Choose Your Own! Wine

Christian Mast

Contents

Overview

This is the final report on my "Chose Your Own!" data science project. Based on the briefing, a dataset named "Wine" was found and chosen from the UCI Machine Learning Repository.

During this project, the following steps have been taken:

- Download the data file from the webpage mentioned above and transform it into a workable data frame
- Explore the data, checking for missing values, NAs, inconsistencies
- Visualization of the dataset - histograms of the different variables and a correlation plot
- Developing and testing of models to predict the winery from the chemical analysis - applying several techniques learned in the course and from the internet

At the end, the model was able to predict the winery with an accuracy of XYZ%.

Furthermore, an outlook on potential further improvements is given.

The dataset "Wine"

Choosing a dataset

Why "Wine"? I was looking for a tidy dataset that I could easily understand in terms of content and topic - as some expertise usually helps to better understand information. The relatively small data set should also allow to run different methods without running into speed or memory issues on my PC. The link to the UCI page was given with the edx briefing.

As a chemist I understand the background of the described data, and as a person living in a wine region I am sufficiently familiar with the type of products described. The fact of having only 48h from briefing to due date of this project might have triggered a fast decision as well.

Source of the data:

Original Owners:
Forina, M. et al, PARVUS - An Extendible Package for Data Exploration, Classification and Correlation. Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy

Donor:
Stefan Aeberhard, email: stefan '@' coral.cs.jcu.edu.au

Downloading the data

The data is provided as a simple csv file using comma as separators. Importing it using read.csv() yields a data frame. Columns where named according to the information of the wine.name file, with the longest name being shortened to get visually more pleasant diagrams.

```
# Getting the data
# Define the urls
wine_data_url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
wine_names_url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.names"

#Download and rename files
download.file(wine_data_url, ".\\wine-data.csv")
download.file(wine_names_url, ".\\wine-names.txt")

#Creating a data frame from wine-data.csv
wines <- read.csv(".\\wine-data.csv", header=FALSE, sep=",")
```

```r
#Adding column names - for the ease of life, this is done manually
#"OD280/OD315 of diluted wines" was shortened to OD280.OD315
#"Nonflavanoid phenols" was shortened to Nonflav. phenols
#Slashes and spaces were removed to make rpart happy - thx to Stackoverflow!
colnames(wines) <- c("Winery", "Alcohol", "Malic_acid",
                     "Ash", "Alcalinity_of_ash", "Magnesium",
                     "Total_phenols", "Flavanoids",
                     "Nonflav._phenols", "Proanthocyanins",
                     "Color_intensity", "Hue",
                     "OD280.OD315", "Proline")
```

Exploring and analyzing the data

General structure

The file size of only 10.7kB already indicates that this data set is not very large. The function head() gives a good first impression:

```
##   Winery Alcohol Malic_acid  Ash Alcalinity_of_ash Magnesium Total_phenols
## 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
##   Flavanoids Nonflav._phenols Proanthocyanins Color_intensity  Hue OD280.OD315
## 1       3.06             0.28            2.29            5.64 1.04        3.92
## 2       2.76             0.26            1.28            4.38 1.05        3.40
## 3       3.24             0.30            2.81            5.68 1.03        3.17
## 4       3.49             0.24            2.18            7.80 0.86        3.45
## 5       2.69             0.39            1.82            4.32 1.04        2.93
## 6       3.39             0.34            1.97            6.75 1.05        2.85
##   Proline
## 1    1065
## 2    1050
## 3    1185
## 4    1480
## 5     735
## 6    1450
```

The data frame has 178 rows and 14 columns. The basic structure is tidy, as each observation is a row and each variable is a column. There are no zeros, empty cells or N/A's, and there are also no negative values. The first column "Winery" is of type int, but is actually categorical - describing which of the three participating wineries made that particular wine. This is also the prediction to make later on:

Can we predict who made a wine if we know the results from the lab analysis?

