Red and White

Made Ots

08/03/2022

INTRODUCTION

This report is part of HarvardX PH125.9x Capstone project, Choose Your Own Submission. My project is inspired from Cassie Kozyrkov's course modules of making Friends With Machine Learning & AI, which are are available freely on Youtube. She is using the wine classification example in the first session of the course. I decided to try and use the wine tasting and wine qualities analysis for my own submission project. I am not sure how much of a use this would have in a real world, being that the wine tasting and qualities, like whiskey tasting and qualities, are actually important for quite a small segments of the whole population.

Datasets for Red & White wine used in this project are available from Kaggle -> https://www.kaggle.com/brendan45774/wine-quality I have uploaded both data sets to my Github page.

GOAL

The goal of this analysis is to build a system which is able to predict & recognize, based on existing ratings, is the wine we are given a red one or a white one. I started out analyzing the available data from both sets to see the structure and what information is available, followed with wine quality ratings overview.

INSPECTING THE DATA

1. We start with looking into the Headers of both data sets to see what type of information is included.

Showing the first 5 rows of each file.

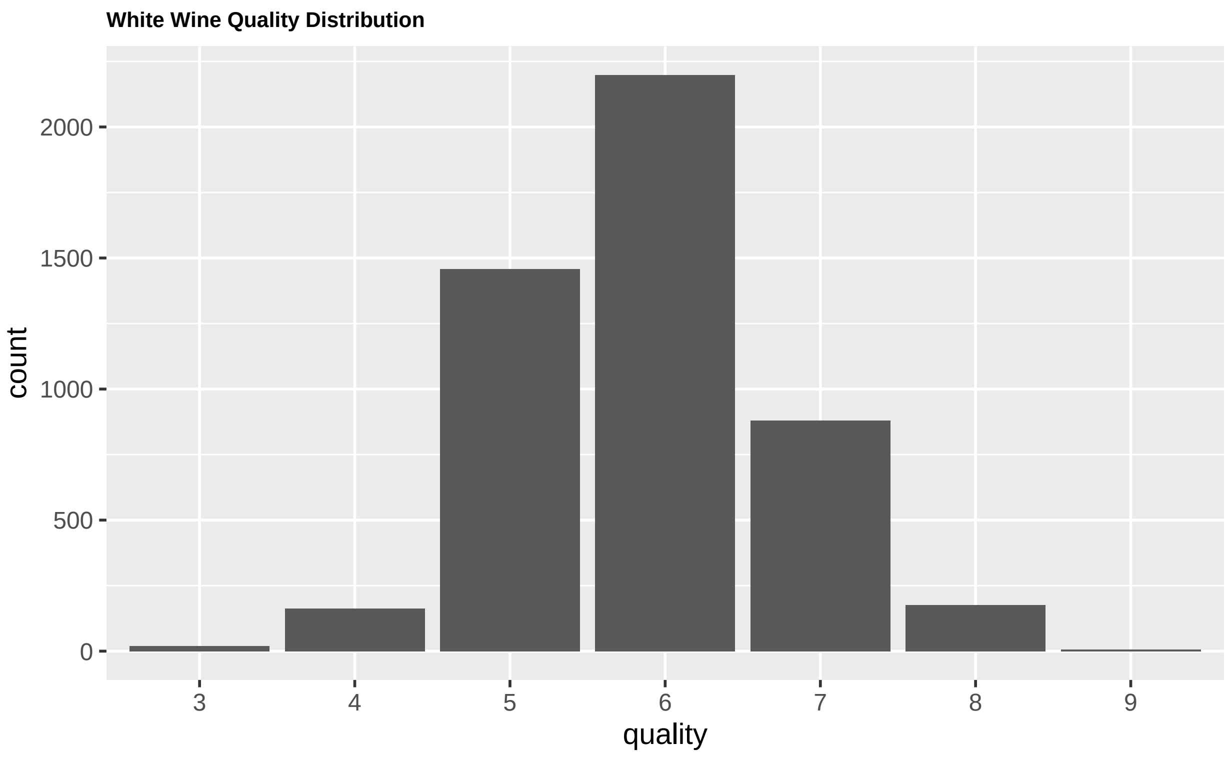
White Wine Quality:

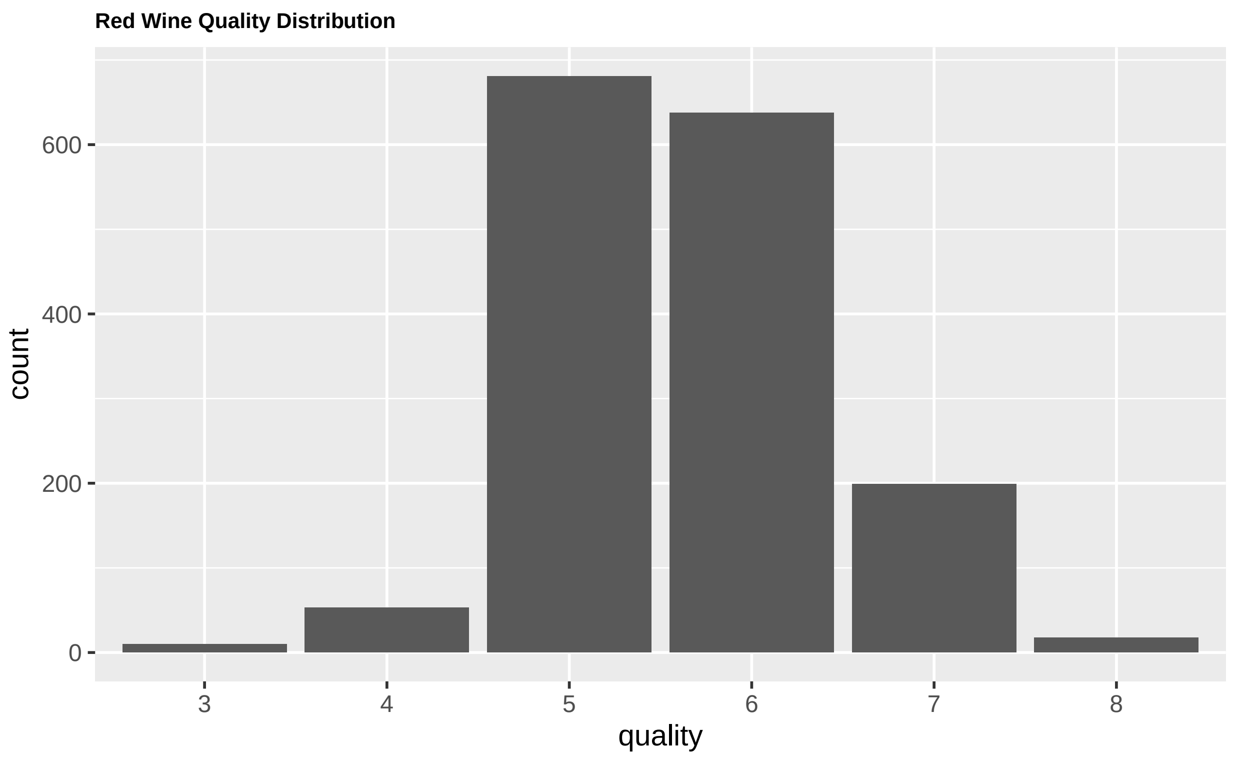
```
## # A tibble: 5 x 12
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
##           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>
## 1           7               0.27        0.36           20.7     0.045
## 2           6.3             0.3         0.34            1.6     0.049
## 3           8.1             0.28        0.4             6.9     0.05
## 4           7.2             0.23        0.32            8.5     0.058
## 5           7.2             0.23        0.32            8.5     0.058
## # ... with 7 more variables: free.sulfur.dioxide <dbl>,
## #   total.sulfur.dioxide <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
## #   alcohol <dbl>, quality <chr>
```

Red Wine Quality:

```
## # A tibble: 5 x 12
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
##           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>
## 1           7.4             0.7         0              1.9     0.076
## 2           7.8             0.88        0              2.6     0.098
## 3           7.8             0.76        0.04           2.3     0.092
## 4          11.2             0.28        0.56           1.9     0.075
## 5           7.4             0.7         0              1.9     0.076
## # ... with 7 more variables: free.sulfur.dioxide <dbl>,
## #   total.sulfur.dioxide <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
## #   alcohol <dbl>, quality <chr>
```

2. We are mostly interested in the wine quality rating as our system should be able to predict the wine being red or white based on it, so next we will inspect the rating distributions in both sets.





The three words sweetness, acidity, and tannin represent three of the major components (parts) of wine. The fourth is alcohol. Besides being one of the reasons we often want to drink a glass of wine in the first place, alcohol is an important player in wine quality. Tannin and acidity are hardening elements in a wine (they make a wine taste firmer and less giving in the mouth), while alcohol and sugar (if any) are softening elements. The balance of a wine in the relationship of the hard and the soft aspects of a wine — and a key indicator of quality.

