Machine learning on homebred beer using R

Junjie Cai

13 June 2018

Abstract

In this paper, machine learning and statistical analysis will be employed on a classification problem about homebred beer recipes, in particular, predicting the color of beers. A variety of models and algorithms, including *logistic regression, decision tree*, and *KNN algorithm*, are used to analyse the dataset. The way to assess the classification model is calculating the *test error rate*, which determines how much the prediction is right on test dataset. As a result, KNN function works best with the lowest error rate at **34.00%** when **K=20**. The result is not very well, due to the accuracy of data source and(i.e. data is uploaded bu users and they can enter whatever they want). Thus, future work will be focused on improvement of accuracy by applying other useful method, such as Support Vector Machines and Random Forests.

Introduction

Machine learning and statistical analysis have been widely studied and applied in a variety of fields in recent years, especially in business area. Companies get benefits from these techniques, thus wining the competition. Therefore, it is common and necessary to analyse features of the product by employing data mining and statistical methods. The types of problems that machine learning algorithms are applied typically consist of *classification* problem, which predict two or more discrete outcomes, and *regression* problems, which predict continues values.

In this project, a set of supervised learning methods are employed in a classification problem related to home-brewed beer recipes. The data comes from Kaggle.com website, and contain a set of variables referring to features in the fermentation of beers, thus it is a good data source to find relationship between these features. Each observation is classified as either *dark* beer or *light* beer. The goal of this project is going to predict the colour of beers.

Data

Beer recipes dataset is hosted in Kaggle.com website mentioned above, which is a website running data science competitions. The dataset is original sourced from the Brewer's Friend website, which is a place for users to share their homebrew beers. Users are allowed to upload their homebrew beer recipes by entering values for each feature. It is possible that users just enter whatever they want.

Data pre-process

This original dataset contains 73,861 observations of 22 variables. However, it is necessary to pre process the initial dataset, as most of 22 variables are either not applicable or have no contribution on the goal of the project (e.g., **URL** just refers to the link of the item in Brewer's Friend website).

```r
library(MASS)
library(tree)
library(knitr)
library(ggplot2)
library(boot)
library(class)
library(reshape2)
rawRecipe = read.csv("recipeData.csv", na.strings = "N/A")
recipes = rawRecipe[,c(4,7:11)]
recipes = na.omit(recipes)
# we delete some outliers in data set
outlier = which(recipes$OG>1.5 | recipes$FG>1.5 | recipes$ABV>15 |recipes$IBU>100 | recipes$Color>40)
recipes = recipes[-outlier,]
```

For pre-processing, 5 most useful variables are selected from the original dataset, and hence make up a new dataset with 63,949 observations of 6 columns, as shown in **Table 1**.

Table 1 – 5 most useful Variables

```r
recipes.X = recipes[,-1] # extract beeer features except the Style
```

ID	Variables	Description
1	OG	original gravity, specific gravity measured before fermentation
2	FG	final gravity, specific gravity measured at the completion of fermentation
3	ABV	alcohol concentration represented by Volume
4	IBU	international bitterness unit, a numeric measure of bitterness of the beer
5	Color	numeric value of the beer color

Further, in order to obtain a good result, the quantitative variable **Color** is transformed to the qualitative variable **Dark** with two values (**yes** or **no**). Each observation with the value of **color** greater than 25 is classified into **yes** label, otherwise **no**, and then we remove **Color** variable, as shown in **Table 2**.

```r
recipes.X$Dark = rep("no",nrow(recipes.X))
recipes.X$Dark[recipes.X$Color>25]="yes"
recipes.X$Dark = factor(recipes.X$Dark)
recipes.X$Color = NULL # remove Color
```

Table 2 – Final variables with a qualititative variable *Dark*

ID	Variables	Types
1	OG	numeric
2	FG	numeric
3	ABV	numeric
4	IBU	numeric
5	Dark	factor

After that, we extract the same numbers of items with **no** label as **yes** label, which benefits to reduce variance. Finally, we obtain the dataset which will be analyzed in this project.

```r
# random choose the same rows of observations from "no" class
no.rows= which(recipes.X$Dark=="no")
set.seed(123)
random.rows= sample(no.rows, length(no.rows)-length(which(recipes.X$Dark=="yes")))
recipes.X = recipes.X[-random.rows,]
# divide training and test set
set.seed(12)
train.rows=sample(1:nrow(recipes.X),0.9*nrow(recipes.X))
recipes.X.train = recipes.X[train.rows,]
recipes.X.test = recipes.X[-train.rows,]
```

