Building a Recommendation System for Beer Ratings

Jiushuang Guo, Carson Zhao, Rui Ling
Stanford University
jguo18@stanford.edu, czhao333@stanford.edu, ruiling@stanford.edu

Abstract

In this project, we use machine learning for two tasks. The first task is to build regression and classification models to predict the overall ratings of a specific brand of beer, based on content-based features (e.g. aroma, appearance, palate, taste). We extract features from beer text reviews such as sentiment score, TF-IDF, word counts, and also use K-modes, an unsupervised learning method, to cluster similar beers and users together. The second task of the project is to build a beer recommendation system on a sparse utility matrix consisting of user, beer, and beer rating, using collaborative filtering and latent factor methods. LightGBM is our best traditional ML model with a test RMSE of 0.5593 and test accuracy of 0.8695, and KNN collaborative filtering is our best recommender algorithm with a test RMSE of 0.5750 and test accuracy of 0.8629.

1. Introduction

The beer industry has expanded exponentially in the last few decades, leading to an overwhelming number of beers for customers to choose from. In this project, we will create models to help breweries predict beer ratings and recommend new beers to customers. Our input is beer reviews, containing beer attributes such as beer's appearance, aroma, palate and taste, all scored in the range of 0 to 5, beer style, beer ABV(alcohol per volume) and id for both beer and brewery. Also, the input includes user's information such as their profile name, gender, age, the time they wrote the review, and their review texts. Since the response variable beer rating can be both discretized and continuous, we can use both regression and classification models to output a predicted overall beer rating.

To build a recommendation system, we implement collaborative filtering algorithms based on a utility matrix of users, beers, and respective beer overall ratings and give beer recommendation based on similarity among the users and beers and recommend highly predicted rating beers.

We also apply sentiment analysis to extract more features from the review texts to help improve the predictions of overall ratings.

2. Related work

Recommender systems are very popular methods that attempt to predict some rating a user would assign to an item [9]. They are widely used commercially. There are three main approaches to recommender systems: content-based, collaborative filtering, and latent-factors. In the collaborative filtering setting, we find a set of other users whose ratings are "similar" to some user x's ratings, and we estimate x's ratings based on the users in that set [7]. This allows CF models to capture local neighborhood effects. We can measure "similarity" through cosine similarity, Jaccard measure, Pearson correlation, etc. Global effects are added to recommenders through baseline estimation and model deviations [7]. Latent-factors methods such as SVD allow the model to learn regional effects, and were used by the Netflix challenge winners to decrease RMSE [6].

In 1983, Breiman, et. al invented the classification and regression tree [1]. This would lead to crucial advances in data mining. Single trees are not powerful and have many disadvantages. They are discontinuous piecewise models, fragment data, and high in variance. However, they are fast, $O(DNlogN)$, where D is the number of predictions and N is the number of observations, handle all types of variables, invariant to monotone transformations, and resistant to noisy variables [4]. Random forests and boosting models build on trees, keeping their advantages while improving on the disadvantages. Breiman first coined the term "bagging", or bootstrap aggregation, in 1996, which led to random forests in 2001 [2]. Random forests are in fact bagged trees, which greatly stabilizes trees. Finally, boosting has been found to be very effective in many ML tasks. The gradient boosting algorithm is an ensemble method that increases the predictive accuracy of trees. Boosting produces a more powerful model with a much larger function space than that of a single tree, and the boosting function is much smoother [3].

3. Dataset and Feature

3.1. Dataset

The dataset is provided by BeerAdvocate and contains 1,586,614 observations and 9 features [8]. We split the dataset into two parts, 80% for training set and 20% for testing set. We then further split the training set and extract 20% for a validation set.

After conducting exploratory data analysis (EDA), we found that there are 104 different beer styles, 5840 different breweries, 56,788 unique beers, and 33,387 unique users. Among those users and beers, about 85% of users gave fewer than 50 reviews and about 90% of beers have fewer than 50 reviews.

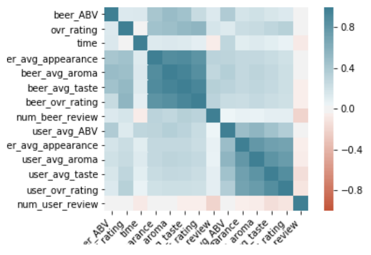

Figure 1: Correlation plot of numeric features

3.2. Feature Engineering

We conduct feature engineering on the original variables to extract more meaningful features which can better support the regression and classification models. We first compute average ratings (overall rating, aroma, taste, appearance, and palate) for each user and each beer to allow the model to adjust for user/item bias. This lets the model learn global effects. Then, we incorporate users' preferences by computing which beer style they rated the most. The correlation plot in Figure 1 shows how correlated our numeric features are to each other.