Keeping this in mind, we will now set out to see can we teach our system to predict is the wine we like red wine or white wine.

3. Analysing the wine ratings

We will work mostly with White Wine set while training our machine. We start with splitting White Wine data to Test set & Train set.

In this project I am using the Random Forest and the Rborist package in R to train the algorithms and get the results. A Random Forest grows many classification trees and for each output from that tree, it is said that the tree "votes" for that class. A tree is growing using the following steps: 1. a random sample of rows from the training data will be taken for each tree; 2. from the sample taken in step 1, a subset of features will be taken to be used for spitting on each tree; 3. each tree is grown to the largest extent specified by the parameters until it reaches a vote for the class.

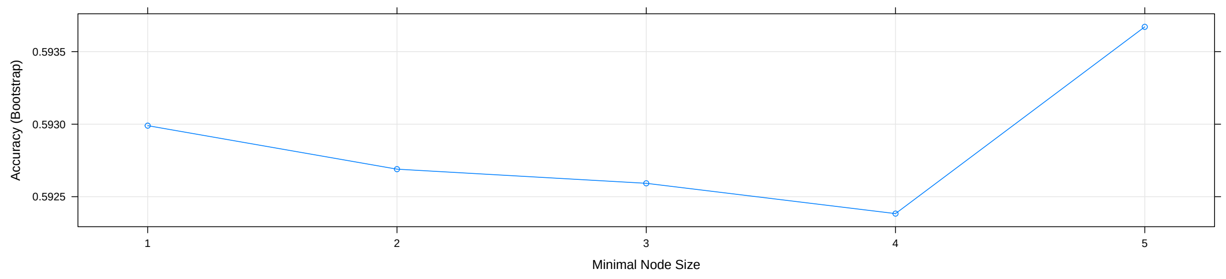
The main reason to use a Random Forest instead of a decision tree is to combine the predictions of many decision trees into a single model. The reasoning is that a single even made up of many mediocre models will still be better than one good model. Also, Randome Forests are less prone to over-fitting because of this.

Over-fitting can occur with a flexible model (decision trees) where the model will memorize the Training data and learn any noise in the data as well. This makes it unable to predict the Test data. A Random Forest can reduce the high variance from a flexible model like a decision tree by combining many trees into one ensemble model.

3.1 Ratings in the White Wine set.

We train our White Wine test set with “caret” function and the Rborist is chosen for the Random Forest as it gives the predFixed and minNode parameters. We already know that the data set mainly focuses on average wines which means that we may encounter a lack of data that describes the wine properties for bad and good wines.

```
## \begin{table}
## \centering\begingroup\fontsize{9}{11}\selectfont
##
## \begin{tabular}{lrr}
## \toprule
##   & predFixed & minNode\\
## \midrule
## \cellcolor{gray!6}{5} & \cellcolor{gray!6}{0} & \cellcolor{gray!6}{5}\\
## \bottomrule
## \multicolumn{3}{l}{\rule{0pt}{1em}\textit{Note: }}\\
## \multicolumn{3}{l}{\rule{0pt}{1em}best tune}\\
## \end{tabular}
## \endgroup{}
## \end{table}
```



```
## Rborist variable importance
##
##                      Overall
## density               1.1531
## alcohol               1.0891
## volatile.acidity      1.0317
## free.sulfur.dioxide   0.9254
## chlorides             0.9253
## total.sulfur.dioxide  0.8864
## residual.sugar        0.8702
## citric.acid           0.8316
## pH                    0.8085
## sulphates             0.7882
## fixed.acidity         0.7571
```

