Fine Food? The Text Knows!

Zhongjian Zhu
A53210727
zhz362@eng.ucsd.edu

Jinhan Zhang
A53211930
jiz530@eng.ucsd.edu

Siqi Qin
A53203046
s7qin@eng.ucsd.edu

Abstract

In this article, we aim to predict rating of food based on data from amazon.com. Empirically, people use different word to describe high rating food and poor rating food. As a result, we can use people's review text as features to predict their ratings about food. The features we use include adjective word, bigram and trigram. Several models are applied to the predictive task, including Linear Regression, Ridge Regression, Logistic Regression, Gradient Boosting and Random Forest. We use training and validation data to optimize each model and evaluate their performance on the test data set in terms of the mean-squared error (MSE). The result shows that the Random Forest leads to the minimal MSE.

Keywords: rating prediction, text, Linear Regression, Ridge Regression, Logistic Regression, Gradient Boosting, Random Forest

1 INTRODUCTION

Dining counts most for people, of which taste comes first. With the development of information technology, an increasing number of people purchase delicious food via online shopping platforms for convenience. But one problem with these platforms is that we cannot foretaste and directly perceive through the senses. Thus, the experiences from previous users become an important reference for us. Amazon, one of the most popular online shopping market in the world, provided a huge number of user reviews on all kinds of products. With these reviews, especially the review text and star rating, we can analysis the user's sentiment towards a certain product and help us to make our own decision.

In this project, we use the dataset downloaded from Amazon Fine Food to predict the star rating based on the review information given by the users. We first do an exploratory analysis of the data and get some inspiration on the feature selection. Then, predictive task is determined and a baseline is set as a reference. We try five different models and evaluate their performance on the dataset based on MSE. Finally, we find some recent literatures related to our project and put forward some future work. Based on the model given in this project, we can realize better prediction of the rating.

2 DATASET

2.1 Description

We use the Amazon Fine Foods Reviews dataset provided by Julian McAuley on the Stanford snap datasets[1]. This dataset consists of reviews of fine foods from Amazon within 1999 to 2012. There are 568462 reviews, in which 256059 users give their ratings towards 74258 products. And 568454 of the reviews have review text. In the review, product and user information, ratings and a plaintext review are given, which can be used for our prediction. We describe the features of each review in Table.1.

Table 1. Feature Description

Feature	Description
productID	asin give for each product
userId	ID of the user
profileName	name of the user
helpfulness	fraction of users who found the review useful
score	rating of the product
time	time of the review (unix time)
summary	review summary
text	text of the review

To select the features that best describe the prediction task we care about, we are interested in examining the structure of these dataset. So, we perform some exploratory analysis to better understanding these reviews. The analysis is based on the entire dataset.

2.2 Exploratory analysis

2.2.1 Rating Distribution

Rating is what we want to predict in this project, so we first take insight into the properties of the rating of the dataset. The rating distribution is shown as Fig. 1.

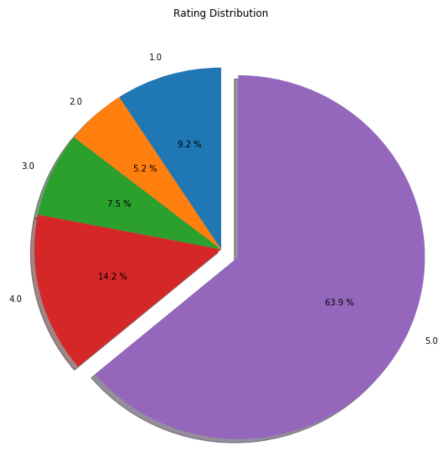


Fig. 1. Rating Prediction of the Dataset

As shown in Fig. 1, the 5-star reviews account for 63.9 % of all reviews. It seems that people are apt to give positive ratings nowadays.

2.2.2 *Length of review*

Longer review texts always provide more information compared to the shorter ones. So, we next analyze the relationship between the length of the review and the rating score. We count the length of review texts in rating score 1 to 5. Results are shown in Fig. 2.

