Wine Cultivar Flavor Profile Analysis

Loren Grooms

05/04/2021

Introduction

Wine is a commodity that has been produced and enjoyed by humans across the globe for thousands of years. It has proven to be far more than a simple inebriant and is regularly employed in culinary, health, and religious contexts. A number of constituents play into the flavor profile of wine, from magnesium levels to protein content and beyond. Using data provided by UCI, this project aims to analyze these variables in wines made from three Italian grape varietals grown in the same region. Our ultimate goal will be to help a consumer predict the levels of several influencing variables in these wines based on one main factor: alcohol content (AC). This was chosen because AC is almost always printed on the wine's label and is visible to the consumer. Looking at each constituent individually, we will use AC to identify several strong predictors of each grape varietal. Then we will form a predictive model to visualize the potential levels of each constituent predicted by its AC. Finally, we will combine the results to form a simplified table of the relevant constituents for consumer reference.

Method

1 - Data Inspection & Cleaning

We will begin our analysis by locating the Wine Data Set on the UCI Machine Learning Repository. The website identifies this data as chemical analysis results from 178 wines made with 3 grape varietals, providing quantities of the following constituents:

1) Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavonoids
8) Nonflavonoid phenols
9) Proanthocyanins
10) Color intensity
11) Hue
12) OD280/OD315 of diluted wines (Protein Concentration)
13) Proline

We'll now download the dataset using R then extract, clean, and label the data.

```
# Download wine dataset from UCI website.
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
# url <- "https://raw.githubusercontent.com/diero42/WineCultivars-Project/main/wine.data"

# Read the table into a list, flatten the list, convert data to characters, and
# split each row along the commas.
wine <- strsplit(as.character(unlist(read.table(url))),split=",")

# Create a dataframe with the names of each column from the data source website.
# Insert zeroes as placeholders.
wineDat <- data.frame("Grape" = 0, "Alcohol" = 0, "MalicAcid" = 0, "Ash" = 0,
                      "AlcalinityOfAsh" = 0, "Magnesium" = 0, "TotalPhenols" = 0,
                      "Flavonoids" = 0, "NonflavonoidPhenols" = 0, "Proanthocyanins" = 0,
                      "ColorIntensity" = 0, "Hue" = 0, "ProteinConcentration" = 0,
                      "Proline" = 0)

# Insert each row of the list into the dataframe.
for(i in 1:length(wine)){
    wineDat <- rbind(wineDat,wine[[i]])
}

# Remove the row of zeroes.
# Renumber the rows.
# Convert all values to numeric using sapply.
wineDat <- wineDat[-1,]
row.names(wineDat) <- 1:nrow(wineDat)
wineDat <- as.data.frame(sapply(wineDat, as.numeric))
```

```
##   Grape Alcohol MalicAcid  Ash AlcalinityOfAsh Magnesium TotalPhenols
## 1     1   14.23      1.71 2.43            15.6       127         2.80
## 2     1   13.20      1.78 2.14            11.2       100         2.65
## 3     1   13.16      2.36 2.67            18.6       101         2.80
## 4     1   14.37      1.95 2.50            16.8       113         3.85
## 5     1   13.24      2.59 2.87            21.0       118         2.80
## 6     1   14.20      1.76 2.45            15.2       112         3.27
##   Flavonoids NonflavonoidPhenols Proanthocyanins ColorIntensity  Hue
## 1       3.06                0.28            2.29           5.64 1.04
## 2       2.76                0.26            1.28           4.38 1.05
## 3       3.24                0.30            2.81           5.68 1.03
## 4       3.49                0.24            2.18           7.80 0.86
## 5       2.69                0.39            1.82           4.32 1.04
## 6       3.39                0.34            1.97           6.75 1.05
##   ProteinConcentration Proline
## 1                 3.92    1065
## 2                 3.40    1050
## 3                 3.17    1185
## 4                 3.45    1480
## 5                 2.93     735
## 6                 2.85    1450
```

```
## Rows:  178
```

```
## Columns:  14
```

Our data is now readable and easily usable within R.

