Beer Style Prediction using the RateBeers Dataset

Akshay Gopalkrishnan, Albert Lin, Akhil Parvathaneni, Karthik Guruvayurappan

Dataset and Exploratory Data Analysis

Basic Statistics and Properties

The RateBeers dataset is a publicly available dataset that contains 2,855,322 beer reviews that span from April 2000 - November 2012. The dataset includes 40,213 unique users and 110,419 unique items. Each beer review included: beer name, beer ID, brewer ID, alcohol by volume (ABV), beer style, appearance review (out of 5), aroma review (out of 10), palate review (out of 5), taste review (out of 10), an overall review (out of 20) which averages the former 4 review types, a review timestamp, a review profile name, and review text.

Distribution of Styles

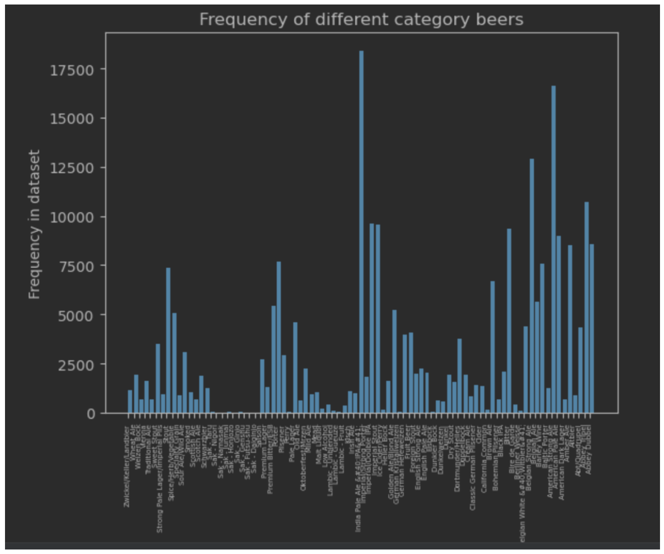

Figure 1: Distribution of review frequencies in the RateBeer dataset. There are 89 unique beer categories present in the RateBeers dataset.

In this dataset, there are 89 unique beer styles. However, most beers appear to fall in the top categories, while numerous beer categories have very few reviews (Figure 1). Therefore, the RateBeers dataset demonstrates class imbalance. To visualize the distribution of review frequencies across the top categories, we visualized review category frequencies for the top 5 beer styles. The five most common beer styles were: India Pale Ale, Pale Lager, Belgian Strong Ale, Imperial Stout, and Imperial/Double IPA (Figure 2).

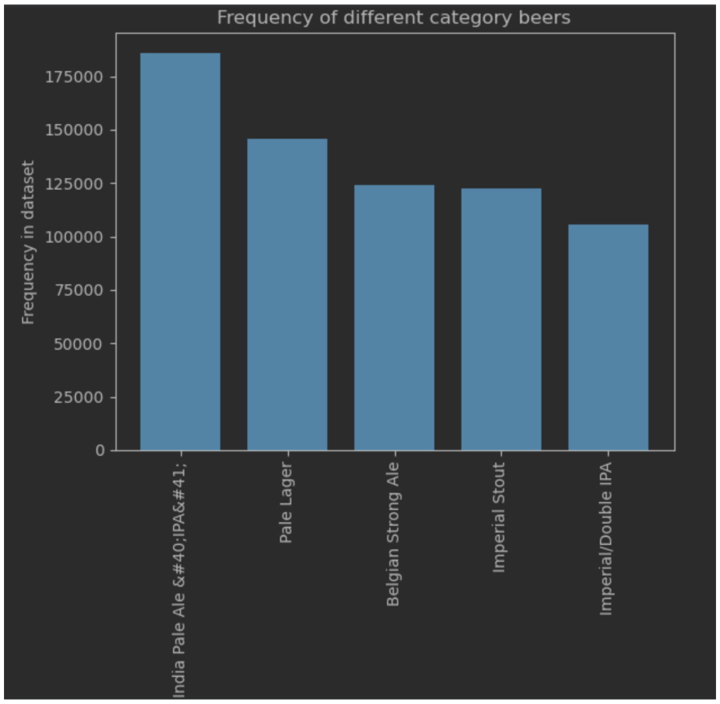

Figure 2: Distribution of top 5 beer categories in the RateBeers dataset.