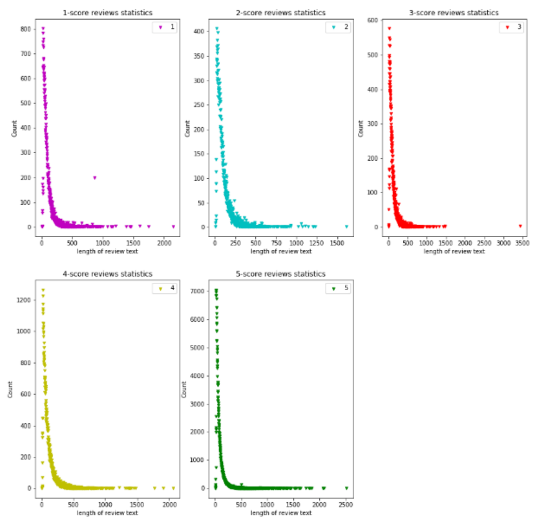


Fig. 2. Length of Review in Different Scores

In different scores, the length distribution all have the same trend. But we may find that the review text in 5-star rating is always shorter compared with the others.

2.2.3 *N-grams*

In the feature description part, we find that the most informative feature is the review text. So here we analyze the reviews in a text mining viewpoint. We count the most popular unigram, bigram and trigram in 1-star rating, 3-star rating and 5-star rating. The results are shown in Fig. 3, Fig. 4 and Fig. 5.

From the most popular N-gram, we found that the bigram and the trigram are very representative for the score level they stand for. But the unigram is not varying very much among score 1, 3 and 5.

Considering that unigram is the most frequently used feature in text mining, we try to classify the unigrams based on the word class.

2.2.4 *Word Classification*

A word can play different a role in a sentence. With natural language processing method, we can classify the words in the text into coordinating conjunction (CC), cardinal number (CD), determiner (DT), preposition (IN), adjective (JJ), noun (NN), pronoun (PR), adverb (RB) and verb (VB). This classification is based on the NLTK library in python. The word class distribution of the dataset is shown as Fig. 6.

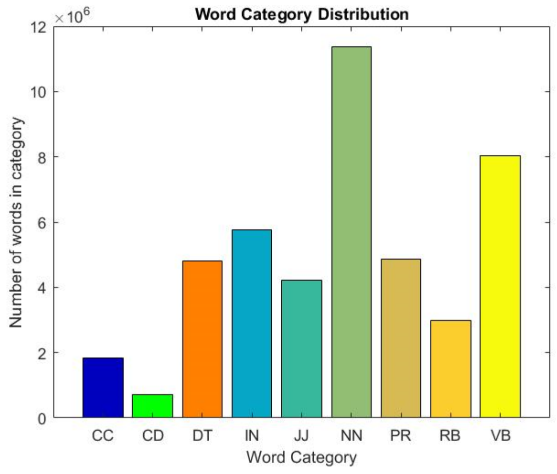
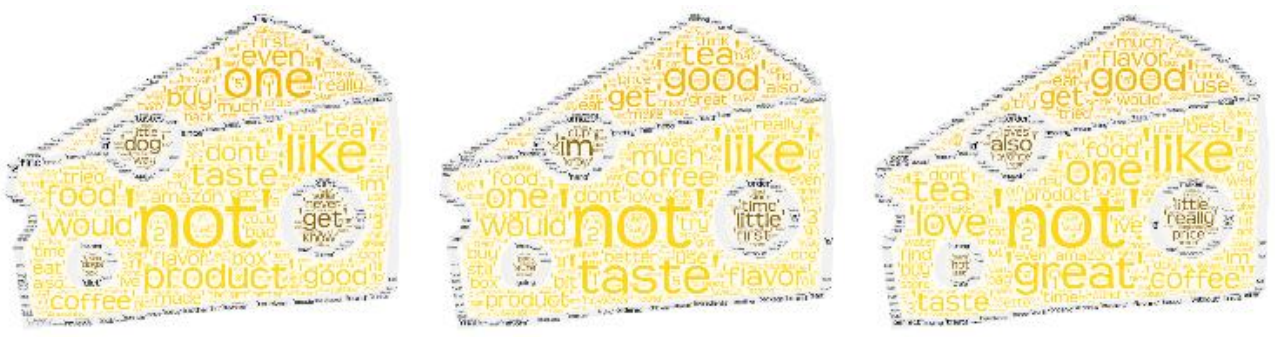


