CIS 472
Yanting Liu
Kaiyu Song
3/19/18

Predicting the Rating of Chocolate Bar

Abstract

Chocolate is one of the most popular foods in the world. However, chocolates are not made equally, and they all have different rating. There are many features that can influence the rating of chocolates, such as the coca percent, producing company and producing location. In this paper, we will figure out the rating of chocolates based on those features. We evaluate three different kinds of classifier: Decision Tree, Random Forest, K-NN, and test the effects of different features on the rating of chocolate. After analyzing, the accuracy of three models are between 69% to 71%.

I. Introduction

There are millions of chocolate bar around the world, but some of them are liked by many people ,and some are not. We want to figure out which kinds of chocolate is more likable. We use the dataset from Kaggle, and it is narrowly focused on plain dark chocolate. This dataset contains information of over 1,700 chocolate bars, which includes their regional origin, percentage of cocoa, the variety of chocolate bean used, where the beans were grown, and their expert rating. This dataset also includes the information about when this rating recorded and its review date. There are 5 level of ratings for the chocolate. The 5 is elite, and 1 is unpleasant. According to the size of our dataset, we will use Decision Tree, its extension: Random Forest, and K-NN. In section II, we will specifically analyze the dataset and explain the methodology we used. In section III, we will analyze our results, and finally we will conduct our conclusion in section IV.

II. Methodology

A. Data exploration

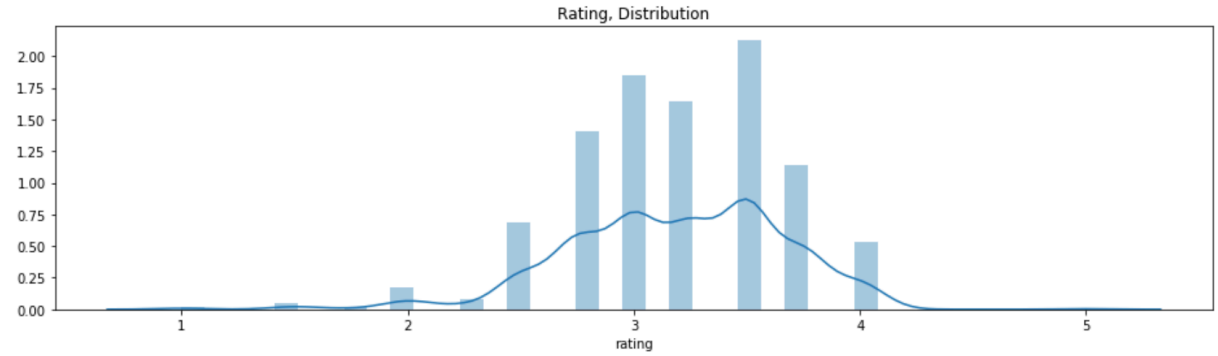
In the dataset, there are total 8 different features and 1 target label, and there are 1795 different examples. [1]
The features are:

1. Company (String): Name of the company make the bar.
2. Specific Bean Origin or Bar Name (String): The of the bar.
3. REF (Numeric): A value linked to when the review was entered in the database. Higher = more recent.
4. Review Date (Numeric): Date of publication of the review.
5. Cocoa Percent (String): Cocoa percentage (darkness) of the chocolate bar being reviewed.
6. Company Location (String): The location of company.
7. Bean Type (String): Cocoa percentage (darkness) of the chocolate bar being reviewed.
8. Broad Bean Origin (String): The broad geo-region of origin for the bean.

The original type of the features are:

Company	BarName	REF	Review Date	Cocoa Percent	Company Location	Rating	BeanType	Broad Bean Origin
objest	object	int64	int64	object	object	float64	object	object
No null	No null	No null	No null	No null	No null	No null	One null	One null

The distribution of our target are :



Source: Kaggle[2]

From the figure, we notice that most of the rating are fall in to 2.75 to 3.75.

B. Data Processing

Before we run the classifiers, we need to process our data so that it can fit our choosing function.

Firstly, we need to choose good features. REF and Review Date are highly correlated, so they are redundant features. Moreover, REF and Review Data are irrelevant features since the recorded date might be largely irrelevant for predicting rating of the chocolate bar. Besides, REF and Review Date are both very large number: Review Date is year which more than 1,000 and most samples' REF are also bigger than 100. Them have huge weight in the model, especially the K-NN. Because most of other value in K-NN data set is small, and if there is a value have more than 100 in it, it would almost decide the whole model. Therefore, we decide to do not use REF and Review Date in the final model.

Then, we process missing values. Since there are just two example have null values, we just drop those two examples and now we have 1793 examples.