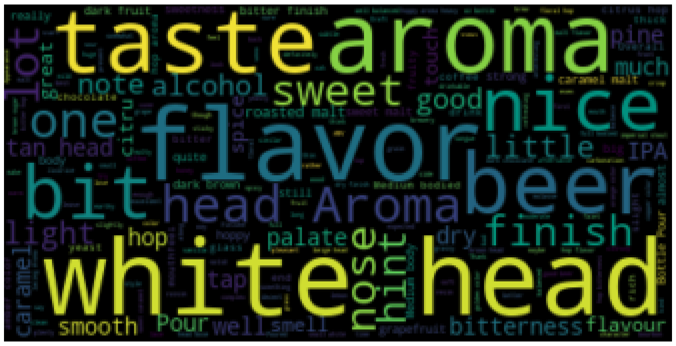

Figure 3: Most common words in review text shown by a Word Cloud chart.

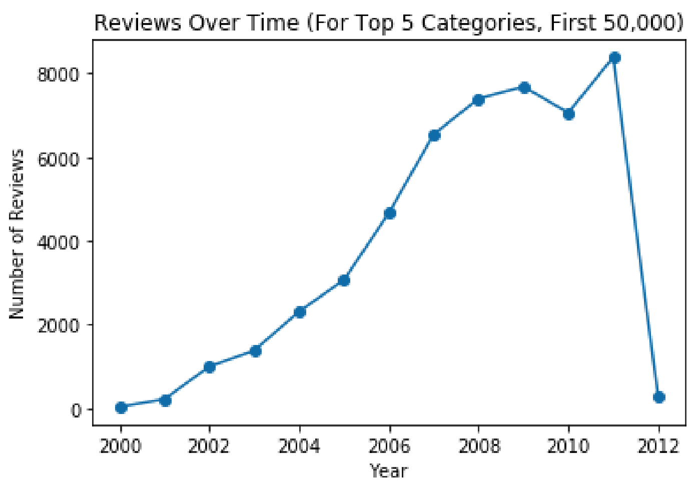

Figure 4: Frequency of Reviews Over Time in the RateBeers dataset. The review counts were binned by year based on their timestamp.

To observe how beer review frequency changed with respect to time, we visualized the frequency of reviews against the timestamp in the dataset. Since the timestamps were provided in seconds, we binned the data by year to visualize it. We observe a general increase in beer reviews in the RateBeers dataset with more recent time, potentially indicating the site's growth in popularity in recent years (Figure 3).

We next explored correlation structures and distributions for the quantitative features in our data: all of the rating categories and ABV (Figure 4).

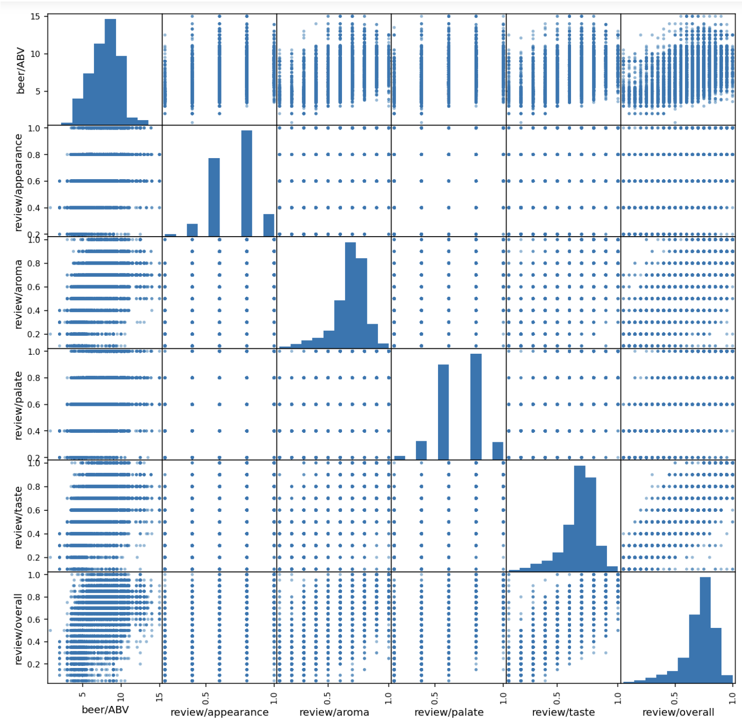

Figure 5. Scatter plots and bar plots for all of the quantitative features present in the RateBeer dataset.

Since the overall rating is computed as a function of all of the other ratings, we observe that the overall rating appears to be correlated with every other review feature in the dataset (Figure 4). Every quantitative feature also appears to follow a distribution that is roughly normal (Figure 4). Including correlated variables would result in multicollinearity when attempting a regression or classification task. Therefore, we excluded the overall review from downstream analysis when predicting beer styles from the dataset. Apart from overall review, it appears that there are no other significant correlation structures present among the quantitative variables in our dataset.