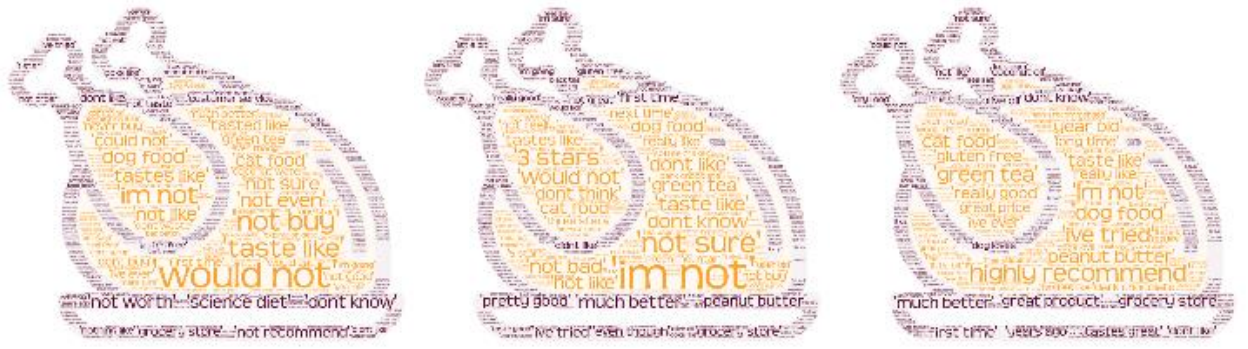
Fig. 6. Word Category Distribution

We can find here that some categories contain little information about a user's attitude, such as the preposition, are in huge amount. Maybe this is the reason that unigram can't represent a certain star level well. So, in our project, we decide to try the adjective, noun and verb as feature, which we think contains more information. We finally count the most popular adjective, noun and verb in our dataset to help us better select the words we can use as features. The results are shown in Fig. 7.



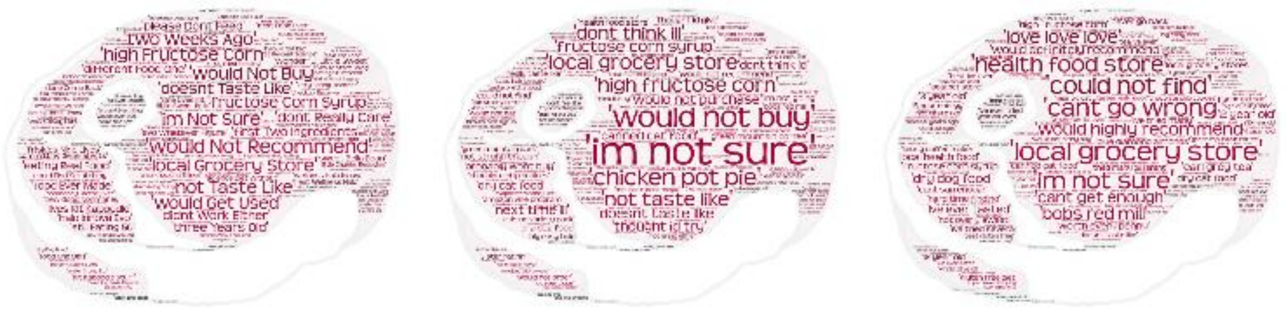
(a) One-star rating (b) Three-star rating (c) Five-star rating

Fig. 3. Most Popular Unigram in Rating 1, 3, 5



(a) One-star rating (b) Three-star rating (c) Five-star rating

Fig. 4. Most Popular Bigram in Rating 1, 3, 5



(a) One-star rating (b) Three-star rating (c) Five-star rating

Fig. 5. Most Popular Trigram in Rating 1, 3, 5

3 PRIDICTIVE TASKS

Naturally, the rating customers give to merchandise would be a reasonable reflection of their evaluation upon that. And since we already have a clear overview about the properties of the Amazon fine food dataset, we can now conduct a predictive task to predict the rating based on other information for every review record. These predictions can be used to help recommend potential attracting products to the customer's taste.

