# Wine Classification Project

Judy Bowen

2/25/2022

## Executive Summary

The purpose of this project is to use R to build a recommendation system that helps people find a good wine, based upon the chemical composition of the wine and wine ratings from third party wine tasters. Modeling wine preferences may be useful not only for marketing purposes but also to improve wine production or support the oenologist wine tasting evaluations. The dataset is obtained as a download from the Kaggle website. As described on Kaggle.com, the dataset is related to red and white variants of the Portuguese "Vinho Verde" wine as compiled in Cortez et al, 2009.[1] This dataset is also available from the UCI machine learning repository[2]. Due to privacy and logistic issues, only physiochemical (inputs) and sensory (the output of 'quality', as ranked from 1 to 10) variables are available (e.g. there is no data about grape types, wine brand, or wine selling price).

Input variables are (based on physicochemical tests): fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates and alcohol content. The output quality variable is based on sensory data, as a score between 0 and 10. In this paper, the quality variable is further defined by a grouping into two broad categories: quality measures of 7 and above are defined as group 1 and measures below 7 group 0. The purpose of the model is to identify 'good' wines in group 1.

The data set is first described by correlation estimates and then by eigenvectors of principal component analysis, to discover relationships between the chemical composition input variables and the categorical quality output variable. Following the exploratory data analysis, several models were selected to find the best way to predict a 'good' wine: a random forest decision tree, a logistic regression model and a boosted gradient model are designed and tested.

The results of the data analysis showed that of all models tested here for identifying 'good' wines, none could satisfactorily identify all of the good wines in the test data set. The options for identifying good wine were either of two: select the models which identify a small portion of good wines but which did not also incorrectly identify a larger number of poor wines as good wines or; identify most of the good wines but also falsely identify a large number of poor wines as good wines. Improving the models by moving the threshold of quality in the models has been identified as an unsatisfactory fix to resolve the problem.

### Dataset Description and Analysis

The dataset was downloaded from the Kaggle website[3] . There are no empty data cells. However, there are duplicate entries in the database. The original dataset contained 12 columns and 1599 rows. After running the R duplicated function to remove duplicate records, the database contained 12 columns and 1359 rows.

There is no evidence that the duplicate entries were more frequently associated with a particular quality variable, when taking the fraction of duplicate records over total record count by each quality group:

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

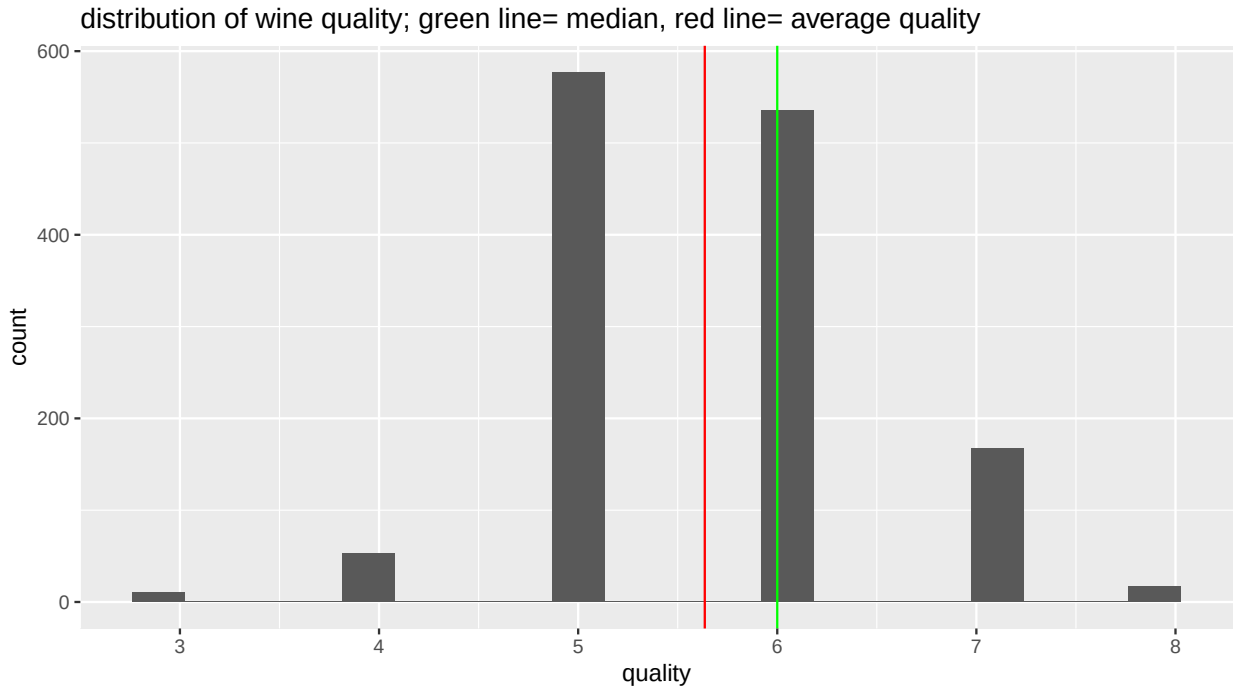
[2] https://archive.ics.uci.edu/ml/datasets/wine+quality

[3] https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

Table 1: Duplicate record portions removed from each wine quality class

| quality | duplicate_count | original_count | Fraction_of_original_count |
|---|---|---|---|
| 3 | 0 | 10 | 0.00 |
| 4 | 0 | 53 | 0.00 |
| 5 | 104 | 681 | 0.15 |
| 6 | 103 | 638 | 0.16 |
| 7 | 32 | 199 | 0.16 |
| 8 | 1 | 18 | 0.06 |

The following paragraphs describe the dataset, with duplicate records removed. A histogram of the wine quality variable indicates that the distribution is expected to be normal with most wines classified in the middle qualities of five and six, with fewer wines of poorer quality 3 and 4 and good quality of 7 and 8 classified at the lower and upper tails, respectively :




In later sections, where classification models are used to predict a 'good' wine (wines with a quality rank of 7 or more) versus a 'poor' wine (wines with a quality rank of 6 or less), a class imbalance is in evidence. It is evident, looking at the histogram that wines ranked as 'good' are far outnumbered by wines ranked as 'poor'. The class imbalance should be acknowledged when estimating and evaluating the models. Further discussion on this matter is presented later in this report.

In general terms, the database is described as having moments as described in Table 2 for the chemical components and the quality descriptor of the wines:

The mean of the distribution of quality in the wine database is 5.62. The median is located to the right of the average rating at 6. As the mean of the distribution is less than the median, the distribution is negatively skewed.

The chemical composition variables were scaled to a z-distribution, to permit reliable estimations of the relations and contributions of each to the prediction models. The quality variable is not scaled. The distributions of the scaled variables are shown in Table 3.

Table 2: Moments of wine quality database, duplicates removed

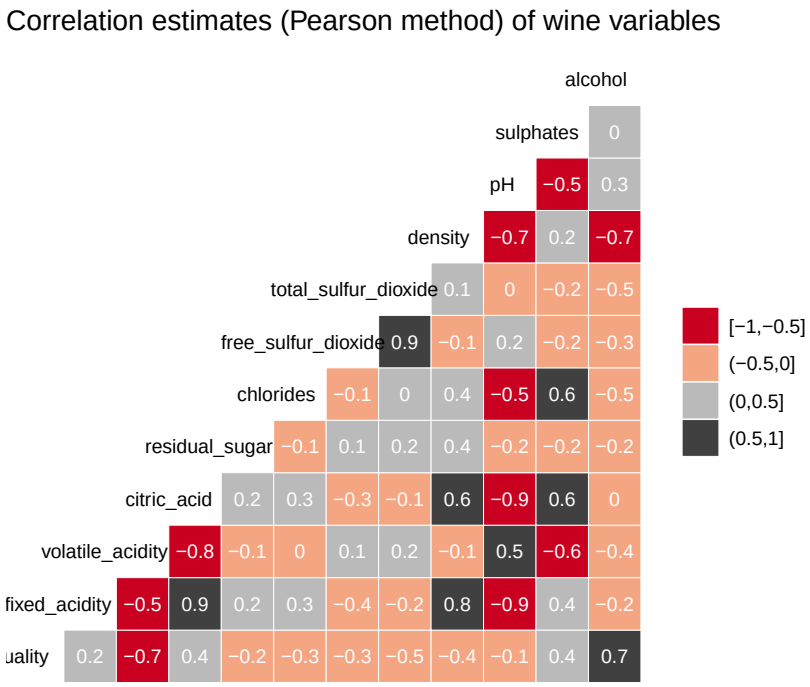
| | mean | median | min | max |
|---|---|---|---|---|
| fixed acidity | 8.31 | 7.90 | 4.60 | 15.9 |
| volatile acidity | 0.53 | 0.52 | 0.12 | 1.6 |
| citric acid | 0.27 | 0.26 | 0.00 | 1.0 |
| residual sugar | 2.52 | 2.20 | 0.90 | 15.5 |
| chlorides | 0.09 | 0.08 | 0.01 | 0.6 |
| free sulfur dioxide | 15.89 | 14.00 | 1.00 | 72.0 |
| total sulfur dioxide | 46.83 | 38.00 | 6.00 | 289.0 |
| density | 1.00 | 1.00 | 0.99 | 1.0 |
| pH | 3.31 | 3.31 | 2.74 | 4.0 |
| sulphates | 0.66 | 0.62 | 0.33 | 2.0 |
| alcohol | 10.43 | 10.20 | 8.40 | 14.9 |
| quality | 5.62 | 6.00 | 3.00 | 8.0 |