Now calculating the accuracy on test set we created:

```
## [1] 0.6568747
```

Our very first model predicted the quality of ratings with almost 60%, but it far from being usable in any way. To see can we do better, let’s see create the confusion matrix.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   0   0   0   0   0   0
##          4   0   9   3   3   0   0   0
##          5   3  46 460 142  12   0   0
##          6   7  27 256 886 198  46   2
##          7   0   0  10  68 227  14   1
##          8   0   0   0   0   3  28   0
##          9   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6569
##                  95% CI : (0.6377, 0.6757)
##     No Information Rate : 0.4484
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4599
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity           0.00000 0.109756   0.6310   0.8062  0.51591  0.31818
## Specificity           1.00000 0.997467   0.8821   0.6036  0.95375  0.99873
## Pos Pred Value            NaN 0.600000   0.6938   0.6231  0.70938  0.90323
## Neg Pred Value        0.99592 0.970033   0.8496   0.7930  0.90005  0.97521
## Prevalence            0.00408 0.033456   0.2974   0.4484  0.17952  0.03590
## Detection Rate        0.00000 0.003672   0.1877   0.3615  0.09262  0.01142
## Detection Prevalence  0.00000 0.006120   0.2705   0.5802  0.13056  0.01265
## Balanced Accuracy     0.50000 0.553612   0.7566   0.7049  0.73483  0.65846
##                      Class: 9
## Sensitivity          0.000000
## Specificity          1.000000
## Pos Pred Value            NaN
## Neg Pred Value       0.998776
## Prevalence           0.001224
## Detection Rate       0.000000
## Detection Prevalence 0.000000
## Balanced Accuracy    0.500000
```

3.2 Good, Bad & Average ratings - meaning?

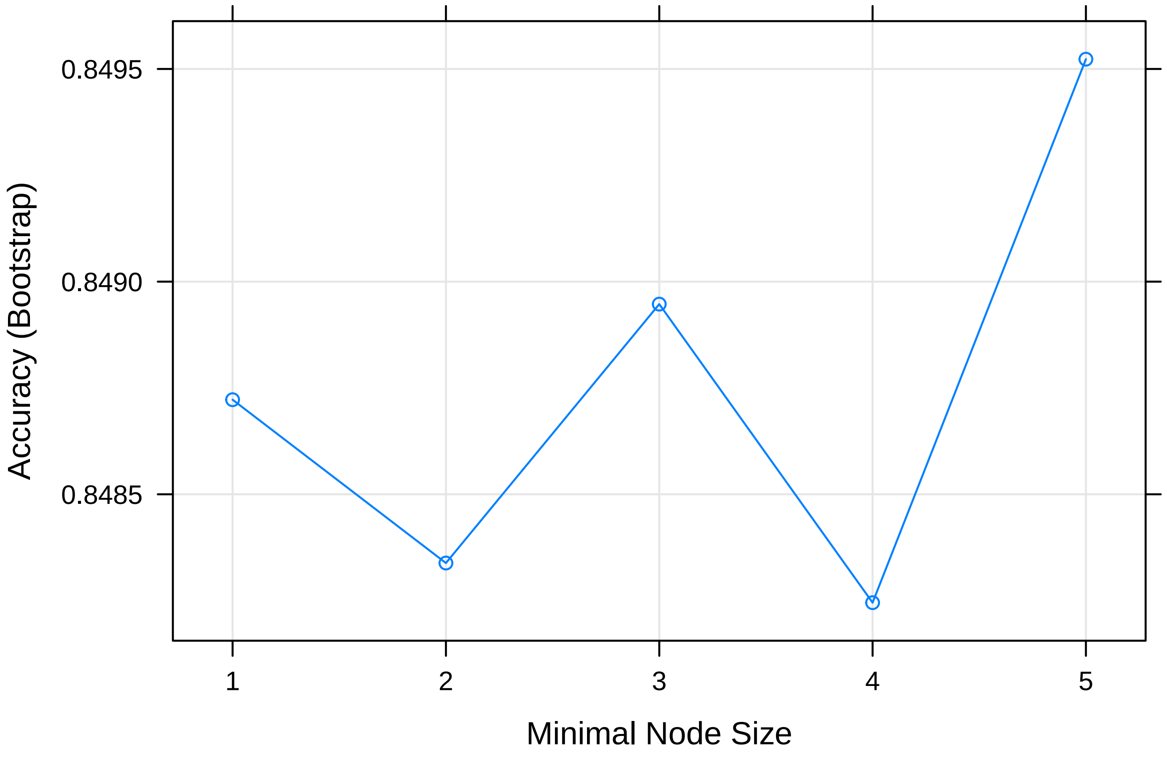
The ratings in the White Wine and Red Wine data sets are given from 0 to 10. To simplify the prediction,we have set the rating parameters to following: Good 7 - 10 Average 4 - 6 Bad 0 - 3 We have no idea of what the given numerical ratings on scale 0 - 10 mean, except that we can guess that rating "0" is probably very bad and rating "10" is probably the best. Because the ratings were not distributed equally, some ratings were given more often than others. The grouping to Good, Bad & Average may help to increase the accuracy of our predictions. White Wine Data set.

	predFixed	minNode
5	0	5

Note:
best tune

We are creating a new Train Set with our newly set quality ratings.

Let's see how our simplified qulity ratings are now performing:



```
## Rborist variable importance
##
##                      Overall
## alcohol               2.0618
## density               1.7448
## pH                    1.1406
## volatile.acidity      1.1022
## free.sulfur.dioxide   1.0545
## chlorides             1.0524
## total.sulfur.dioxide  1.0499
## residual.sugar        1.0165
## sulphates             0.9129
## citric.acid           0.8254
## fixed.acidity         0.7738
```

```
## [1] 0.845243
```

Our Random Forest prediction accuracy improved significantly after grouping the numerical ratings to Good, Bad & Average. However, some of the information is lost because of the grouping of the quality ratings.

Our accuracy is now around 85%. Confusion matrix will let us see in more details:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction average  bad good
##    average    1803   10  263
##    bad           0    0    0
##    good        106    0  267
##
## Overall Statistics
##
##                Accuracy : 0.8452
##                  95% CI : (0.8303, 0.8594)
##     No Information Rate : 0.7795
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4947
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: average Class: bad Class: good
## Sensitivity                  0.9445   0.000000      0.5038
## Specificity                  0.4944   1.000000      0.9448
## Pos Pred Value               0.8685        NaN      0.7158
## Neg Pred Value               0.7158   0.995917      0.8733
## Prevalence                   0.7795   0.004083      0.2164
## Detection Rate               0.7362   0.000000      0.1090
## Detection Prevalence         0.8477   0.000000      0.1523
## Balanced Accuracy            0.7195   0.500000      0.7243
```

In total we have fewer predictions wrong than with previous model which used the numeric ratings instead of grouped ratings. Aso after using the grouping, we can see that the group "Bad" barely exists anymore and majority of ratings are falling into the "Average" category. The overall accuracy can be a deceptive measure and the Random Forest was not very good in predicting the "bads" and was very bad in predicting the "goods". It only seemed to work with the averages.

3.3 White Wine - Good or Bad?

When we grouped the wine ratings to Good, Bad & Average, even befor this the Bad raings were almost non-existent. Perhaps we should balance the missing ratings out using only 2 groups for more balanced approach - Good & Bad? The Bad group will include ratings from 0 - 5 and Good group will include ratings from 5 - 10.

Grouping the Good and Bad:

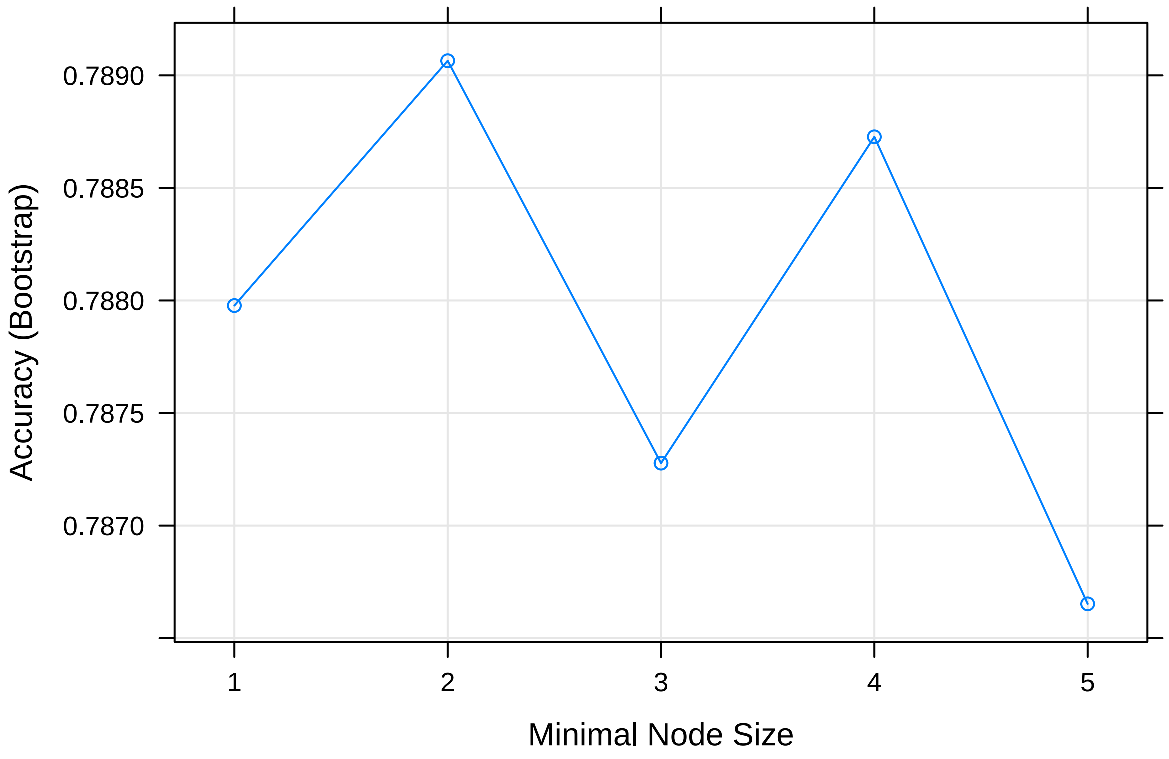
Creating the Test and Train set:

	predFixed	minNode
2	0	2

Note:
best tune

```
# Creating train set & test set for "good" and "bad" ratings
test_index_GoodBad <- createDataPartition(winequality_white_GoodBad$quality_GoodBad, times = 1, p = 0.5
test_set_GoodBad <- winequality_white_GoodBad[test_index_GoodBad, ]
train_set_GoodBad <- winequality_white_GoodBad[-test_index_GoodBad, ]
```

Predicting the White wine quality with simplified Bad & Good ratings.



```
## Rborist variable importance
##
##                        Overall
## alcohol                  3.687
## density                  3.245
## volatile.acidity         3.148
## free.sulfur.dioxide      2.543
## citric.acid              2.374
## total.sulfur.dioxide     2.371
## chlorides                2.249
## residual.sugar           2.178
## sulphates                1.924
## pH                       1.842
## fixed.acidity            1.743
```

```
## [1] 0.8260514
```

Creating confusion Matrix to see the accuracy:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  bad good
##       bad   532  138
##       good  288 1491
##
##                Accuracy : 0.8261
##                  95% CI : (0.8104, 0.8409)
##     No Information Rate : 0.6652
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5909
##
##  Mcnemar's Test P-Value : 5.234e-13
##
##             Sensitivity : 0.6488
##             Specificity : 0.9153
##          Pos Pred Value : 0.7940
##          Neg Pred Value : 0.8381
##              Prevalence : 0.3348
##          Detection Rate : 0.2172
##    Detection Prevalence : 0.2736
##       Balanced Accuracy : 0.7820
##
##        'Positive' Class : bad
##
```

The goal was to optimize the overall accuracy which is simply defined as the overall proportion that is predicted correctly. It seems that changing the groupings and rating variables didn't cause major improvement in accuracy of the prediction. The new Bad group now has some data in it, but it is still far less that the Good group.

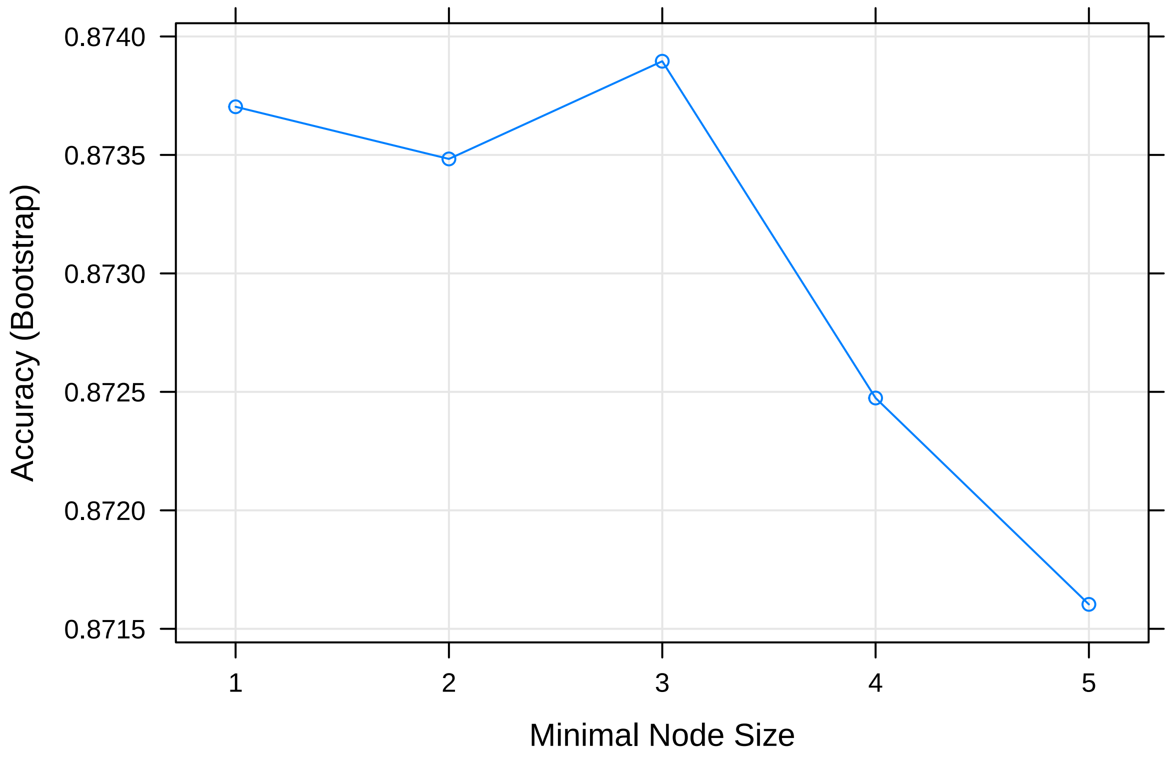
3.4. Can we balance the White Whine - Good & Bad?

Often the classification problems turn out to be imbalanced so it is only fair to assume that the data sets may be imbalanced as well. The grouping of the quality rating into 2 classes already seems to be resulting a more balanced data set. We will try to further balance it by over-sampling the minority class.

We apply the same caret and Rborist functions as previously, except we are now trying to balance the two quality classes of Good and Bad.

	predFixed	minNode
3	0	3

Note:
best tune



