Choose Your Own! - Dataset: 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 or inconsistencies
- Visualization of the dataset - histograms of the different variables and a correlation plot
- Defining the challenge - can we predict the winery based on the measurements?
- Developing and testing of models to predict the winery from the chemical analysis - applying several models learned during the data science course and from the internet.
- Forming an ensemple prediction based on the best three models found
- Giving a summary of findings

At the end, the model was able to predict the winery with an accuracy of 98.9%.

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")
```

```r
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=",")

#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 for the tip!
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 data structure is tidy, so 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?

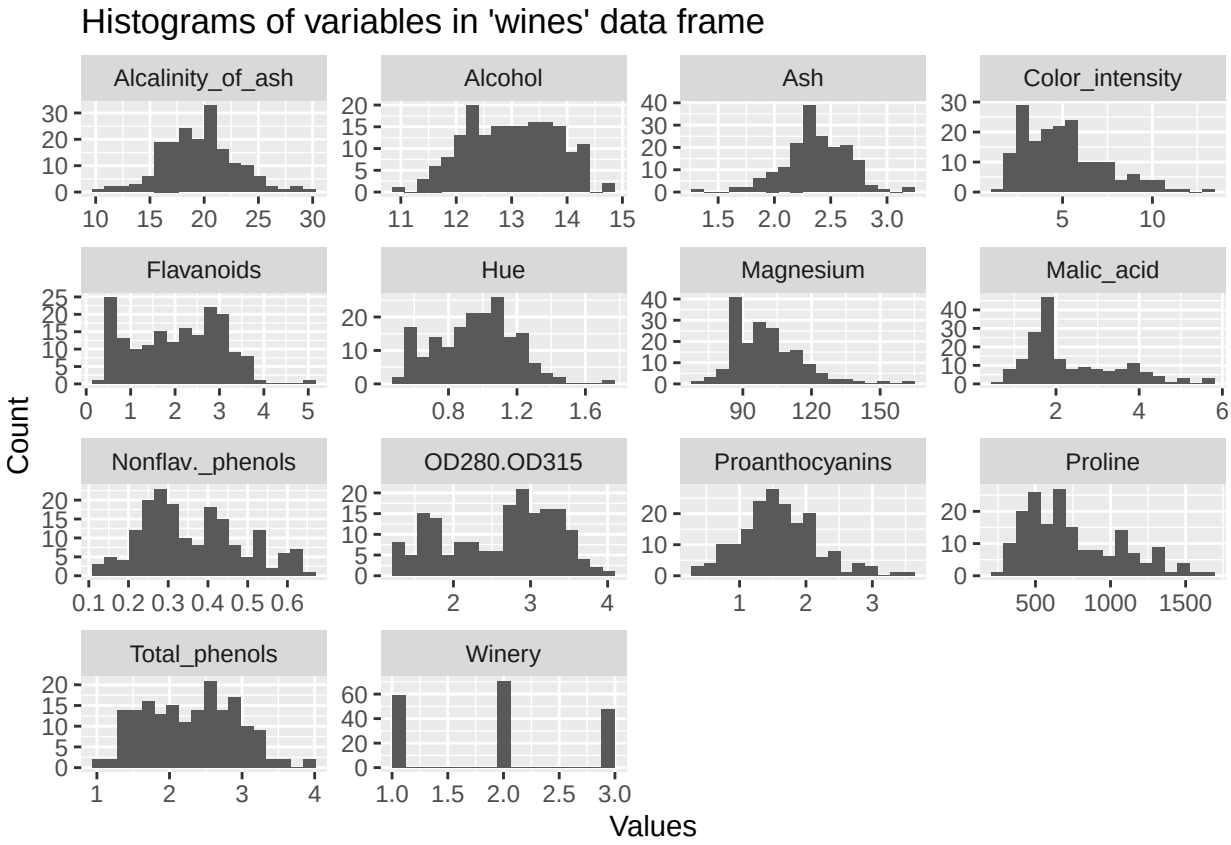
With the summary() function we can get a good first impression on the range of values for each variable:

```
##      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” also have much higher absolute values than the others.

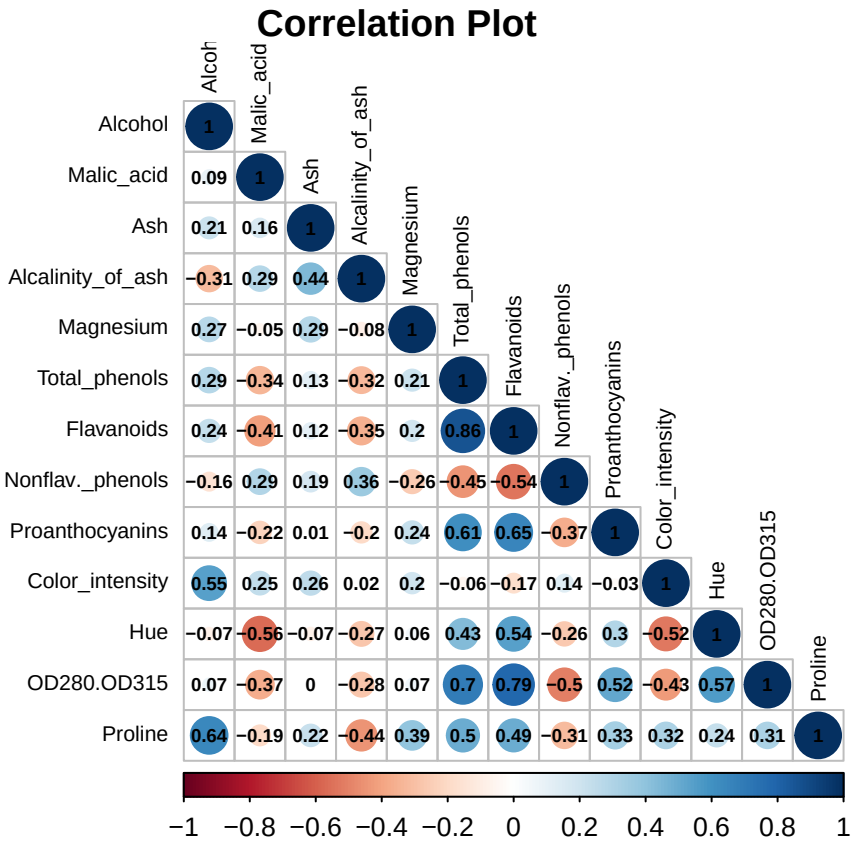
Visualizations

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

Preprocessing

In order to use the data in the dataframe for methods like knn, we will scale it. This will make sure that e.g. high levels of Proline will not distort measurements for "distance".