3.1 Data Pre-processing

As introduced in section 2.1, the Amazon fine food database has 568462 reviews as total, and after we eliminate those without review text we end up with 568454 reviews left. And we find that many of these review texts contain HTML formatting tags such as $<$ br/$>$ and they will dominate in our word frequency statistics but they contain no information, so we have to remove these HTML formatting tags by BeautifulSoup Python library firstly.

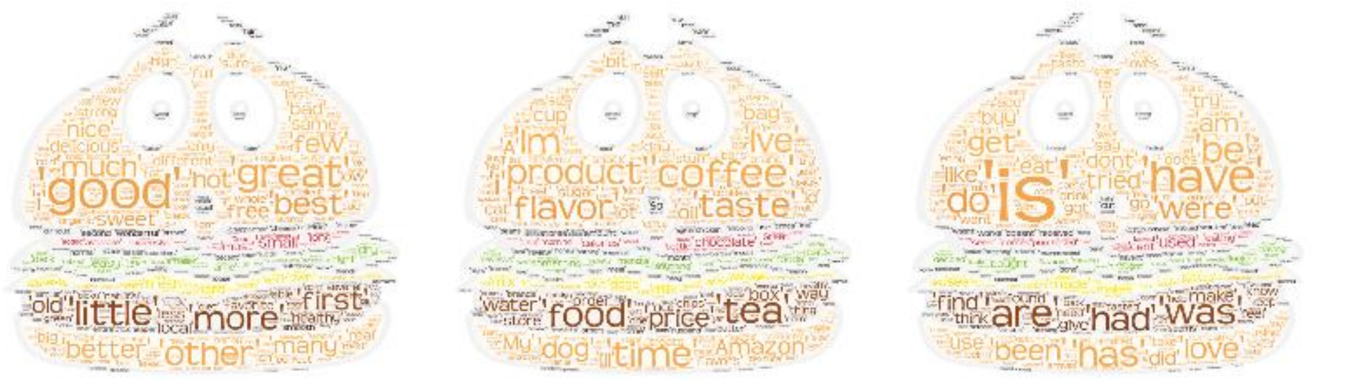
Since the uppercase and lowercase of the same word usually don't contain much different information, we convert the review text to all-lowercase form for each piece of review. And also we remove the punctuations due to that they contain little information. Finally, we shuffle the dataset, split the dataset and use the 1-100000 reviews to be our training set, 100001-200000 reviews to be validation set and 200001-300000 reviews to be the test set.

3.2 Model Evaluation

In order to evaluate how well our models perform in the predictive task, we will use Mean Squared Error (MSE). Compared to Mean Absolute Error(MAE), MSE has the property of penalizing large magnitude errors much more greatly, thus we can end up with a model whose errors are distributed more evenly across the whole training set. The MSE is defined as below:

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

where y_i is the true (or label) value and $\hat{y}_i$ is the value predicted by our model.



(a) adjective (b) noun (c) verb

Fig. 7. Most Popular Adjective, Noun and Verb

3.3 Baseline

In order to better evaluate our models, we set up a baseline which makes prediction intuitively by using only the number of words in each review text and bias term as the feature and use linear regression with no regularization to get the parameters. This is one of the most straightforward machine learning method, but has a reasonable overall performance to act as a baseline. Then we will use mathematical equation to represent the baseline, and it is defined as below.

$$\text{score} \simeq \theta_0 + \theta_1 \times (the\ number\ of\ words\ in\ review)$$

Now we can apply our baseline model on the data we obtain as section 3.1. On the test dataset, the MSE achieves to be 1.7103 using the baseline model. And then we can assess the validity of the prediction of our models by calculating how much the MSE of our model exceeds that of the baseline on test set.

3.4 Feature Selection

In this section, we analyze the dataset to choose features, use these features to train models and apply the models on validation set and pick features by their performance concerning MSE.