With the summary() function we can get a good first impression on the range of values for each variable:

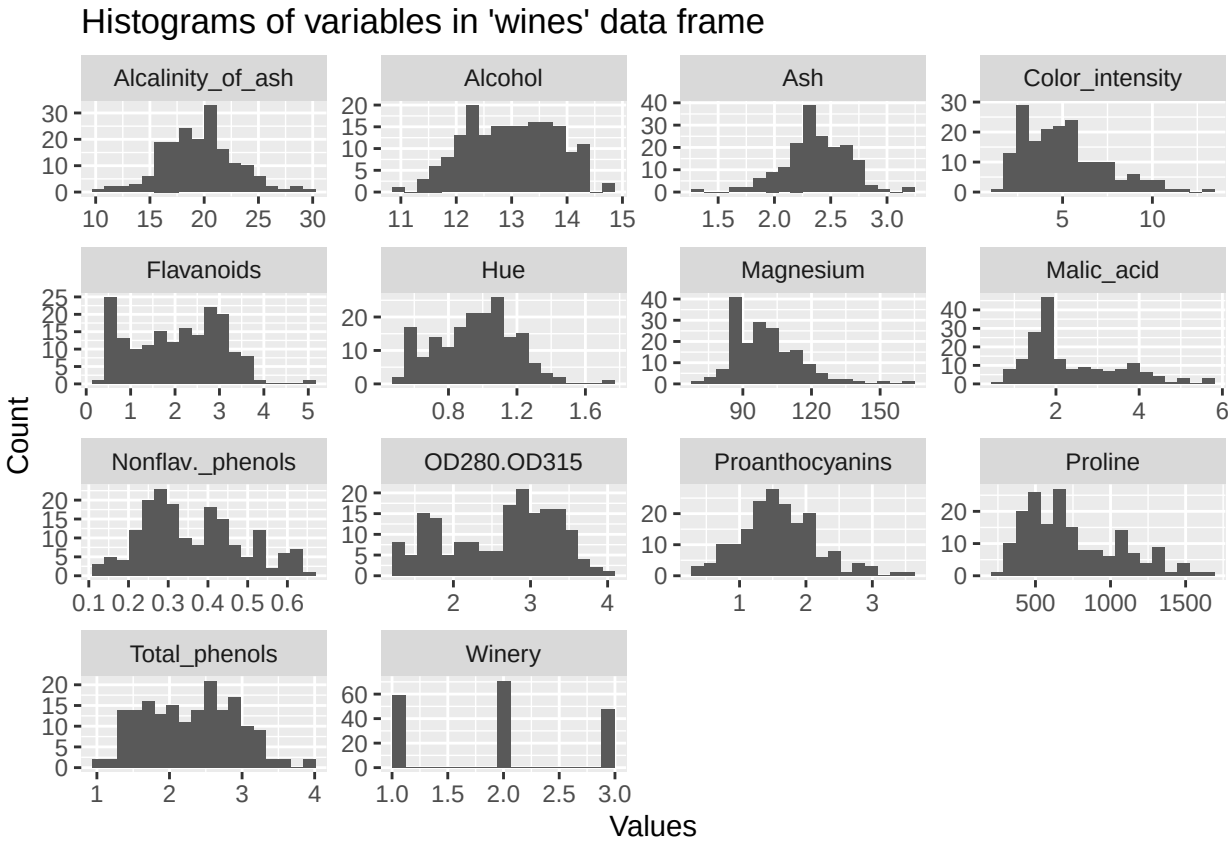
```
##      Winery          Alcohol        Malic_acid         Ash       
##  Min.   :1.000   Min.   :11.03   Min.   :0.740   Min.   :1.360  
##  1st Qu.:1.000   1st Qu.:12.36   1st Qu.:1.603   1st Qu.:2.210  
##  Median :2.000   Median :13.05   Median :1.865   Median :2.360  
##  Mean   :1.938   Mean   :13.00   Mean   :2.336   Mean   :2.367  
##  3rd Qu.:3.000   3rd Qu.:13.68   3rd Qu.:3.083   3rd Qu.:2.558  
##  Max.   :3.000   Max.   :14.83   Max.   :5.800   Max.   :3.230  
```

```
##  Alcalinity_of_ash   Magnesium      Total_phenols     Flavanoids
##  Min.   :10.60     Min.   : 70.00   Min.   :0.980   Min.   :0.340
##  1st Qu.:17.20     1st Qu.: 88.00   1st Qu.:1.742   1st Qu.:1.205
##  Median :19.50     Median : 98.00   Median :2.355   Median :2.135
##  Mean   :19.49     Mean   : 99.74   Mean   :2.295   Mean   :2.029
##  3rd Qu.:21.50     3rd Qu.:107.00   3rd Qu.:2.800   3rd Qu.:2.875
##  Max.   :30.00     Max.   :162.00   Max.   :3.880   Max.   :5.080
##  Nonflav._phenols Proanthocyanins Color_intensity       Hue
##  Min.   :0.1300   Min.   :0.410   Min.   : 1.280   Min.   :0.4800
##  1st Qu.:0.2700   1st Qu.:1.250   1st Qu.: 3.220   1st Qu.:0.7825
##  Median :0.3400   Median :1.555   Median : 4.690   Median :0.9650
##  Mean   :0.3619   Mean   :1.591   Mean   : 5.058   Mean   :0.9574
##  3rd Qu.:0.4375   3rd Qu.:1.950   3rd Qu.: 6.200   3rd Qu.:1.1200
##  Max.   :0.6600   Max.   :3.580   Max.   :13.000   Max.   :1.7100
##   OD280.OD315       Proline
##  Min.   :1.270   Min.   : 278.0
##  1st Qu.:1.938   1st Qu.: 500.5
##  Median :2.780   Median : 673.5
##  Mean   :2.612   Mean   : 746.9
##  3rd Qu.:3.170   3rd Qu.: 985.0
##  Max.   :4.000   Max.   :1680.0
```