Table 3: Moments of standardized wine quality database, duplicates removed

| | mean | median | min | max |
|---|---|---|---|---|
| fixed acidity | 0 | -0.236 | -2.14 | 4.37 |
| volatile acidity | 0 | -0.052 | -2.24 | 5.74 |
| citric acid | 0 | -0.063 | -1.39 | 3.72 |
| residual sugar | 0 | -0.239 | -1.20 | 9.60 |
| chlorides | 0 | -0.185 | -1.54 | 10.59 |
| free sulfur dioxide | 0 | -0.181 | -1.43 | 5.37 |
| total sulfur dioxide | 0 | -0.264 | -1.22 | 7.25 |
| density | 0 | -0.005 | -3.55 | 3.74 |
| pH | 0 | 0.001 | -3.68 | 4.52 |
| sulphates | 0 | -0.227 | -1.93 | 7.86 |
| alcohol | 0 | -0.215 | -1.88 | 4.13 |

Understanding the database includes an understanding of the explanatory variables: the explanatory variables are chemical components that have been selected in this model to be representative of wine quality. But other studies of chemical analysis of wine are looking at a analyses that do not necessarily replicate the list offered in this database. The interaction of the chemical components in the wine is not perfectly understood; otherwise, chemists would be designing wines to perfection. There would be no need for tasting wines. But, for a data scientist there remains a problem: should one or any of the chemical components in the database should be removed? As I explain later on, data analysis does not easily reveal which of the chemical explanatory variables is 'redundant' and therefore easily removed to improve a model.

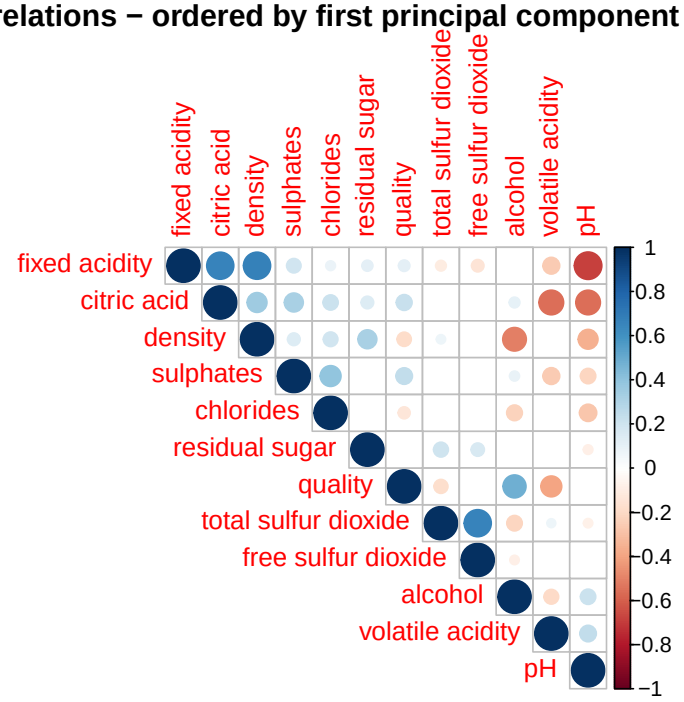
Corrplot is used to describe the correlation patterns between the chemical composition variables and the subjective quality variable (which, for correlation analysis, remains in the range between 3 to 8):




As can be observed in above correlation diagram, there are a few very strong correlations (as indicated by the red and black coloured squares) and many weaker relationships between the variables. Noted is that alcohol is showing strong linear correlation to the quality measures, with sulphates showing correlation to a lesser degree. The correlation diagram below indicates which of the correlations between the variables are statistically significant; blank spaces indicate no statistical significance. The presence of significant correlations between explanatory variables might indicate that multicollinearity would be a problem in models created to identify 'good' wines.

```
 ##keep##significance level for correlation is at 0.01 for p.mat
testRes <- cor.mtest(winequality_numeric, conf.level = 0.95)

Fig3_winequality <- corrplot(cor(winequality_numeric), type="upper", order="FPC",
                    p.mat=testRes$p, sig.level = 0.01, insig='blank',
                    title = "Correlations - ordered by first principal component",
                    mar=c(0,0,.75,0), number.cex = .7)
```




Principal component analysis is used to describe which of the variables are contributing the most towards describing the variance in a classification model, so that such variables could be dropped to reduce multi-collinearity or redundancy in the data.

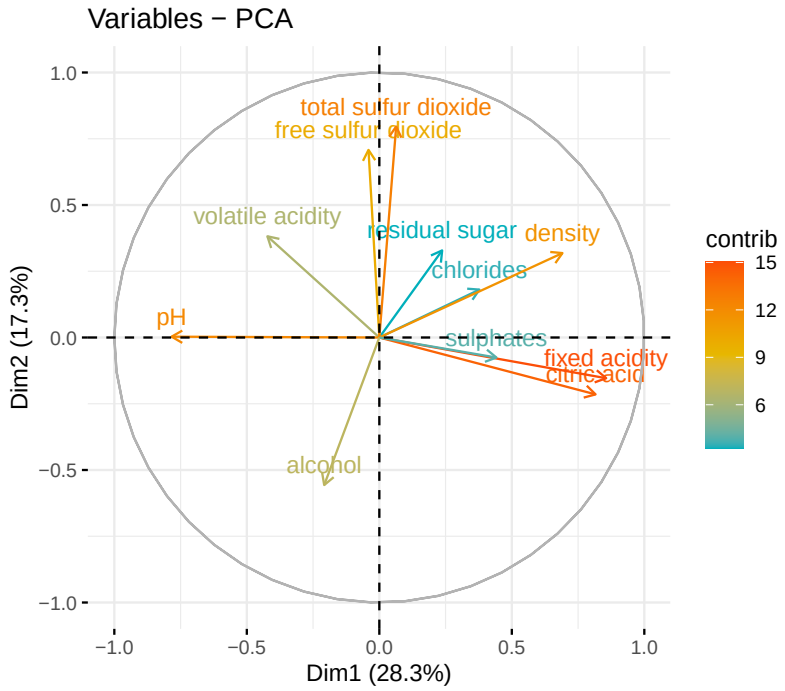
Principal component analysis is carried out using the R packages FactoMineR and factoextra[4]. Many authors have described the mathematics of principal component analysis and the interpretation of the eigenvalues and eigenvectors produced[5], [6]. In this paper, my purpose is limited to identifying the variables which are most likely to contribute to identification of a good wine. However, principal component analysis (as modelled in this paper) is limited to an understanding as it relates to linear relationships between the variables; I suspect that the chemical components of wine may not be easily explained by linear equations and therefore must view the principal component analysis results with caution.

Graphing the PCA components reveals that the first component explains 28 % of the variance while the second component explains 17% of the variance, representing 45% of the information of the original dataset. Variables positively correlated as they contribute to the principal components, are grouped together and negatively correlated variables are on the opposed quadrants. The colours of the variable vectors indicate their contribution to the principal components (principal component 1 and principal component 2). A higher percent indicates a greater contribution to the Principal component, towards explaining variance in the dependent variable.