Firstly, we count the top 500 frequent adjectives used in 1-5 score reviews respectively, thus we get 2500 adjectives in total. We use the $word\_tokenize$ in $nltk$ Python library to distinguish the part of speech of word. But there are many duplicates in them, after eliminating the duplicates we end up with about 760 adjectives. Then we use the number of each adjective as features and use these 760 features plus a bias term to train a linear regression model. We use this model to predict the rating on validation set. Then we do the same process for top 500 frequent nouns and verbs, and their number of features and corresponding MSE results on validation set are listed as Table 2.

Table 2. Comparison of MSE Using Different Kinds of Words

Category	The number of features	MSE
Adjective	761	1.2248
Verb	748	1.2547
Noun	738	1.3174

Since the number of features are all around 750, we can see from Table 2 that adjectives are most informative, thus we will use adjectives as part of our features in the following explorations.

Then we try to test if it is useful to add bigrams and trigrams as our features. Similarly, we count the top 100 frequent bigrams in review text used in 1-5 score reviews respectively, and remove the duplicates and use them as extra features plus adjectives to train models and run tests on validation set. And then we do the same thing for trigrams. And also we test $adjectives + bigrams + trigrams$ model. The results are shown below as Table 3.

Table 3. Comparison of MSE Using Bigrams and Trigrams

Category	Features	MSE
Adj.	761	1.2248
Adj.&Bi.	988	1.2239
Adj.&Tri	1097	1.2174
Adj.&Bi.&Tri	1324	1.2084

From Table 3, we can see that bigram and trigram features do have some extra information, so we will use them as well as the high frequent adjectives as the features.

And choosing these features is also accord with common sense. The adjectives are usually used to describe something instead of narrating a plain fact, which has more things to do with subjective factors. Thus they are likely to be more informative when predicting one's evaluation upon a product. Introducing bigrams and trigrams will decrease the loss of information by splitting phrases (e.g. "not bad" into "not" and "bad", "would not recommend" into "would", "not" and "recommend").

4 MODEL DESCRIPTION

The model we use can be divided into two categories, regression and classification. For regression, we treat the output (ratings) as a continues variable. For classification, since there are only five possible rating values, one to five, we treat this as a multi-classification problem. For each model, we use the most informative features selected in 3.4. So the length of θ is 1324 plus a bias term. For every review text, we find all adjectives, bigrams and trigrams. Their contributions are the times they appear in the training set. Then we train different models to get the optimal θ and apply the models on the test set to see the result. Finally, we choose the one with the best performance on the test set.

4.1 Linear Regression

Linear regression is one of the simplest supervised learning approaches to learn relationships between input variables and output variables. We assume the output variables which we want to predict is a linear combination of the input features. The relationship can be described as the following equation:

$$y = X\theta$$

where X is the matrix of features derived from data, y is the vector of output and θ is the unknowns. θ represents the relationship between input and output. Using training data, we can solve for θ which minimize the MSE.

In python, we can use the function, $numpy.linalg.lstsq$, to optimize the model. Linear regression is very simple and it usually works well. However, this model might not generalize. This is called overfitting. To avoid this problem, we can use the following model.

4.2 Ridge Regression

To avoid this overfitting, we can add a regularization term to the MSE equation as the process of penalizing model complexity during training. So we can get the optimal θ by solving the following equation:

$$\theta_0 = arg\min_{\theta} \frac{1}{N} \|y - X\theta\|_2^2 + \lambda\|\theta\|_2^2$$

where λ is the weight we penalize the model. So there is a trade-off between accuracy and complexity.

The optimal θ can be solved using gradient descent. The algorithm is as follow.

ALGORITHM 1: Gradient descent

```
Choose an initial vector of parameters $\theta$ and learning rate $\alpha$
while not converged, do
        $\theta \leftarrow \theta - \alpha f'(\theta)$
end
```

where $f(\theta) = \frac{1}{N}\|y - X\theta\|_2^2 + \lambda\|\theta\|_2^2$. To get the optimal λ we can use the validation set. For every λ, we get the optimal θ through the training set and see their MSE on the validation set. By doing this, we find that when $\lambda = 7$ the MSE is minimal on the validation set, so its corresponding θ is the optimal model.

