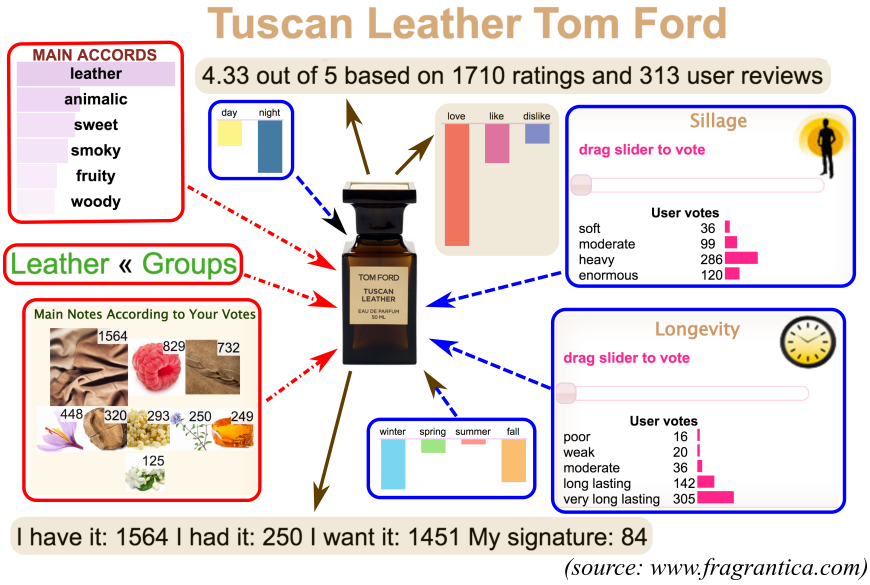
Predicting Perfume Rate and Popularity

Yao Li (yaoliphy@stanford.edu)

1. Introduction

The fragrance and perfume industry is experiencing a growth of approximately 7% to 8% annually during 2014 and 2020[1], and the size of the global fragrance and perfume market is as large as $40.1 billion in 2016[2]. The continuous growth and huge size of this market is due to the worldwide popularity of perfume use. For example, 52.54% U.S. households use or buy perfume, cologne, and toilet water in 2016[3]. In this work, we apply different Machine Learning (ML) techniques to analyzing this huge market and explore the possibility of predicting the rate and popularity of a perfume based on its various properties and features.

The ultimate goal of this work is to provide advice for both perfume buyers and manufactures. For perfume buyers, this work is supposed to help them choose perfumes that will help them smell on-trend (high rate, high popularity) or unique (high rate, low popularity). For perfume manufactures, we would like to provide advice regarding how to produce perfumes that will become next best sellers.



(source: www.fragrantica.com)

Figure 1: *An example of input data*

2. Dataset and Features

Get raw data and select features. Datasets are obtained through web scraping of www.fragrantica.com using Python library BeautifulSoup[4]. www.fragrantica.com is a perfume encyclopedia website, containing information of over 38,000 perfumes. Figure 1 shows an example of various information that can be found on this website for each perfume. The rate score and the number of ratings and user reviews represent the rate and popularity of the perfume. The website also displays user votes for longevity, sillage (the degree to which a perfume's scent lingers in the air), day/night, and seasons. Besides, we can also find main accords and main notes of a perfume. Notes are descriptors of scents that can be sensed upon the application of a perfume, and are very important and fundamental features of a perfume. In Table 1, we list all input and

target features selected in this work. The target features are rate and popularity. The input features include season, day/night, longevity, sillage, notes, and accords.

	Features/Variables	Description	Example (Tuscan Leather Tom Ford)
Input features	season	percentage of votes for four seasons	(45.0%, 12.1%, 4.4%, 38.5%)
	day/night	percentage of votes for "day" and "night"	(32.8%, 67.2%)
	longevity	percentage of votes for different levels of perfume longevity	(3.1%, 3.8%, 6.9%, 27.4%, 58.8%)
	sillage	percentage of votes for different levels of perfume sillage	(6.7%, 18.3%, 52.8%, 22.2%)
	notes	indicator vector of the perfume notes	(0 0 1 0 0 1 0 …)
	accords	indicator vector of the perfume accords	(1 0 0 0 1 0 0 …)
Target featurs	rate	rate score	4.33 out of 5
	popularity	# of ratings and user reviews	1710, 313