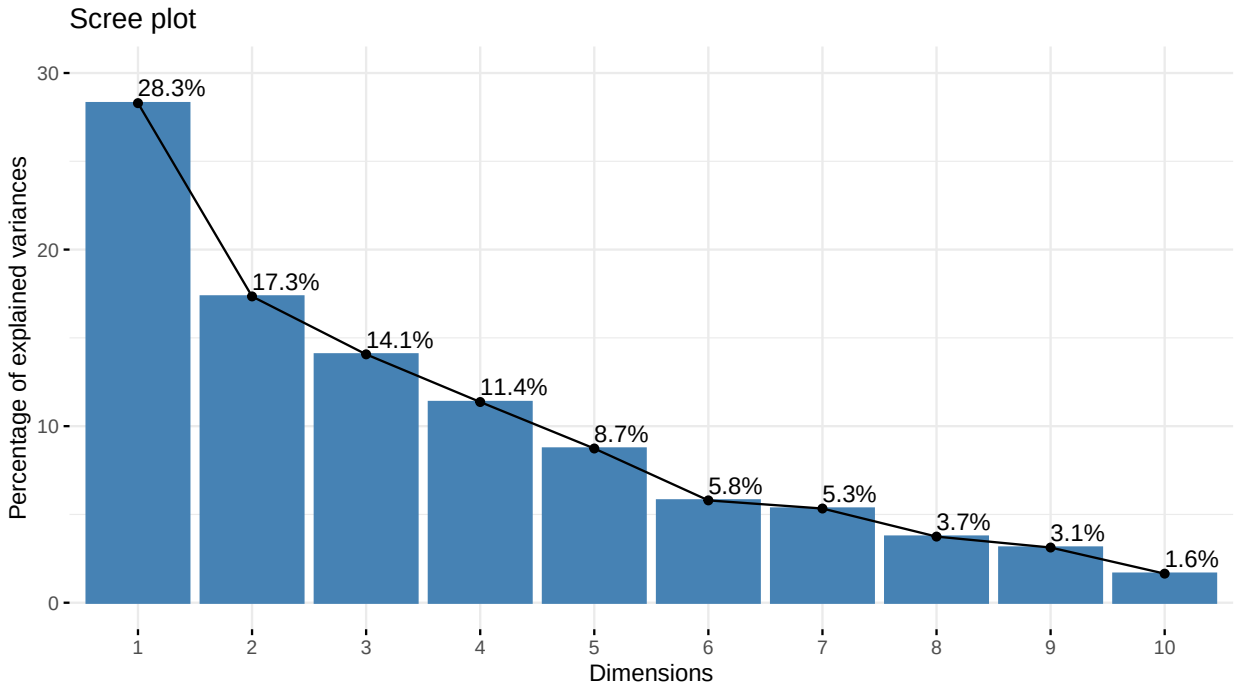
[4]http://www.stdth.com/english/rpkgs/factoextra as described in analytical methods presented in Practical Guide to Principal Component Methods in R, Alboukadel Kassambara, 2017

[5]A nice chatty discussion is found at https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

[6]https://en.wikipedia.org/wiki/Principal_component_analysis



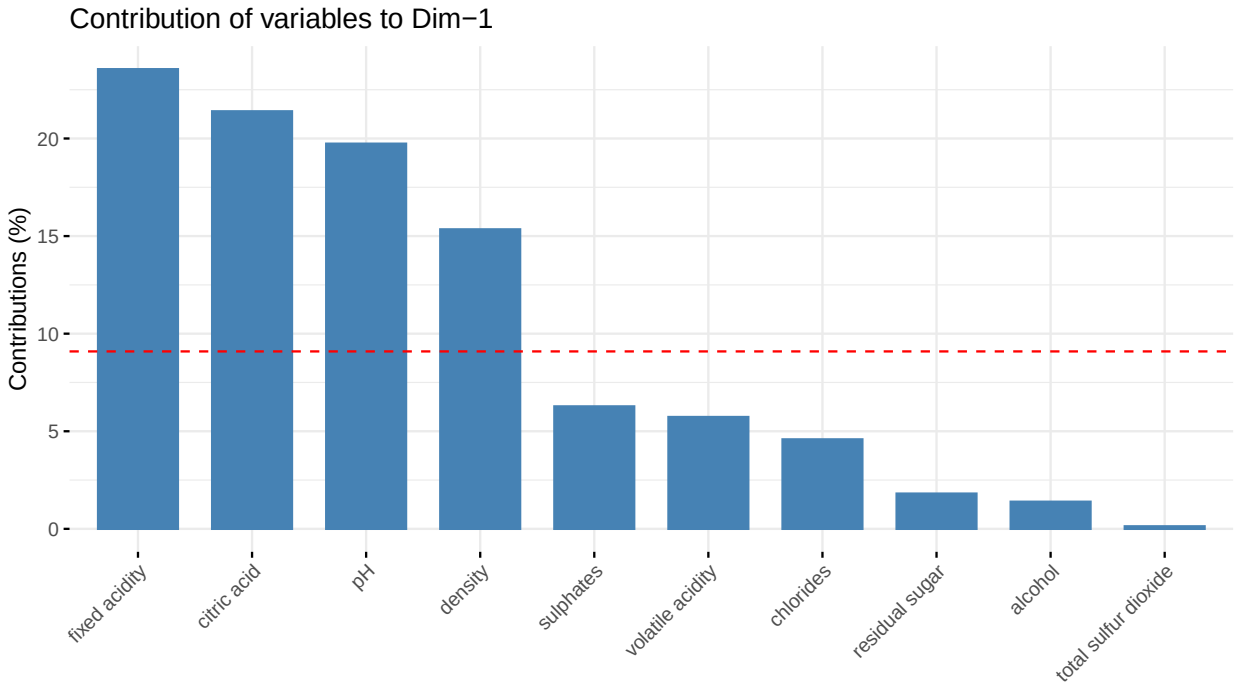

Looking at all dimensions of the principal components, we look at the scree plot of the principal component dimensions to find the contribution (toward explaining the variance in the wine quality) of each dimension. Of course, the contribution of each successive dimension diminishes:



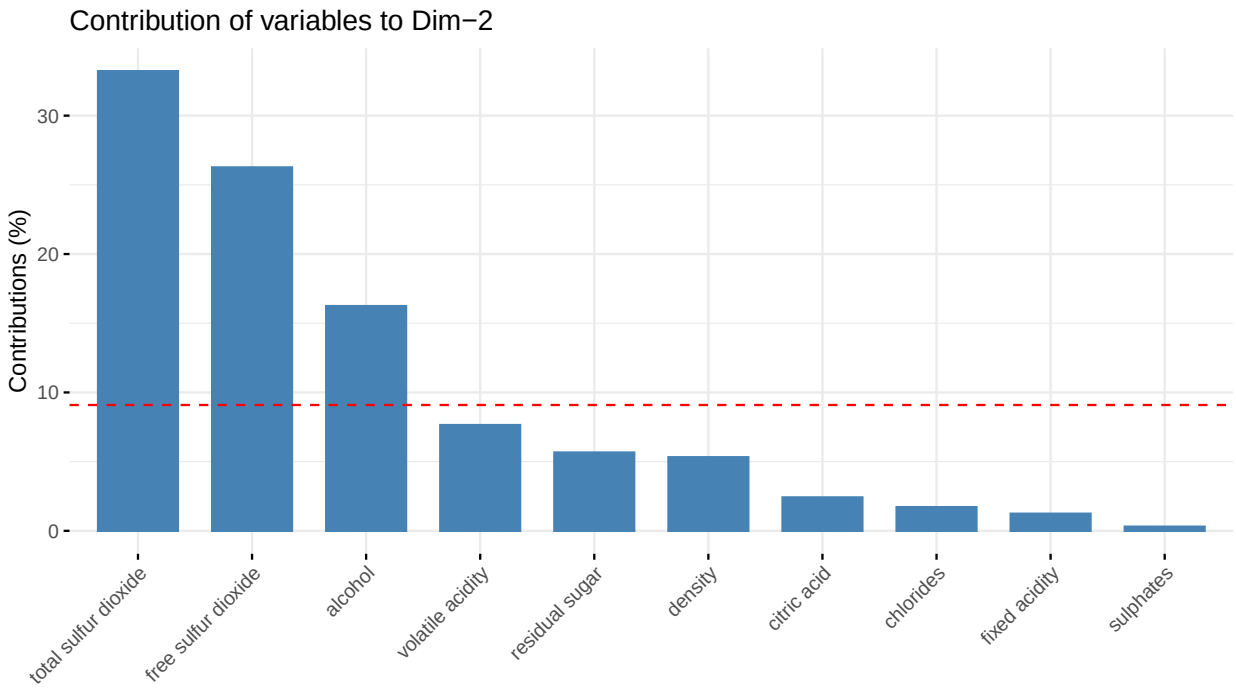

Kassambara (2017)[7] states that the contribution of variables in accounting for variability of principal components can be expressed as a percentage, whereby variables that are correlated with PC1 (in dimension 1) and PC2 (in dimension 2) are the most important in explaining the variability in the dataset. Variables that are not well correlated with any principal component or which are correlated only with