```
#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, using 50% of the data for training and 50% for testing:

```
# Set a seed to get reproducible train & test sets
set.seed(101, sample.kind = "Rounding")
index <- createDataPartition(scaled_wines$Winery, p = 0.5, list = FALSE)
train_set <- scaled_wines[-index, ]
test_set <- scaled_wines[+index, ]
```

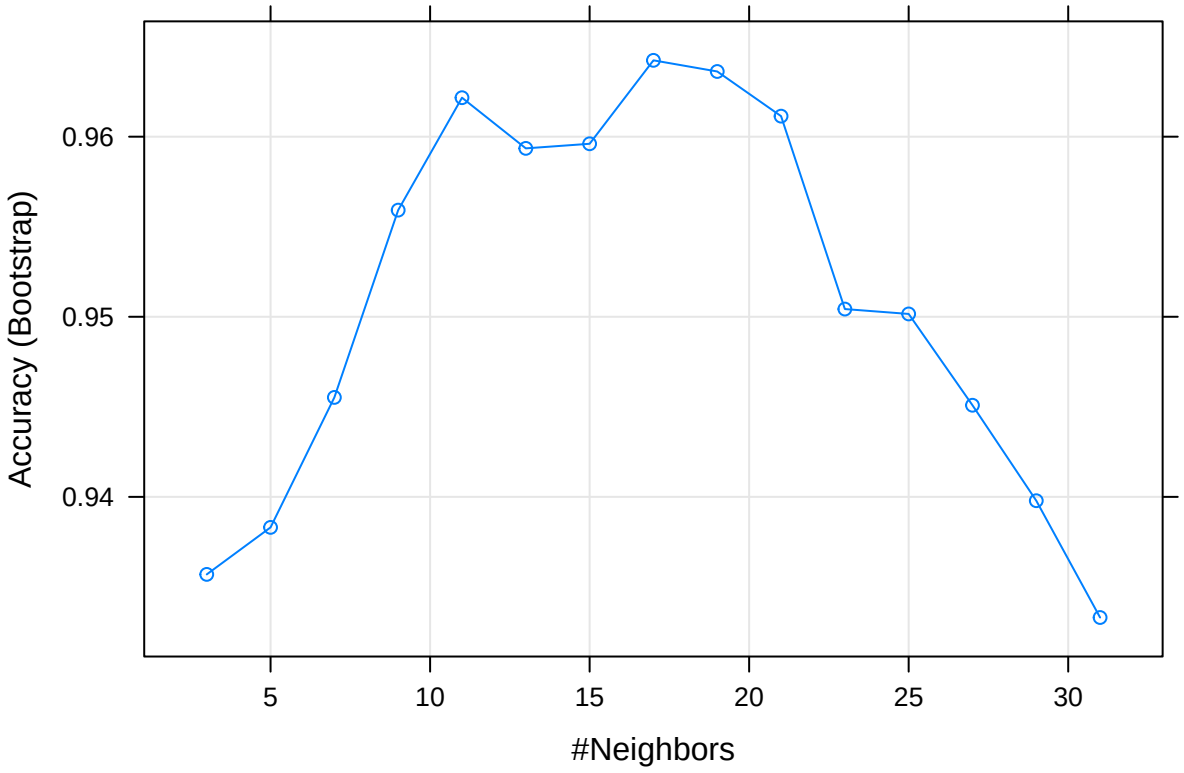
Now we can start training models!

Training

KNN

In order to create a first benchmark - a basic knn model is trained and the resulting outcome is used to make predictions on the test set. Based on some initial trials, a rather broad range of k values was offered for optimization via the tuneGrid parameter.

```
## k-Nearest Neighbors
##
## 88 samples
## 13 predictors
##  3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 88, 88, 88, 88, 88, 88, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    3  0.9356958  0.9010364
##    5  0.9383063  0.9052952
##    7  0.9455232  0.9163458
##    9  0.9559221  0.9323191
##   11  0.9621636  0.9415459
##   13  0.9593504  0.9375782
##   15  0.9596061  0.9380923
##   17  0.9642380  0.9451534
##   19  0.9636189  0.9441999
##   21  0.9611397  0.9406107
##   23  0.9504281  0.9244820
##   25  0.9501579  0.9238390
##   27  0.9450883  0.9160372
##   29  0.9397873  0.9076572
##   31  0.9333016  0.8988175
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 17.
```