Table 1: *Selected input and target features*

Filter data and preprocess non-standard data. There is no meaning including extremely unpopular perfumes in our dataset. Therefore, we have removed these extremely unpopular perfumes from our dataset by applying filtering criteria. The filtering criteria include that the rate score cannot be none, user votes for season, day/night, longevity, and sillage cannot be zero, and main accords and notes cannot be none. After filtering, 22,857 perfumes are left. To apply ML techniques, we have also preprocessed non-standard data, as shown in the last column of Table 1. First, the user votes for season, day/night, longevity, and sillage have been converted to percentage of the total votes. Second, the notes and accords have been converted to indictor vectors. In the dataset, a complete list of all perfume notes includes approximately 700 notes, and only the 100 most common notes are selected to form a set of note tokens. The indicator vector of a perfume's notes describes whether each token note appears in this perfume. This is similar to the spam classification problem, in which we first choose a set of tokens and then find the indicator vector for each message or email. The indicator vector of accords is computed similarly after choosing the 30 most common accords among all 67 accords to form a token list. Third, before applying classification models, we have discretized the continuous-valued target features. For example, the perfumes with the 33% most ratings and user reviews are labeled as "popular" ("2"); the perfumes with the 33% least ratings and user reviews are labeled as "unpopular" ("0"); the rest perfumes are labeled as "median" ("1").

3. Method

We first apply six different classification models to predicting both rate and popularity of a perfume. Because classifications models do not work very well for the rate prediction, we then apply three different regression models to the prediction of rate. We have used the implementation of these models in scikit-learn[5] in Python.

Classification models. The first classifier we have tried is Support Vector Machines (SVM). We choose the kernel to be Gaussian radial basis function with parameter γ:

$$K(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|_2^2\right), \quad \gamma > 0$$

A small γ means two vectors far away from each other will still influence each other, leading to under-fitting. A large γ means that the support vector does not have widespread influence, leading to high bias and low variance, namely over-fitting. The second classifier we have applied is boosting. Gradient tree boosting has been chosen with decision stumps as weak learner. Using boosting, we have also compute the relative importance of different input features. The relative importance of input features is computed based on the number of times a feature is selected for splitting, weighted by squared improvement to the model. The third classifier we have tried is K nearest neighbors (K-NN). K-NN works by finding k training samples closest in distance to the new data point and then predicting the output from its k nearest neighbors. A very large value of k would lead to slow simulation, and we stop at using k equivalent 12. We have also applied Decision tree (DTs) classifier. A large value of maximum depth of the tree leads to high bias and low variance, and vice-versa. In addition to these four classifiers, we have also tried logistic regression and Naïve Bayes (NB) classification.

Regression models. Decision tree (DTs) regression, linear regression, and boosting regression have been applied to predicting rate. Similar to DTs classifier, the performance of DTs regression also depends on the maximum depth of the tree. For boosting regression, we use least squares loss function and 200 boosting stages to perform.