[7] in Practical Guide to Principal Component Methods in R, Alboukadel Kassambara, 2017, Section 3.4.2.4 Contributions of variables to PCs, page 24.

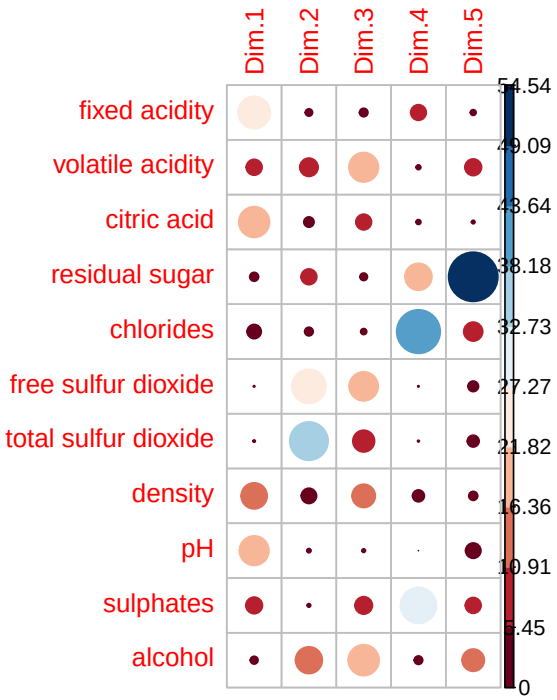
the last dimension principal components are variables with low contribution and may (under the right circumstances) be removed to simplify the analysis. In the diagram above (Variables – PCA), the variables of low contribution are represented by a short vector towards the circumference of the circle representing the first two dimensions. The variables closer to the circumference may be more important in explaining the variability in the dataset. These concepts are represented again below. The contribution (in percentages) for a variable is described for the first two dimensions. For the first dimension, the top contributors are fixed.acidity, citric.acid, pH, and density. The red line on the illustration indicates the expected average contribution of the variables, if the contribution of each variable were uniform.




We look at the top contributors to the second dimension, below:




Viewing the contribution of each variable to the top five dimensions:



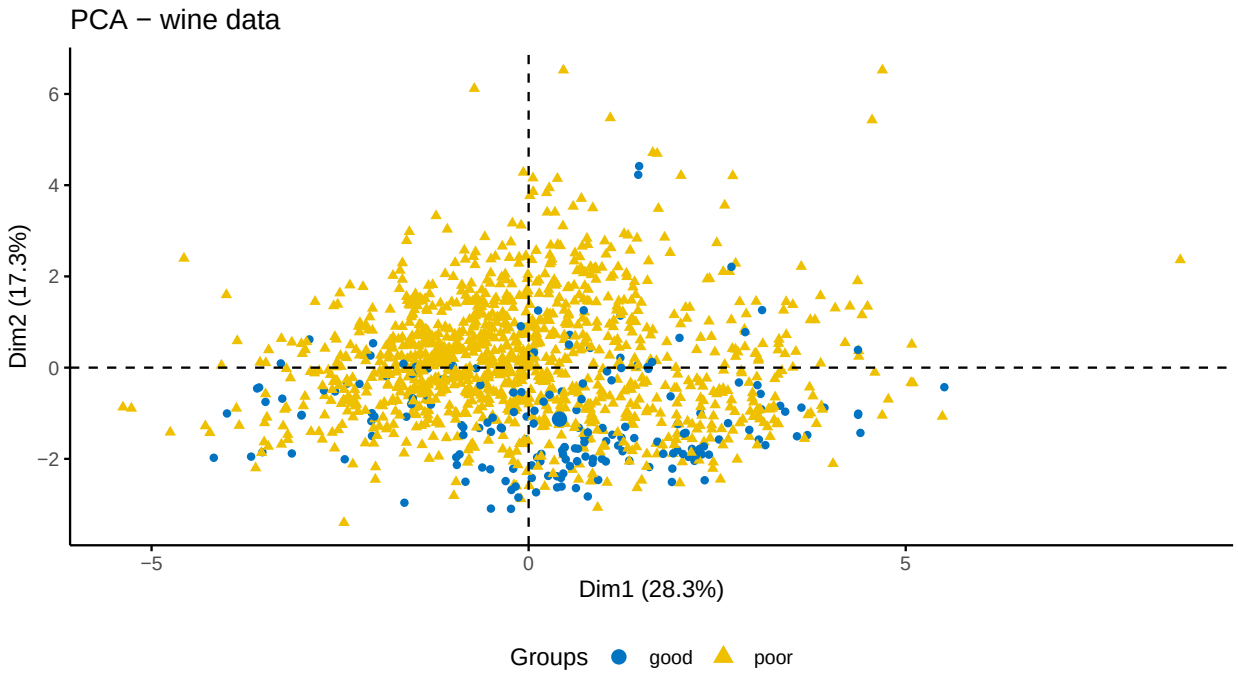

In dimension 2, we see a shift in the contribution of the variables: total.sulfur.dioxide, free.sulfur.dioxide and alcohol contribute above the expected average contribution. If we look at the contributors in in each of the dimensions, we see that chlorides and sulphates are consistently below the expected average contribution. Recalling from the PCA graph of variables that 1) the contribution of chlorides (to explaining variance in the dataset) are highly correlated with density and 2) the contribution of sulphates are highly correlated with

fixed.acidity. Should these two variables be dropped from the classification models in hopes of eliminating 'noise'? We see in the above diagram that chlorides and sulphates are the highest contributors to dimension 4; we can also see that dimension 4 contributes only 11% towards explaining the variance in the model (a similar line of reasoning applied when viewing the contribution of residual.sugar in dimension 5). If the contributions of variables in the dimensions further from the first orthogonals are weak, are they merely contributing to redundancy in the model and do we gain by including these variables in a model? (no answers yet…but, in later sections we find that alcohol is the leading contributor to our models and that sulphates are often ranked second in importance).

Of course, there remains possible complications from correlations between other variables which will kept in mind when evaluating the classification models. Reviewing the correlation coefficients as estimated using corrplot, sulphates and chloride columns were not showing exceptional correlation estimates with each other; sulphates showed weak correlation to quality. The highly correlated variables, including total.sulfur.dioxide and free.sulfur.dioxide show in PCA analysis as good contributors towards explaining the variance in the dataset. But, this is as expected: Variance does not capture the inter-column relationships or the correlation between variables . Underlying all analyses of correlation and of the principal components is the assumption that the relationships between the explanatory variables is linear and that these linear relationships can be rearranged to be represented by the linear correlation and principal component functions. If the relationship between a few or many of the variables in this model are non linear, then linear analytical tools may be misleading.

Finally, we take a look at the distribution of the wine quality groups (group 1 is all wines with quality assignments greater than or equal to 7; group 0 is all other quality groups) on the principal component dimensions 1 and 2:

Assessing the distribution over the dimensions, it seems possible to identify groups of good quality wines (blue dots) versus the poor quality wines (yellow triangles). There is overlap in evidence across the two dimensions. However, the good wines seem to occupy the dimension 1 quadrant with a greater presence than the poor wines. Also in evidence are the outliers – which were not removed from the sample but perhaps should be if a linear model is preferred. However, these outliers are showing up in a mapping of the first two principal components. What is identified as an outlier on this diagram may not be identified as such in a multidimensional space, which I do not attempt to illustrate.