4.3 Logistic Regression

Logistic regression is a regression model where the dependent variable is categorical.

$$y_i = \begin{cases} 1, & if\ X_i\theta > 0 \\ 0, & otherwise \end{cases}$$

To convert the real value expression, $X_i\theta$, into a probability, we can use the sinusoid function.

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

In the training step, $X_i\theta$ should be maximized when y_i is positive and minimized when y_i is negative.

In this cases, we have five classes. The most common way to generalize binary classification to multiclass classification is to train a binary predictor for each class.

In python, we use *sklearn.linear_model.LogisticRegression* function to optimize the model. By changing the parameter C in the function, we can avoid overfitting. By testing, we choose $C = 5.6$.

This model does not perform well based on the MSE criterion because the difference between the true ratings and the predicted ratings are always integers.

4.4 Gradient Boosting

The gradient boosting combines weak "learners" into a single strong learner, in an iterative fashion. The goal is to "teach" a model F to predict values in the form $\hat{y} = F(x)$, by minimizing the MSE.

At each stage $1 \le m \le M$ of gradient boosting, it may be assumed that there is some imperfect model F_m (at the outset, a very weak model that just predicts the mean y in the training set could be used). The gradient boosting algorithm does not change F_m in any way; instead, it improves on it by constructing a new model that adds an estimator h to provide a better model $F_{m+1}(x) = F_m(x) + h(x)$. The question is now, how to find h?

The gradient boosting solution starts with the observation that a perfect h would imply $F_{m+1}(x) = F_m(x) + h(x) = y$. Therefore, gradient boosting will fit h to the residual $y - F_m(x)$. Like in other boosting variants, each $F_{m+1}(x)$ learns to correct its predecessor $F_m(x)$. A generalization of this idea to other loss functions than squared error (and to classification and ranking problems) follows from the observation that residuals $y - F_m(x)$ are the negative gradients of the squared error loss function $\frac{1}{2}(y - F_m(x))^2$. So, gradient boosting is a gradient descent algorithm; and generalizing it entails "plugging in" a different loss and its gradient.

$sklearn.ensemble.GradientBoostingRegressor$ function in python can directly give us the desired model. The parameter $n\_estimators$ is the number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting so a large number usually results in better performance. Here we choose 100.

4.5 Random Forest

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, x_2, \dots, x_n$ with responses $Y = y_1, y_2, \dots, y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

ALGORITHM 2: Random Forest

```
for $b$ in $1,2,\dots,B$ do
        Sample, with replacement, $B$ training examples from $X$, $Y$, call these $X_b$, $Y_b$ convert neuron_orientation to vector
        Train a decision or regression tree $f_b$ on $X_b$ $Y_b$
end
```

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

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

or by taking the majority vote in the case of decision trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the out-of-bag error: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample. The training and test error tend to level off after some number of trees have been fit.

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated. An analysis of how bagging and random subspace projection contribute to accuracy gains under different conditions is given by [3].

Typically, for a classification problem with p features, $\sqrt{p}$ (rounded down) features are used in each split. For regression problems the inventors recommend $p/3$ (rounded down) with a minimum node size of 5 as the default[4].

We use $sklearn.ensemble.RandomForestRegressor$ in python, to train the model. Similar to the previous model, the $n\_estimators$ still influences the performance. Here we choose $n\_estimators = 100$. Through comparison, the Random Forest model leads to the best performance. So we finally choose this model.

5 LITERATURE

Rating prediction is widely studied worldwide. The dataset we use is given by Julian McAuley in [1]. Julian uses this dataset to test the model he builds for evaluation prediction through online reviews. Traditional latent factor model is firstly tried and then temporal information is added into the model. The entire dataset is used and evaluation is based on MSE. Julian's result is shown in Table. 4 and Table. 5.

Table 4. Julian, results on user's most recent reviews

model	MSE
Latent-factor	1.582
Community at uniform rate	1.527
User at uniform rate	1.548
Community at learned rate	1.529
User at learned rate	1.475

Table 5. Julian, results on randomly sampled reviews

model	MSE
Latent-factor	1.425
Community at uniform rate	1.382
User at uniform rate	1.383
Community at learned rate	1.374
User at learned rate	1.189

Julian uses the latent factor model in his article, while, almost nothing about features is considered. Latent factor model works well with huge data and will give satisfactory result if the user or item exists in the training data. While in our method, we use a more general method, which considering the features in the text. This will perform equally on existed or new customers.

