Red wine quality prediction through machine learning

HarvardX PH125.9x Data Science: Capstone - Choose your own project

Priscila Treviño Aguilar

December, 2020

Acknowledgement

I would like to thank professor Irrizary and the whole HarvardX staff for their superb work on this course, I was very pleased with the content and assignments and I consider that the capstone project did an excellent job of summarizing the skills learned and encouraging practice of the most relevant aspects of the course. As a food scientist and an enthusiast of biological and chemical sciences in general, I'm glad that I was able to incorporate an issue of my choice and that is relevant in the still emerging relation between data science and the food and beverage industry. Completing this capstone project was an exciting and challenging experience in my path to becoming a data scientist.

1. Introduction

This capstone project consists in employing the R-programming and statistics skills obtained throughout the course as well as applying the learned machine learning techniques to predict red wine quality. A few machine learning algorithms were fit to the data to predict red wine sensory quality (a score between 1 to 10) in a test set using the inputs of a provided train set.

Nowadays data science and analytics are widely used to support and optimize many different disciplines, food science is not the exception, although the relation with the food and beverage industry is still nascent there are already several ways data science is being applied to improve food and beverage products' development and manufacturing. One such way is the application of data analysis to evaluate and predict the sensory quality of food and beverage products. Sensory analysis is an important food science sub discipline that applies principles of experimental design and statistical analysis to the use of human senses to evaluate the sensory properties of consumer products (Jantis, 1992). This discipline relies on both physicochemical parameters and human expert evaluation to perform its analysis (trained sensory judges and biostatisticians), although those aspects are not completely replaceable, data analysis can aid in better understanding the complex relationship between food physicochemical attributes and sensory quality, providing relatively accurate predictions while also minimizing labor and costs (Cortez et al., 2009). Thus, this application presents an advantage of scientific and economic relevance.

The dataset used in this project is related to a red variant of the Portuguese "Vinho Verde" wine. It consists of 11 input variables (wine physicochemical parameters) and one output variable (wine quality based on sensory analysis data), it's publicly available at UCI machine learning repository and Kaggle and it was obtained from a research paper by Cortez et al., (2009) in which data mining was used to perform wine quality assessment.

The analysis approach in this project was more simple. Exploratory data analysis was performed and, although the data set was already clean for its analysis, highly correlated variables were eliminated to promote better model fitting. Since this data set can be seen as a regression or a classification task, both approaches were taken in this project. Two regression models were fit to the data (Regression tree and Random Rorest regression), as well as two classification models (Naive Bayes and Random Forest classification). The regression models were evaluated by computing their root mean squared error (RMSE). For the classification models, the integer output variable was converted into a dichotomous categorical variable by setting a cutoff to determine if a wine was considered moderate or high quality based on hedonic scales commonly used in sensory analysis tests and they were evaluated through their accuracy and as well as their ROC curve and AUC calculation.

2. Methods

Exploratory data analysis

The structure of the data set showed that it contained 1599 distinct observations, 11 numeric input variables and one integer output variable.

```
str(wine, width=80, strict.width="cut")
```

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 ..
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ..
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8..
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

A look at summary and class of the variables gave a more clear idea of their properties and values.

```
summary(wine)
```

```
##  fixed.acidity   volatile.acidity  citric.acid    residual.sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
##  Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
##        pH          sulphates         alcohol         quality
##  Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.310   Median :0.6200   Median :10.20   Median :6.000
##  Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636
##  3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000
```

```
sapply(wine, class)
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##            "numeric"            "numeric"            "numeric"
##       residual.sugar            chlorides  free.sulfur.dioxide
##            "numeric"            "numeric"            "numeric"
## total.sulfur.dioxide              density                   pH
##            "numeric"            "numeric"            "numeric"
##            sulphates              alcohol              quality
##            "numeric"            "numeric"            "integer"
```