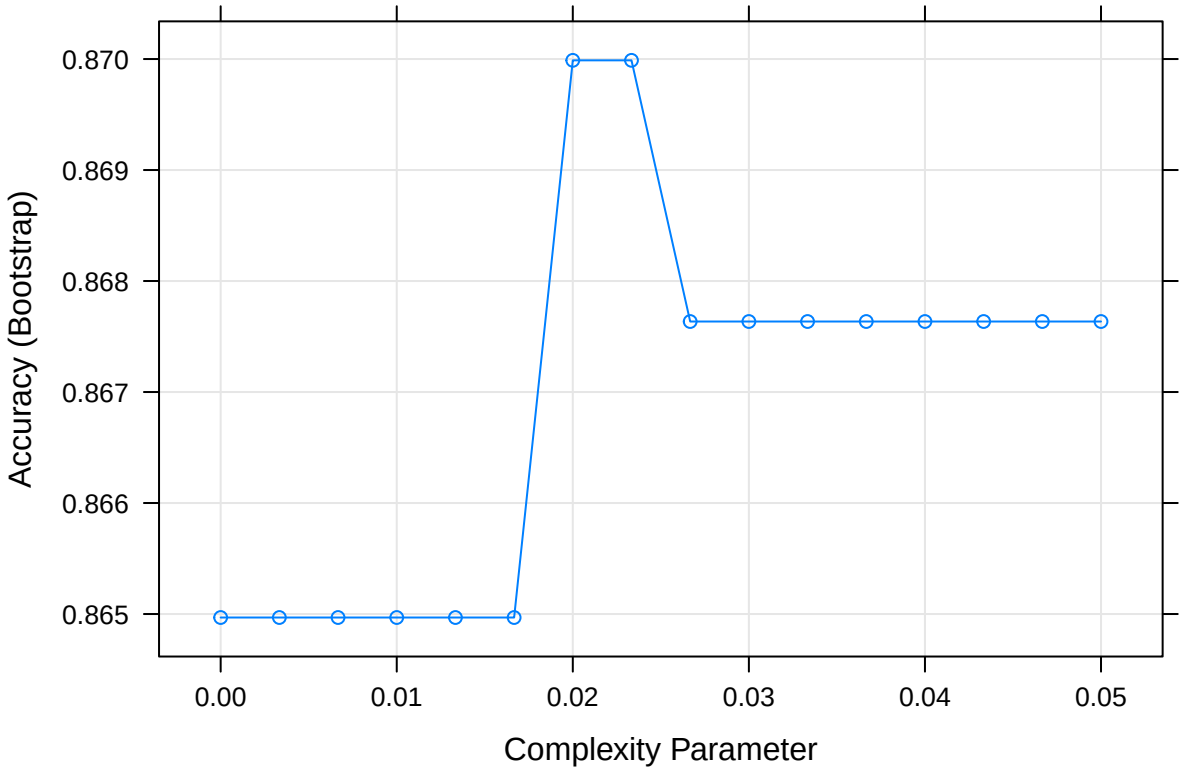

```
## Accuracy of the knn model: 0.9666667
```

The knn model is already very good - an accuracy of 0.9666667 is achieved when predicting the wineries of the test set. This is quite a good start and might be difficult to improve.

Classification Tree

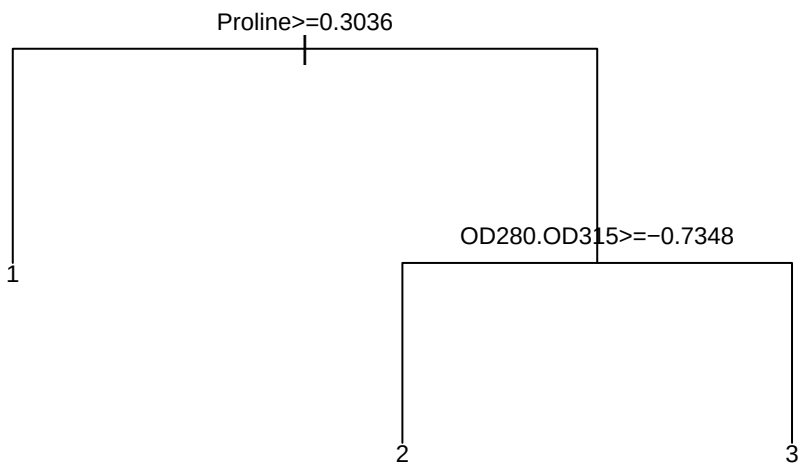
A classification tree depends on rather simple decisions, leading along a branched structure to come to the classification result. The package "rpart" has been used to optimize such a classification tree on the given training set.

```
## CART
##
## 88 samples
## 13 predictors
##  3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 88, 88, 88, 88, 88, 88, ...
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.000000000  0.8649684  0.7931875
##   0.003333333  0.8649684  0.7931875
##   0.006666667  0.8649684  0.7931875
##   0.010000000  0.8649684  0.7931875
##   0.013333333  0.8649684  0.7931875
##   0.016666667  0.8649684  0.7931875
##   0.020000000  0.8699880  0.8004035
##   0.023333333  0.8699880  0.8004035
##   0.026666667  0.8676351  0.7967965
##   0.030000000  0.8676351  0.7967965
##   0.033333333  0.8676351  0.7967965
##   0.036666667  0.8676351  0.7967965
##   0.040000000  0.8676351  0.7967965
##   0.043333333  0.8676351  0.7967965
##   0.046666667  0.8676351  0.7967965
##   0.050000000  0.8676351  0.7967965
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.02333333.
```




```
## Accuracy of the rpart model: 0.8222222
```

How does the resulting classification tree look like?

