Predicting wine quality

Statistical Learning Methods for Decision Making

Yoann Boget, Florian Hochstrasser

2018-06-14

Contents

1 INTRODUCTION

The famous *vinho verde* is a tasty wine from northern Portugal. The number of producers, and by consquence the number of different *vinho verde*, is very large. Therefore, it is hard to have an exhaustive overview of all the varieties. So that, it can be of great interest to find the physicochemical properties of a good wine and to identify a system able to predict the quality of a wine.

The aim of this report consists in developping a model that may be used to identify the most influential explanatory variables to predict the wine quality. Then, we are looking for the best model in predicting wine quality.

Classification and predictive modeling is a broad field of research in statistics and a lot of different methods are available to deal with that kind of issues. In this work, we will test and compare four among the most common methods: general linear model (GLM), decision tree, neural net and random forest. By comparing the results of those techniques, we will be able to identify the method providing the best result for the issue of wine classification. However, the best prediction depends on what we are looking for. The accuracy or the kappa are good indicators of the general quality of a model, but according to the aim of the analysis, other indicators such as sensitivity, specificity, positive and negative predictive values can be interesting.

From the point of view of potential purchasers, the interest of the model consists in giving the best chances to pick a good wine while minimizing the possibility to classify poor wines as good. As we coded *good* as *1* and *poor* as *0*, the probability of choosing a good wine is given by the negative predicted value (the reference category is the one that is naturally first, hence, *0*). In consequence, we decided to put an emphasis on the negative predicted value. However, following the instructions, we will always keep an eye on the general accuracy of the model. So, we will compare the strength of the models looking mainly at the accuracy and the negative predicted value.

In order to have a better idea of the issue, we will first present the data we use for the analysis and provide a first explanatory analysis. In a second part, we will present the methods we have used and explain the main principle of the general linear model, the decision tree, the neural network and the random forest. In the third part, we will present the results for each method. Finally, we will conclude by a summary of the strength and the weakness of each model and propose some ideas for further investigation.

2 DATA

In this section, an exploratory data analysis is conducted. The results help to draw first conclusions about how to employ the learning methods later on.

2.1 STRUCTURE

The data at hand originates from a famous dataset on Portuguese red and white wines and is available on the website of the UCI Machine Learning repository.[1] In this analysis, a subset is used. It comprises 1599 observations.

The dataset studied consists of 11 continuous explanatory variables and a categorical response variable with 8 levels. The 11 explanatory variables describe physicochemical properties of the wines. Their respective values range in the orders of 1e-01 to 1e02.

2.2 RESPONSE

The response variable describes the quality of wines. It is an ordered categorical variable attributing a quality grade. The range in the data lies between 3 and 8. The cutoff for the quality is arbitrarily set at 7, meaning that we classified as *good* wines graded 7 or 8. The wines with a lower score are classified as *poor*. Before recoding, most of the wines are actually of medium quality, there are only few wines with a rating of 4 or less (see Figure 1, left). After recoding, the response is markedly unbalanced, since only 217 of the observations fall into the category *good* and 1382 in the category *poor* representing respectively 13.57 % and 86.43 % (see Figure 1, right).

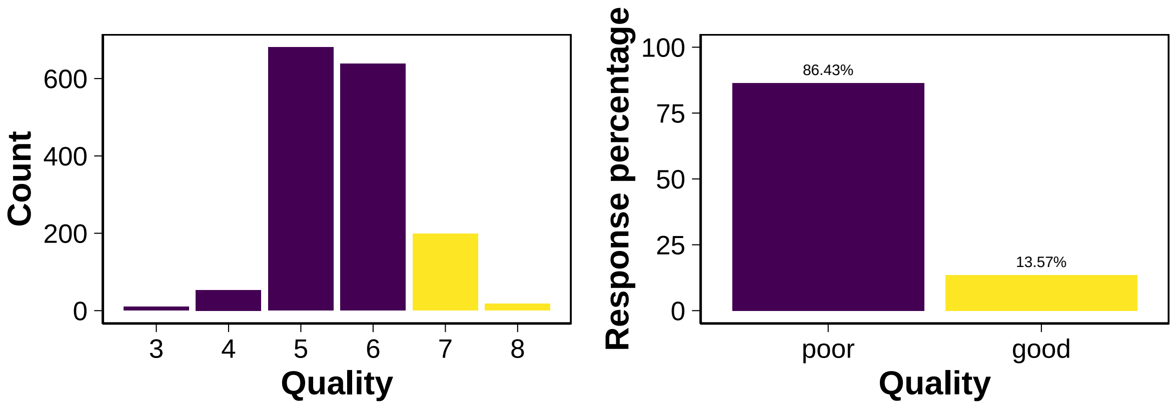

Figure 1: Frequency of response categories before recoding (left) and after recoding (right).

[1] https://archive.ics.uci.edu/ml/datasets/Wine+Quality

2.3 EXPLANATORY VARIABLES

The explanatory variables contain information about the physicochemical composition of the wines. The dataset provides measures for the following 11 constituents: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates and alcohol. All these variables are numerical. There is no missing value.