3.2.1 K-modes

The beer style category by itself does not account for beer quality and popularity. In order to better cluster similar beers, we implement K-modes, an unsupervised method to re-cluster the beers. K-modes is similar to K-means, but with categorical variables. We implement k-modes for various of k and find the best k based on an elbow plot of the K-modes dissimilarity metric $D(X,Y) = \sum_{h=1}^{m} d(X_h, Y_h)$ where $d(X_h, Y_h) = \begin{cases} 1, & \text{if } X_h = Y_h \\ 0, & \text{otherwise} \end{cases}$

3.2.2 Natural Language Processing

We extract sentiment scores of users' reviews using natural language processing, and compute the average sentiment score for each beer and each user. Sentiment scores are calculated using the NLTK sentiment polarity score and are in the range -1 to 1 where negative values imply negative sentiment and positive values imply positive scores.

Beyond that, we also apply TF-IDF (term frequency–inverse document frequency),a statistic that can reflect the importance of a word in a document. $TF_{ij} = \frac{f_{ij}}{max_k f_{kj}}$ where f_{ij} is the frequency of term i in document j. $IDF_i = log\frac{N}{n_i}$ and finally $TF\text{-}IDF_{ij} = TF_{ij} \times IDF_i$. We apply this method on the beer review to find the top 1000 key words that can be beneficial in rating prediction. We also use a count vectorizer to convert our reviews to a matrix of token counts, and keep the top 1000 features. Appending the full TF-IDF and counts features would add 2000 columns to the dataset, so we use truncated SVD to reduce the dimension of the extracted features to 250 (150 TF-IDF and 100 count features). The truncated SVD retains around 50% of the total variance explained.

4. Methods

Overall beer rating is a discrete value ranging from 0 to 5, incrementing by 0.5. Thus, we can treat this response variable as both multi-class labels or continuous values (since a prediction of 4.33 makes intuitive sense). Therefore, we train the inputs with both regression models and classification models to see which performs better in our task. Later, we discuss CF and latent-factor recommendation system algorithms.

4.1. Regression and Classification

4.1.1 Lasso and Ridge Regression

The linear regression model calculates parameter estimates to minimize the residual sum of squares, and is the best linear unbiased estimator. Shrinkage methods like lasso and ridge regression reduce the variance of linear regression by introducing a little bias [4]. They shrink the parameters toward zero and have a complexity hyperparameter that controls the amount of shrinkage. Lasso and ridge regression add L1 norm and L2 norm to the ordinary least squares criterion, respectively. Elastic net is a compromise of lasso and ridge regression and adds the complexity term $\lambda \sum_{j=1}^{p} \alpha|\beta_j| + (1-\alpha)\beta_j^2$. Here, λ is the regularization strength and $\alpha = 0$ recreates ridge regression and $\alpha = 1$ recreates lasso.

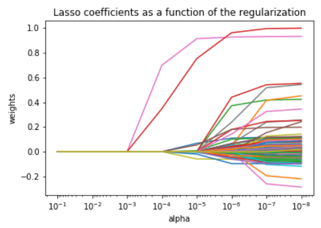

Figure 2: Lasso Path, shows the entry point of parameters with respect to the regularization strength alpha

4.1.2 Logistic Regression and Naive Bayes

We wanted to explore baseline classification algorithms to see what works well for our task. Multi-class logistic regression is a discriminative method and predicts the probability of assigning examples to every class. Gaussian Naive Bayes is a very good initial algorithm to try. Naive Bayes is a generative model that assumes that the features are conditionally independent given the response variable y [4].

4.1.3 Decision Trees

Before discussing random forests and gradient boosting, we first introduce their backbone - decision trees. The tree method at each step chooses the variable which makes the best split and then decides the split point. The best split is the split that best reduces the risk (error) and improves the purity of the leaf nodes [1]. Decision trees allow us to calculate variable importance [1]. Single tree models are intuitive to understand but not powerful for prediction. This is because they are high in variance and are very unstable.

4.1.4 Random Forest

Random forest and gradient boosting algorithm are ensemble methods to improve the performance of trees (by achieving higher predictive accuracy). Random forest is a bagging (bootstrap aggregation) technique and reduces variance by building a large collection of de-correlated trees and then averaging them. Bagging stabilizes unstable tree models, and is more robust to small changes in the training data. [4].