2 - Data Preparation

We will be using the **k-nearest neighbors (kNN) algorithm** to explore our data further, so we must **normalize** the data so it is usable in the algorithm. We will normalize our data using the following equation:

$$N_{norm} = \frac{n - n_{min}}{n_{max} - n_{min}}$$

We will then separate our data into several sets:

Test - Used within the algorithm to test the effectiveness of its components.
Training - Our core data, used as the input to our kNN algorithm.
True Classifications (CL) - The set of true values corresponding to the algorithm's predictions.

We want our Test set to be a reasonable fraction of the greater dataset so we can use it to confirm our results reliably without removing too much of the core training data. Beyond that, this is somewhat of an arbitrary decision so we will be making our Test set 33% of the overall data and Training 66%. The algorithm will use the CL to validate the final results.

```
# Set a standard normalization function.
norm <- function(x){
    (x -min(x))/(max(x)-min(x))
}

# Normalize the data by applying the norm function to all columns.
normWine <- as.data.frame(lapply(wineDat[,2:14], norm))

# 66% and 33% have been selected as the respective sizes of the training and
# test sets.
# This is an arbitrary decision influenced by observed custom.
# Calculate 66% of the normalized data entries.
# Sample the normalized data to select a test index.
# Divide normalized data along test index.
train_size <- floor(0.66 * nrow(normWine))
test_index <- sample(seq_len(nrow(normWine)), size = train_size)
train_set <- normWine[test_index,]
test_set <- normWine[-test_index,]

# Isolate the column of the value to be predicted (grape cultivar).
trainGrape <- wineDat[test_index,1]
testGrape <- wineDat[-test_index,1]
```

3 - Data Exploration

We can now explore our dataset by running our kNN analysis with our normalized data. We are attempting to identify the constituents in our wine that correlate most strongly with grape varietal and alcohol content. To accomplish this, we will be running this algorithm 12 separate times, each time pairing AC with one other constituent. The algorithm will attempt to use the levels of these two variables to predict which grape varietal was used to make the wine, and will then return the accuracy of each pair's prediction.

```
# Apply the knn function from the class package to the train and test sets
# in order to predict the grape type.
# 13 regions will be generated as this is roughly the square root of the
# total number of observations (178).
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
kResults <- " Accuracy:\n"
for (x in 2:13){
    kWine <- knn(train=train_set[,c(1,x)],test=test_set[,c(1,x)],cl=trainGrape,k=13)
    tab <- table(kWine,testGrape)
    kResults <- c(kResults,paste(names(train_set)[x],"=",accuracy(tab),"%\n"))
}
```

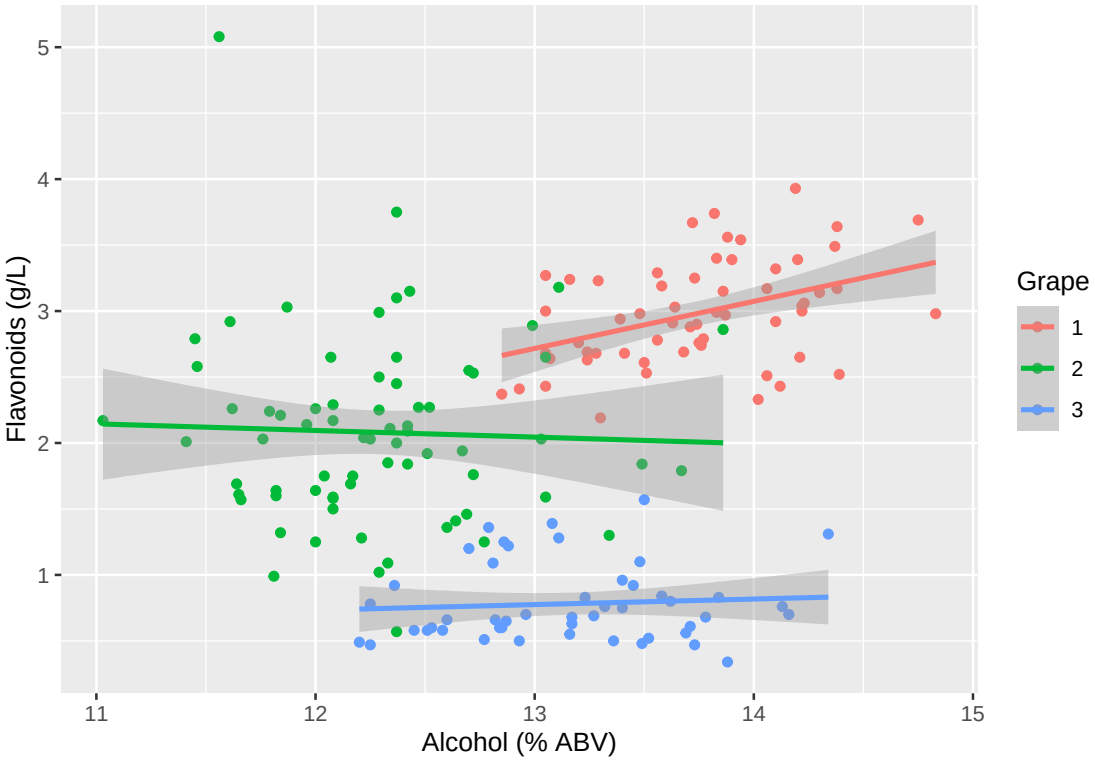
```
##  Accuracy:
##  MalicAcid = 83.6065573770492 %
##  Ash = 73.7704918032787 %
##  AlcalinityOfAsh = 78.6885245901639 %
##  Magnesium = 73.7704918032787 %
##  TotalPhenols = 81.9672131147541 %
##  Flavonoids = 93.4426229508197 %
##  NonflavonoidPhenols = 86.8852459016393 %
##  Proanthocyanins = 78.6885245901639 %
##  ColorIntensity = 88.5245901639344 %
##  Hue = 88.5245901639344 %
##  ProteinConcentration = 96.7213114754098 %
##  Proline = 85.2459016393443 %
```