Boxplots for all variables are shown in Figure 2. It is evident that most variables have outliers. From the shape of the IQR (inner quartile range), we can see evidence that some variables seem to be clearly not normally distributed.

The boxplots allow to identify important variables where a marked difference in the distribution of variables is visible. For alcohol, this is most obvious. Volatile acidity and sulphates also appear to enable a good distinction between poor and good wines. In the opposite, pH, density and residual sugar look very similar for the two groups.

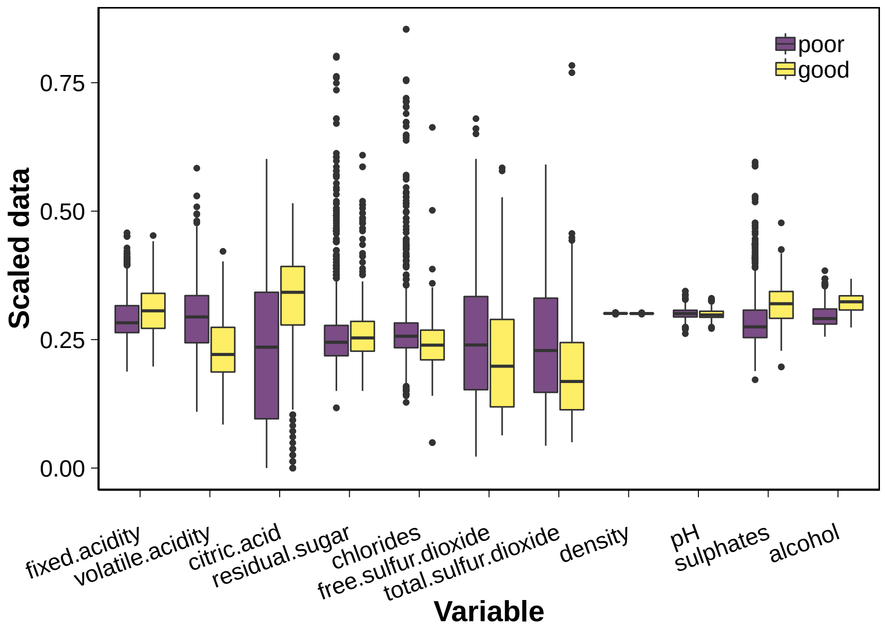

Figure 2: Boxplots of the explanatory variables by response category. The values were scaled to the interval [0:1]. Each variable is grouped by response category. The better the separation of the inner quartile range of the boxplots of a given variable, the better the variable is suited to separate observations by category.

2.4 CORRELATIONS

In Figure 3, the correlations between explanatory variables are shown. The correlations can be easily deduced with chemical reasoning. There is a cluster of negative correlations between variables that describe acidity and pH. Lower pH means more acidity. Free sulfur is positively correlated with total sulfur as the former is a fraction of the total sulfur. We also see correlations between density and various variables. Molecules that are heavier than water have a positive correlation with density, those lighter have a negative correlation.

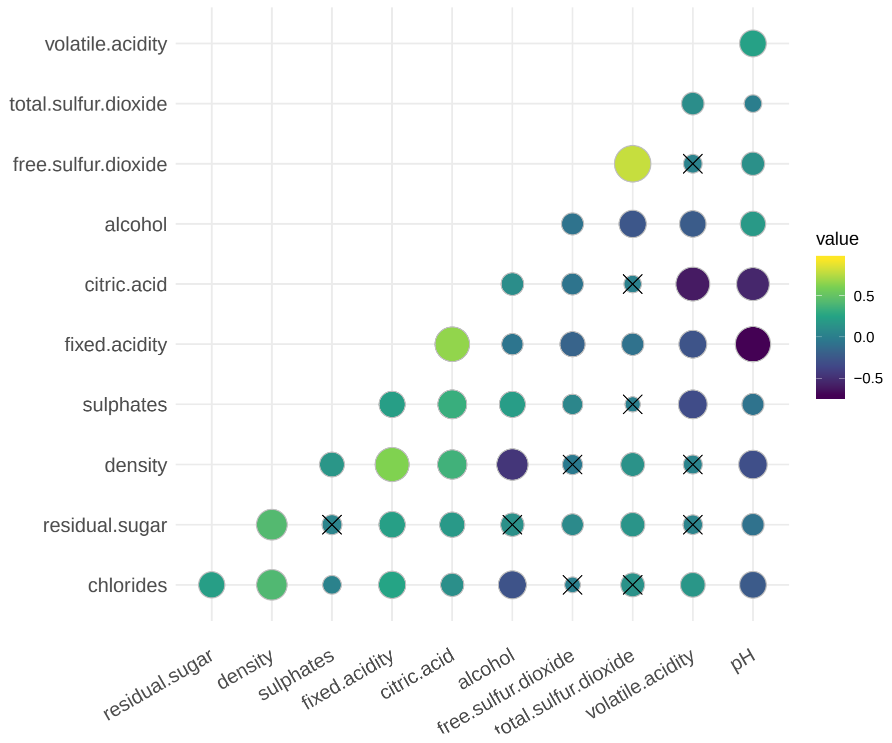

Figure 3: Correlations between explanatory variables. The size of the circles as well as their color indicates the strength and nature of correlation (positive = purple, negative = yellow, bigger = stronger). Correlations not significant at $\alpha = 0.05$ are crossed out.