Feature analysis

Here we are going to roughly look at the relationship between 4 features and **Dark** variable, as shown in **Figure 1**. **Figure 1 – boxplot of variables and *Dark***

Methods

Analysis methods

In this project, we employ various classification methods to analyze the dataset and try to predict the color of the beer. Firstly, 90% of the dataset is divided into training subset and the others are test subset. Secondly, we apply classification algorithms to build the regression models on our training dataset. After that, the error rate on test dataset (i.e. Test Error Rate) is calculated in order to assess the accuracy of the model. Finally, all test error rates are compared together to find the optimal model for our dataset. By the way, a method called **backforward stepwise selection** is used to find optimal predictors in each regression model.

Models and algorithm

Random guessing

We can simply guess whether the beer is dark or not, according to the percentage of the value of **Dark** in the whole dataset. Since there are same numbers of observations with **yes** and **no** labels, the test error rate is just **50%**.

Logistic regression

In this project, logistic regression is used to predict the *probability* that the response (**Dark**) belongs to a particular category (**yes** or **no**). Therefore, the response of the logistic regression belongs to *[0,1]*, so that logistic regression is a good way to reduce the range of response, hence reduce the errors. Another advantage of the logistic regression is that it is easy to interpret the relationship between the response and variables.

We build the model of logistic regression and then predict the probability of being the dark beer. Firstly, we use all features as predictors of the model.

Table 3 – Coefficients of the logistic regression model

```r
m.log = glm(Dark~., data = recipes.X.train, family = binomial)
gl.probs = predict(m.log,type = "response", newdata = recipes.X.test)
gl.pred=rep ("no" ,nrow(recipes.X.test))
gl.pred[gl.probs > 0.5]="yes"
#mean(gl.pred!= recipes.X.test$Dark)
kable(summary(m.log)$coefficients)
```

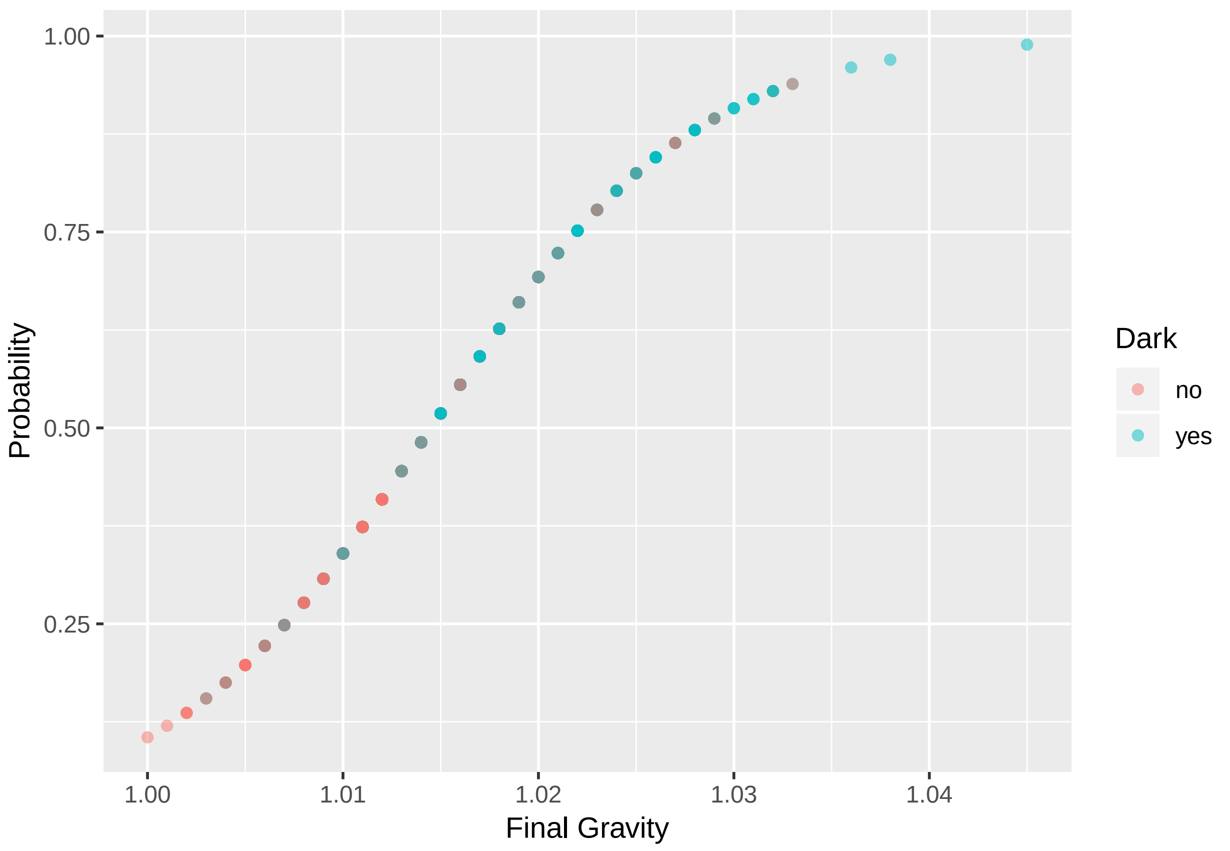
	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-150.2234183	4.9178327	-30.5466714	0.0000000
OG	-6.9263468	25.1766309	-0.2751102	0.7832316
FG	155.0163294	25.4997961	6.0791204	0.0000000
ABV	0.0450538	0.1910081	0.2358736	0.8135308
IBU	0.0008346	0.0009008	0.9264778	0.3541977