```
## Rborist variable importance
##
##                       Overall
## alcohol                 6.143
## density                 4.671
## volatile.acidity        4.373
## free.sulfur.dioxide     4.007
## chlorides               3.635
## residual.sugar          3.311
## total.sulfur.dioxide    3.282
## pH                      2.946
## citric.acid             2.715
## sulphates               2.571
## fixed.acidity           2.449
```

```
## [1] 0.8248265
```

Checking our results with confusion matrix:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  bad good
##       bad   549  158
##       good  271 1471
##
##                Accuracy : 0.8248
##                  95% CI : (0.8092, 0.8397)
##     No Information Rate : 0.6652
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5928
##
##  Mcnemar's Test P-Value : 6.394e-08
##
##             Sensitivity : 0.6695
##             Specificity : 0.9030
##          Pos Pred Value : 0.7765
##          Neg Pred Value : 0.8444
##              Prevalence : 0.3348
##          Detection Rate : 0.2242
##    Detection Prevalence : 0.2887
##       Balanced Accuracy : 0.7863
##
##        'Positive' Class : bad
##
```

The Random Forest prediction for quality of wine after grouping the ratings into two balances classes is with a slightly lower accuracy when we hoped. But, the accuracy is now similar in both the Bad and the Good class.

HOW THE PREDICTION MODEL PERFORMS

We have tried different groupings and it is time to try is our model fitting well. All the training and testing in the modelling phase was done on the White Wine Quality data set. From our investigation of both data sets at the beginning of the project we know that they have the same structure and same variables. This means that we are able to validate our model on the Red Wine Quality data set to make sure it's working properly.

We start with joining both data sets.

```
# Predicting the type of the wine - joining 2 databases, red and white
winequality_white_winetype <- winequality_white %>% mutate(wine = "white")
winequality_red_winetype <- winequality_red %>% mutate(wine = "red")
winequality_whiteANDred <- winequality_white_winetype %>% full_join(winequality_red_winetype)
```

```
## Joining, by = c("fixed.acidity", "volatile.acidity", "citric.acid",
## "residual.sugar", "chlorides", "free.sulfur.dioxide", "total.sulfur.dioxide",
## "density", "pH", "sulphates", "alcohol", "quality", "wine")
```