2.5 PCA

In order to have a better representation of the data, we decided to plot a principal component analysis (PCA), presented in Figure 4. Since the classical two dimensional PCA represented a quite poor proportion of the variance, we decided to plot a 3D PCA, representing about 60% of the variance. The yellow spheres represent good wines, the purple ones poor wines. The PCA shows a region where good wines are concentrated and a region with almost only poor wines. However, we see there is an overlap of good and poor wines. This suggests that it could be quite difficult to separate the two categories. The PCA also suggests that some variables are more important than others to explain wine quality. The alcohol rate variable points in the region of good wines. It may be the case for the sulphates and citric acidity, but it is less clear. The volatile acidity points in the opposite direction suggesting an inverse correlation with wine quality.

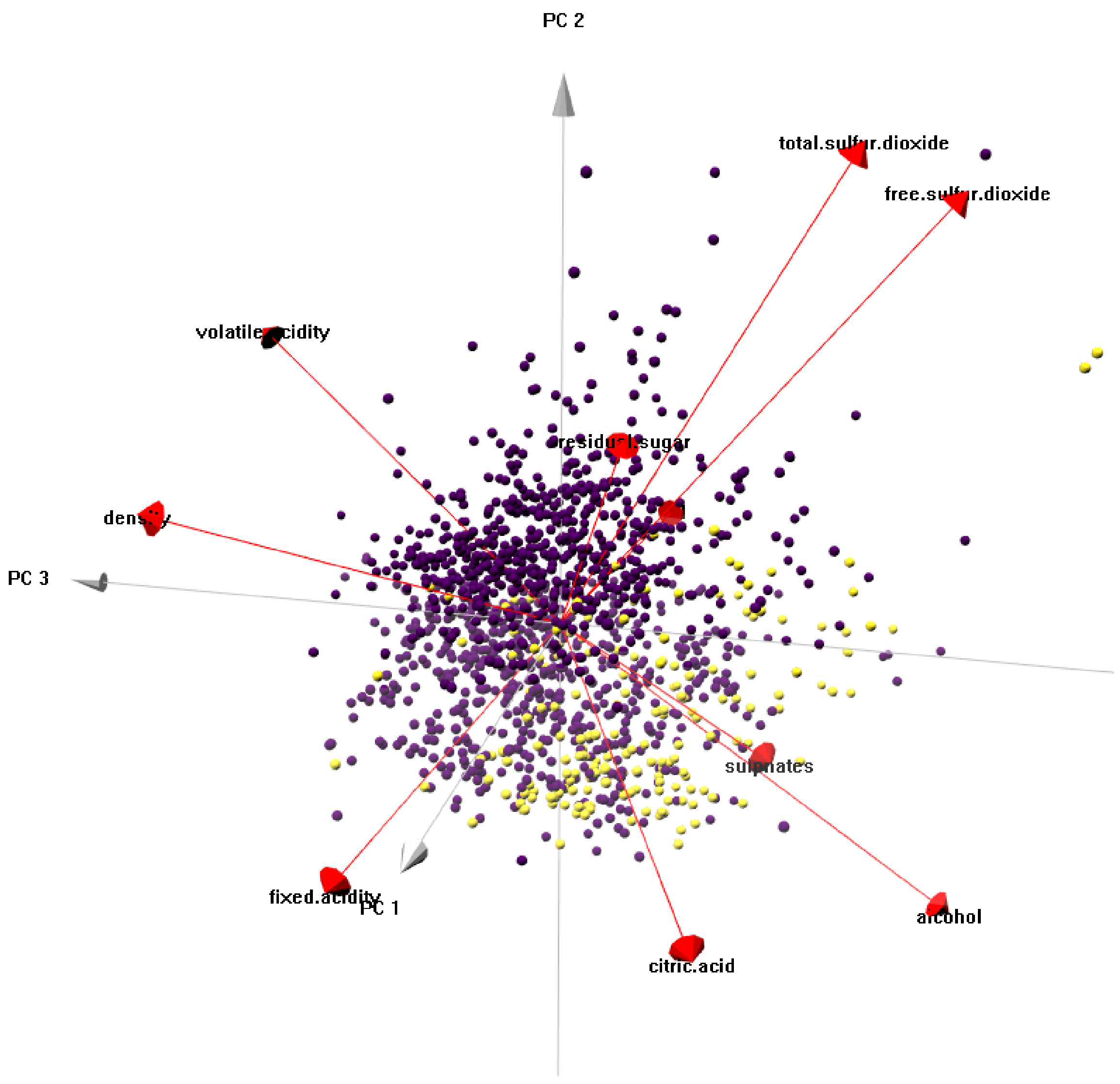

Figure 4: Three-dimesional principal component analysis. The purple spheres represent poor quality wines, the yellow spheres indicate wines of good quality. Clearly, there are two regions with mostly poor and mostly good wines and an overlapping region. We can identify alcohol, sulphates and citric acid and in the opposed direction volatile acidity and density as important variables to separate the categories.