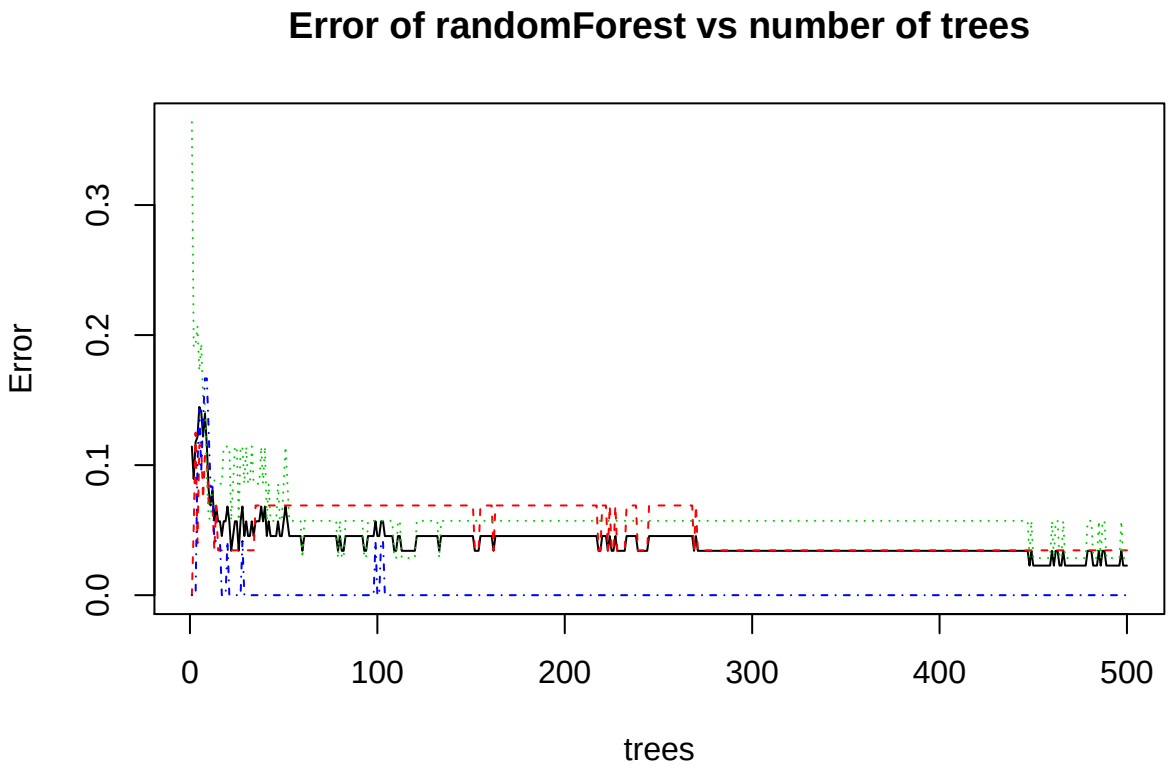


The resulting classification tree is surprisingly simple and based on only two predictors - Proline and the OD280/OD315 on diluted wines. However, the accuracy of 0.8222222 for the predicted wineries is much worse than the results from the knn approach.

Random forests

The logical consequence of using a classification tree and the package rpart is to have a look at a random forest method. A quick experiment with the randomForest function shows that 100 trees are enough to deliver a constant error.