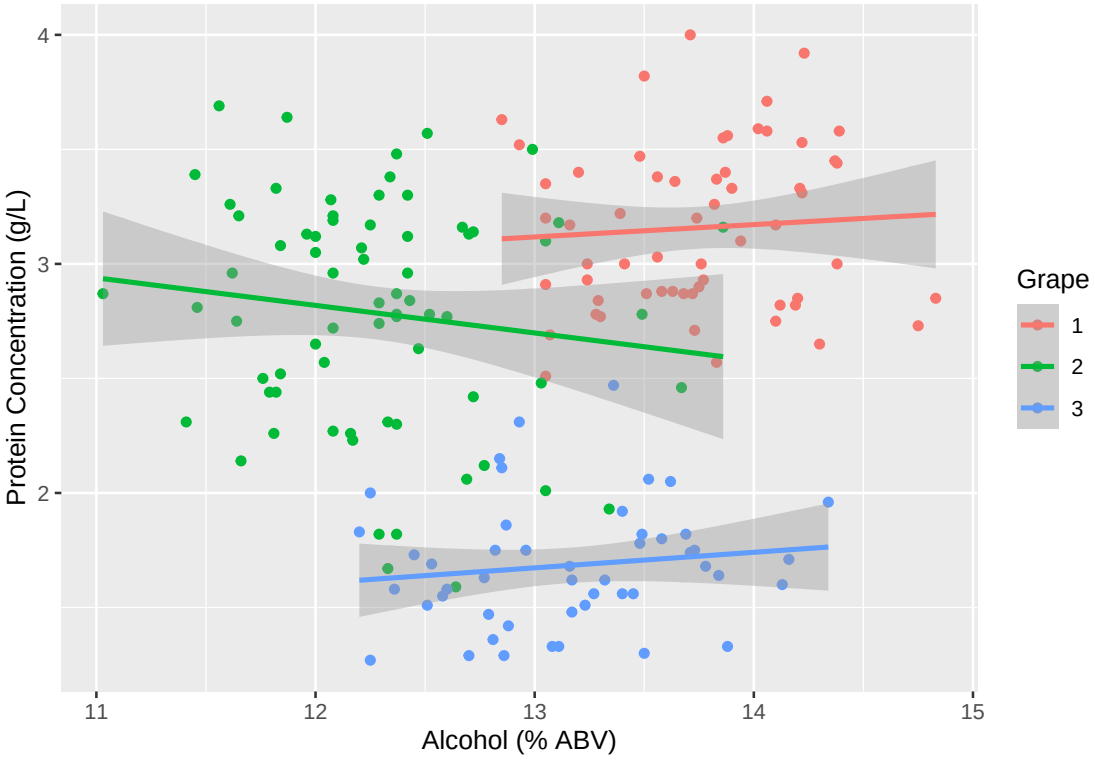
4 - Data Interpretation

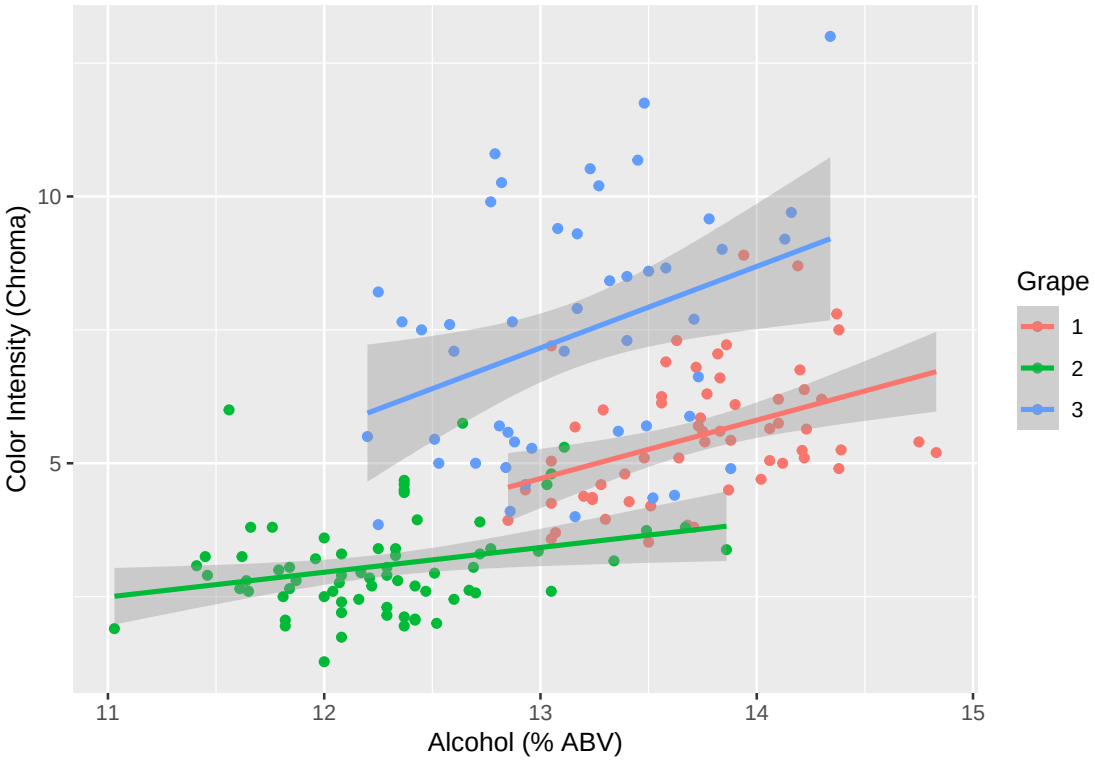
Our results look excellent. We can see that, when paired with AC, our constituents all correlate with grape varietal to varying degrees, but there are some that stand out with especially high accuracies. Looking at our four top performers, this analysis tells us that these constituents are the most distinct between the varietals in question, but not much beyond that. Since our end goal is to give the consumer simple, helpful information about the constituents in their wine, we will form a linear model to visualize the relationship between AC and the other constituents.