3 METHODS

The following section describes the considered learning methods along with the respective settings used. The problem at hand suggests using a supervised learning method, as the structure of the data and hence the response (target attribute) are known. We decided to study decision trees, neural networks and random forests. Additionally, we studied a GLM model for comparison with the learners.

Special attention was also paid to the topic of sampling from the data to obtain training and test datasets.

3.1 SAMPLING

3.1.1 TRAINING AND TEST DATASETS

The same training and test datasets were used for all methods. The training dataset used for fitting the models comprises 70 % of the initial data, leaving the remaining 30 % for testing the models for their prediction performance.

The samples were drawn using the function `createDataPartition()` from package `caret` (Kuhn 2008). This function preserves the response frequencies of the original data as it creates data splits within the groups of a categorical variable, as opposed to using the generic `sample()` method from base R, which samples completely random.

3.1.2 DEALING WITH IMBALANCED RESPONSE VARIABLES

As was shown in Section 2.2, the response in our case is very imbalanced. When models are fitted naively on such data, the prediction results are usually not satisfactory because the minority category is not well captured by the model. In these cases, accuracy tends to correspond to the fraction of the majority class.

There exist different strategies to deal with imbalance. On algorithm-level, cost-sensitive approaches are popular. Less represented classes are assigned a higher cost for misclassification, which boosts their importance during the learning process. On data-level, different sampling approaches exist to mitigate imbalance. One approach is oversampling, where more observations are drawn from the minority class so as to better balance the response classes. It can be problematic to do so because there is a risk of over-fitting, i.e. it is possible that properties of the training sample are baked into the model and thus it will perform bad on predictions of new observations (Kotsiantis et al. 2006).

In this analysis, a data-level approach was implemented for all methods that were fitted through `caret`. When training the models through `train()`, the option `sampling="up"` was passed in through the `trainControl` mechanism. This balances the response categories by oversampling the minority category *good*.

3.2 GLM

Generalized liner model (GLM) extends the classical linear regression . In particular, it allows to consider categorical response variables. As for the classical framework, the model is based on the minimization of the mean squared error. For a binary response, the model provides the probability of being of one category for some given values of the explanatory variables.

The probabilities are then used to assign the observation to the corresponding category. Usually, if the probability is over 0.5, the observation will be predicted as 1 and as 0 if the probability is lower than 0.5. The cutoff at 0.5 will always provide the model with the best accuracy. However, it is possible to move this cutoff in function of the aim.

In our case, we have tried three different cutoffs in order to see how the prediction evolves. In the first case, we take the classical 0.5 cutoff, predicting as good wines with a probability higher than 0.5 and as poor the wines with lower probability. In addition, we also considered two other cutoffs: one at 0.7 and one at 0.1.

3.3 DECISION TREE

Decision trees are an intuitive approach that can be used for classification problems. Starting with the whole dataset, an initial split is searched that best reduces the impuritiy of the data (the variable that best separates the data based on a logical condition). On the resulting two branches, this procedure is repeated until only a small number of observations remain at the terminal nodes. Subsequently, the tree is pruned back. Pruning is usually done with either the so-called “1-SE method” or through cross-validation. Decision tree fitting for this case-study was performed with repeated cross-validation through `caret`. We used 10-fold cross-validation (training dataset is split into 10 groups). One run of cross-validation consists of holding each of the groups back for testing once while using the remaining groups to fit a model, then using the retained group for testing. This procedure was repeated 10 times.

The only additional option passed to `caret::train()` was the complexity parameter which was set to 2e-5. This effectively is a pre-pruning measure, limiting initial three growth.

3.4 NEURAL NETWORK

The neural network is a learning method of trial and error. The idea is to run the neural net and to improve progressively the result by changing the weights between layers. The inconvenience of the neural net is that it can be very difficult, not to say impossible, to understand the links and correlations done by the network. The neural network can be very efficient, but is in most of the cases hardly interpretable.

Neural net is a network of simulated neurons. It can also be considered as a nonlinear regression. Each layer consists in several neurons. A first layer contains as many neurons as input variables. These neurons take the values of the variables. Then, there are one or more hidden layers. Each neuron computes a weighted sum of the input values from the previous layer and from an intercept. The last layer is the response, each neuron corresponding to an answer. In our case, the last layer is composed of only one neuron indicating either *poor* or *good* as the predicted outcome.

To obtain the best result, it is possible to play with the number of hidden layers, the number of neurons in each layer and the decay. The decay is a penalization parameter to avoid overfitting.

For this report, we only tested networks with one hidden layer. The fitting was done using `caret::train()` with 10-fold cross-validation. The data was pre-processed using the option `preProcess="range"` which scales values to the interval [0:1]. As neural networks are sensitive to differences in value ranges between input variables, this is a necessary step. Caret tries several combinations of number of neurons in the hidden layer and decays. The metric used to decide on the best network was accuracy.