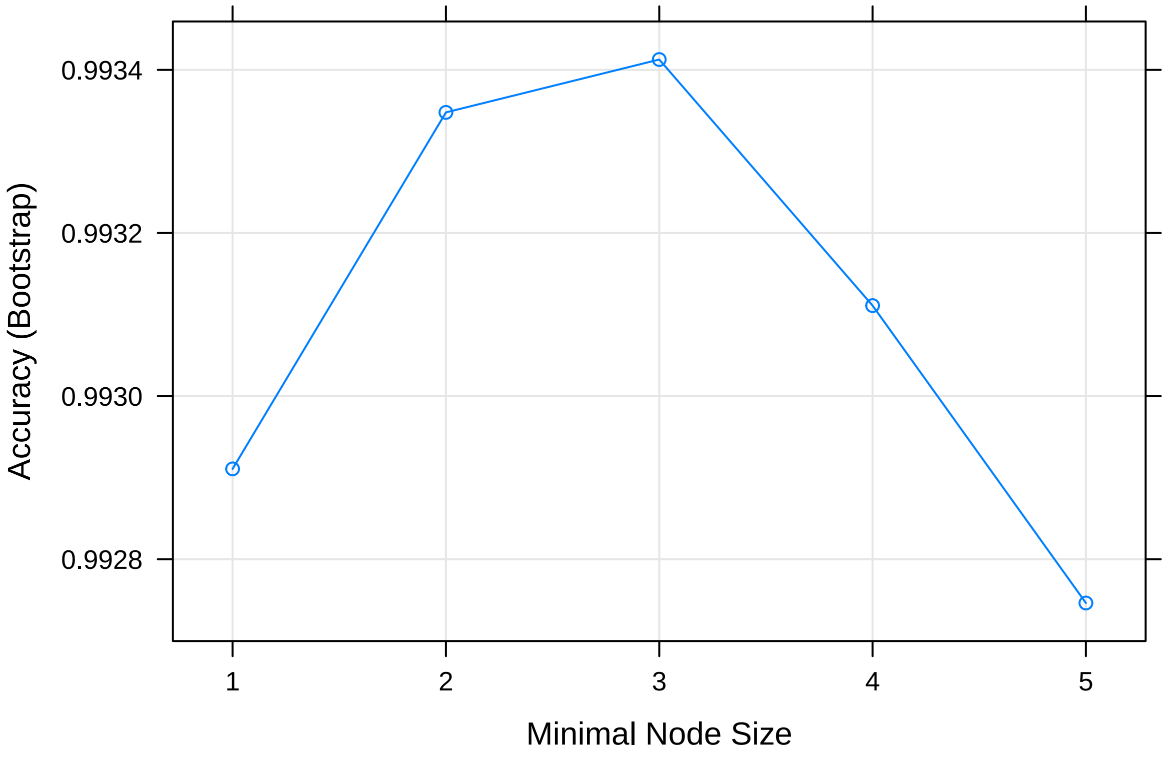
Splitting the joined data set into Train and test sets:

	predFixed	minNode
3	0	3

Note:
best tune

```r
# Creating the train set & test set for the wine type, red or white
test_index_winetype <- createDataPartition(winequality_whiteANDred$wine, times = 1, p = 0.5, list = FALS
test_set_winetype <- winequality_whiteANDred[test_index_winetype, ] %>% dplyr::select(-quality)
train_set_winetype <- winequality_whiteANDred[-test_index_winetype, ] %>% dplyr::select(-quality)
```

We are now trying to fit our Random Forets model to the new data set and see how it performs.



```
## Rborist variable importance
##
##                       Overall
## chlorides              8.4197
## total.sulfur.dioxide   7.9080
## volatile.acidity       4.8882
## fixed.acidity          2.0544
## sulphates              1.6818
## density                1.5248
## free.sulfur.dioxide    1.2061
## residual.sugar         1.0662
## pH                     0.5932
## citric.acid           0.2807
## alcohol               0.2503
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  red white
##      red    786     4
##      white   14  2445
##
##                Accuracy : 0.9945
##                  95% CI : (0.9913, 0.9967)
##     No Information Rate : 0.7538
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.985
##
##  Mcnemar's Test P-Value : 0.03389
##
##             Sensitivity : 0.9825
##             Specificity : 0.9984
##          Pos Pred Value : 0.9949
##          Neg Pred Value : 0.9943
##              Prevalence : 0.2462
##          Detection Rate : 0.2419
##    Detection Prevalence : 0.2432
##       Balanced Accuracy : 0.9904
##
##        'Positive' Class : red
##
```

The Random Forest managed to predict the type of wine with a very high overall accuracy which shows good prediction capabilities.

CONCLUSIONS

The random forest predicted with the quality scores from 1 to 10 with an accuracy of approximately 65%. It was mainly good at predicting the average wines. When we advanced and modified our data, the forest started perform with higher accuracy. Random Forest Analysis is widely used in almost every sector - banking,medicines, stock market, e-commerce and etc.