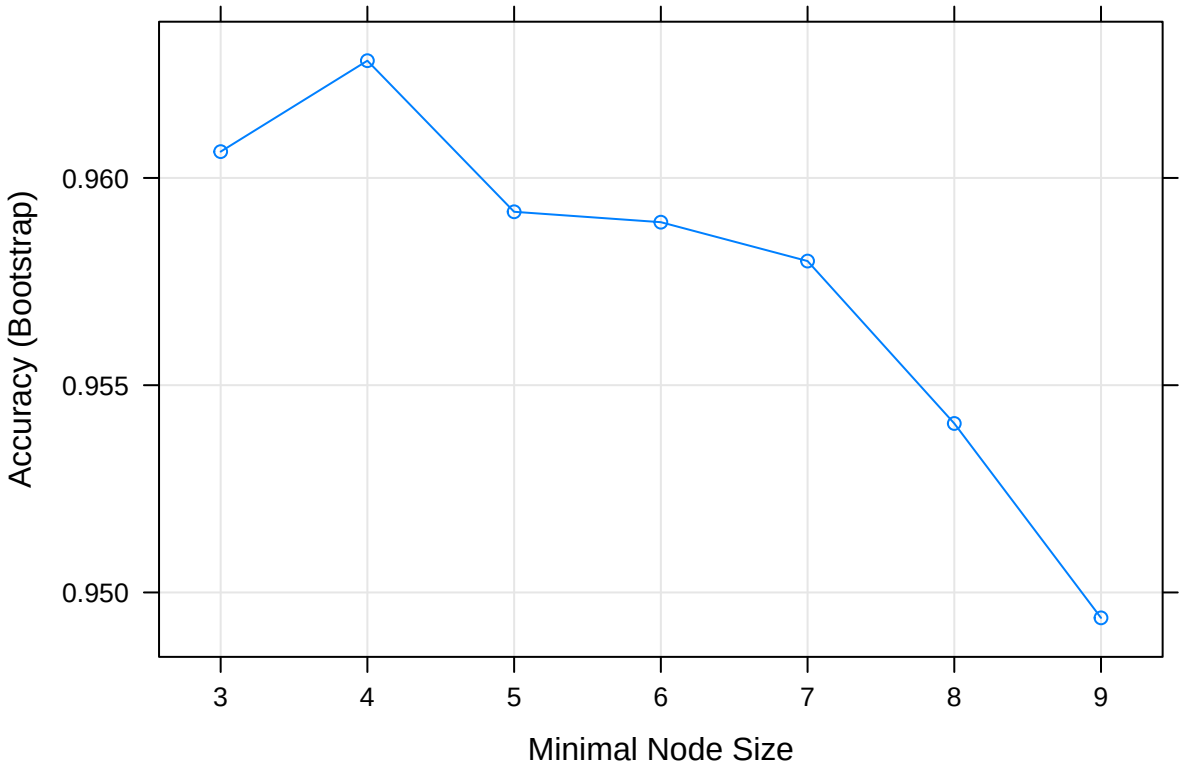
```
fit_rf <- randomForest(Winery~., data = train_set)
plot(fit_rf, main="Error of randomForest vs number of trees")
```



For the random forest model, the method Rborist is used, with the number of trees limited to 100 - however, the speed of the calculation is still reasonable.

```
## Random Forest
##
## 88 samples
## 13 predictors
##  3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 88, 88, 88, 88, 88, 88, ...
## Resampling results across tuning parameters:
##
##   minNode  Accuracy   Kappa
##   3        0.9606347  0.9397985
##   4        0.9628282  0.9429117
##   5        0.9591832  0.9373893
##   6        0.9589326  0.9370955
```

```
##   7         0.9579945  0.9355094
##   8         0.9540770  0.9296548
##   9         0.9493871  0.9222109
##
## 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 = 4.
```




```
## Accuracy of the Rborist model: 0.9888889
```

The approach using Rborist provided an accuracy of 0.9888889 for the prediction of wineries on the test set, the best result so far.