3.5 RANDOM FOREST

Random forests are a collection of decision trees. The trees are grown slightly different than when growing a single decision tree. In random forests, the splits are only chosen from a random subset of fixed size of the available explanatory variables. This limits computation time. By growing many trees, it can be expected that still, all explanatory variables will be used in the overall forest. Predictions in a random forest are a majority vote of the whole forest on a supplied observation.

For this report, the random forest was fitted using 10-fold cross-validation. An initial run with a size of 1000 trees showed that the error rates stabilised at just over 500 trees, so the forest size was reduced to 601 trees. Taking an odd number of trees eliminates ties in the votes.

Initially, we calculated the number of variables to be used in splits as 3 (argument `mtry` in caret, by taking `floor(11/3)` as is recommended in literature). These results were not satisfactory, so we let `caret::train()` determine the optimal number of variables.

4 RESULTS

This section describes the obtained models and gives some of their perfromance measures. A comparison of their performance and an ultimate rating can be found in the section 5.2 Comparison of method results.

4.1 GLM

4.1.1 MODEL

The logarithmic regression confirms the tendencies of the exploratory analysis. The coefficients with the highest significant differences between good and poor wines are sulphates, alcohol, fixed- and volatile acidity. After fitting a full model, the three variables citric acid, free.sulfur.dioxide and pH not significant at $\alpha = 0.05$ were removed to build a reduced model for predictions. Table 1 shows variables with their coefficients and the p-values of the final model:

Table 1: Coefficients and their p-values for each variable of the final GLM model.

Variable	Coefficient	P-value
Intercept	2.730e+02	0.014192
fixed acidity	3.640e-01	0.000150
volatile.acidity	-2.740e+00	0.000287
residual.sugar	2.351e-01	0.002487
lchlorides	-1.039e+01	0.034468
total.sulfur.dioxide	-1.007e-02	0.005895
density	-2.881e+02	0.009994
sulphates	3.513e+00	5.44e-09
alcohol	7.905e-01	6.13e-09

4.1.2 COMPARISON OF DIFFERENT CUTOFF LEVELS

A method to tune the resulting GLM model is to change the cutoff level. This is the threshold value for predictions where they are separated into the response categories.

First, we analyzed the data with the classical cutoff at 0.5. The predictions provided by this model are quite good. The accuracy lays at 0.875.

Second, we tried to change the cutoff in order to see if we were able to improve some of the indicators. Of course, moving the cutoff up will mechanically increase the negative predictive value and reduce the specificity. In the opposite, moving it down will weaken the negative predictive value and growth the specificity.

We present here two attempts with cutoff values at 0.7 and 0.1, so that it illustrates the changes. Surprisingly, moving the cutoff to 0.7 does not change the accuracy (0.875). Changing the cutoff does not increase the number of mistakes in that case. In the opposite, taking a cutoff at 0.1 lets the accuracy fall to 0.72. The changes in negative predicted value are very important. Putting the cutoff at 0.7 increases it to 0.67, making this model the best one with respect to this criterium. In the opposite, the cutoff at 0.1 has a terrible negative predicted value. We observe the reverse phenomenon for specificity (also compare to 5.2 Comparison of Method Results for details).

Table 2: Comparison of confusion matrices for the GLM approach at different cutoffs. Predictions are given in rows, while reference is shown in columns.

	Reference					
	Cutoff: 0.5		Cutoff: 0.7		Cutoff: 0.1	
	poor	good	poor	good	poor	good
poor	394	40	409	55	286	6
good	20	25	5	10	128	59

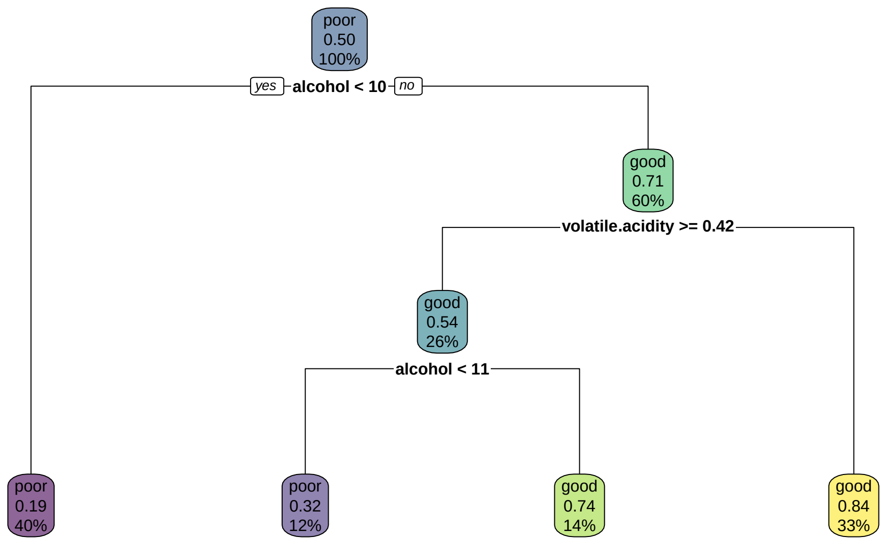

Figure 5: Pruned decision tree.