## The Classification Models

I expect, looking at the correlation analysis and the principal component analysis, the model describing the classification of wines between poor and good may be nonlinear. I decided to first try a cross validated logistic regression classification model expecting this classification model to perform less well than other models. I also attempt a random forest classification model and a gradient boosted algorithm classification model.
The purpose of all classification models is to identify a 'good' wine, as identified by a quality rank of 7 or more.

In the red wine data set we are working with, we know that 'good' wines (as identified by a quality of 7 or over) are far outnumbered by 'poor' wines (identified by a quality of 6 and under): there are 1175 poor wines and 184 good wines. The dataset has been randomly split into the train set (80% of the total dataset with 145 good wines and 942 poor wines) and the test set (20% of the total data set with 39 good wines and 233 poor wines). After creation, the same train set and the test set were used in all models in this report. For the logistic model and the gradient boosted models, the train set was subsequently reweighted (from itself) and adjusted to account for class imbalance, as poor wines clearly outnumber good wines in all data sets. The test set remains in its original form (39 good wines, 233 poor wines). As the test sets are the same across all models, comparisons of the performance of the models are easily interpreted if I decide that my models are well defined (which is questionable, once I try to understand everything there is to know about the caret package and to explain all the metrics and tuning of model parameters, beginning with the train control functions).

### The Logistic Model

The logistic classification model was created to predict a classification of wine to good or poor based upon the chemical components in the red wine database. A cross validated model was used, as cross validation is recommended as the best modelling structure to avoid misrepresentation of the actual relationships due to a 'lucky' sample of the population of regressors[8] . Cross validation is an attempt to make use of repeated sampling from a subset of a population, to create a better portrait of the unseen population. As there is some evidence that a class imbalance exists in the data set (where the good wines are far outnumbered by poor wines in the sample), I include a logistic regression where the train set was resampled using 'down' sampling (the count of poor wines was reduced to equal the count of good wines in the train set).

The cross validated logistic classification model was created using the train and predict functions of the caret package.

```
cntrlspecs_cv <- trainControl(method="cv",
                              number=10,
                              savePredictions="all",
                              classProbs=TRUE)


set.seed(5627)

logistic_model_cv <- train(winequal1 ~ .,
                           data=data_train2,
                           method="glm",
                           family="binomial",
                           trControl=cntrlspecs_cv)
```

Tests were conducted to determine whether multicollinearity should be considered as a problem in the models. The tests were conducted using the multiColl package. The results of the test shown indicate the variable

[8] https://rafalab.github.io/dsbook/cross-validation.html

Table 4: VIF measures of multicollinearity

| | |
|---|---|
| fixed acidity | 8.13 |
| volatile acidity | 1.73 |
| citric acid | 3.13 |
| residual sugar | 1.66 |
| chlorides | 1.59 |
| free sulfur dioxide | 2.03 |
| total sulfur dioxide | 2.33 |
| density | 6.33 |
| pH | 3.34 |
| sulphates | 1.50 |
| alcohol | 3.02 |

inflation factor (VIF) of each independent variable in the dataset. The rule of thumb is that the VIF should not exceed 10. The VIF of the variables does not indicate a variance inflation factor greater than 10 for any of the chemical components, and therefore adjustments to correct the classification models for multicollinearity are not productive. However, it is noted that fixed.acidity has a VIF of 8.13 and density is at 6.33; these measures are a little high. For the most part, the VIF measures remain well under 5. Multicollinearity, by the measures available, does not seem to be a serious issue.

**The Random Forest Model**

Random forest models are a learning tool for classification[9] [10]. A random forest is a supervised machine learning algorithm that is constructed from decision tree algorithms [11] . It can be more accurate than a decision tree algorithm as it takes iterations of many decision trees before an optimal classification algorithm is obtained, without the overfitting encountered by decision trees. Random forests are forgiving when encountering outliers [12].

The random forest algorithm was tuned to find the best mtry (the number of predictors used in the decision node of each tree), number of trees (Ntree), node size (per Probst et at (2019): The nodesize parameter specifies the minimum number of observations in a terminal node. Setting it lower leads to trees with a larger depth which means that more splits are performed until the terminal nodes) and the best decision for maximum nodes of the trees (the maximum number of terminal nodes that the trees in the forest can have). Cross validation was used to select these features (with the caveat: as if I were fully aware of what I was doing to the model). The tuning of the random forest model was done using the caret package for train functions. Noted from a paper by Probst et al (2019)[13] :

> "Note that the convergence rate of RF does not only depend on the considered dataset's characteristics but possibly also on hyperparameters. Lower sample size (see Section 2.1.2), higher node size values (see Section 2.1.3) and smaller mtry values (see Section 2.1.1) lead to less correlated trees. These trees are more different from each other and are expected to provide more different predictions. Therefore, we suppose that more trees are needed to get clear predictions for each observation which leads to a higher number of trees for obtaining convergence."

After reading Probst et al, I decided to go with a model with a higher nodesize (at 8) and for a higher number of trees (450) than my (inexpert) tuning exercises indicated. To be noted: if I 'tune' mtry and then

[9] https://www.blopig.com/blog/2017/04/a-very-basic-introduction-to-random-forests-using-r/
[10] https://www.listendata.com/2014/11/random-forest-with-r.html
[11] https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/
[12] https://datascience.stackexchange.com/questions/54751/when-to-use-random-forest
[13] https://arxiv.org/pdf/1804.03515.pdf

nodesize and then number of trees, is number of trees optimal if there were unknown interactions between the first test on mytry and the second tuning test on nodesize? I ran my final random forest model using a tuning grid to select the best mtry. I also discovered that 'tuning' my model was 'over training' because as I tuned, without really understanding all the interactions between the metrics, I get what I identified as 'the best' random forest model. And then proceeding further, I created a tune free nodesize/tune free maxmode/tune free ntree model, to have a look at what all my tuning had wrought. Apparently, increasing the node size has the model select more good wines, but also has the model falsely identifying the poor wines in close proximity (to good wines) as good wines. I am still not sure what the effects of mtry and ntrees have on the outcome of the model (I don't have time to test everything; that would take another project).

The random forest model, with tuned metrics for nodesize, maxnodes, ntree and mtry on a tune grid from 1 to 10:

```
set.seed(5627)
tuneGrid <- expand.grid(.mtry = c(1: 10))
trControlRF <- trainControl(method = "cv",
                            number = 10,
                            search = "grid",
                            classProbs=TRUE,
                            summaryFunction=twoClassSummary)

rf_wine <- train(winequal1 ~ ., data=data_train2,
                 metric="ROC",
                 tuneGrid = tuneGrid,
                 trControl=trControlRF,
                 importance=TRUE,
                 nodesize = 8,
                 maxnodes=19,
                 ntree = 450)
```

The random forest model with 'no tuning' of nodesize, maxnodes or ntree, with mtry on a tune grid from 1 to 10:

```
set.seed(5627)
tuneGrid <- expand.grid(.mtry = c(1: 10))
trControlRF2 <- trainControl(method = "cv",
                             number = 10,
                             search = "grid",
                             classProbs=TRUE,
                             summaryFunction=twoClassSummary)

rf_wineNT <- train(winequal1 ~ ., data=data_train2,
                   metric="ROC",
                   tuneGrid = tuneGrid,
                   trControl=trControlRF2,
                   importance=TRUE,
                   #nodesize = 8,#
                   #maxnodes=19,#
                   #ntree = 450#)
)
```

### The gradient boosted algorithm

Gradient boosted algorithms[14] are a decision tree algorithm. Unlike the random forest algorithm which builds an ensemble of deep decision trees, the gradient boosted model algorithm (gbm) takes an ensemble of shallow and weak successive trees with each tree learning and improving on the previous tree. Each iteration works toward a better understanding of the unexplained variance of a previous tree. In this way, gradient boosted algorithms are sometimes more powerful than a random forest model. The gradient boosted algorithm performs well with weak models as it is permitted to learn slowly from past iterations. In this report, the gradient boosted algorithm is a cross validated model which uses the caret package. The model was trained preselecting the Bernoulli distribution as the probability model as the selection between a good and a poor wine is binomial. The r programming for the gradient boosted model was adapted from a paper by Martin (2016)[15] and Martin(2017)[16]. In the 2016 paper, Martin describes efforts to correct for class imbalance in the models and describes the r programming required to use sample weights to deal with the challenges of dealing with an outcome where one class heavily outweighs another. Four gradient boosted models were created (lengthy descriptions of each adjustment are included in Martin(2016) and other resources (a Silicon Valley Data Science blog post[17] by T. Fawcett)): 1) a gradient boosted model with no adjustment for class bias; 2) a gradient boosted model using an 'up' adjustment to the minority class to have the minority (good wine) class on equal count to the majority (poor wine) class; 3) a 'down' adjustment to the majority class to have the majority class count equal to the minority class count ; 4) a gradient boosted model using adjusted weights to impose a heavier penalty when errors are made in the minority class; in the weighted sample, a heavier weight on a class disposes the model to classify an entity to the more heavily weighted class.

