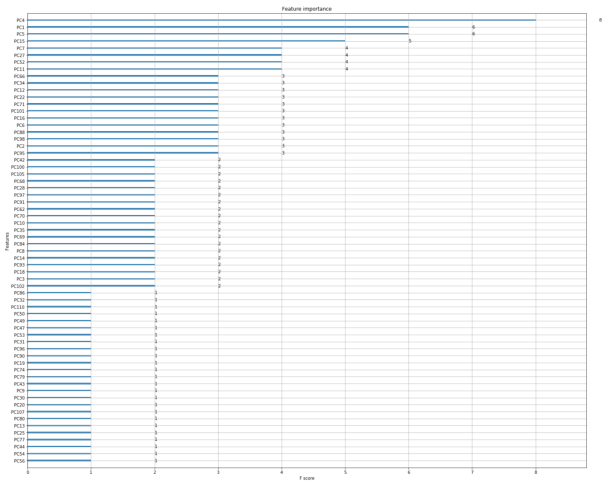

Fig. 9. feature importance

F. XGBoost

Xgboost is another popular machine learning algorithm in binary classification. Xgboost can provide a much better result than classical methods. Thus, we choose it as one of models that help us solve the problem. After parameter tuning, we choose 10 as estimator tree, 0.5 as the learning rate and 3 as the max depth. As a result, we achieved a testing error of 0.192 and a F1 score of 0.870.

Overall, there is a table summarizing the outputs of the methods mentioned above. We can see the training error, test error, precision, recall, and F1 score of these different methods and compare with them easily.

Model	Train error	Test error	Precision	Recall	F1 score
Naive Bayes	0.223	0.228	0.843	0.853	0.848
LDA	0.190	0.218	0.796	0.951	0.867
QDA	0.166	0.212	0.853	0.864	0.859
Logistic Regression	0.189	0.222	0.788	0.961	0.866
Neural Network (MLP)	0.198	0.252	0.749	0.998	0.860
SVM	0.002	0.282	0.795	0.840	0.817
RF	0.003	0.192	0.848	0.894	0.870
XGBoost	0.115	0.203	0.827	0.956	0.875

Fig. 10. result conclusion

VII. Conclusion

In terms of both the test error and F1 score, Random Forest and XGBoost are the best models at predicting review scores from review comments. Therefore, we conclude that tree models have better performance than other models on this specific text data. Moreover, all of our models except neural network and svm, outperformed the dummy classifier and achieved a test accuracy close to or higher than 80%. Thus, we could use our model to choose the house we want to live by looking through the reviews of this house when we are traveling.

VIII. Future work

- We could try some other deep learning models, such as LSTM model, which has been used a lot in NLP field.
- We could use other descriptions of airbnb location to predict the price of airbnb, such as the number of bedroom.
- We could consider some other interesting features like the locations of each hotel and try applying some clustering methods to get meaningful results.

References

[1] Lyric Analysis with NLP & Machine Learning with R. https://www.datacamp.com/community/tutorials/R-nlp-machine-learning
[2] Data sources. https://www.kaggle.com/airbnb/boston
[3] R Interface to 'Keras'.https://keras.rstudio.com/index.html
[4] Stats503 lab