4.2 DECISION TREE

The obtained decision tree (see Figure 5) is of size 3. We see a non-linearity with alcohol appearing twice as the criterion. The prediction performance of the tree was not satisfactory. Achieved accuracy was at 0.777, which was low compared to the best performers, and negative predicted value only reached 0.358. This means that many poor wines were predicted good.

The three most important variables found are shown in Table 3. The measure “Importance” refers to the mean decrease in gini coefficient due to each variable.

Table 3: Important variables determined through the decision tree method.

explanatory	Importance
alcohol	297.65506
volatile.acidity	87.01844
citric.acid	73.74184

4.3 NEURAL NETWORK

We obtained the best accuracy for a neural net containing 5 neurons in the hidden layer and a decay of 1e-4. The network is shown in Figure 6, the line weights representing the magnitude of weights for the connections between the neurons. Purple lines are poor wines, the good wines are coded yellow.

Neural networks are often very efficient, but in our case-study the result is quite disappointing (see Table 4). As will be shown in the conclusion, the neural network is the second worst for accuracy and negative predictive value. Its results are not bad for specificity and sensitivity without being exceptionally good neither. Since neural networks are hard to interpret, we are not able to explain its bad performance in our case.

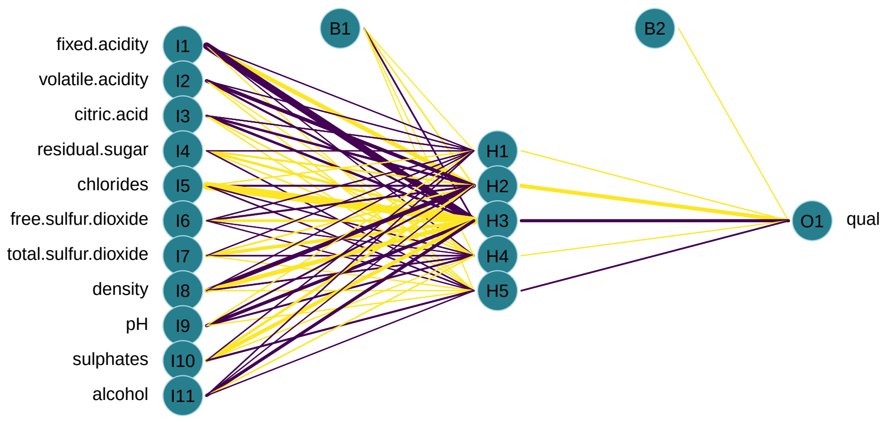

Figure 6: Structure of the neural net.

Table 4: Confusion matrix for predicted wines using the neural network.

	poor	good
poor	315	8
good	99	57

The important variables identified by the neural network are shown in Table 5.

Table 5: Important variables determined through the neural network method.

Explanatory	Overall
chlorides	89.14445
density	61.31375
sulphates	100.00000

4.4 RANDOM FOREST

For the random forest, the results with the training dataset look very promising . The error rate for good wine prediction dropped to zero very fast, meaning that on the training dataset, no misclassification for good wines was present (see Figure 7).

With accuracy as the performance measure, mtry was found at 6, producing an accuracy of 0.905.

Prediction performance measured on the test data was no longer perfect (see Table 6). This is an indication of overfitting. This means that the random forest was trained very well on the exact features of the good wines in the training dataset, but is not able to accurately predict good wines in the test dataset. Still, the results are among the best compared to all other methods studied, as will be discussed in detail in the conclusions.

The important variables found for the random forest are shown in 7.

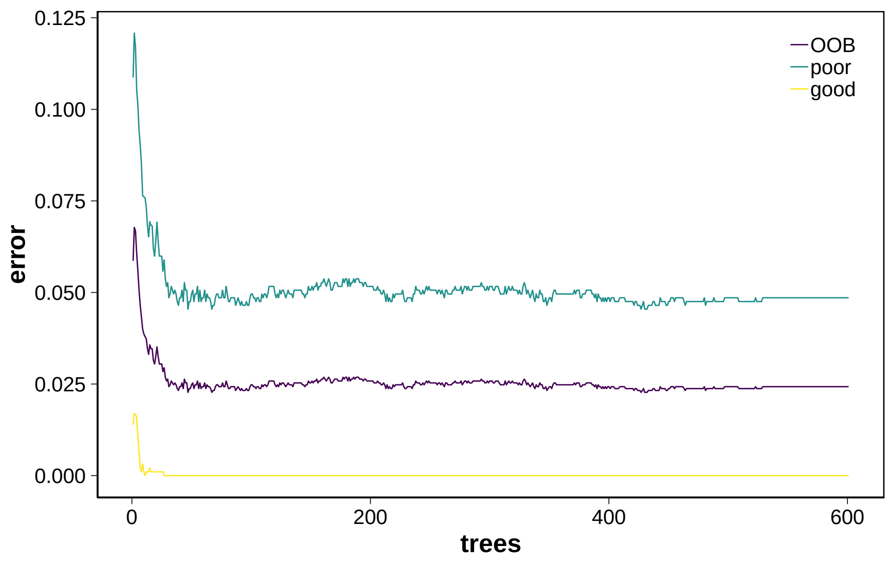

Figure 7: Convergence of error rates for the fitted random forest. For good predicitions, the error rate drops to zero very fast. At slightly above 500 trees, the error rates stabilize for poor prediction and out-of-box.