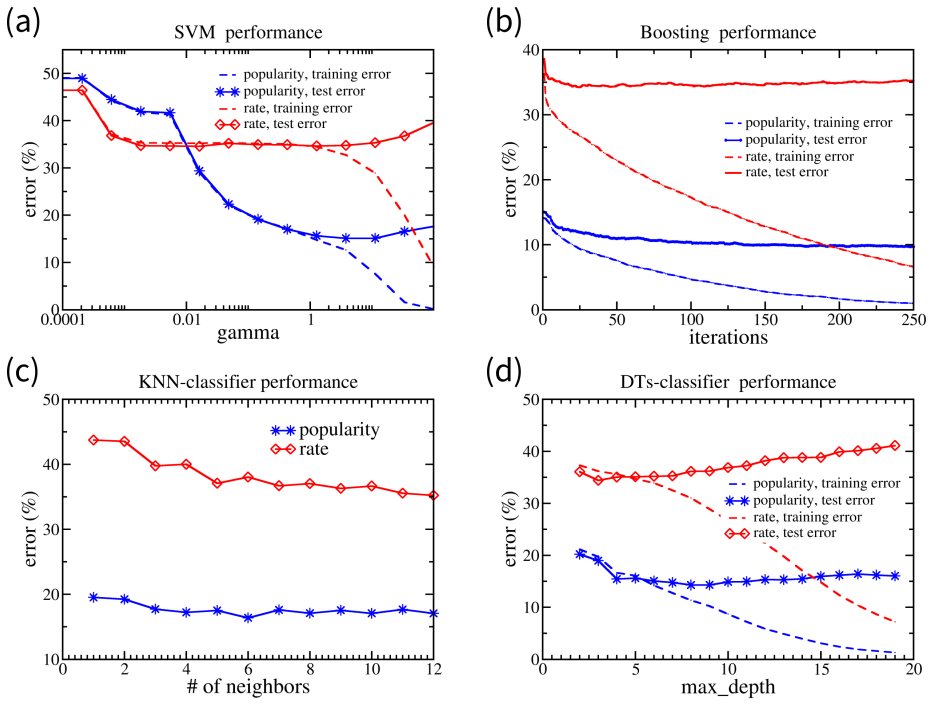


Figure 2: *Performance of different classifiers*

4. Results and Analysis

In Figure 2, we plot the performance of different classifiers for both popularity (blue curves) and rate (red curves). The solid curves are test errors and dashed curves are training errors. In Figure 2a, we plot the error by SVM as a function of the parameter γ. As shown in Figure 2a, a small

value of γ leads to under-fitting and a large value leads to over-fitting. After choosing an optimized value of γ, SVM predicts popularity with a test error of 15.1% and rate with an error of 34.2%. Figure 2b shows that boosting model leads to a smaller test error for popularity prediction (9.12%) and a test error of 32.8% for rate prediction. As shown in Figure 2c, when the number of nearest neighbors is chosen to be 12, the test error is 17.1% for popularity prediction and 34.8% for rate prediction. Figure 2d shows that DTs classifier is easy to be over-fitted if the maximum depth of the tree is large, and it predicts popularity with a test error of 14.2% and rate with an error of 34.4%. Comparison of these different classifiers has been plotted in Figure 3. Figure 3 also includes the test errors of logistic regression and Naïve Bayes. As shown in Figure 3, the error of popularity prediction is the smallest using boosting model. SVM, KNN, and DTs can also predict popularity with a small error of approximately 15%. However, the error generated by logistic regression and Naïve Bayes is much larger, approximately 40%. The error of Naïve Bayes is large because our features are not independent of each other given the class; therefore they do not satisfy the Naïve Bayes assumption.