4.1.5 Gradient Boosting

Boosting is an effective ensemble method based on trees, combining a large number of weak learners that can outperform individual stronger learners [3]. During the training process, boosting assigns more weight on misclassified examples in the current learners to let the future learners focus more on these examples to better classify them. Gradient boosting implements gradient descent to minimize the loss function.

LightGBM is a tree-based gradient boosting framework for both regression and classification that can train large datasets with great speed while being memory-efficient. This is because LightGBM uses histogram-based algorithms that can bucket continuous feature values into discrete bins [5].

4.1.6 Neural Networks

In order to find out whether another non-linear architecture can help with overall rating predictions, we construct a two-layer neural network regression model. The network has two hidden layers with 40 nodes in the first hidden layer and 30 nodes in the second hidden layer. The layers are separated by a sigmoid activation function σ. The output of the model is a vector $\hat{y} \in \mathbb{R}^{n\times 1}$. The formulation is below:

$$Z_1 = XW_1 + b_1$$
$$Z_2 = \sigma(Z_1)W_2 + b_2$$
$$\hat{y} = \sigma(Z_2)W_3 + b_3$$

where $X \in \mathbb{R}^{n\times d}, W_1 \in \mathbb{R}^{d\times 40}, W_2 \in \mathbb{R}^{40\times 30}$, and $W_3 \in \mathbb{R}^{30\times 1}$. We minimize mean squared error loss between the predictions $\hat{y}$ and the true values y.

4.2. Recommendation System

Collaborative Filtering (CF) is a recommender algorithm built only on an utility matrix, a sparse matrix of product ratings with columns unique users and rows unique products. Thus, it is an algorithm free of extra features and excludes users' historical item preferences. Collaborative Filtering is based on the assumption that users' future behaviors and preferences are consistent with the past. In our project, we implement the K-Nearest Neighborhood (KNN) CF model and SVD, a latent-factor model.

4.2.1 KNN Methods

There are user-based and item-based KNN algorithms. KNN user-user calculates the similarities between target user i and all other users and choose the top K similar users by the Pearson correlation coefficient similarity:

$$sim(x,y) = \frac{\sum_{s\in S_{xy}}(r_{xs}-\overline{r_x})(r_{ys}-\overline{r_y})}{\sqrt{\sum_{s\in S_{xy}}(r_{xs}-\overline{r_x})^2}\sqrt{\sum_{s\in S_{xy}}(r_{ys}-\overline{r_y})^2}}$$

It predicts the rating for user x on item i, denoted r_{xi}, by weighted average (based on Pearson correlation coefficient) of item j's ratings of the K similar users. It also adds user and beer baselines (bias) to adjust for different beers' and users' deviance from the global values. The predicted rating is formulated as:

$$r_{xi} = \mu + b_x + b_i + \frac{\sum_{y\in N_x} S_{xy}\cdot(r_{yi}-b_{yi})}{\sum_{j\in N(x)} S_{xy}}$$

where S_{xy} is the similarity between users x and y, N_x is the K-nearest neighbors of user x, and $b_{yi} = \mu + b_y + b_i$ is the baseline adjustment for user y and item i.

We can also define the item-item KNN predicted rating similarly with the following formula:

$$r_{xi} = \mu + b_x + b_i + \frac{\sum_{j\in N(i)} S_{ij}\cdot(r_{xj}-b_{xj})}{\sum_{j\in N(i)} S_{ij}}$$

where S_{ij} is the similarity measure of item i and item j, N_i is the K-nearest neighbors of item i, and $b_{yi} = \mu + b_y + b_i$ is the same as defined above.

4.2.2 SVD Methods

"SVD", or Matrix Factorization, is a latent factor model to solve the sparsity challenges of standard CF methods like KNN. SVD decomposes the sparse utility matrix to low-dimensional matrices with latent factors to reduce the sparsity. In this way, SVD can compute similarities between low-dimensional latent factor vectors, instead of high dimensional original sparse rows, making it possible to compute similarity measures between two users, regardless of whether they rated the same beer. The rating prediction in the SVD algorithm is computed by

$$r_{xi} = \mu + b_x + b_i + q_i \cdot p_x$$

where μ is the overall mean rating, b_x is the bias for user x, b_i is the bias for beer i, and $q_i \cdot p_x$ is the user-beer interaction.