Table 6: Confusion matrix for predicted wines using the random forest.

	poor	good
poor	382	17
good	32	48

Table 7: The 3 most important variables in the random forest.

explanatory	Importance
volatile.acidity	102.0780
sulphates	129.0025
alcohol	282.0074

5 CONCLUSIONS

5.1 OVERSAMPLING

The approach of oversampling from the minority class yielded very good results in the case of the random forest. Comparison with initial trials without oversampling (not shown in this report) showed a large improvement in specificity and overall accuracy. However, the issue of oversampling became evident by comparison of the confusion matrices for training and test data. Still, the results with oversampling are promising.

5.2 COMPARISON OF METHOD RESULTS

As stated in the introduction, our goal was to find the best model for predicting a good wine while minimizing the probability of rating a poor wine as good. This corresponds to the best negative predictive value as the category good is coded as *1* in the data and is thus interpreted as the negative case when fitting the different models.

The results of the learning methods studied are presented in Table 8. The color code helps to identify the methods with overall high values.

Considering the negative predicted value, the best method is GLM with a cutoff at 0.7. The high cutoff, however, comes at a price: Overall, only few wines are predicted to be of good quality (see 2)indicated by the very poor specificity. Even if the results are quite good compared with other methods, the negative predictive value lays quite low.

Overall, the random forest performed quite good. It's negative predicted value is lower than GLM with a cutoff of 0.7, which in our definition of the criterum to determine the winner means it is on second rank. It's superior accuracy and good performance on specificity however mean that it might be the better choice.

It is obvious that GLM with a cutoff at 0.1, the decision tree and the neural network performed not satisfactory. GLM with the cutoff at 0.5 is somewhere between, which is noteworthy. A relatively simple model, naively employed, performed better than or at least comparable to elaborate learning methods.

Table 8: Direct comparison of the performance of all the learning methods studied. Darker cells indicate better values, green and yellow cells lower ones.

	GLM	GLM 0.7	GLM 0.1	Dec. Tree	Rand. For.	N-Net
Neg Pred Value	**0.556**	**0.667**	**0.316**	**0.358**	**0.6**	**0.365**
Accuracy	**0.875**	**0.875**	**0.72**	**0.777**	**0.898**	**0.777**
Sensitivity	**0.952**	**0.988**	**0.691**	**0.771**	**0.923**	**0.761**
Specificity	**0.385**	**0.154**	**0.908**	**0.815**	**0.738**	**0.877**

5.3 IMPORTANT VARIABLES

Sulphates was present in the top 3 of each method, while alcohol was among the 3 most important variables in all methods but the neural network. Following these two, we have seen chlorides, citric.acid, density, fixed.acidity and volatile acidity once (see Figure 8). The only method that is lacking sulphates is the neural network. Given the opaque nature of neural network fitting, it is not possible for us to indicate possible reasons for this observation.

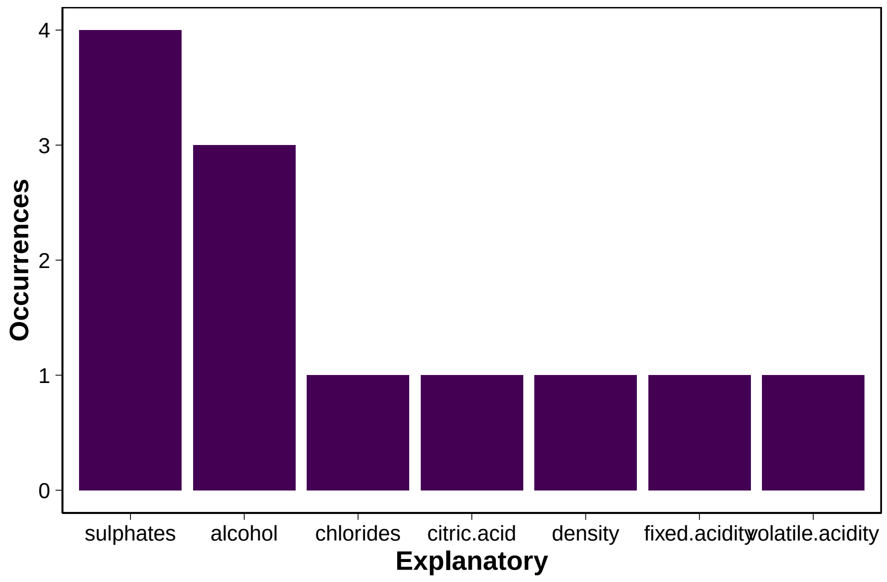

Figure 8: Aggregated important variables over all methods studied. For each method, the top 3 variables were extracted and their total occurrence counted.

5.4 OUTLOOK

In future analyses, it would be interesting to also look at other sampling techniques often advocated for imbalanced response categories, like SMOTE (synthetic minority oversampling technique) (Chawla et al. 2002) or ROSE (random over-sampling examples) (Lunardon, Menardi, and Torelli 2014).