The larger difference between Median and Mean for some variables (e.g. Proline) already indicates that the distribution of values for each variable are not necessarily normal. The variables “Magnesium” and “Proline” are have higher values than the others.

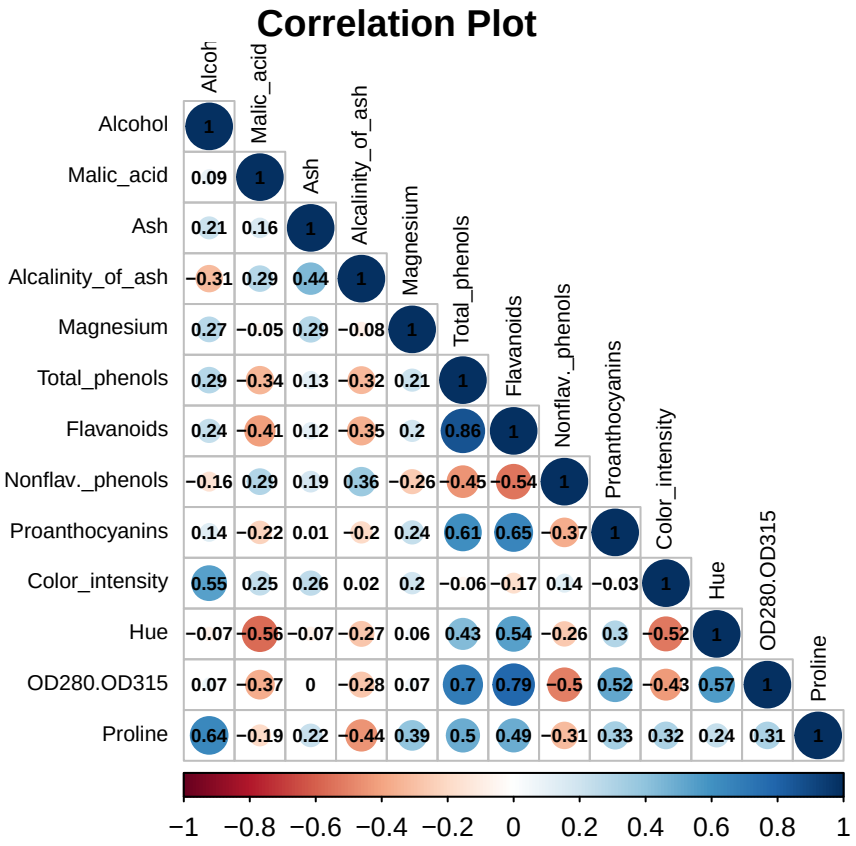
Visualizations

Histograms are a good tool to get an overview about the distribution of data - visualizations are often much easier to understand than numbers.