The loss function is

$$\min_{Q,P} \left(\sum_{(x,i)\in R} \Big(r_{xi} - (\mu + b_x + b_i + q_i \cdot p_x) \Big)^2 + \lambda_1 \sum_i \|q_i\|^2 + \lambda_2 \sum_x \|p_x\|^2 + \lambda_3 \sum_x \|b_x\|^2 + \lambda_4 \sum_i \|b_i\|^2 \right)$$

All the parameters are updated by stochastic gradient, e.g. $Q \leftarrow Q - \mu \nabla Q(r_{xi})$.

4.2.3 Other Recommenders

We also explore Non-negative Matrix Factorization (NMF), which is very similar to the SVD method above. The only difference is that the user and item factors are forced to be positive. Another method is the SlopeOne method, where the prediction for user x on item i is user x's average rating plus the average deviation among the set of relevant items to item i. Finally, our last recommender method is CoClustering, where users and items are assigned to some clusters C_u, C_i, and co-clusters C_{ui}. The prediction is the average rating of the co-cluster plus the deviation of the average rating of C_u from the global user rating μ_u, plus the deviation of the average rating of C_i from the global item rating μ_i.

5. Experiments/Results/Discussion

5.1. Evaluation Methods

The evaluation methods we use to measure the performance of the regression models are root mean squared error (RMSE) and accuracy. RMSE measures the square root of the average squared difference between the predicted responses and true responses. Regression accuracy level is calculated as 1 - MAPE.

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

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

For classification models, we use accuracy score which gives the fraction of correctly classified samples.

5.2. Experimental Details

For the two-layer neural network, we tried various combinations of hyperparameters and chose the best set via validation set performance. We found that hidden sizes of 40 and 30 work well, along with a batch size of 2048. We used SGD with momentum 0.9 as our optimizer, with initial learning rate 10^{-2} and weight decay 10^{-6}. Also, an exponential learning rate decay below decreases the RMSE and MAPE slightly, $lr = lr \times 0.93^{epoch}$. We trained the model for 11 epochs.

For the light gradient boosting model, we used cross-validation and a grid search to tune the many hyperparameters. The optimal parameter configuration for LGBM is bagging fraction of 1, feature fraction of 0.7, minimal number of data in one leaf of 30, maximum number of leaves 40, L2 regularization of 10^{-7}, and max number of bins of 265.

The recommender algorithms also had many hyperparameters to tune. We found that SVD with 22 factors and 0.03 regularization strength, CoClustering with 10 user clusters and 2 beer clusters, KNN item-item with K=100, and KNN user-user with K=70, leads to the best validation set performance.

5.3. Quantitative Results

On the traditional ML side, the best model is Light GBM with a test RMSE of 0.5593 and test accuracy of 0.8695. The results of Ridge and Lasso have very similar and relatively high results, which is due to the correlation between the response variable and the engineered features, and the fact that linear models can capture the linear relations well. PCA with ElasticNet has worse performance, since dimensionality reduction of PCA loses some information, and therefore is not as powerful. The neural network and random forest models are able to improve on linear models by capturing non-linear relationships and complex interactions.

The best results from the recommender algorithms are KNNBaseline User-User with 0.8629 test accuracy and KNNBaseline Item-Item with 0.5750 test RMSE. The results indicate that without any extra features, recommendation system algorithms are almost as powerful as the best regression methods with engineered features. NMF makes all the latent factors positive, and thus there is a decrease in performance, since we lose dissimilarity information between users or beers. Both KNN and SVD have relatively high performance, but there is a trade off between memory and training time. KNN is very memory-intensive while SVD has relatively longer training time. This is because

SVD must calculate a matrix factorization of the original utility matrix, which takes a lot of computational time, and KNN needs memory to store the nearest neighbors.

Algorithm	Test RMSE	Test Accuracy
SVD	0.5773	0.8620
NMF	0.5985	0.8595
SlopeOne	0.5893	0.8614
CoClustering	0.6346	0.8551
KNNBaseline Item-Item	**0.5750**	0.8624
KNNBaseline User-User	0.5860	**0.8629**

Table 1: RMSE and Accuracy (1-MAPE) of different recommendation system algorithms on the test set.

Method	Test RMSE	Test Accuracy
Ridge	0.5746	0.8656
Lasso	0.5746	0.8655
PCA + ElasticNet	0.6087	0.8566
Random Forest	0.5712	0.8680
Light GBM	**0.5593**	**0.8695**
Neural Network	0.5718	0.8664

Table 2: RMSE and Accuracy of different regression methods on the test set.