The coefficients of the model is shown as **Table 3**. We could see that P values of **OG**, **IBU** and **ABV** are pretty large, which indicates that these three variables are not statistically significant with the response, while the P value of "FG" is small enough to be a good predictor. The positive coefficient suggests that a higher final gravity more likely results in a dark beer. The test error rate of the model using all predictors is **37.09%**, which is better than random guessing but not very well. Therefore, I want to find the best features to improve the accuracy of the model.

Since there are only four features in the dataset, therefore it is easy to find optimal predictors using best subset selection. Here, I use **backforward stepwise selection**, and finally obtain the optimal predictor is **FG** with the test error rate **36.88%**.

Figure 2 – The logistic regression model with the predictor *FG*

```
m.log = glm(Dark~FG, data = recipes.X.train, family = binomial)
gl.probs = predict(m.log,type = "response", newdata = recipes.X.test)
gl.pred=rep ("no" ,nrow(recipes.X.test))
gl.pred[gl.probs > 0.5]="yes"
#table(gl.pred,recipes.X.test$Dark)
#mean(gl.pred!= recipes.X.test$Dark)
# plot the model with data points.
qplot(recipes.X.test$FG, gl.probs, data = recipes.X.test, colour= Dark, alpha=I(0.5), xlab = "Final Gra
```



The **Figure 2** shows how final gravity affect the probability that a beer is dark. We could see that observations with higher final gravity have a higher probability of being a dark beer.

