Help Yelp Predict The Rating

Group 18 - The Grand Truthers
Omid Eghbali, Garvit Pugalia, Kanisha Shah, Hengda Shi, Prabhjot Singh

1 ABSTRACT

Online reviews have shown to have significance influence on consumer shopping behavior. Yelp, Google Reviews, TripAdvisor, and other review websites allow users to post their reviews for plethora of businesses, services, and products. An online review typically includes a rating in the form of a number of stars out of five and a free-form text. We can thus formulate a problem: given a user, their review of a business, and the business information on Yelp, are we able to accurately predict the number of stars that user will give the business?

In order to solve this problem, we extracted certain features and analyzed them through seven machine learning algorithms: (i) Linear Regression, (ii) Multi-class Logistic Regression, (iii) Decision Tree, (iv) Random Forest, (v) Neural Network, (vi) Gradient Boosted Regressor, and (vii) KNN.

We came up with the best model of predicting a user's rating for a particular business by analyzing the performance of the selected features and our seven chosen models. Using the provided Yelp dataset, we noticed that the Gradient Boosted Regressor consistently produced the lowest RMSE error, and was thus declared the best model.

2 OVERALL GOAL AND BACKGROUND

Reviews written by users are an essential part of websites like Amazon, Google Reviews, TripAdvisor and Yelp, which allow users to post their opinions and feedback about businesses, products, and services. These reviews serve as an online form of "word of mouth" and can influence other users to buy a product, or try out a service. Yelp, in our case, has collected over 135 million reviews worldwide, forming a huge database of customer information, business information, and their interrelationships. With such a large amount of data, there is potential for data mining and analysis. On such a popular website, however, a user may get overwhelmed by the all text written in thousands of reviews. That is why a user may prefer a star rating over a text rating. Usually given out of five, the stars indicate how much a user likes that product, business, or service. The higher the number of stars given, the more he or she likes that product.

The overall goal of our project was to access a sample of the Yelp review dataset and design a predictor that can forecast a customer's rating of a business based on the customer's history and the business' previous reviews. The type of data available to the predictor included business' rating and a number of reviews, customer's average ratings, and a large bank of existing reviews. In our approach, the data was processed first to retrieve the relevant features before fitting to our model. After training the model on a sample of the dataset, the RMSE of the prediction was computed to tweak and optimize the model.

In order to accomplish this task, we used a well-known machine learning library, Scikit-Learn, as well as NumPy, matplotlib and Pandas. These libraries helped us analyze the large Yelp dataset and form models for classification, regression, and clustering. Furthermore, due to the fact that Scikit-Learn is built on top of NumPy and matplotlib, it is highly compatible with the other preprocessing and visualization tools used in our project. We used the following models for the predictor which are all provided by scikit-learn:

1 Linear Regression

We can view the rating [1, 5] as a continuous variable, predicted by business and customer information, and then adjust to a class value.

2 Multiclass Logistic Regression

The algorithm is an extension of Logistic Regression to multiple classes. It is an attractive option as it doesn't assume linearity or normality of the dataset (Starkweather, 2011).

3 Decision Tree

This intuitive, rule-based classifier requires low data preparation and works with nominal and numeric features. Decision trees also provide a good foundation for feature selection, which can enhance other classifiers.

4 Random Forest

Given the large scale of our dataset, notoriety of random forest for being good at classification tasks, and resilience to overfitting (Donges, 2018), this model will likely be key in getting a lower RMSE.

5 Neural Network

With a large amount of features and potential interrelationships, most classifiers are unable to detect highly complex patterns in the data. However, neural networks, along with a range of available optimizations, can provide a robust, accurate solution.

6 Gradient Boosted Regressor

Instead of using a majority system between multiple classifiers (such as in Random Forest), Gradient Boost builds multiple classifiers sequentially where each classifier learns from the mistakes of the other. Given a cost function, the Gradient Boost algorithm can efficiently minimize bias.

7 K-Nearest Neighbors

KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.

3 RELATED WORK

A paper titled “Applications of Machine Learning to Predict Yelp Ratings” published by Stanford University aimed to identify the most important features affecting a business’ Yelp rating. By using SVMs, logistic regression and many other models, they achieved “an accuracy of ~ 45%” (Carbon et al). While this paper can provide a foundation for our work, there are many differences that need to be considered. The paper focuses on extracting important features, with little consideration given to reducing the error. They also work with a subset of the provided data, focusing on restaurants in Phoenix, AZ. By focusing on a subset, they are able to extract more detailed, location-specific information. Finally, they employ data mining techniques such as Naive Bayes for Sentiment Analysis that were not considered due to time constraints.