Furthermore, a comparison with cost-sensitive approaches could be performed as a different way of dealing with imbalance.

It would also be interesting to consider different performance metrics. Especially ROC seems to be a popular metric for imbalanced response problems.

6 REFERENCES

Chawla, Nitesh V, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. "SMOTE: Synthetic Minority over-Sampling Technique." *Journal of Artificial Intelligence Research* 16:321–57.

Kotsiantis, Sotiris, Dimitris Kanellopoulos, Panayiotis Pintelas, and others. 2006. "Handling Imbalanced Datasets: A Review." *GESTS International Transactions on Computer Science and Engineering* 30 (1):25–36.

Kuhn, Max. 2008. "Building Predictive Models in R Using the Caret Package." *Journal of Statistical Software, Articles* 28 (5):1–26.

Lunardon, Nicola, Giovanna Menardi, and Nicola Torelli. 2014. "ROSE: A Package for Binary Imbalanced Learning." *R Journal* 6 (1).

7 APPENDIX

7.1 R SESSION INFO

The following output details the exact environment used for the calculations and result display of this report.

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] bindrcpp_0.2.2       NeuralNetTools_1.5.1 nnet_7.3-12
##  [4] gmodels_2.16.2       randomForest_4.6-14  rattle_5.1.0
##  [7] pca3d_0.10           FactoMineR_1.41      ggcorrplot_0.1.1
## [10] GGally_1.4.0         reshape2_1.4.3       Hmisc_4.1-1
## [13] Formula_1.2-3        survival_2.41-3      forcats_0.3.0
## [16] stringr_1.3.1        purrr_0.2.4          readr_1.1.1
## [19] tidyr_0.8.1          tibble_1.4.2         tidyverse_1.2.1
## [22] caret_6.0-79         lattice_0.20-35      rpart.plot_2.2.0
## [25] rpart_4.1-13         viridis_0.5.1        viridisLite_0.3.0
## [28] htmltools_0.3.6      rgl_0.99.16          data.table_1.11.4
## [31] kableExtra_0.9.0     knitr_1.20           gridExtra_2.3
## [34] ggplot2_2.2.1.9000   dplyr_0.7.5
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.1.0          backports_1.1.2       plyr_1.8.4
##   [4] lazyeval_0.2.1        RGtk2_2.20.34         splines_3.5.0
##   [7] crosstalk_1.0.0       digest_0.6.15         foreach_1.4.4
##  [10] gdata_2.18.0          magrittr_1.5          checkmate_1.8.5
##  [13] cluster_2.0.7-1       sfsmisc_1.1-2         recipes_0.1.2
##  [16] modelr_0.1.2          gower_0.1.2           dimRed_0.1.0
##  [19] colorspace_1.3-2      rvest_0.3.2           haven_1.1.1
##  [22] crayon_1.3.4          jsonlite_1.5          bindr_0.1.1
##  [25] iterators_1.0.9       glue_1.2.0            DRR_0.0.3
##  [28] gtable_0.2.0          ipred_0.9-6           kernlab_0.9-26
##  [31] ddalpha_1.3.3         DEoptimR_1.0-8        abind_1.4-5
##  [34] scales_0.5.0          miniUI_0.1.1.1        Rcpp_0.12.17
##  [37] xtable_1.8-2          htmlTable_1.11.2      magic_1.5-8
##  [40] flashClust_1.01-2     foreign_0.8-70        stats4_3.5.0
##  [43] lava_1.6.1            prodlim_2018.04.18    htmlwidgets_1.2
##  [46] httr_1.3.1            RColorBrewer_1.1-2    acepack_1.4.1
##  [49] pkgconfig_2.0.1       reshape_0.8.7         labeling_0.3
##  [52] tidyselect_0.2.4      rlang_0.2.0           manipulateWidget_0.9.0
##  [55] later_0.7.2           munsell_0.4.3         cellranger_1.1.0
##  [58] tools_3.5.0           cli_1.0.0             broom_0.4.4
##  [61] evaluate_0.10.1       geometry_0.3-6        yaml_2.1.19
##  [64] ModelMetrics_1.1.0    robustbase_0.93-0     nlme_3.1-137
##  [67] mime_0.5              RcppRoll_0.2.2        leaps_3.0
##  [70] xml2_1.2.0            compiler_3.5.0        rstudioapi_0.7
##  [73] e1071_1.6-8           stringi_1.2.2         Matrix_1.2-14
##  [76] psych_1.8.4           pillar_1.2.3          httpuv_1.4.3
##  [79] R6_2.2.2              latticeExtra_0.6-28   promises_1.0.1
##  [82] codetools_0.2-15       MASS_7.3-50            gtools_3.5.0
##  [85] assertthat_0.2.0       CVST_0.2-2             rprojroot_1.3-2
##  [88] withr_2.1.2            mnormt_1.5-5           parallel_3.5.0
##  [91] hms_0.4.2              grid_3.5.0             timeDate_3043.102
##  [94] class_7.3-14           rmarkdown_1.9          scatterplot3d_0.3-41
##  [97] shiny_1.1.0            lubridate_1.7.4        base64enc_0.1-3
## [100] ellipse_0.4.1
```