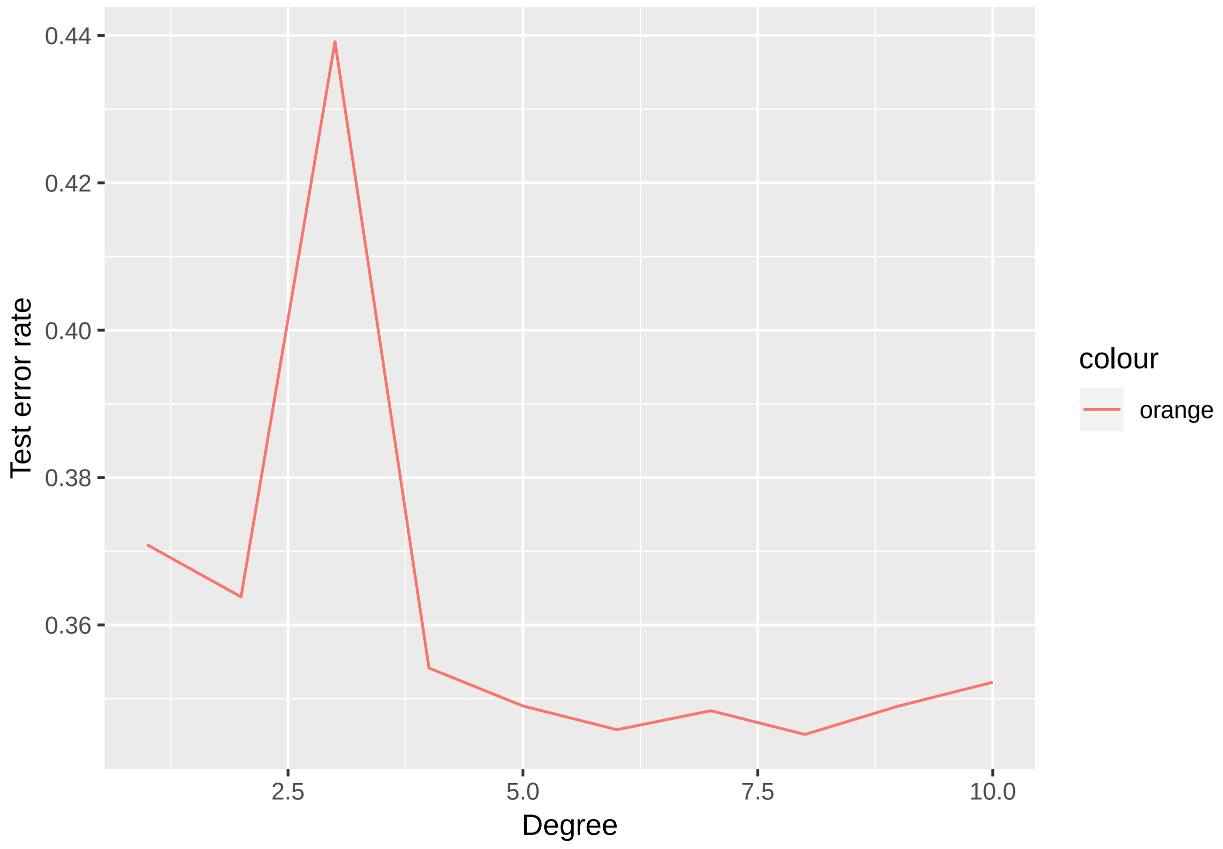
Logistic regression with polynomials regression

Polynomials regression is an easy way to extend the model by adding extra predictors, which are raised to a power of the original predictors. This method benefits to fit the data, thus reduce the error rate.

Now we are going to use all predictors with their non-linear transformations as our predictors to improve the accuracy of our logistic regression. The best polynomial degree is shown in **Figure 3**.

Figure 3 – Logistic regression with polynomials

```
test.error = rep(0,10)
for (i in 1:10) {
    m.log.poly = glm(Dark~  poly(FG,i)+poly(OG,i)+poly(ABV,i)+poly(IBU,i), data = recipes.X.train, famil
    gl.probs = predict(m.log.poly,type = "response", newdata = recipes.X.test)
    gl.pred=rep ("no",nrow(recipes.X.test))
    gl.pred[gl.probs > 0.5]="yes"
    test.error[i]=mean(gl.pred!= recipes.X.test$Dark)
}
#summary(m.log.poly)
qplot(1:10, test.error, xlab = "Degree", ylab = "Test error rate", geom = "line", colour="orange")
```



```
#test.error
```

The plot shows the best polynomial degree is 8, at which the test error rate has a significant decrease at **34.51%**. The result is not very good but is much better than the logistic model without polynomials and random guessing. In conclusion, the logistic regression using all predictors with their polynomials degree 8 as predictors has the lowest test error rate, which is **34.51%**.