The data set did not contain any NA values.

```
any(is.na(wine))
```

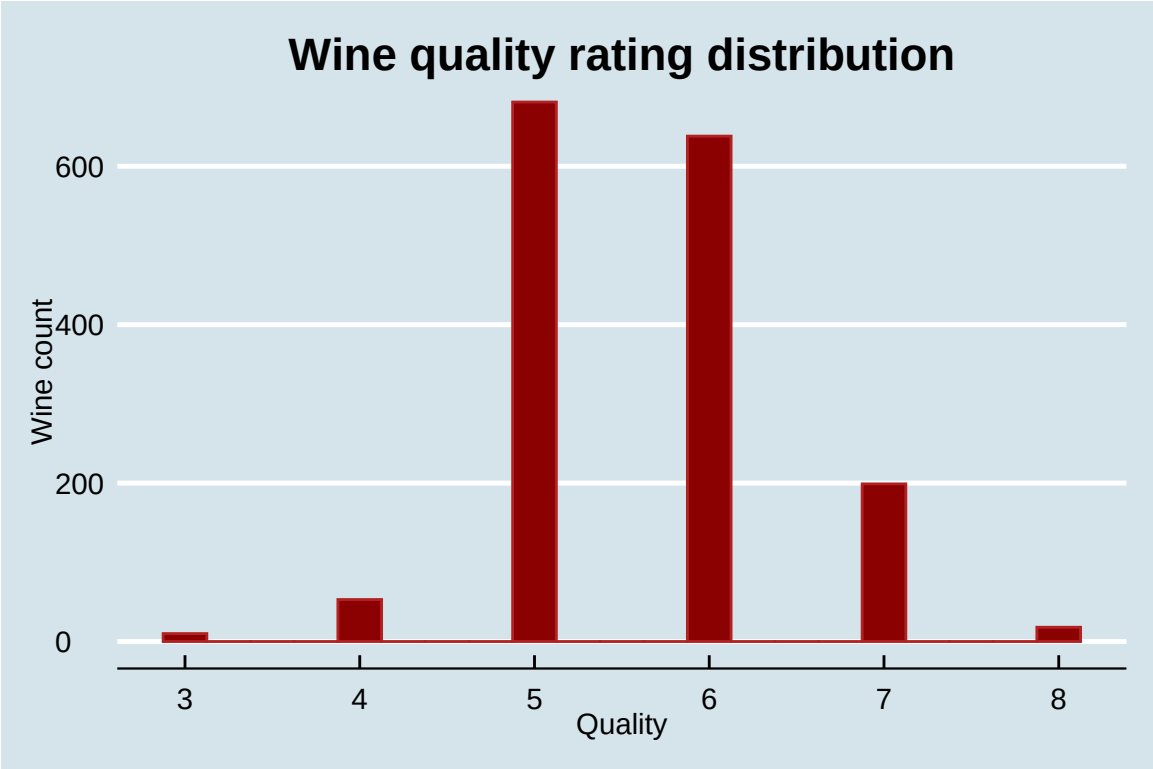
```
## [1] FALSE
```

Technical particularities of the data set's variables were studied further to have a better understanding of their possible influence on the final product.

Table 1: Variable description

Column/Variable	Description
fixed.acidity	Nonvolatile acids (do not evaporate readily)
volatile.acidity	Acetic acid in wine (high levels cause unpleasant flavor)
citric.acid	Adds 'freshness' and flavor to wines, present in small quantities
residual.sugar	Remaining sugar after fermentation stops
chlorides	The amount of salt in the wine
free.sulfur.dioxide	Prevents microbial growth and the oxidation of wine
total.sulfur.dioxide	Free and bound forms of sulfur dioxide, usually present in low concentrations
density	Dependent on the alcohol percentage and sugar content
pH	Describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic)
sulphates	A wine additive which acts as an antimicrobial and antioxidant
alcohol	Alcohol percentage
quality	Output variable (based on sensory analysis data, score between 0 and 10)

By visualizing the distribution of the quality data below it was observed that the majority of the wines in the data set are rated between 5 to 6 and there are few wines with very low or very high rating. The distribution is slightly left skewed and the tall peak indicates that the data is close to the mean (low standard deviation).



This was confirmed by computing the quality mean, median and standard deviation.

```
mean(wine$quality)
```

```
## [1] 5.636023
```

```
median(wine$quality)
```