While the paper sheds light on relevant classification techniques, the contrasting objectives and difference in datasets make it a good starting point rather than a guide.

4 PROBLEM FORMALIZATION

This problem can be formalized as: Given a previously unseen {customer_id, business_id} pair, with given attributes for each customer and business, predict a rating from [1, 2, 3, 4, 5] for the customer’s review of the business.

5 DATA PREPROCESSING

Data preprocessing has been our largest focus over the past couple of weeks. Given how our features are spread out across multiple files, we have been using pandas to convert the data into a more manageable form which we can then feed into our models. Further, we have implemented what is it needed to normalize feature values to make comparisons between the normalized and un-normalized dataset. Using what we have so far we can train basic models, but to improve on accuracy more work is needed to clean the data further and finalize feature choices.

Missing data is a factor that can affect many classification models. To improve accuracy, some features were disregarded due to the high percentage of missing data. For example, for attributes_Music, 82% of the data was missing:

Figure 01: Taken from Kaggle, the figure states the proportion of valid, missing and unique values in the attributes_Music column.

However, it would be too costly to remove all rows that have missing data. Therefore, we decided to mainly use the mode for the specific attribute to fill all the NaNs in the columns.

There are also many features that, intuitively, do not affect the rating such as a business’ opening hours, location, or zip code. For example, if we look at the distribution of ratings for the entire training dataset:

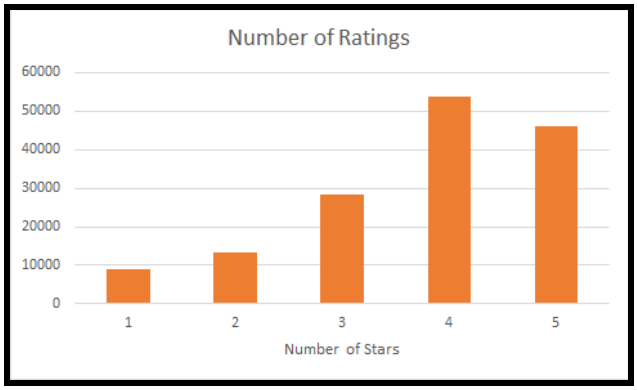

Figure 02: The distribution of ratings (stars) for the entire training dataset.

If the ratings are restricted to certain feature values, and the corresponding distribution looks similar to the overall rating distribution, we can conclude that the feature doesn't affect the rating. As an example, we take ratings from specific states (Nevada, and Ohio) and plot the distributions:

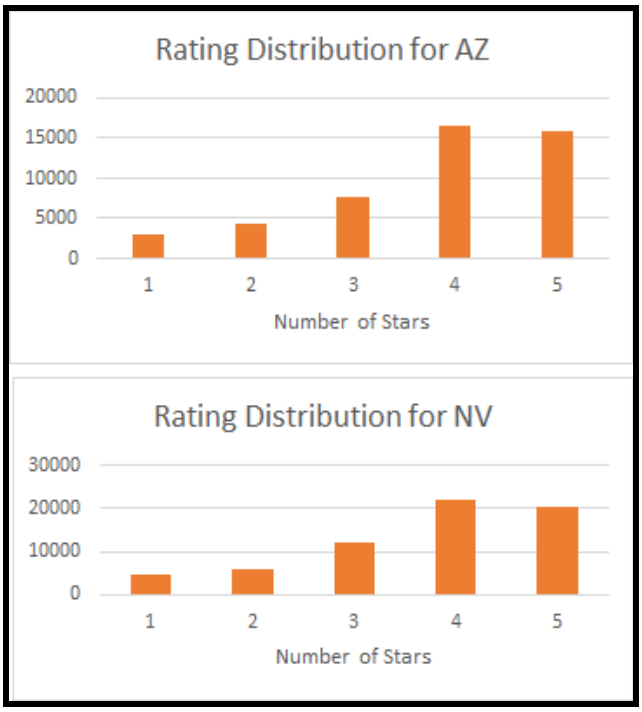

Figure 03: Figure captures the rating distribution for states Arizona and Nevada

Clearly, the same distribution is observed and we can conclude that "State" has no effect on the rating. (While this method was not directly used in code, it demonstrates the basic reasoning for selecting important features)

Apart from these, we also decided to drop most categorical variables. This decision was made to avoid tedious parsing and over-complication of the classifier, with a possible loss in accuracy. There is one categorical data that we decided to keep, which is "attributes_NoiseLevel". Intuitively, we believed that this attribute would contribute to our model because noise level is really important while rating a place.