Method	Test Accuracy
Naive Bayes	0.3695
Logistic Regr	0.3695
Random Forest	0.4153
Light GBM	**0.4287**

Table 3: Accuracy of different classification methods on the test set.

5.4. Qualitative Results

5.4.1 NLP Analysis

The sentiment score histogram in Figure 3 shows that review sentiments are left skewed with a spike in the slightly negative scores. This is aligned with the distribution of the beer ratings since ratings are concentrated around four out of five.

From the top-30 important words based on TF-IDF scores shown in Figure 3, we notice that the beer attributes users care most about are taste, appearance, and aroma. Also, many users like describing the color, carbonation, and hops of the beer.

5.4.2 LightGBM

The Boosting model feature importance plot shows that the beer cluster obtained from k-modes clustering, beer ABV,

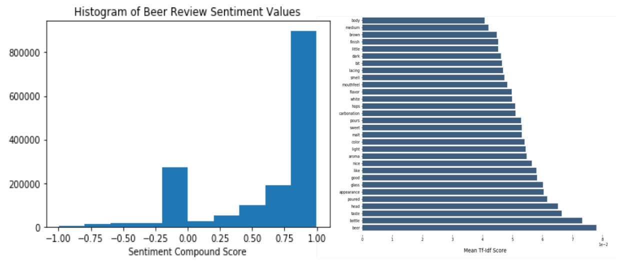

Figure 3: Histogram of NLTK sentiment scores generated from beer reviews and top-30 words based on TF-IDF

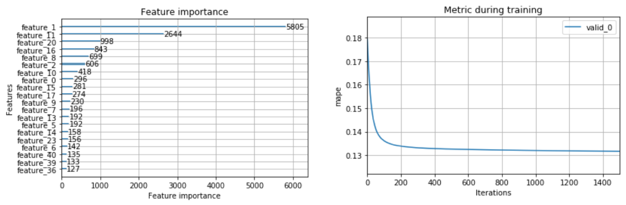

Figure 4: LightGBM plots. Left: top 20 most important features. Right: validation MAPE metric vs number of trees

avg beer palate, avg beer ABV grouped by user, and beer style are the most important. It also shows that users usually prefer a specific cluster/style of beer, and deviations in attributes for beers and users make good predictors.

6. Conclusion/Future Work

We explored many types of models in the context of recommendation systems. In the traditional ML sense, LightGBM did the best in RMSE, MAPE, and classification accuracy because it reweights the results based on the performance of each tree, assigning higher weight on stronger learners and lower weight on weaker learners. Random Forest and two-layer network also performed well in those three metrics. In the recommendation system algorithm context, the data is transformed into a sparse utility matrix, consisting of only user, beer, and overall rating to train the models. KNN collaborative filtering methods accounting for baseline ratings perform the best. KNN user-user (similarities computed between users) does the best on accuracy and KNN item-item (similarities computed between beers) does the best on RMSE. This makes sense since a user will likely enjoy beers with very similar attributes to their favorite beers.

There are multiple more experiments we would like to try concerning this project. There are more recommendation algorithms to explore such as SVD++, TimeSVD, and a hybrid model that combines collaborative filtering with content-based models. We could also try ensembling our different results by weighted average to improve the performance on the test set.

7. Contributions

The three of us contributed to the project equally. Jiushuang Guo and Carson Zhao implemented the recommendation system algorithms, NLP, feature engineering, light gradient boosting, and two-layer neural network. Rui Ling is in charge of Exploratory Data Analysis, domain knowledge of beers, and training the dataset with general regression and classification models. All team members contributed to writing the final report.

References

[1] Breiman, Leo. Classification and regression trees. Routledge, 2017.

[2] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[3] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.

[4] Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. "The elements of statistical learning. Springer series in statistics." :. Springer, 2001

[5] Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." Advances in Neural Information Processing Systems. 2017.

[6] Koren, Yehuda. "The bellkor solution to the netflix grand prize." Netflix prize documentation 81.2009 (2009): 1-10.

[7] Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets. Cambridge university press, 2014.

[8] Leskovec, Jure. “BeerAdvocate Reviews.” Https://Snap.stanford.edu/Data, Stanford Network Analysis Project, snap.stanford.edu/data/web-BeerAdvocate.html.

[9] Ricci, Francesco, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook." Recommender systems handbook. Springer, Boston, MA, 2011. 1-35.