As we can easily see, "Winery" is indeed categorical and not continuous. The other values show all kind of distributions, being it normal, skewed or bimodal.

How do these variables correlate with each other? To get a quick overview, corrplot() is used:



The plot shows nicely that some values are not correlating much with the others, e.g. Ash and Magnesium. Some correlations are actually expected, e.g. the rather high correlation of Alcohol and Proline - as both indicate high ripeness of the grapes due to a lot of sun.

Modelling

Processing

In order to use the data in the dataframe for methods like knn, we will scale it. After that operation,

```
#scaling wines, then rebuilding the winery column
scaled_wines <- as.data.frame(scale(wines))
scaled_wines$Winery <- as.factor(wines$Winery)
```

We will then create a test and a training set:

```
# Set a seed to get reproducible train & test sets
set.seed(1, sample.kind = "Rounding")
index <- createDataPartition(scaled_wines$Winery, p = 0.25, list = FALSE)
train_set <- scaled_wines[-index, ]
test_set <- scaled_wines[+index, ]
```

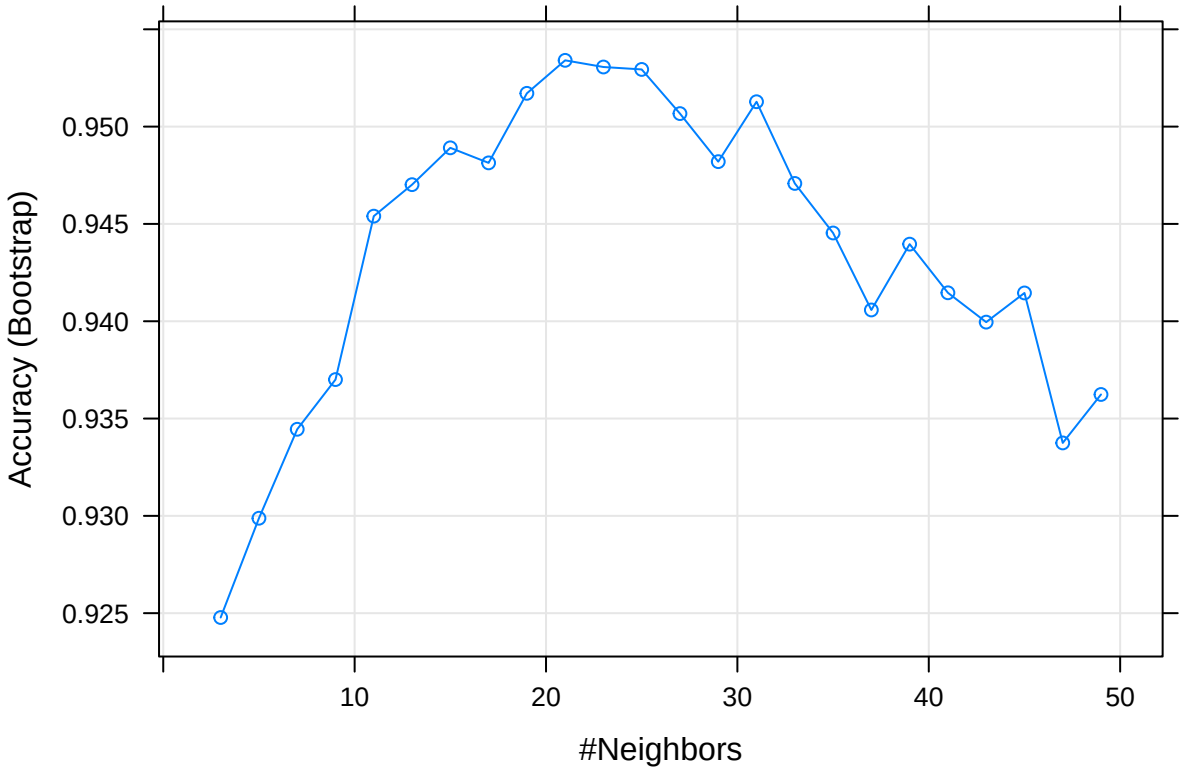
Now we can start training models!