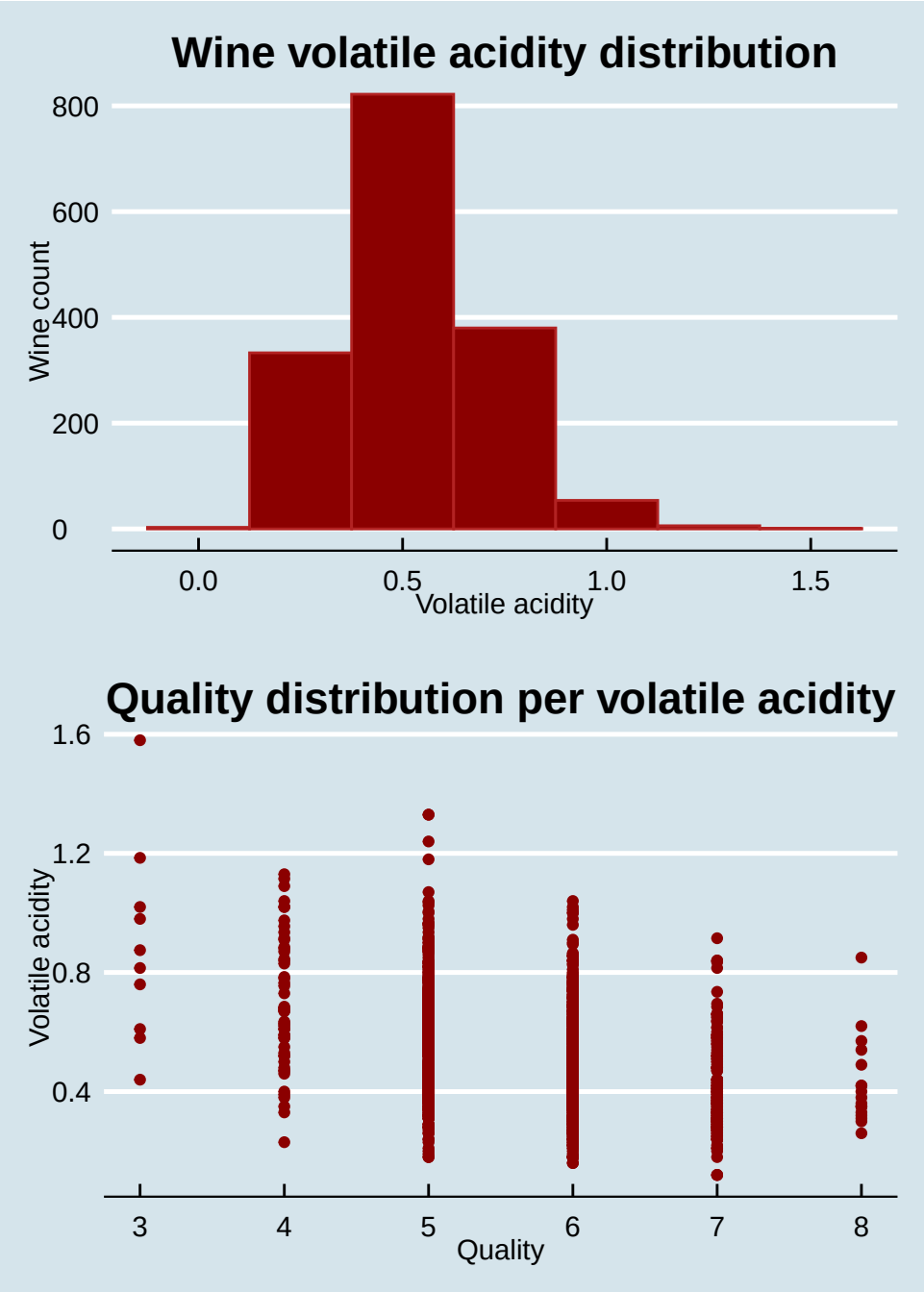
```
## [1] 6
```

```
sd(wine$quality)
```

```
## [1] 0.8075694
```