Recently, Rob Castellano, a researcher in New York University Data Science Academy, also performs an analysis on this dataset. Rob found that positive reviews are very common, which is the same with our result. He also found that positive reviews are shorter and longer reviews are more helpful. Also, he points out that frequent reviewers are more discerning in their ratings, write longer reviews and write more helpful reviews[5]. His analysis gives us some intuition on feature selection.

This dataset is also widely used in college projects. Fang & Lin[6] use (Gated) recurrent neural network to classify the sentiment of food reviews and get best accuracy of 93.35 % on training set and 68.75 % on test set. Wu & Ji[7] propose recursive neural network for multiple sentences to analyze sentiment in the dataset and get best precision of 0.75 and recall of 0.75. But they use different evaluation method with us, so that the result can't be compared directly. While, they still give us some intuition to make a regression task into a classification one. David[8] use Latent Dirichlet Analysis to extract topics from this dataset, which helps us to put forward the idea of word categorization. Chu etc.[9] select unix time, length of text and length of summary as features and get MSE of 1.694953. Our result is much better than theirs.

Many new methods are put forward for rating prediction. Eissa etc.[10] propose an extended text classifier based on Vector Space Model and sentiment analysis. Zhang & Wang[11] give an integrating topic and latent factors model, which combines topic and latent factors in a linear way to make them complement each other for better accuracies in rating prediction. Lei etc.[12] describe a framework to indicate how and how much user's preferences change and influence each other with trust propagation over the network. These methods can be used in our future works.

6 RESULTS AND DISCUSSION

Due to the imbalanced rating distribution (e.g. more than 50% 5-score review) we have observed in section 2.2.1, the model would be inclined to focusing on 5-score review if we just use the top frequent words as features regardless of rating. Thus we choose to combine the top frequent words/phrases in 1-5 score review text to be our features. In section 3.4, we have compared different types of features involving review text, so we finally choose 1324 features (761 adjectives, 227 bigrams and 336 trigrams) plus a bias term to be used in our models. Then we use the models in section 4 to run test on test set.

6.1 Results of Linear/Ridge Regression

The results of linear/ridge regression models are shown as Table 6 below.

Table 6. The results of linear/ridge regression models

Model	Baseline	Linear Regression	Ridge Regression (λ=7)
MSE	1.7103	1.2185	1.2182

We can see from Table 6 that the result of using our text features outperforms that of baseline a lot. That is mainly due to that the length of review text is not as informative as the content in it as we have analyzed in section 2.2.2. And the ridge regression with regularizer λ=7 has almost the same performance with that of no regularizer.

6.2 Results of Logistic Regression

Since score has only the value of 1, 2, 3, 4 and 5, so we can try to treat this problem as a multi-class classification problem and solve it with classification method. So we try to use logistic regression to build our model, and the results are shown as Table 7.

Table 7. The results of logistic regression models

Model	Baseline	Logistic Regression (C=5.6)
MSE	1.7103	1.6984

The logistic regression model seems not to have a much better performance than the baseline. This is probably because we treat this problem as classification problem and that tends to increase the MSE.

6.3 Results of Gradient Boosting

In this section, we try to use gradient boosting regressor to build our models. Since gradient boosting is fairly robust to resist overfitting so a large number of boosting stages will usually give the model better performance. Thus we choose the number of boosting stages to be 100. And the maximum depth of the individual regression estimators is chosen to be 10. Other parameters are used by their default value. The results are shown as Table 8.

Table 8. The results of gradient boosting regressor models

Model	Baseline	Gradient Boosting
MSE	1.7103	1.1377

We can see that the MSE has exceeds that of the baseline by 33% and outperforms the linear regression model by 6.7%. Thus gradient boosting has a fairly good result.

6.4 Random Forest

Then we use random forest regressor to build our model and do test on test set. Similarly, we use the 1324 features chosen before, and run test with the number of trees in the forest (i.e. n_estimators) to be 100. The results are shown in Table 9 as below.