Training

Training a knn model

Its time to set a first benchmark - a basic knn model is trained and the resulting outcome is used to make predictions on the test set.

```
## k-Nearest Neighbors
##
## 133 samples
##  13 predictor
##   3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 133, 133, 133, 133, 133, 133, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    3  0.9247766  0.8849566
##    5  0.9298781  0.8928042
##    7  0.9344483  0.8997253
##    9  0.9370034  0.9036919
##   11  0.9453990  0.9163086
##   13  0.9470173  0.9188341
##   15  0.9489098  0.9217057
##   17  0.9481372  0.9204056
##   19  0.9517109  0.9258908
##   21  0.9534048  0.9285461
##   23  0.9530628  0.9280109
##   25  0.9529377  0.9277053
##   27  0.9506691  0.9243317
##   29  0.9481995  0.9207518
##   31  0.9512790  0.9252890
##   33  0.9470811  0.9190383
##   35  0.9445342  0.9150863
##   37  0.9405794  0.9092346
##   39  0.9439605  0.9141543
##   41  0.9414609  0.9104708
##   43  0.9399537  0.9079087
##   45  0.9414526  0.9102509
##   47  0.9337474  0.8986054
##   49  0.9362372  0.9023440
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 21.
```



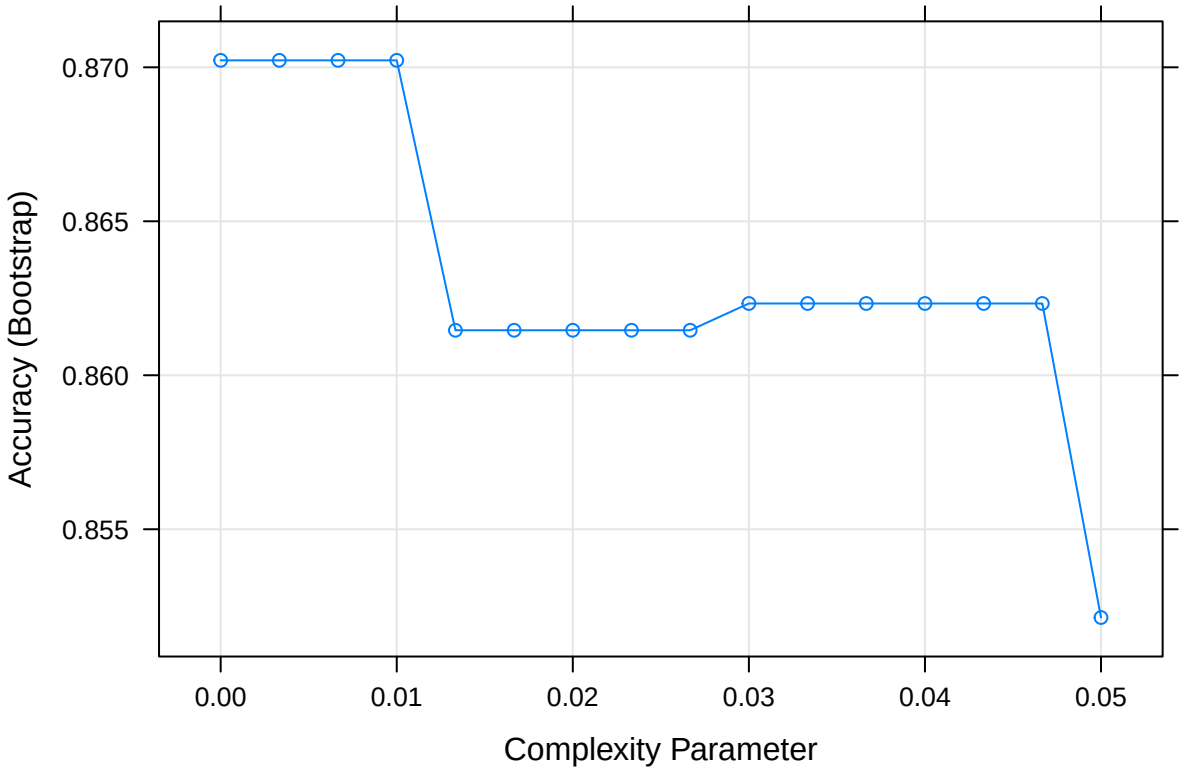

```
## Accuracy of the knn model: 1
```