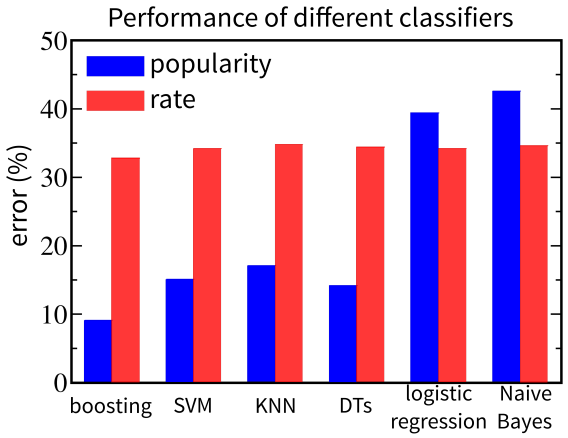


Figure 3: *Comparison of performance of different classifiers*

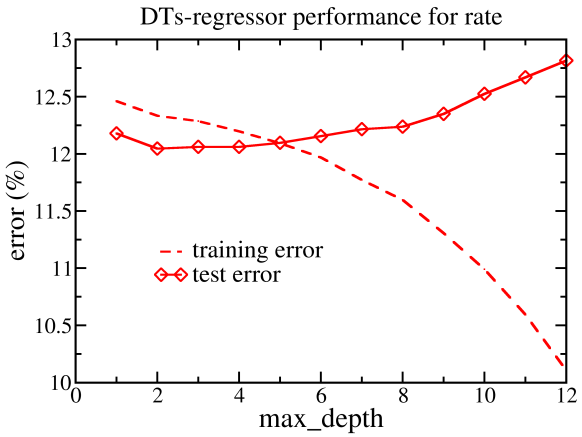


Figure 4: *Performance of DTs regression for rate prediction*

As shown in Figure 3, for the prediction of rate, all these classifiers generate an error of approximately 35%. To achieve a smaller error, we have applied different regression models to rate prediction. Figure 4 shows the performance of DTs regression. Similar to DTs classifier, a large value of maximum depth of the tree will result in over-fitting. As shown in Figure 4, for rate prediction, DTs regression generates an error of approximately 12.0%. We have also tried linear regression and boosting regression. After parameter optimization, linear regression gives a test error of 12.0% and boosting regression gives an error of 11.9%.

Through the comparison of different classification and regression models, we have shown that classification models such as boosting work the best for the prediction of perfume popularity and can give a test error as small as 9%, and regression models work better for the prediction of perfume rate and can lead to a test error of 12%.

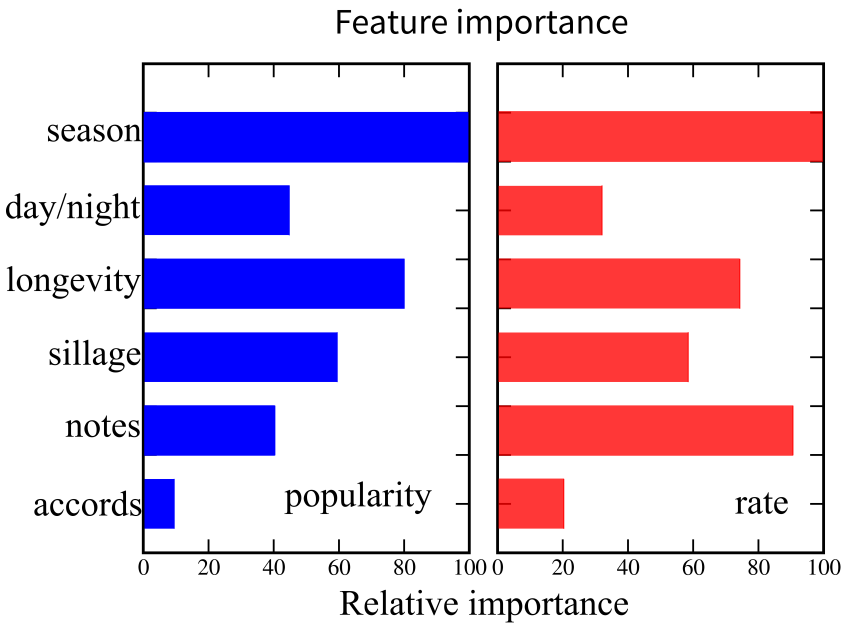


Figure 5: *Relative importance of different input featuers*

Figure 5 shows the relative importance of different input features we have selected in our models. The relative importance is computed using boosting algorithm. As shown in Figure 5, season is the most important input feature for both popularity and rate prediction. Longevity and sillage are also important for both. Day/night is less important. Perfume notes are very important for the prediction of rate, and less important when predicting popularity. Main accords are the least important feature for both.

5. Conclusions and Future Work

In summary, we have applied different ML techniques to predicting the rate and popularity of a perfume using its various features. We have shown that some classification models such as boosting can predict perfume popularity with a test error as small as 9% and regression models can be applied to predicting perfume rate with a test error of 12%. Furthermore, we show that some features such as season are more important than other features such as accords.

To continue this work, we would like to include the perfume ingredients as another input feature. This will help our work provide guidance for perfume manufactures regarding how to produce a popular perfume. Finally, we want to interpreter our ML results and try to provide customized advice. For example, we would like to predict the most popular perfumes for different seasons. Also, we would like to show what combination of perfume notes would have the highest rate.

References

1. http://www.futuremarketinsights.com/reports/global-fragrances-market
2. http://www.statista.com/statistics/259221
3. http://www.statista.com/statistics/276493
4. https://www.crummy.com/software/BeautifulSoup/bs4/doc/
5. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.