Table 9. The results of random forest models

Model	Baseline	Random Forest Regressor
MSE	1.7103	1.0585

The MSE of model built by random forest regressor is much lower than that of the model built by linear regression using the same features. This model also improves the baseline's MSE by around 38.1%, which is a fairly satisfying model.

6.5 Summary

In this part, we have tested different models on test data and also compared the results. Our random forest model has the best MSE performance, its performance exceeds that of the baseline essentially and also be greater than linear regression model by about 13% concerning MSE. This is mainly due to 2 reasons. Firstly, we introduce the numbers of top frequent adjectives as our features, and adjectives are naturally more informative since they are used to convey subjective evaluation in many cases. And also we introduce the bigram and trigram features to compensate the loss of information for splitting phrase into words. But other features are not as informative such as nouns, verbs and the length of reviews since they are less dependent on the rating of a review. Secondly, random forest has a good capability to resist overfitting due to its randomness and also has a good performance on imbalanced data due to its insensitivity to multicollinearity. This model has provided a method to do text mining and applies it on predicting people's rating.

For linear/ridge regression models, they have a relatively good performance although they are essentially simple models. This is mainly because they have used good features, but they cannot be outstanding due to the limit of the model's principle. The logistic regression's performance is very close to baseline because it's a classification method and we are using MSE to be the evaluation standard.

7 FUTURE WORK

Given more time, we can work on further improvement to our models in the following aspects:

- **Introducing dimension reduction before using features to train model**

  The feature matrix may contain redundant information, so we can apply dimension reduction. Since the features are sparse, we can use SVD instead of PCA to do dimension reduction. This can eliminate some redundant information and reduce the computational cost.

- **Exploring more interesting features in the review text**

  There are still more interesting features in the review text need to be explored. For example, we can try adverbs, the number of all-uppercase words (which is a highly emotional feature) and so on.

- **Choosing better parameters for our models**

  We can try parameters of our models more precisely. We can use better parameters such as the $n\_estimators$ in Random Forest model (which involves more time to compute) to get better performance.

REFERENCES

[1] McAuley, J. J., & Leskovec, J. (2013). From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. (pp.897-908).

[2] James, G., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. (Doctoral dissertation, Springer New York). (pp. 316–321).

[3] Ho, T. K. (2002). A data complexity analysis of comparative advantages of decision forest constructors. Pattern Analysis and Applications, 5(2), 102-112.

[4] Hastie, T., Tibshirani, R., & Friedman, J. (2008). The Elements of Statistical, Second Edition. Springer. ISBN 0-387-95284-5.

[5] Castellano, R., Exploratory visualization of Amazon fine food reviews. http://blog.nycdatascience.com/student-works/amazon-fine-foods-visualization/

[6] Feng, H., & Lin, R. (2016). Sentiment classification of food reviews.

[7] Ji, T., & Wu J. (2016). Deep Learning for Amazon Food Review Sentiment Analysis. http://cs224d.stanford.edu/reports/WuJi.pdf

[8] Tssukiyama, D. (2016). Amazon Fine Food Reviews…wait I don't know what they are reviewing. https://cseweb.ucsd.edu/~jmcauley/cse190/reports/sp15/056.pdf

[9] Chu, Y., Pham, Y., Duong, N., Wu, Y. (2015). Predicting Rating of Amazon Fine Food from Reviews. https://cseweb.ucsd.edu/~jmcauley/cse190/reports/fa15/021.pdf

[10] Alshari, E. M., Azman, A., Mustapha, N., Doraisamy, C., & Alksher, M. (2016). Prediction of Rating from Comments based on Information Retrieval and Sentiment Analysis. International conference on information retrieval and knowledge management.

[11] Zhang, W., & Wang, J. (2016). Integrating topic and latent factor for scalable personalized review-based rating prediction. , 28(11), 3013-3027.

[12] Lei, Y., Chen, Q., Chen, C., & Li, W. (2015). User Preference Modeling by Trust Propagation for Rating Prediction. IEEE International Conference on Smart City/socialcom/sustaincom (Vol.119, pp.500-506). IEEE Computer Society.