The knn model is already very good - depending on the seed, an accuracy between 0.95 and 1 (!) is achieved. Predictions cannot get much better. Most interesting, the best value for k differs quite a lot - sometimes its 3, sometimes 29. That is the reason the tunegrid for k has been set to a rather broad range, which seems counterintuitive - after all the training set has only 133 entries, so looking at the 49 nearest ones is already looking at more than a third of the total population.

Training a classification tree

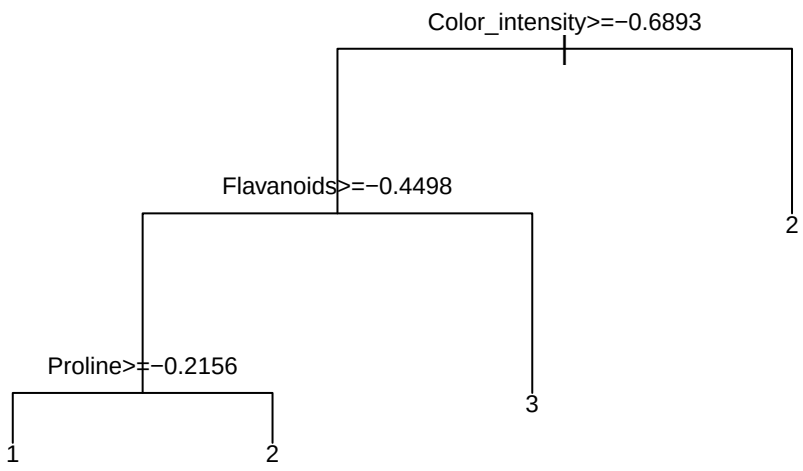
A classification tree depends on rather simple decisions, leading along a branched structure to come to the classification result. The package "rpart" has been used to optimize such a classification tree.

```
## CART
##
## 133 samples
##  13 predictor
##   3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 133, 133, 133, 133, 133, 133, ...
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.000000000  0.8702253  0.7994060
##   0.003333333  0.8702253  0.7994060
##   0.006666667  0.8702253  0.7994060
##   0.010000000  0.8702253  0.7994060
##   0.013333333  0.8614609  0.7875876
##   0.016666667  0.8614609  0.7875729
##   0.020000000  0.8614609  0.7875729
##   0.023333333  0.8614609  0.7875729
##   0.026666667  0.8614609  0.7877884
##   0.030000000  0.8623305  0.7891154
##   0.033333333  0.8623305  0.7891154
##   0.036666667  0.8623305  0.7891154
##   0.040000000  0.8623305  0.7891154
##   0.043333333  0.8623305  0.7891154
##   0.046666667  0.8623305  0.7891154
##   0.050000000  0.8521338  0.7733762
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.01.
```




```
## Accuracy of the rpart model: 0.9333333
```

```
plot(fit_rpart$finalModel, margin = 0.1, main="Classification tree")
text(fit_rpart$finalModel, cex = 0.75, pos=3, offset=0)
```



The resulting classification tree is surprisingly simple and based on only three predictors - color intensity, as well as flavanoids and proline concentrations/amounts. Especially the color intensity can be easily modified with wines - so this does not look like the most stable model to roll out to include further manufactures.