Finally, we decided to use the following features, which can also be seen in *constants.py*.

Business Features:
"stars",
"review_count",
"attributes_RestaurantsPriceRange2"
"attributes_BikeParking",
"attributes_BusinessAcceptsCreditCards",
"attributes_GoodForKids",
"attributes_HasTV",
"attributes_OutdoorSeating",
"attributes_RestaurantsDelivery",
"attributes_RestaurantsGoodForGroups",
"attributes_RestaurantsReservations",
"attributes_RestaurantsTableService",
"attributes_RestaurantsTakeOut",
"attributes_WheelchairAccessible"
"attributes_NoiseLevel"

User Features:
"average_stars",
"review_count",
"useful"

While processing train_review.csv, we treat each row as a training data with label as the actual rate given by the user to the business. We extracted the user_id, business_id, and stars, and then replace the user_id and business_id with the corresponding features in user.csv and business.csv. By concatenating them together, we make a training dataset that is ready to use for our selected regression models.

6 METHOD DESCRIPTION

In order to tackle this large data mining problem, we have set a plan to break down the problem into manageable units. Firstly, we start with data visualization in which we make histograms of the different features and look at the frequency of the ratings that are given. This has already proven to be tremendously helpful in our task of feature selection. Next, we move onto preprocessing the data sets, which is further described below. We plan to employ a forward selection approach of the features, since training time using all the features to start would take up a lot of our time. Finally, we move onto training our different models of interest, more about this can be seen in the relevant section below. The best model will be chosen based on the lowest RMSE we find and will be reported on accordingly.

Different methods learned in the class were used to achieve our goal of predicting the ratings by any user. The methods we used for them are Decision tree, SVM, KNN, Random Forest, Gradient Boosting, Stochastic Gradient Descent Regression, NN, and Logistic Regression. All the methods were imported from scikit-learn library. The methods we focused on to get optimal models are explained below.

6.1 Decision Tree

It was critical to find out the best depth for a given feature selection. So, we iterated over all the possible depths and did the following. We first trained our model using the provided training data for a given depth. We then tested it and used our loss function (RMSE: Root Mean Square error) to check the accuracy of our model. We selected the model with the least error for predicting the ratings.

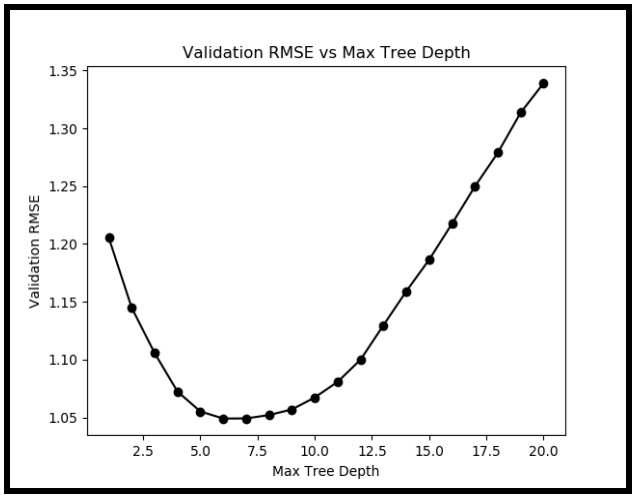

Figure 04: Validation RMSE for a decision tree with different max depths.

6.2 Neural Networks

We decided to have 100 hidden layers with a maximum of 200 iterations and learning rate of 1e-4. More the hidden layers are, more accurate is the prediction. After 100 hidden layers, increasing the layers did not show any substantial increase in the accuracy. Hence, we kept 100 hidden layers. We keep our learning rate constant throughout the process. If our learning rate does not help reduce the training loss twice or more, we divide our learning rate by 5 for better validation score.

6.3 Linear Regression

We used one simple model and another one with polynomial features. In the simple model, we passed our training data to train the model. We then used the model to predict the values for the testing data. The error made by this model was calculated using RMSE loss function. To implement linear regression with polynomial features, we first had to transform our training data and the testing data. Just like the previous case, we trained our model by passing our transformed training data and checked the accuracy of our model by passing the testing data and computing the RMSE function over it.