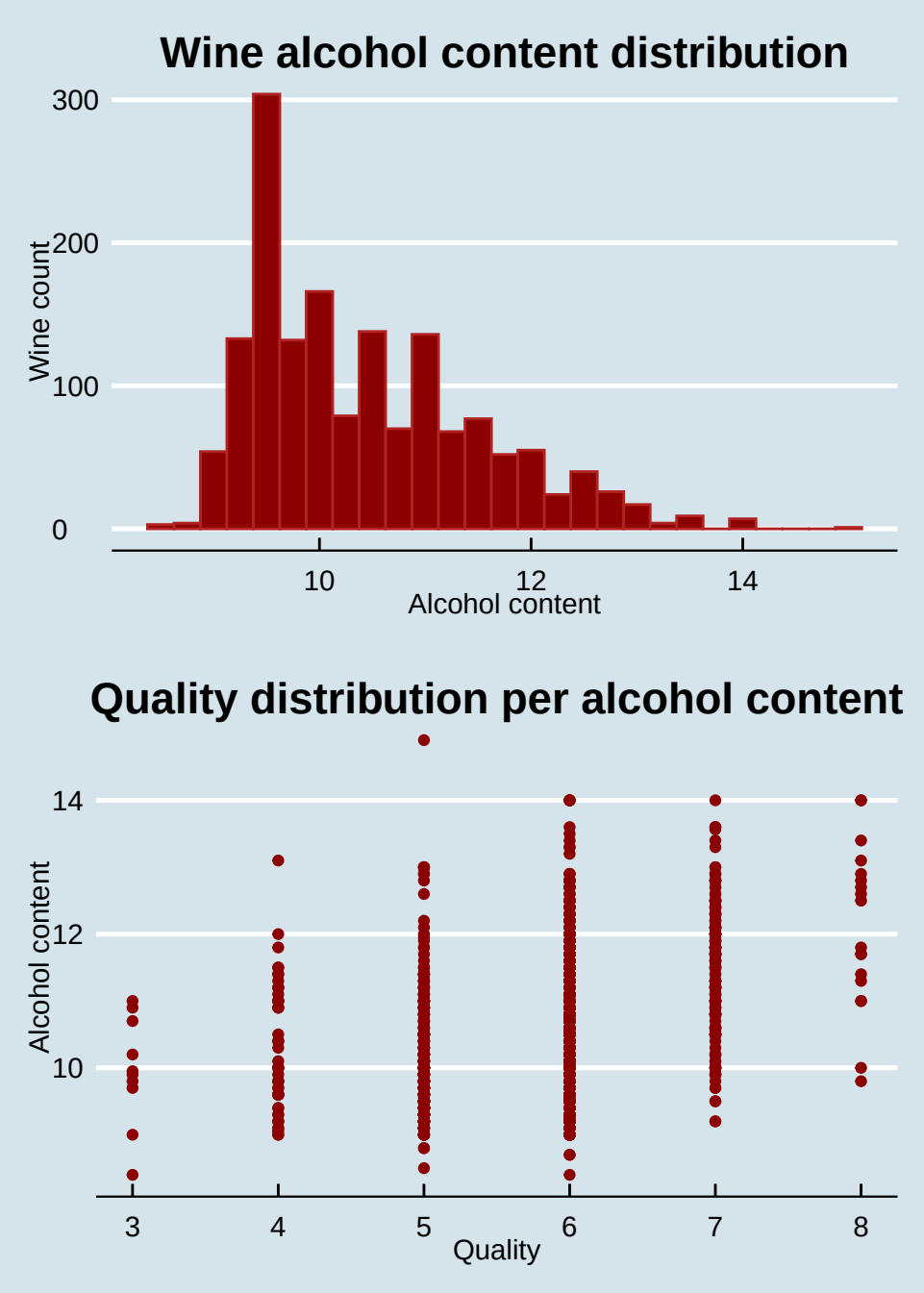
One of the most important attributes that influence the quality of wines is the aroma. Wine aroma is extremely complex, determined by volatile compounds present in very low concentrations that have a synergic effect on the final quality. These volatile compounds generally include alcohols, esters, aldehydes, acids, phenols and sulfur compounds (Sanchez-Palomo et al., 2018). Upon consulting scientific literature on wine sensory properties and its general flavor profile, it was found that among the physicochemical parameters available in this project's data set, volatile acidity and alcohol content might be the ones with the most substantial effect on wine sensory quality. To attempt to better understand what makes a high quality wine, these variables were studied further through literary research and visualization.

The absolute amount of acids in wine may affect other parameters, such as pH, sugar and ethanol concentrations. Maintaining a low pH is also crucial to the color stability of red wines (which is also a relevant aspect of the sensory experience). Volatile acidity itself refers to acetic acid which is essential for conveying a vinegary and pungent aroma (sourness) (Villamor, 2012) (Sanchez-Palomo et al., 2018).



The plots above showed the distribution of volatile acidity among wines and seemed to indicate that higher quality wines tend to contain smaller values of volatile acidity.

Alcohol content can directly affect the aroma perception in wine, for example, at low ethanol concentration (10.0 to 12.0%) it has been described as fruity while at higher ethanol concentrations it has been defined as herbaceous. Alcohol content has also been shown to increase the perceived viscosity in wine, which in turn has an effect on the intensity of how volatile compounds in wine are perceived (Villamor, 2012) (Yildirim et al., 2005).