Random forests

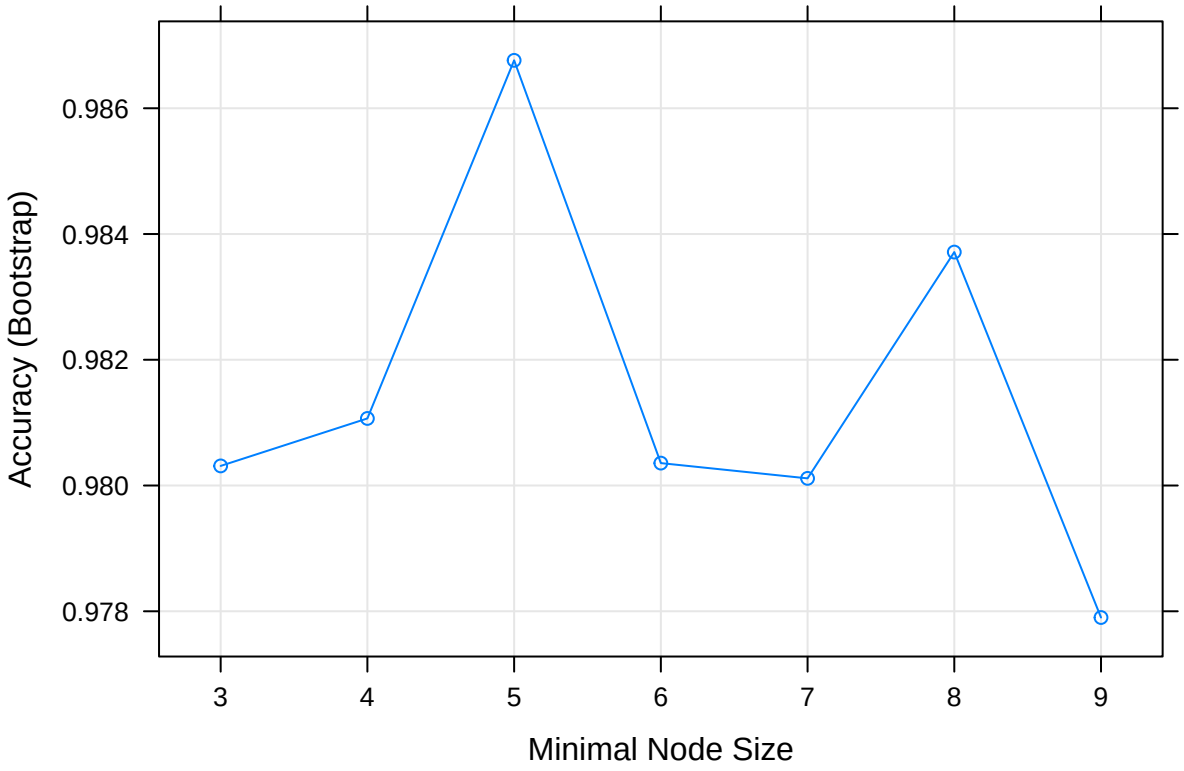
The logical consequence of using a classification tree and the package rpart is to have a look at a random forest method. In this case, the method Rborist is used.

```
# Train a rborist model
fit_rb <- train(Winery ~.,
                method="Rborist",
                tuneGrid = data.frame(predFixed=2, minNode=seq(3,9,1)),
                nTree=100,
                data = train_set)

# Output the fit results
fit_rb
```

```
## Random Forest
##
## 133 samples
##  13 predictor
##   3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 133, 133, 133, 133, 133, 133, ...
## Resampling results across tuning parameters:
##
##   minNode  Accuracy   Kappa
##   3        0.9803119  0.9699529
##   4        0.9810669  0.9710535
##   5        0.9867620  0.9797026
##   6        0.9803564  0.9698973
##   7        0.9801139  0.9695957
##   8        0.9837129  0.9750162
##   9        0.9779006  0.9660717
##
## Tuning parameter 'predFixed' was held constant at a value of 2
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were predFixed = 2 and minNode = 5.
```

```
# Plot the fit results to show the best k
plot(fit_rb)
```




```
# Create a prediction, calculate the confusion matrix and print the accuracy
prediction_rb <- predict(fit_rb, test_set)
cm_rb <- confusionMatrix(prediction_rb, test_set$Winery)
cat("Accuracy of the Rborist model:", cm_rb$overall["Accuracy"])
```

```
## Accuracy of the Rborist model: 1
```