6.4 Logistic Regression

We created a simple model which was optimized using L2-regularization and could handle the multinomial loss. We used 5-fold cross-validation technique for training our model i.e. one of the five sets was used for validation and the rest for training the data. The training data was passed in to train such a model. The same model was used to predict the yelp ratings for the validation data (different from the 5-fold training data). We could not calculate the accuracy of our model using the RMSE loss function since it did not converge.

6.5 Gradient Boost

For finding the best model for Gradient Boosting, we tried different parameters to see which parameters give us the best model. We tried different boosting stages, learning rates, and the maximum depth of the individual regression estimators. Since gradient boosting is robust to overfitting data, a higher number of boosting stages usually give us better results. We saw that better accuracy was obtained by having 127 boosting stages, maximum depth of 4 to limit the number of nodes in the tree and learning rate of 0.1. This model trained using the training data gave an RMSE of roughly 1.04 on the validation data set.

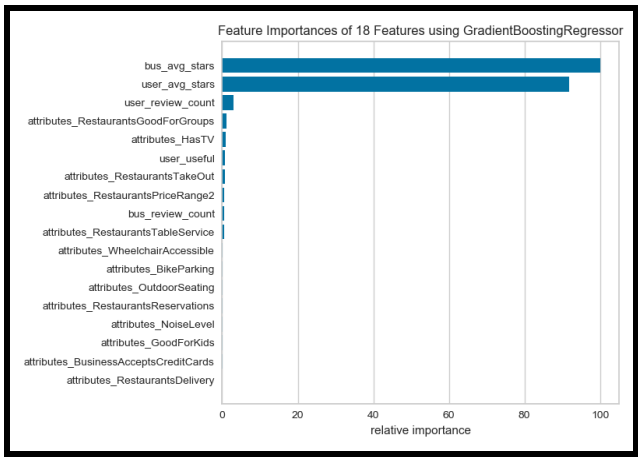

Figure 05: Feature importance of different features extracted from Gradient Boosting Regressor. Features such as business average stars, and user average stars are most important in forming an accurate model.

6.6 Random Forest

We created different models to see which model would give the best prediction for the yelp ratings. The parameters we changed to obtain different models are the number of trees in the forest and the condition for the minimum of nodes required for splitting an internal node. We tried the model with 150, 200, 250 trees and 2, 3, 4 for the minimum number of samples for splitting. We kept the depth of the tree constant which was 8. We used the training data to train the model and checked the accuracy of each model on the validation data. The model with the best accuracy for prediction was selected. The accuracy for the validation data was calculated using the RMSE loss function.

6.7 KNN

For KNN, the model with the best accuracy was chosen from all the different models of K. We tried various Ks using GridSearch and we plotted the following graph indicating validation RMSE associated with different values of K:

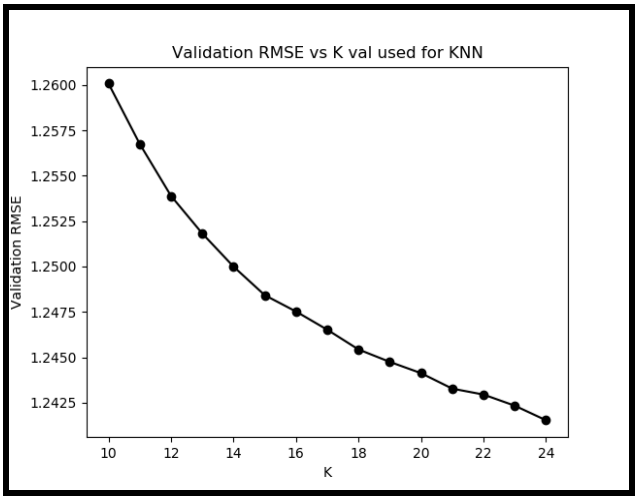

Figure 06: Validation RMSE for different values of K in K-Nearest Neighbors. The validation RMSE after K = 24 was negligibly decreasing.

7 EVALUATION

The Root Mean Squared Error (RMSE) can be calculated with the following equation:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{N}}$$

The RMSE describes the standard deviation in the differences between the predicted and real values. There are a few benefits from its alternative, MSE, that need to be taken into consideration.

The RMSE brings the error back into the units of the data (i.e. value on the y-axis), whereas MSE is in the square units of the data. Therefore, the RMSE is a more easily interpretable and accurate measure of the goodness of the fit. The RMSE normalizes by the amount of training data provided, therefore, a relatively large sample of the Yelp dataset should be used.