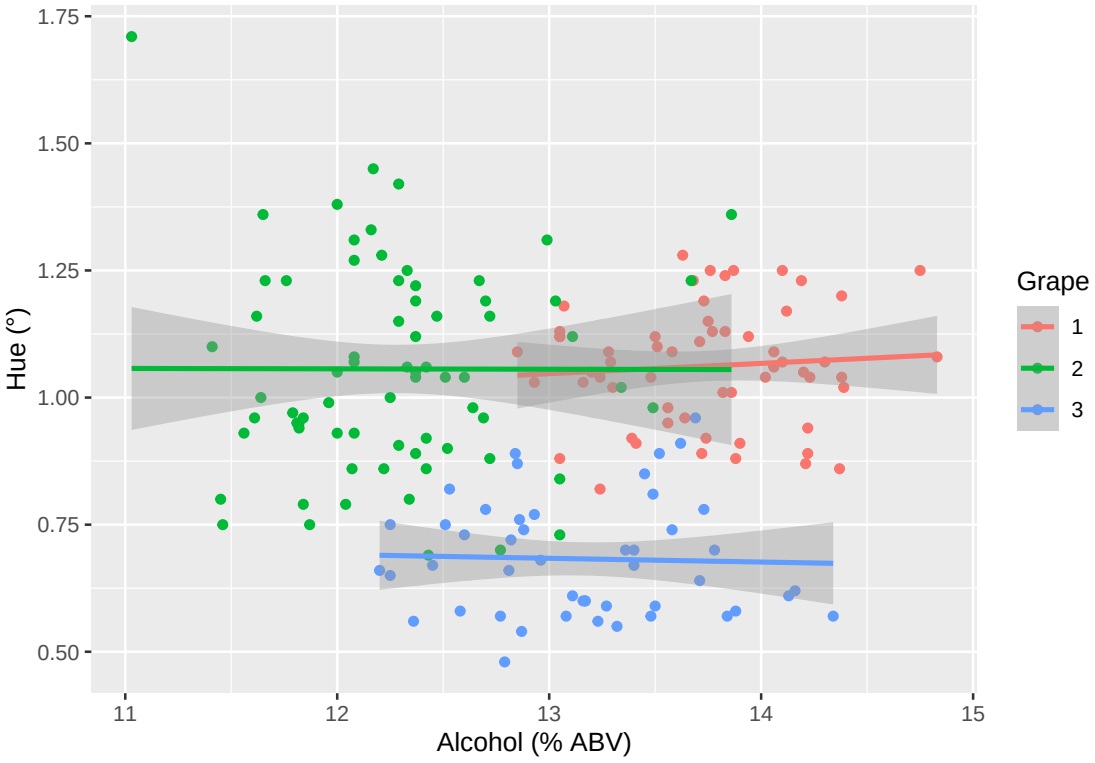
5 - Data Visualization

In order to extract useful, accurate information about our top four constituents using AC, we will graph a **generalized linear model** for each one which uses AC to predict its levels.






Color Intensity (Chroma)
10
5
11
12
13
14
15
Alcohol (% ABV)
Grape
1
2
3


Hue (°)
1.75
1.50
1.25
1.00
0.75
0.50
11
12
13
14
15
Alcohol (% ABV)
Grape
1
2
3

Results & Application

Our final graphs show us the levels of our top four constituents as predicted by alcohol content. This readable information allows the consumer to locate a bottle of wine made with one of these types of grape, look at its alcohol content, and be able to know the likely levels of each of these constituents inside. This helps everyday people who are not connoisseurs or sommeliers engage more deeply with their wine and start to put names to the common flavors they experience in their favorite bottles.

The graphs may be sufficient for dispersal to the consumer as-is, or we can attempt to improve readability with something like the following table:

```
## Grape 1
```

	Low AC	- Increases? -	High AC
Alcohol Content (% ABV)	12.8	+	14.8
Flavonoids (g/L)	2.7	+	3.3
Protein Concentration (g/L)	3.2	x	3.2
Color Intensity (Chroma)	4.5	+	6.5
Hue (°)	1.1	x	1.1

Further Refinement

While our results are very promising, there are still many things that need improvement.
Most significantly, we do not know what types of grapes these are as this data is anonymous. The method we developed here would need to be applied to wines from a known grape varietal and/or region in order to have actionable data.
When applying this method elsewhere, our sample size should be much larger. Our core data consisted of 178 observations which, while decent, is not large enough to be a reliable reflection of the overall trends for these grapes.
If sufficient data quality were attained, we could improve this method further by broadening the scope of our analysis. Looking back to the UCI repository webpage, the author mentions that around 17 variables from the original set were not included. If we increased the amount of constituents observed by finding supplemental or more complete data, we would increase the quality and utility of our final results.
We could also take more on-the-bottle predictors into account, such as the vintage and winery. Enhancing our GLM function this way would increase our accuracy and allow the consumer even more control and understanding about how years, regions, and weather affect their favorite wines.

Conclusion

x