The original (no adjustment for class imbalance) gradient boosted model is as follows:

```
set.seed(5627)


ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)



orig_fit <- train(winequal1 ~ .,
                  data = data_train2,
                  method = "gbm",
                  distribution="bernoulli", ##for binomial##
                  verbose = FALSE,
                  metric = "ROC",
                  trControl = ctrl)
```

### Evaluation metrics

As the models in this exercise are classification models, it has been determined after reading quite a few articles that the best evaluation metrics to evaluate the performance of my models are the recall and precision measures. Recall is the number of correct positive predictions made out of all positive predictions that could have been made (all the positive predictions that could have been made is True Positives plus False Negatives). Precision is the number of positive class predictions that actually belong to the positive class

[14] http://uc-r.github.io/gbm_regression
[15] https://dpmartin42.github.io/posts/r/imbalanced-classes-part-1
[16] http://dpmartin42.github.io/posts/r/imbalanced-classes-part-2
[17] https://www.svds.com/learning-imbalanced-classes/

(True Positives/(True Positives plus False Positives)). A false negative is when a success (good wine) is labelled as a failure (poor wine). A false positive is when a failure (poor wine) is labelled as a success (good wine). The F1 score is the harmonic mean of precision and recall; for the F1 score to be high, both precision and recall need to be high.

Martin(2016) argues that many evaluation metrics for classifiers exist, and can generally be divided into two main groups (quoting directly from the paper):

> Threshold-dependent: This includes metrics like accuracy, precision, recall, and F1 score, which all require a confusion matrix to be calculated using a hard cutoff on predicted probabilities. These metrics are typically quite poor in the case of imbalanced classes, as statistical software inappropriately uses a default threshold of 0.50 resulting in the model predicting that all observations belong in the majority class.

> Threshold-invariant: This includes metrics like area under the ROC curve (AUC), which quantifies true positive rate as a function of false positive rate for a variety of classification thresholds. Another way to interpret this metric is the probability that a random positive instance will have a higher estimated probability than a random negative instance.

The measures of recall, precision and the F1 score are easily produced by the R program and are fairly straightforward to understand. A measure of the area under the ROC curve can be produced but must be viewed with other metrics. The Kappa[18] coefficient is sometimes a useful indicator, when viewed along with other indicators. Its value is interpreted from 'rule of thumb', as described in McHugh(2012)[19]. The kappa measure is an indicator of how much the classifications of the model versus the true classification of the wine variable could be attributed to chance. I also viewed the estimates of the contribution of the independent variables to the outcome of the model, looking for some coherence in the choice of independent variables to predict good wine.

Thanks to the programming models in Martin(2016) I was able to create ROC curves (y axis is true positive rate , x axis is false positive rate) which compares the performance of the different classification models on a common grid. The best models for identifying the 'good' wines will be in the regions of the plot which show the greatest area under the curve for the smallest false positive rate value (at the false positive rate between 0% and 25%). That is, per Martin(2016), the most successful algorithm better identifies the true positives as a function of false positives for instances that are predicted as having a high probability of being in the minority class. In my analyses, I found that examining the ROC curve in this manner corroborated somewhat with the results of the model as represented in the output confusion matrices and the estimates of recall and precision or other performance measures.

## Results

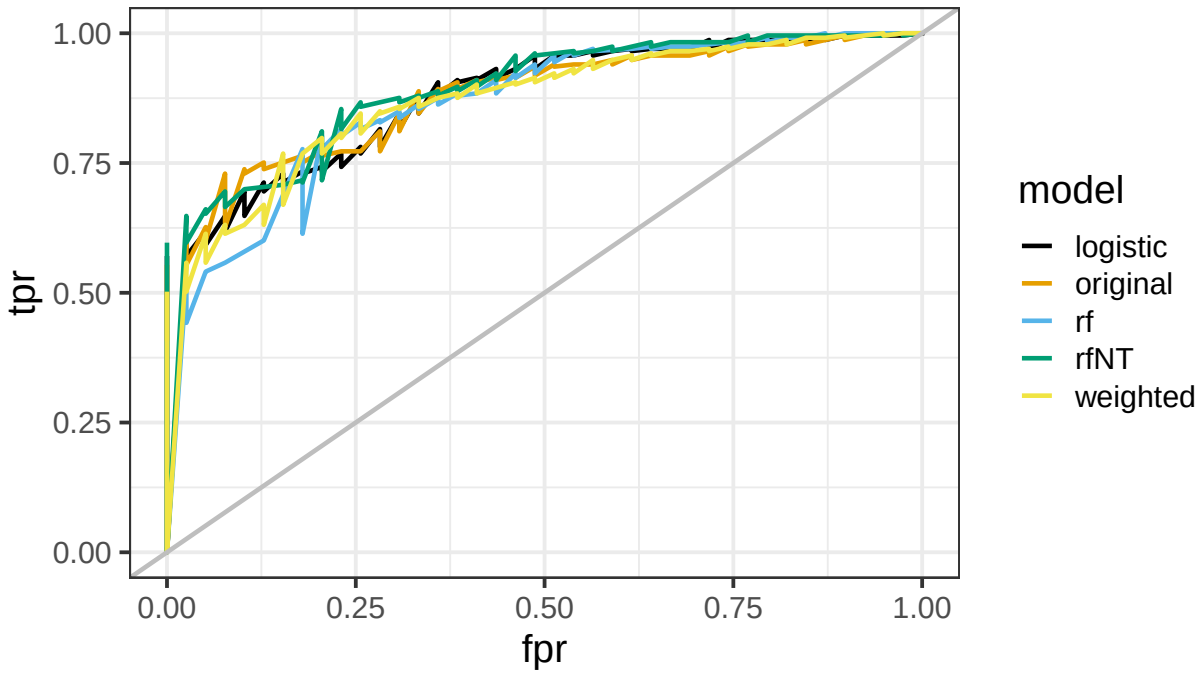
### Descriptions of Model Results

The results of the models are presented in Table 5, below (for reference, in the test sample, the total count of good wine is 39; the total count of poor wine is 233). The ROC curves, which I created to rate a selection of the models based upon the area of the curve between 0 – 25% on the false positive rate axis is presented immediately after. I will refer to these when discussing the results from all models.