After that, we transfer the datatype so that it can fit our choosing machine learning algorithms. As we can see from above table, 6 of the remaining 7 features are object, and they are all categorical variables. Categorical variables are convenience for people to understand but not machine learning algorithms. Moreover, many of our learning algorithm requests numerical input. Therefore, we need to convert categorical variables into numerical variables. We have tried two ways to do this.

- The first method is encoding every categorical variable to an ordinal one. For example, when convert the company location. We represent France as 1, U.S.A. as 2 and Fiji as 3. This methods does not make sense because we are assuming that Fiji is more similar to U.S.A than France.
- Since the first method does not make sense, we turn to our second method, which is encoding every categorical variable to an dummy one. We will use Company as an example. First, we count the number different kinds of company. And let each of them be unique feature of data. Then we walk through each sample again. And for each sample I will sign the company which produce it integer 1 and 0 to other companies. This can also be implemented using pandas.

Finally, we need to covert our target: rating. Since we are using classifier, our target must be integer. We also tried two ways to cover the data. The first one is that we multiply 100 to the rating, so it now convert to 100,200 and so on. However, by doing this our accuracy is rather low, which is about 20%.Then we use the second method. We simplify the problem by changing rating to stars, which means if the rating falls between 100 to 199, it now will all be 1.0 star. By doing this, we are now only have 5 classes, and it highly improve our accuracy. We also tries other logic to change rating to stars. For example, if the rating falls between 0 to 150, it should be 1.0 star and 2.0 star for rating between 150 to 250. However, the accuracy is much lower than our first method to convert the rating.

C. Classifiers

Since our dataset is small, we decided to use Decision Tree classifier, Random Forest classifier, and K-NN for this problem. Besides that, we use dummy classifier as our baseline.

- Baseline:
  We use dummy classifier with satisfied method as our baseline. DummyClassifier is a classifier that makes predictions using simple rules and "stratified": generates predictions by respecting the training set's class distribution[5].

- Decision Tree:

Same with we learn from class. Each time using features with best gain to build node and leaves to decide which rating it should be.

- Random Forest:

Random forest is the extension of Decision Tree. Its theory is to build multiple decision trees to evaluate sample and use the mean prediction of the individual trees as the output result.

- K-Nearest Neighbors:

K-nearest neighbors is the one from class. First, it need to normalize the dataset. Then it will calculate the distance from each test sample with training data and get most close k sample. Finally, it will let these k samples to void to get the class of test data.

III. Experiments

In the following sections, we detail the experiments we did including learning and various tuning parameters and analyze the results we found.

A. Procedure

We split our data into two with 80% of the content being treated as training data while the other 20% of the data as testing data. We use precision, recall and accuracy to evaluate our results.

Firstly, we run all the algorithms with their default values:
Our Baseline Accuracy: 53.76%

Decision Tree(with min_samples_leaf=1, which represents the minimum number of samples required to be at a leaf node, as default value):

	precision	recall	f1-score	Accuracy
Test 1	0.64	0.68	0.65	68.245
Test 2	0.62	0.68	0.64	67.967
Test 3	0.63	0.69	0.65	68.802
Average	0.63	0.68	0.65	68.338

Random Forest(with 10 trees in the forest as default):

	precision	recall	f1-score	Accuracy
Test1	0.61	0.68	0.63	67.688
Test2	0.61	0.69	0.63	69.359
Test3	0.61	0.68	0.63	67.688
Average	0.61	0.68	0.63	68.245

K-NN(with K = 5 as default):

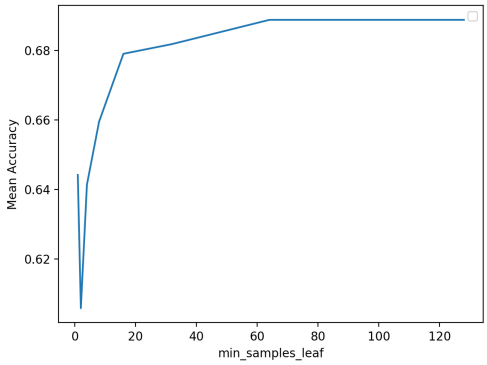
	precision	recall	f1-score	Accuracy
Test1	0.61	0.68	0.63	68.245
Test2	0.61	0.68	0.63	68.245
Test3	0.61	0.68	0.63	68.245
Average	0.61	0.68	0.63	68.245

With default values for each classifier, their performance are similar. All of them performs better than our baseline, so we are going to tune the hyperparameters of all three models to get better result.

B. Cross-validation: Hyperparameter tuning

We use 10-fold cross validation to tuning the hyperparameters. In cross-validation, we break our training data up into 10 equally-sized partitions and the train a learning algorithm on 9 of them and test it on the remaining data. We do this 10 times and finally average the performance over all then parts[3]. Cross-validation also help us avoiding overfitting.