For the algorithms mentioned, we obtained the following RMSE scores:

Algorithm	Validation Set RMSE
Random Probabilistic Model	1.712
Decision Tree	1.049
Random Forest	1.043
KNN	1.215
Linear Regression with Polynomial Features	1.056
Logistic Regression	No convergence
Gradient Boost	1.042
Neural Networks	1.047

We selected the model with the lowest, most consistent RMSE error, out of all our models. This gave us a **Kaggle public score (Test RMSE) of 1.05352.**

8 CONCLUSION

Given the number of features we are provided, we have experienced that feature selection is taking lot more work. In order to gain coherent results, it was necessary that we improve on our chosen features which we worked on for last few weeks. Also, after initial data processing, we realized that there are possible unique customers and businesses in the validation set. Therefore, more advanced techniques with user correlation were added to our preliminary model. In our initial model, we used the business average stars and user average stars as the two features, so we can get a feel for the necessary complexity and confirm our models work as intended.

Previously, we tested our first version of the preprocessed data, using only the two aforementioned features, to build

logistic regression, linear svm, and random forest models. We observed that with linear svm, the time taken to complete the model scales heavily with the number of features included. Hence we were careful with adding more features while trying to lower our RMSE. Linear SVM and logistic regression both got an RMSE of roughly 1.2, which was a satisfying start but needed improvement. Surprisingly, our first run of the random forest had the worst RMSE out of everything tried thus far.

These results improved significantly in the subsequent weeks as more features were selected and model tuning was done. We build a decision tree, neural networks, and gradient boost models. Decision tree model, neural network and gradient boosting model got an approximate RMSE of 1.05, 1.05 and 1.04 which was a notable improvement form our previous linear SVM and logistic regression model. Since the gradient boosting model performs better than all the other classifiers, we chose that model for the final submission.

For future work, we can consider an additional Naive Bayes classifier and expand on feature selection to further decrease the RMSE error.

REFERENCES

[1] Starkweather, J. and Moske, A. (2011). Multinomial Logistic Regression. [online] It.unt.edu. Available at: https://it.unt.edu/sites/default/files/mlr_jds_aug2011.pdf

[2] Mohamed Aly (2005). Survey on Multiclass Classification Methods. https://www.cs.utah.edu/~piyush/teaching/aly05multiclass.pdf.

[3] The Random Forest Algorithm – Towards Data Science: 2018. https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd. Accessed: 2018- 12- 12.

[4] Andreea Salinca (2017). Convolutional Neural Networks for Sentiment Classification on Business Reviews. https://arxiv.org/pdf/1710.05978.pdf

[5] Ben Gorman (2017). A Kaggle Master Explains Gradient Boosting. http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

[6] Scikit-learn Documentation: https://scikit-learn.org/stable/user_guide.html

[7] Jiliang Tang, Salem Alelyani and Huan Liu. Feature Selection for Classification: A Review. https://pdfs.semanticscholar.org/310e/a531640728702fce6c743c1dd680a23d2ef4.pdf

[8] Kyle Carbon, Kacyn Fujii, Prasanth Veerina. Applications of Machine Learning to Predict Yelp Ratings. https://pdfs.semanticscholar.org/d924/40fbdc144828468ec0a2b2e13df534ceedec.pdf

[9] Yelp Dataset Challenge [Online]. Available: https://www.yelp.com/dataset/challenge

[10] NumPy documentation: https://docs.scipy.org/doc/numpy/reference/routines.html

[11] Jiawei Han, Micheline Kambe, Jian Pei (2011). Data Mining: Concepts and Techniques:The Morgan Kaufmann Series in Data Management Systems. 3rd Edition.

[12] Adi Bronshtein (2017) A Quick Introduction to K-Nearest Neighbors Algorithm. https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7

TASKS DISTRIBUTION FORM

Tasks	**Members**
Clean up / Finalize Preprocessing Code	Hengda
Test current Linear SVM/Logistic Reg/Random Forest model further	Hengda, Omid, Garvit
Feature selection and model tuning	Omid, Prabhjot, Garvit
Linear Regression Model Implementation, Further model tuning	Prabhjot, Kanisha
Researching what we can do with review contents	ALL
Finalize Model tweaking / features chosen for each	Hengda, Omid, Garvit
Looked further into review contents after learning naive bayes	ALL
Try more advanced correlation tools to deal with unique user issue	Hengda, Kanisha
Begin outputting graphs for report, writing report	Prabhjot, Omid, Garvit
Finish Report and Submit final submission(s) to Kaggle early week 10	ALL