[18] https://en.wikipedia.org/wiki/Cohen%27s_kappa
[19] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/

Table 5: Evaluation metrics of all models

| method | Kappa | Recall | Precision | F1 | misclassification | true_good | false_good | false_poor | true_poor |
|---|---|---|---|---|---|---|---|---|---|
| logistic | 0.47 | 0.44 | 0.68 | 0.53 | 0.11 | 17 | 8 | 22 | 225 |
| logistic-down sample | 0.32 | 0.77 | 0.33 | 0.46 | 0.26 | 30 | 62 | 9 | 171 |
| random forest | 0.49 | 0.46 | 0.67 | 0.55 | 0.11 | 18 | 9 | 21 | 224 |
| random forest-not tuned | 0.49 | 0.21 | 0.89 | 0.33 | 0.12 | 8 | 1 | 31 | 232 |
| gbm_original | 0.40 | 0.41 | 0.55 | 0.47 | 0.13 | 16 | 13 | 23 | 220 |
| gbm_weighted | 0.39 | 0.85 | 0.37 | 0.51 | 0.23 | 33 | 57 | 6 | 176 |
| gbm_up | 0.36 | 0.82 | 0.35 | 0.49 | 0.25 | 32 | 60 | 7 | 173 |
| gbm_down | 0.32 | 0.82 | 0.32 | 0.46 | 0.27 | 32 | 67 | 7 | 166 |

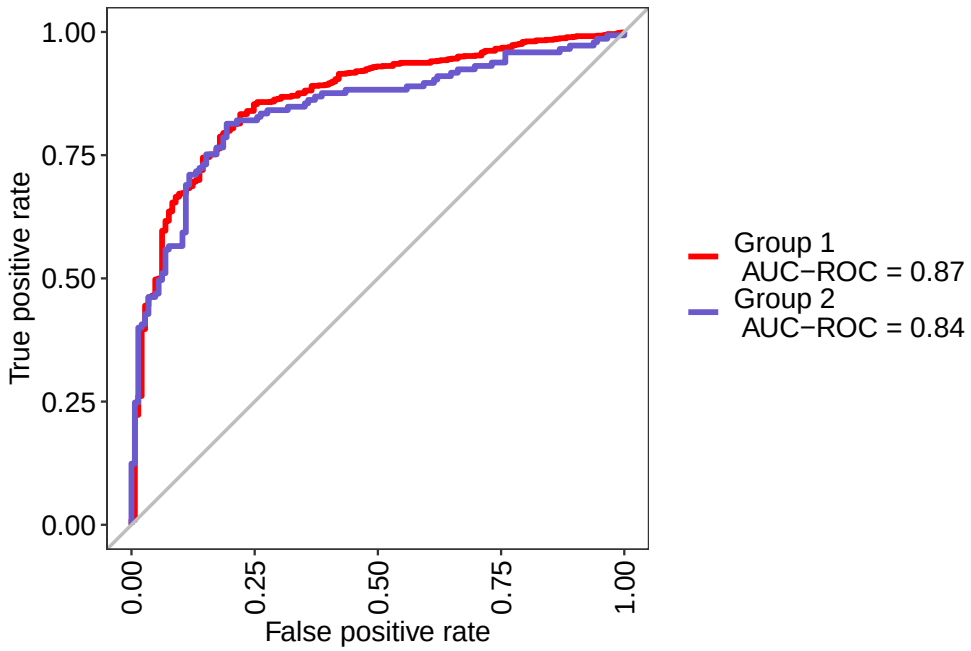



It is difficult to identify which model performed 'best' when looking at the table. I dive into the discussion with a comparison between the two logistic models.

The logistic model performed decently. Attempting to adjust the train set to take into account class imbalance did improve the performance of the logistic model as the AIC dropped from 607.11 in the unadjusted train set model to an AIC of 275.07 in the 'down' sampled train set model. However, taking only the AIC as an indicator of the ability of the down sampled model to predict good wine may be in error. At this point, tests of multicollinearity have indicated that multicollinearity is not a problem in the dataset. Looking at a comparison of the two logistic curves, the AUC of the ROC curve below reveals a) the unadjusted train set model (Group 1 in the diagram) has a larger AUC at 0.87 versus the AUC of the down sampled model at 0.84 (Group 2 in the diagram). The down sampled model also performed more poorly (less AUC) in the 0 to 25% region of the curve (possibly indicating a lesser ability to correctly identify the minority class/good wine – although the down sampled logistic model identified 30 of 39 good wines, it also incorrectly identified 62 poor wines as 'good'). In the comparison between the two logistic models, the indicators for Recall and Precision are supported by the visual check on the area under the ROC in the area between 0 and 25% of fpr; larger AUC in this region supports the notion that model is more likely to find 'good' wines and fewer false 'good' wines than the curve with less AUC in this region; larger AUC also implies a better true positive rate (better odds of correctly identifying a good wine).

What I find interesting about the ROC curves for the two logistic curve was how easy it was to produce the diagram using the MLeval package. But if I compare the output of the MLeval package with what was

produced using ggplot and the programming to create the curves (as described in Martin(2016)), I think it is evident that the MLeval package is 'smoothing' the curves a bit. Which is ok for a quick and dirty visual. But, if i use these curves to identify a better performing model (looking at the bottom percent of the fpr), I should bear in mind that smoothing the rough places might have me mislead about the performance of my models and the data set underlying the curve.




Moving on to look at the other models and the other performance indicators: If we look at the Kappa coefficient, we say that the cross validated logistic model, the two random forest models have weak performance, as the values of the Kappa are 0.40 and 0.59 indicating that 15 – 30% of the data are reliably classified. For the gradient boosted models, the Kappa values are around 0.39 or under (but over 0.2), indicating minimal performance with 4 -15% of the data reliably classified. If we look only at the Kappa statistic, we do not see that any of the models have performed well at identifying good wine.

Next we have measures of Recall and Precision and the F1 measure. Precision answers the question: What proportion of positive identifications was actually correct? How many of the wines identified as 'good' are truly 'good'. Recall answers the question: What proportion of actual positives was identified correctly? How many of the total count of good wines were identified as good. A good model will have both recall and precision values as high as possible towards 1, to obtain an F1 measure as high as possible towards 1. If we look only at the F1 measure, we discover, to a bit of surprise, that the cross validated logistic model (with no adjustment for class imbalance in the train set), performed second best and the 'tuned' random forest model performed the best of all models! (Imagine my surprise upon discovering that I am so very good at tuning a random forest model).

If wine sellers want to be most sure, based upon a chemical analysis of wine (using the criteria described in this report), the random forest model not tuned seems the best for identifying 'good' wines, with only one poor wine misclassified as good (the trade off is that only eight wines are correctly classified as good, with the remainder (31) classified to poor). And the random forest model not tuned is the best at identifying the poor wines correctly, with 232/233 correctly identified. Precision for this model is very high at 0.888. However, recall is very low at 0.205; a very low portion of total good wines were identified as good wine (a larger portion of total good wines were classified to poor in this model).

Opposite to the performance of the random forest not tuned model is gbm_down (a gradient boosted model with an adjustment for class imbalance). The recall is very powerful but the precision is very low for this

Table 6: Variables of importance (roughly defined), all models

| method | sulphates | alcohol | volatile.acidity | fixed.acidity | density | total.sulfur.dioxide | residual.sugar | chlorides | pH | citric.acid | free.sulfur.dioxide |
|---|---|---|---|---|---|---|---|---|---|---|---|
| logistic - down sample | 100 | 69 | 52 | 44 | 36 | 36 | 34 | 28 | 10 | 3 | 0 |
| random forest | 75 | 100 | 43 | 28 | 31 | 25 | 0 | 6 | 10 | 13 | 11 |
| random forest Not Tuned | 85 | 100 | 58 | 32 | 46 | 47 | 0 | 34 | 6 | 26 | 2 |
| gbm-original: relative influence | 18 | 29 | 13 | 6 | 5 | 9 | 3 | 4 | 3 | 6 | 3 |
| gbm-weighted: relative influence | 26 | 48 | 16 | 3 | 1 | 4 | 0 | 1 | 0 | 2 | 0 |
| gbm-down: relative influence | 13 | 43 | 11 | 3 | 1 | 6 | 1 | 9 | 2 | 7 | 3 |
| gbm-up: relative influence | 24 | 48 | 13 | 2 | 1 | 4 | 1 | 2 | 1 | 4 | 1 |

model. The gbm_down model correctly identifies 32 of 39 good wines. Unfortunately, it also misclassifies 67 of the poor wines as 'good', so that less than half of wines identified as good are truly good.

The gradient boosted models were set up with very little tuning, as I have limited experience or understanding of these models as well. I had expected them to perform better than the logistic model and the random forest model. The gradient boosted models were very responsive to the efforts to correct for class imbalance. More expert tuning of these models might yield better results for classifying the wines in this database.

The ROC curves for models in the diagram are all showing a bit of discontinuity in the lower fpr values, which could mean that the train set and test set sizes are too small to create a smooth function. In the diagram, the orange curve describes the gbm-original (no adjustment for class imbalance). Looking at the area under the curve for false positive rate less than 0.25, the gbm_origin model shows up as the best for identifying good wines. The tuned random forest model shows the greater AUC at the 0-5% place on the x-axis. But these observations are taken not all that seriously here: the table above, has already indicated that none of the models in this project perform well (or, if I were to put it more positively: each of the models has its strengths…it would be up to the user to place value on the various outputs, depending on preferences to identifying a good wine with certainty versus identifying a poor wine with certainty).