Since the beers are largely categorized into the top categories among the 89 unique categories, we elected to simplify our prediction task by focusing on the *top 5* beer categories, as this would be an easier task for a classifier to learn. Based on the features provided in the dataset, it appears that all of the quantitative features except for overall review could be predictive of beer style, and the review text could be as well. We also excluded time from downstream analysis since we assumed that the distribution of beer styles has not changed drastically in recent years. Furthermore, the larger scale of the timestamp data could affect the optimization of the other weights, resulting in decreased model accuracy. Therefore, for downstream analysis, we focused on models that use these features.

To understand how review text may be predictive in beer style prediction, we visualized the most common words in the RateBeer dataset (Figure 5, Figure 6). For visualizations, the stopwords were removed using the nltk.corpus default stop words set to visualize words that may be more predictive of beer style.

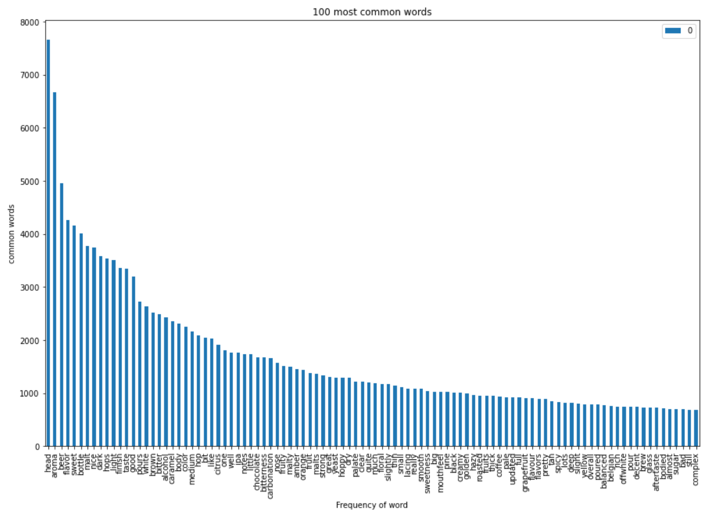

Figure 6: 100 most common words in the RateBeers review text.

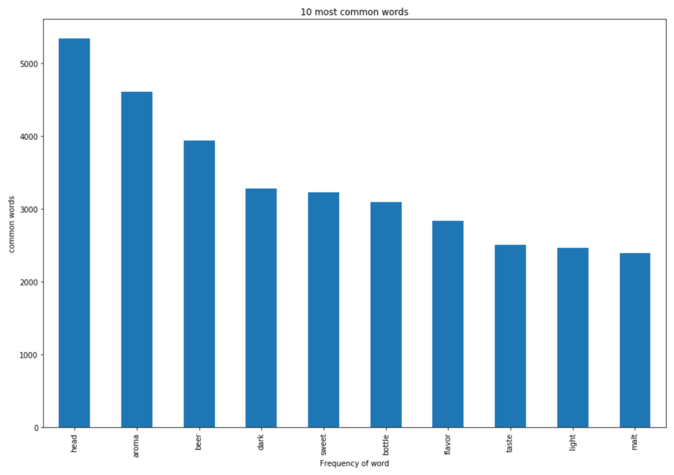

Figure 7: 10 most common words in the RateBeers review text.

The 10 most common words that we observed in the review text corpus were: head, aroma, beer, flavor, sweet, bottle, malt, nice, dark, and hops.

Predictive Task

For the predictive task of this dataset, we chose to perform categorical prediction on the beer style. This was a good task for RateBeer because it offers many features tailored to predictive tasks such as review text, the ABV for the beer and beer aspect ratings.

Our model was evaluated using the accuracy on the validation set, which is defined as $\frac{\#\ of\ correct\ class\ predictions}{\#\ of\ examples\ in\ validation\ set}$.

As a baseline model, we elected to use a model consisting of all of the quantitative features present in the dataset, excluding overall review rating. For the classifier, we first elected to use a logistic regression classifier as a baseline that could be further improved.

A lot of our models were centered on using the review text to perform the beer style category prediction. As a result, we performed preprocessing on the text so it could be used as part of the model. We converted all the review text to lowercase and removed any stop words using the stopwords library from nltk.corpus. All punctuation was also removed. This reduced redundancy in our word frequencies and also removed words that are not predictive from our bag-of-words and TF-IDF analyses.