Support Vector Machine

The train function in the caret package provides lots of different machine learning algorithms. By simply changing the name of the applied method, hundreds of algorithm can be tested despite a lack of deeper understanding. So this method has been chosen as it was described for classification tasks and as it worked well right out of the box.

```
# Train a "Support Vector Machine with linear Kernel" model
fit_svml <- train(Winery ~.,
                  method="svmLinear",
                  tuneGrid = expand.grid(C = c(0.001, 0.01, 0.1, 1, 10)),
                  data = train_set)

# Output the fit results
fit_svml
```

```
## Support Vector Machines with Linear Kernel
##
## 88 samples
## 13 predictors
##  3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 88, 88, 88, 88, 88, 88, ...
## Resampling results across tuning parameters:
##
##   C      Accuracy   Kappa
##   1e-03  0.3868992  0.02849231
##   1e-02  0.9640167  0.94478345
##   1e-01  0.9661669  0.94822201
##   1e+00  0.9546636  0.93090213
##   1e+01  0.9546636  0.93090213
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.1.
```

```
# Create a prediction, calculate the confusion matrix and print the accuracy
prediction_svml <- predict(fit_svml, test_set)
cm_svml <- confusionMatrix(prediction_svml, test_set$Winery)
cat("Accuracy of the svmLinear model:", cm_svml$overall["Accuracy"])
```

```
## Accuracy of the svmLinear model: 0.9666667
```

With an accuracy of 0.9666667, we found another good prediction result.

Forming an Ensemble

We can now combine the predictions of the three best models (knn, Rborist and svmLinear) and create a prediction on the rounded average of all three models to see if there is an improvement.

```
#Convert predictions to numbers, calculate the average
prediction_ensemble <- as.factor(round((
    as.numeric(prediction_knn)+
    as.numeric(prediction_rb)+
    as.numeric(prediction_svml))/3))

# Calculate the confusion matrix and print the accuracy
cm_ensemble <- confusionMatrix(prediction_ensemble, test_set$Winery)
cat("Accuracy of the Ensemble:", cm_ensemble$overall["Accuracy"])
```

```
## Accuracy of the Ensemble: 0.9888889
```

With an ensemble accuracy of 0.9888889, the ensemble is delivering the same accuracy as the model based on the Rborist method. So at least in this case, there is no improvement by combining the predictions.

Results & Summary

Findings & Outlook

During this project, a promising data set called "Wine" was identified, retrieved and examined. It was then tried to create prediction models for the one categorical variable "Winery".

In total four different models have been tried on the dataset, leading to accuracy values of 0.8222 to 0.9889 on the test set:

- knn, k-nearest neighbours: Accuracy 0.9666667
- rpart, a classification tree: Accuracy 0.8222222
- Rborist, a random forest approach: Accuracy 0.9888889
- svmLinear, a support vector machine: Accuracy 0.9666667

The best value of 0.9888889 has been observed with Rborist model. This results reflects one wrong prediction for a test set of 90 observations. So, to answer the initial question:

"*Yes, we can predict the winery based on the lab analysis with sufficient accuracy!*".

A trial to further improve the accuracy by creating an ensemble prediction out of the best three models did not result in further improvement - the same accuracy as with Rborist is observed.

As it turned out to be rather easy to try further models with the train function and the caret package, I might try more models and search the internet about their approach, tuning options and limitations.

Further findings and comments

- As the dataset is rather small, at first a larger training set (75%) was used - hoping for better results due to more training. This approach gave surprising high accuracy values up to 1 (depending on the seed used). This indicates that the data can be classified actually pretty well, so the training set was decreased in size to challenge the different methods a bit more.
- Looking at the one prediction where the ensemble went wrong, it turned out that a different voting mode (e.g. median) would not have changed the result - as all four models predicted the same, wrong result.
- It would be interesting to look at larger wine databases, as many parameters that are observed in real life were not included in this data set, e.g. type of grape, type of ground, vintage. At least when adding more wineries, an increased difficulty can be expected.
- I was surprised how easy it is to access and apply methods that I was not aware of and that I did not fully understand yet. This also includes the risk of errors and mistakes - and makes one feel less like a data scientist and more like a script kiddie.
- Overall a good exercise, learning about sources of data sets, useful websites (stackoverflow, kaggle) and about how much is achievable in short time if necessary.