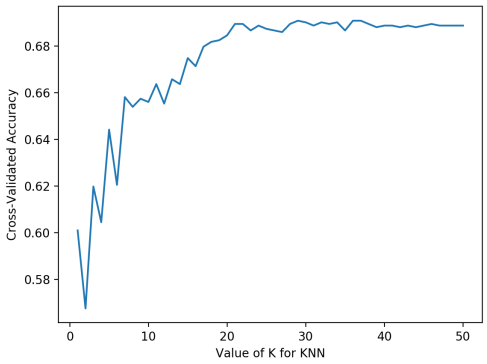
- Decision trees
  For Decision tree, we need to find the best number for min_samples_leaf, which represents the minimum number of samples required to be at a leaf node. Although there are numbers of parameters that we can tune, here we only tune the min_samples_leaf parameter. The candidate values of min_samples_leaf are [1,2,4,8,16,32,64,128]. Here is the figure to represent the result.



The best value for min_samples_leaf is 64 and the result on testing data is

precision	Recall	f1-score	Accuracy
0.51	0.72	0.60	71.588

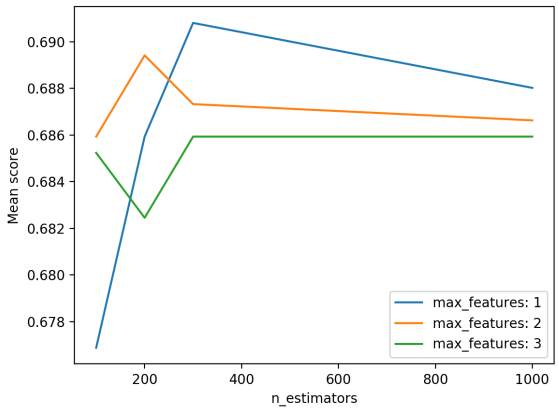
- K-Nearest-Neighbors
  For KNN, we mainly tune the parameter K, the number of nearest neighbors. The other parameter that we can tune is the distance metric function. Since 10-fold cross validation takes too long to run, we use 5-fold cross validation for KNN and Random Forest. The candidate value for K are from 1 to 50.



As the figure shows, K should be 29 and the result on testing data is

precision	Recall	f1-score	Accuracy
0.59	0.71	0.60	71.0306

- Random Forest
  For Random Forest, the hyperparameters we tuned are max_features and n_estimators.
  n_estimators represents the number of trees in the forest.
  max_features when it is integer, it represent the max number of features to consider when looking for the best split.[5]
  The candidate value for n_estimators are 100,200,300,1000 and 1,2,3 for max_features.



The mean accuracy is optimal when n_estimator = 300 and max_features = 1, and following is the result on test set.

precision	Recall	f1-score	Accuracy
0.64	0.71	0.65	70.752

C. Result Analysis

After hyperpramerter tuning, all the three classifier works better than before.
The accuracy is around 71% for Decision tree, but the precision for it is lower than other two model. The accuracy for KNN is around 70% and the accuracy for Random Forest is also 70%.
The performance of the three model are similar and their accuracy are all around 70%. One reason for that is our relatively small dataset and the distribution of it. Most of the data falls into 2.75 to 3.75. After converting our rating to stars, most of data falls into 3.0 stars. There is little example falls into 1.0 and 5.0 stars, about 1%.

IV.Conclusion

In this work we apply three learning algorithms, which are Decision Tree, Random Forest, and K-NN, on the chocolate bar rating dataset. We compare various ways to clean our dataset, and tuning the hyperparameter of each classifier. Due to the relatively small size of our dataset, our accuracy for three models are all around 70%.
There are much more work we can do. In the future, we will use other methods to evaluate our models, such as ROC. Also, we will try other ways to clean our datasets and we will tune more hyperparameters. If there is enough time, we will also explore other learning algorithms.

Reference

[1] "Chocolate Bar Ratings : Expert rating of over 1700 chocolate bars", Tatman, Rachael, Kaggle, https://www.kaggle.com/rtatman/chocolate-bar-ratings, 2017

[2] "Chocolate Bar Ratings: Python EDA& DataViz", Samoylov, Alexey, Kaggle, https://www.kaggle.com/tibhar940/chocolate-bar-ratings-python-eda-dataviz, 2018

[3] "A Course in Machine Learning" Daumé III, Hal, http://ciml.info

[4] "Using categorical data in machine learning with python" Hada, Yonatan, YellowRoad, https://blog.myyellowroad.com/using-categorical-data-in-machine-learning-with-python-from-dummy-variables-to-deep-category-66041f734512, 2017

[5] Scikitlearn Documentation
http://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html