For the quantitative features, since they were provided as ratings out of different maximum values, we first converted each rating fraction into a decimal value between 0 and 1. We used the ABV values by converting the values presented in the dataset to decimal values so that they were also values between 0 and 1. There were also NaN values present within the quantitative values present in our dataset. Since these values would not work in downstream analysis, we removed training examples with NaN values, reducing the number of training examples from 50,000 to 48, 370.

Model Description

For designing our model, we evaluated 4 different models and chose the one with the highest validation accuracy as our final model:

Baseline Model: Quantitative Features

For this model, we used all of the quantitative features available in the dataset, but excluded the timestamp information due to the limitations mentioned in the exploratory data analysis. For the baseline model, we implemented logistic regression using the *LogisticRegression()* function defined in the scikit-learn library, with a default regularization parameter of C = 0.1. The logistic regression model uses the quantity of a linear predictor to best predict which style a beer would belong to, and could find optimal weights associated with each quantitative feature being used to determine beer style. Using this model, we obtained an accuracy of 30.66%. To further improve the model, we attempted changing the regularization parameter of 0.01, 5, and 10, 15, and 20. Changing the regularization term to 15 resulted in the highest validation accuracy of 46.21%.. These changes would have addressed any potential overfitting or underfitting by the logistic regression model. Based on our results, it appears that the model was initially underfitting, as a higher regularization parameter in the library is indicative of less regularization being applied to the logistic regression model. We also experimented with balancing the class weights using the *class_weights* parameter. When doing so with the logistic regression model and the aforementioned regularization parameter values, we observe a subtle improvement in validation accuracy with a regularization parameter of C=20, resulting in an accuracy of 48.71%.

To attempt a different modeling approach under the assumption that the relationship between beer styles would be better modeled by a decision tree than a linear function, we implemented a random forest classifier using the *RandomForestClassifier()* function defined in the scikit-learn Python library. We used the default number of estimators (number of trees) of 100 to train the model. Using this approach, we achieved a validation accuracy of 29.58%. Adjusting the number of estimators (down for overfitting, up for underfitting) did not result in any substantial improvements to the model over the baseline accuracy obtained by logistic regression. Values of 10, 50, 150, and 200 were tested for the *n_estimators* parameter. Based on the low accuracies of both the logistic and random forest models, we concluded that the quantitative features may not be as predictive of the beer style as the review text itself, which impacted our future model selections.

TF-IDF Model

For this model, we used TF-IDF to vectorize the review text so they could be represented as features for the classifier model. TF-IDF is a strong approach because it selects all the relevant words to the reviews and documents which allows us to learn weights for words that are related to the beer categories we are trying to predict. For the TF-IDF model, we used sklearn's TF-IDF vectorizer library to create the vectorized versions of the review text and sklearn's Logistic Regression library for the model prediction. For the Logistic Regression, we use a C-value of 0.1 and balanced class weights. To preprocess text, we lowercase all words and remove stop words. This approach results in a validation accuracy of 84.56%. An issue with sklearn's TF-IDF implementation is that there was a 30,000 feature dimension. Such a large feature dimension may have been difficult to learn and have had low weights for irrelevant words.

Bag-of-words: Logistic Regression Model

In this model, we began by taking the top 5 beer categories by beer/style for the first 50,000 data samples from the RateBeer dataset. Utilizing this dataset we isolate review/text into a frequency of words list; while further filtering out punctuations, lower-casing all words, and removing all **stopwords**. By removing stopwords, common words that do not add much value to the classification are not considered when building the model. Therefore words such as "a", "and", "this", which are among the most common words in the bodies of the review text are removed. Based on this filtration and beer/style onto our 50,000 data sample, we did a 90%/10% training and validation split. Through logistic regression of an arbitrary C-value of 0.1 and balanced weight, we achieved a very decent accuracy of 86.58%.

Bag-of-words: Naive Bayes Model

Same as the Bag-of-words: Logistic Regression Model except the classification technique utilized is Naive Bayes classification. Naive Bayes is a probabilistic classification technique. Naive Bayes works by calculating the posterior probability, which in this case is the probability of a certain beer style given the features from the Bag of Words. It then selects the class with the highest probability given the features. An important item to note is that Naive Bayes assumes that features are mutually independent. This, however, is never true in text data where the presence of words Despite the common violation, Naive Bayes is still widely used for the task of text classification in machine learning. This is due to Naive Bayes being one of the faster classification algorithms that performs very well. There are many types of Naive Bayes classifiers that we could choose to implement and all work well for the task of text classification.