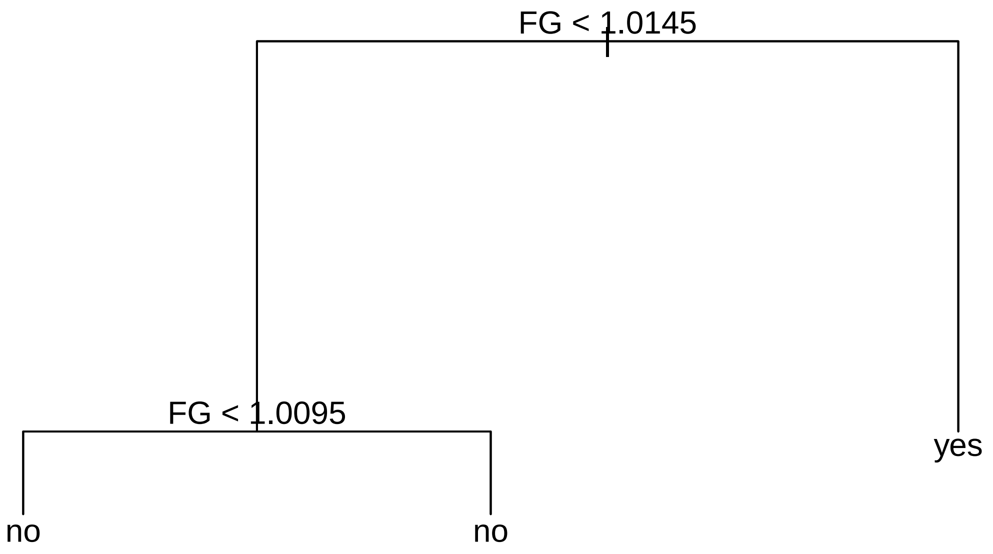
Decision tree

Tree-based methods can be used to solve both regression and classification problems, depending on what types of responses (i.e. *quantitative* or *qualitative*). For classification tree, each observation is classified into *the most common occuring* class in that region where it belongs to.

One obvious advantage is that tree-based methods are easy to explain to people. Further, trees can easily handle qualitative predictors without the need to create dummy variables. However, tree does not have a

same level of accuracy as other classification methods. The model of decision tree is shown in **Figure 4**.
Figure 4 – Decison Tree

```
m.tree = tree(Dark~., recipes.X.train)
#summary(m.tree)
plot(m.tree)
text(m.tree, pretty = 0)
```



```
# accuracy of decision tree
#p=predict(m.tree, recipes.X.test, type = "class")
#sum(p!=recipes.X.test$Dark)/length(p)
#p=predict(m.tree, recipes.X.train, type = "class")
#sum(p!=recipes.X.train$Dark)/length(p)
```

The result shows that train error rate is **35.97%** and test error rate is **37.09%**. The plot can be interpreted that data items with the final gravity greater than 1.0145 is predicted as dark beer, otherwise are light beer. Hence. the final gravity is the only feature considered to classify the beers.

K-nearest neighbor algorithm

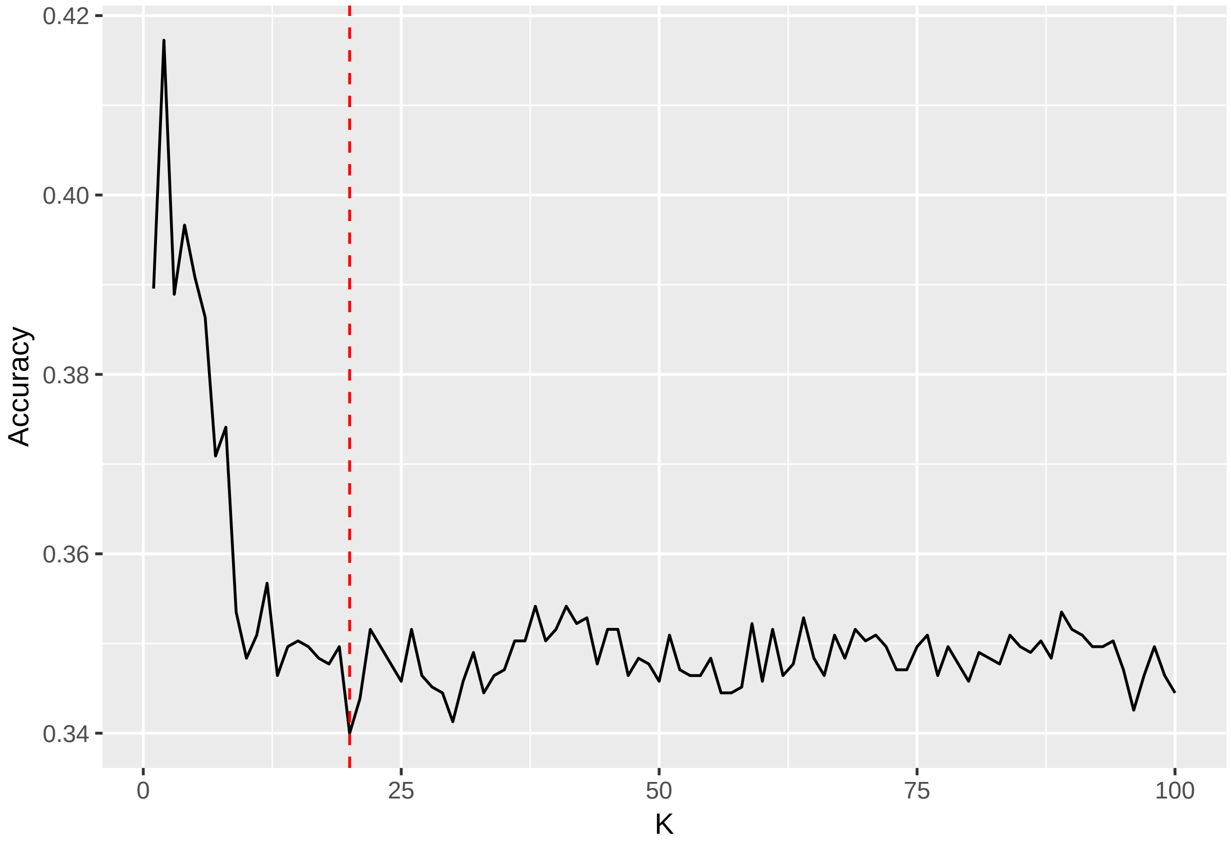
K-nearest neighbor algorithm is a common and useful way to to classification. simply to say, for each observation, the algorithm considers K nearest neighbors, and directly classify this observation into the most class, and then repeat this step. Therefore, this method does not build a particular model explain the relationship between variables and response.A large K will result in a huge amount of computing process, which consumes more memory and computing resources.