**Variables of importance**

We can look at the Variables of Importance/relative influence of each variable in the models (Table 6), acknowledging that such a comparison between models is a bit sketchy. Does each model use the same criteria for ranking importance? No. The logistic model is best analyzed for importance of contribution using the dominanceanalysis package[20] - the ranking I have listed is pretty rough. For random forest models, the variable of importance is based upon the Accuracy measure[21]. For the gbm, the output includes the relative influence of each variable in training the model[22]. Nevertheless, I have gone ahead and put all the measures of 'importance' into one table. What is discovered here is that there is no clearcut identification of 'redundancy' when comparing such a list across all model forms.

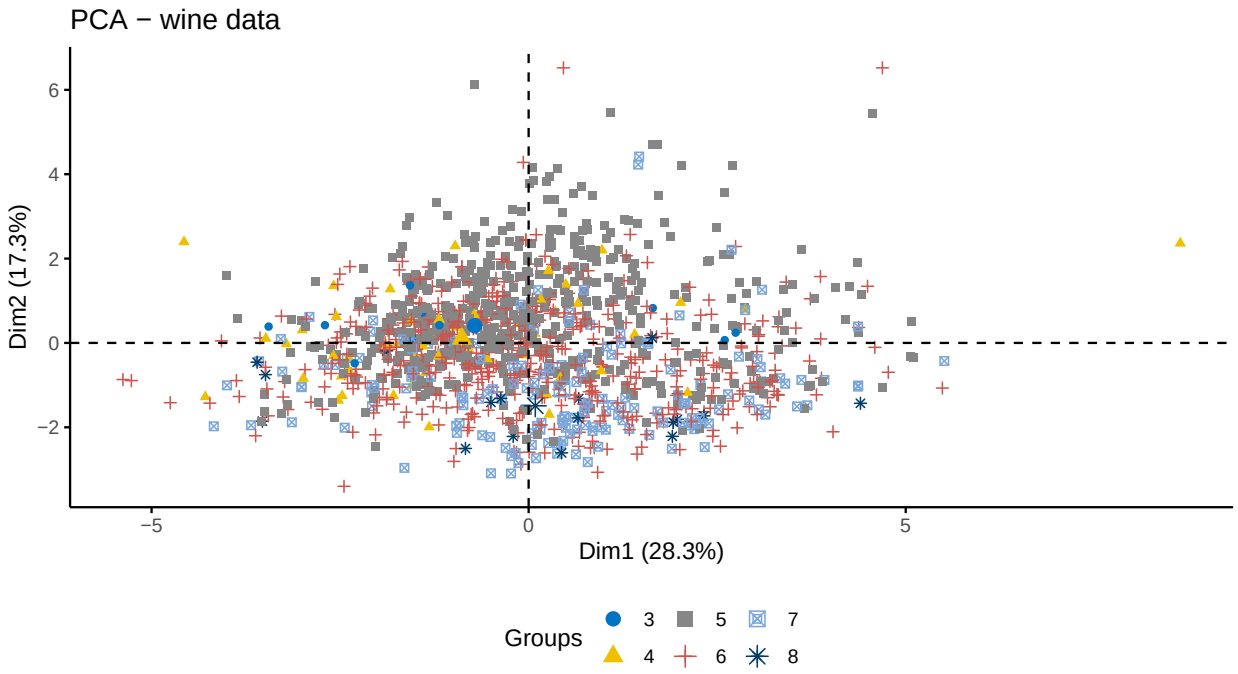
We can see that in all model forms, sulphates, alcohol and volatile.acidity are the top three in 'importance' to the modelling processes. But out of 11 explanatory variables, eight are inconsistent in importance, depending on the form of the model. This inconsistency is important: I cannot clearly identify which of the eight lesser variables could be dropped to reduce redundancy in the model. All variables contribute in some models. But, for example, in three model forms, residual.sugar is not a contributor, with zero importance. But in the cross validated model, residual.sugar ranks 6th in importance. And the table above is in conflict with the principal component analysis which identified fixed.acidity, citric.acid, pH and density as the largest contributors of the first dimension explaining the variance in the quality variable. The inability to clearly identify rank or importance of explanatory variables in the models is a problem to be solved by the chemist who creates the database, not the data scientist modelling data on mathematical principals. A data scientist cannot decide why a variable should be critical to the quality of a wine. If an important explanatory chemical variable does not show up as important in the model, then the chemist has made an error and therefore the predictive model is in error. Without consultation with the chemist, the data scientist cannot evaluate the model for accuracy in predicting a good wine.

[20] https://cran.r-project.org/web/packages/dominanceanalysis/vignettes/da-logistic-regression.html
[21] https://www.displayr.com/how-is-variable-importance-calculated-for-a-random-forest/
[22] https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

**Adjusting the threshold for predicting a good wine**

Reviewing ways to improve classification models, I learned that moving the threshold for identifying a good wine might be the best way to improve the performance of the model[23] [24].

But for the wine database, another look at the distribution of wine quality across principal component dimension 1 and dimension 2 shows no indication that including the quality = 6 group in the 'good' wines would improve the ability of the existing models to identify good or poor wines. The quality = 6 group is spread far and wide across the principal component dimesion grid: the inability to find a defined pattern in the quality of the wine might be due to the database itself. The quality variable could be poorly defined. Per the information I have, the quality was assigned by taste test. Taste tests could be faulty and inconsistent with chemical composition of the wine. And then, as discussed above, the chemical constituents of a good wine must be clearly defined and identified otherwise the database is incoherent and modelling chemical composition using mathematical instruments will not ever identify good wine.




# Conclusions

The models tested the ability of several classification tools (logistic regression, random forest, gradient boosted algorithm) to correctly identify and classify good and poor wines. The database for wine classification was down loaded from the Kaggle site. The results of modelling the quality of wine using the chemical components of the wine was a very instructive exercise, even as it was discovered that the models finally produced and running would not identify good wines in any outstanding manner.

The random forest models in this project were confounded by tuning parameters. The two random forest models came up with different results, based only upon decisions for nodes, maxnodes or number of trees selected for the model. Mtry was offered a grid to select for the best model in each case. The random forest model seemed like a nice option able to handle a difficult data set, but as it turned out, there are many aspects to modelling with random forest that need to be understood if the model is to have reliable

[23] https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification
[24] https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

interpretation. If I understood better how to set up a random forest model, I would have better faith in the results. I would prefer to see the outcome of the model defined by the dataset itself, rather than by how much guessing I could do to the depth of trees or the number of trees.

As I have not any experience coding classification models I had quite a bit to learn. Correlation analysis and principal component analysis produced interesting interpretations of the structures in the dataset. But moving forward to modelling, I discovered that I could not find enough coherence between how a model behaved and what I thought I understood (from correlation analysis and principal component diagrams) about the importance of each chemical component and how these components relate to wine quality. Without an understanding of the chemical components in the database, I cannot possibly decide which should be eliminated or more heavily weighted so that quality wine identification runs more smoothly. Alcohol, sulphates and fixed.acidity were important leading contributors to every model but were not identified as major contributors in principal component analysis; other components in the database had varying importance, depending upon the mathematical structure of the model.

What I also learned from this project: the effects of a change in the "seed" on the output of the models. I meticulously programmed to avoid a change in the "seed", particularly in the cross validated models. A change in the "seed" can, in some cases, move the performance of the model enough to conclude that a model is not all that reliable. The performance measures were roughly the same, but, in the case of the cross validated logistic curves, the change was enough to reverse the AUC measures. The results of that 'mistake' were not documented as time no longer permits more rabbit holes of investigation.

Martin(2016)[25] has suggested that rather than using the area under the ROC curve to identify a good model, that the area under the precision recall curve should be used. Future projects should consider the construction of the precision recall curve.

I did not use a 'validation' dataset in this project. It did not seem to be necessary. And, I was concerned that if I could not produce a good classification model after splitting between test and train sets, further reducing the size of the train and test set to create a validation set did not seem to be all that worthy. I am hoping I am not heavily penalized for having no validation set.

[25] http://dpmartin42.github.io/posts/r/imbalanced-classes-part-2