Gaussian Naive Bayes

First, we utilized Gaussian Naive Bayes. Gaussian Naive Bayes assumes that the frequency of words in the text are distributed according to a Gaussian distribution, or a normal distribution

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Figure 8: Gaussian Naive Bayes Likelihood of Features

To implement a Gaussian Naive Bayes, we used sklearn's GaussianNB. The hyperparameter of the model tuned was the var_smoothing parameter which is how much the variance is altered. The variance affects the likelihood as shown in Figure 4 with the denominator of the first term having the variance. This is done to smooth the curve of the model and also takes into account data points that are farther away from the mean of the distribution. To determine the best var_smoothing value, we iterated through multiple values. The best value we found was the natural log of 2 yielding an accuracy of 86.94%.

Bernoulli Naive Bayes

A Naive Bayes method that worked better on the Bag-of-Words features is the Bernoulli Naive Bayes which works well on binary features.

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

Figure 9: Decision formula for Bernoulli Naive Bayes. x_i is either 1 or 0.

Since it works well on binary features and our Bag-of-Words has binary features for the presence of the top words in the text, Bernoulli Naive Bayes is a great candidate for this text classification task. Once again, we utilized sklearn to implement Bernoulli Naive Bayes with BernoulliNB. By doing so we got an accuracy of 89.17%

Related Literature

The RateBeer dataset that we utilized originates from a publicly sourced review website (ratebeer.com) dedicated to compiling beer ratings, as introduced in [1] and [2], under subsection "Datasets".

To understand other techniques applied to this dataset, we looked at recent literature. In [1], McAuley et al use this RateBeer dataset to try to learn language models capable of identifying what sentences discuss each of the rated aspects (appearance, aroma, etc.). This is similar to our task in that they are trying to categorize review text. To perform this, they create a model that outputs the probability a sentence discusses a particular review aspect k while also given the aspect ratings associated with the review. This model takes in two parameter vectors, which differentiate words that discuss an aspect from words from words that discuss a certain sentiment. For our purposes, we could use a similar strategy in which we are trying to optimize separate weights related to the beer category and weights related to the review aspects. We did use the review rating aspects as part of our model, which is what this paper was trying to learn. However, our model performance based on these values was not as robust as our model performance using feature vectors derived from the review text.

Furthermore, we also drew inspiration from [2], in which McAuley et al. develop a recommender system with the purpose of predicting a user's *current tastes* and recommending products based on these results. McAuley et al. also account for how user *tastes* change-over-time. It is important to note that being able to factor in a user's taste based on the user's subjective experiences is a crucial aspect of the recommender system in-which McAuley et al. has built. McAuley et al. find that including a factor for *experience*, and modeling how user's taste preferences change substantially with respect to this experience significantly improves recommender systems performance. In our work, we are able to effectively predict beer style using review text, indicating that determining a user's beer style preferences is certainly possible to develop an effective recommender system. Our work does not attempt to make user-specific predictions, and therefore does not require the user-specific experience parameter. Future work could attempt to make style predictions based on user information, and could benefit from the inclusion of experience factors.

Results

The table below shows the validation accuracy of all the models we experimented with:

Model Type	Validation Accuracy
Baseline Model (Quantitative Features)	48.70%
TF-IDF	84.56%
Bag-of-words	86.58%
Naive Bayes: Gaussian	86.94%
Naive Bayes: Bernoulli	89.17%

Table 1. Summary of validation accuracies obtained from various models for beer style classification. Best performance was obtained using the Naive Bayes: Bernoulli model with a bag-of-words feature vector.

Conclusions

The Bernoulli Naive Bayes model is the best model of the ones we have chosen. Bag-of-words outperformed TF-IDF and our baseline model (quantitative features). Since Bag-of-words outperformed TF-IDF its indicative that word frequencies were better predictors than word relevance in this case. Gaussian Naive Bayes worked well and would work better if the word frequencies better fit a normal distribution. The increase in the smoothing parameter indicates that the data does not fit a normal distribution the best. Bernoulli Naive Bayes works the best with binary features and when given binary features the model did much better than the aforementioned model types. This model takes into account the content of the review text when determining the type of beer the reviewer is writing about. This can be useful for classifying a beer when the reviewer has not inputted the beer type of the item they have reviewed. Some future work would be to implement a deep learning solution that could yield better results than the Bernoulli Naive Bayes model we used.

References

[1] **Learning attitudes and attributes from multi-aspect reviews**
Julian McAuley, Jure Leskovec, Dan Jurafsky
International Conference on Data Mining (ICDM), 2012

[2] **From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews**
Julian McAuley, Jure Leskovec
WWW, 2013

[3] **Naive Bayes and Text Classification I - Introduction and Theory**
Sebastian Raschka
arXiv, 2017