Figure 5 – Test error rate of KNN function

```
# standardized dataset
recipes.X.std.train= data.frame(scale(recipes.X.train[,-5]),Dark =recipes.X.train$Dark)
recipes.X.std.test= data.frame(scale(recipes.X.test[,-5]),Dark =recipes.X.test$Dark)

accuracy <- rep(0,100)
knn.errors = rep(0,100)
for (i in 1:100) {
    predicted.color = knn(recipes.X.std.train[,-5], recipes.X.std.test[,-5], recipes.X.std.train$Dark, 
    accuracy[i]=sum(recipes.X.std.test$Dark==predicted.color)/length(predicted.color)
    knn.errors[i]=1-accuracy[i]
```

```
}
#qplot(1:100,accuracy,geom="line", xlab="K", ylab="Accuracy", main = "Accuracy of KNN 2")+geom_vline(ae
qplot(1:100,knn.errors,geom="line", xlab="K", ylab="Accuracy")+geom_vline(aes(xintercept =20), linetype
```



The **Figure 5** shows the accuracy of KNN rapidly increases when K increases up to 20. After that, the test error rate fluctuates around at 35.00%. The lowest test error rate is only **34.00%**, which is the lowest one among previous results.

Results and evaluation

The test error rates of each regression method are shown in **Table 4**.

Table 4 – Comparison of test error rate of regression methods

```
Method.name = c("Random guessing","Logistic regression", "Logistic regression with polynomials", "Decis
Predictors = c("NA", "FG","all predictors with their polynomials","FG","NA")
Test.error.rate = c("50%","37.09", "34.51%", "37.09%","34.00%" )
comparison = data.frame(Method.name,Predictors,Test.error.rate)
kable(comparison)
```

Method.name	Predictors	Test.error.rate
Random guessing	NA	50%
Logistic regression	FG	37.09
Logistic regression with polynomials	all predictors with their polynomials	34.51%
Decision tree	FG	37.09%
KNN	NA	34.00%

It is clear that for the problem of determining whether a given observation is dark or not, the KNN function with the $\mathbf{K = 20}$ works best, and the test error rate is around **34.00%**, which seems acceptable.

Conclusion

In this project, we evaluated a variety of classification models on *homebrew beers* dataset hosted on Kaggle website, and tried to find the features which will affect the color of beers. The KNN function is the best method to predict the color, since it has the lowest classification error rate (**34.00%**) when **K=20**. The result is not very well but acceptable, because the original data is uploaded by users on Brewer's Friend, and they can enter whatever they want about beer features. In other words, employed variables are not good enough to predict the color of beers. Future work will be focused on improving accuracy by employing more variables or using other classification methods, such as Support Vector Machines and Random Forests.