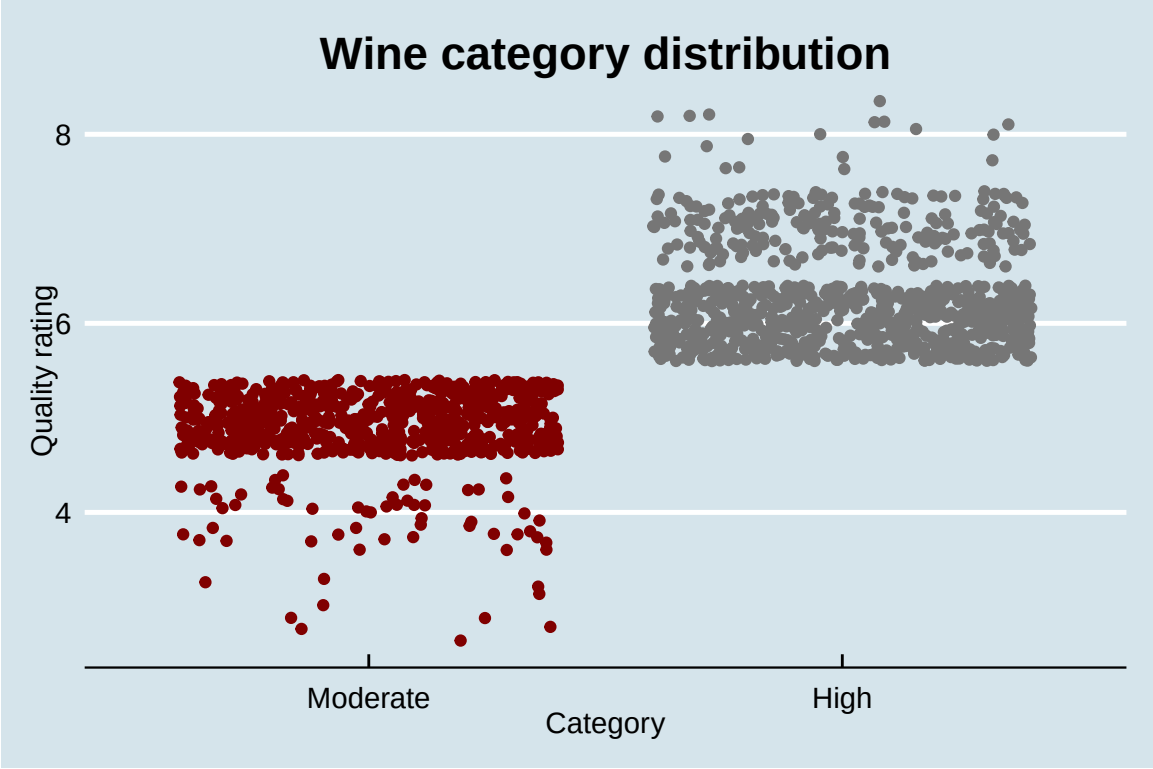
In the plots above it was observed that most wines belong to the range of low ethanol concentration. The quality distribution indicates that high quality wines with ratings between 7 to 8 tend to have higher alcohol content, however the relationship between alcohol content and quality was not entirely clear.

The integer output variable, quality, was transformed into a dichotomous categorical variable to make a distribution plot of wines considered to be of moderate or high quality (as well as for subsequent model fitting). The cutoff value decision was taken based on hedonic scales commonly used in sensory analysis. The wine rating sentiment and description were extrapolated to this project's data set quality scale, so it was considered that a rating of 5 or lower represented that the trained sensory judge neither liked nor disliked the wine and thus the wine's quality was below average to average (moderate quality). Whereas a rating of 6 or higher meant that the wine's quality was good to extraordinary since it represented that the judge liked it (high quality) (Marks, 2019).

Table 2: Comparison of the hedonic scale and the trained sensory judge sentiment description table for wine quality

Rating	9-point hedonic scale	Rating	Sentiment
9	Like extremely	96-100	Extraordinary
8	Like very much	90-95	Outstanding
7	Like moderately	80-89	Very good to barely above average
6	Like slightly		
5	Neither like nor dislike	70-79	Average, soundly made, little distinction
4	Dislike slightly		
3	Dislike moderately	60-69	Below average, notable deficiencies
2	Dislike very much		
1	Dislike extremely	50-59	Unacceptable

The strip plot below shows the distribution of moderate and high quality wines.



The amount of wines in each category was computed below.

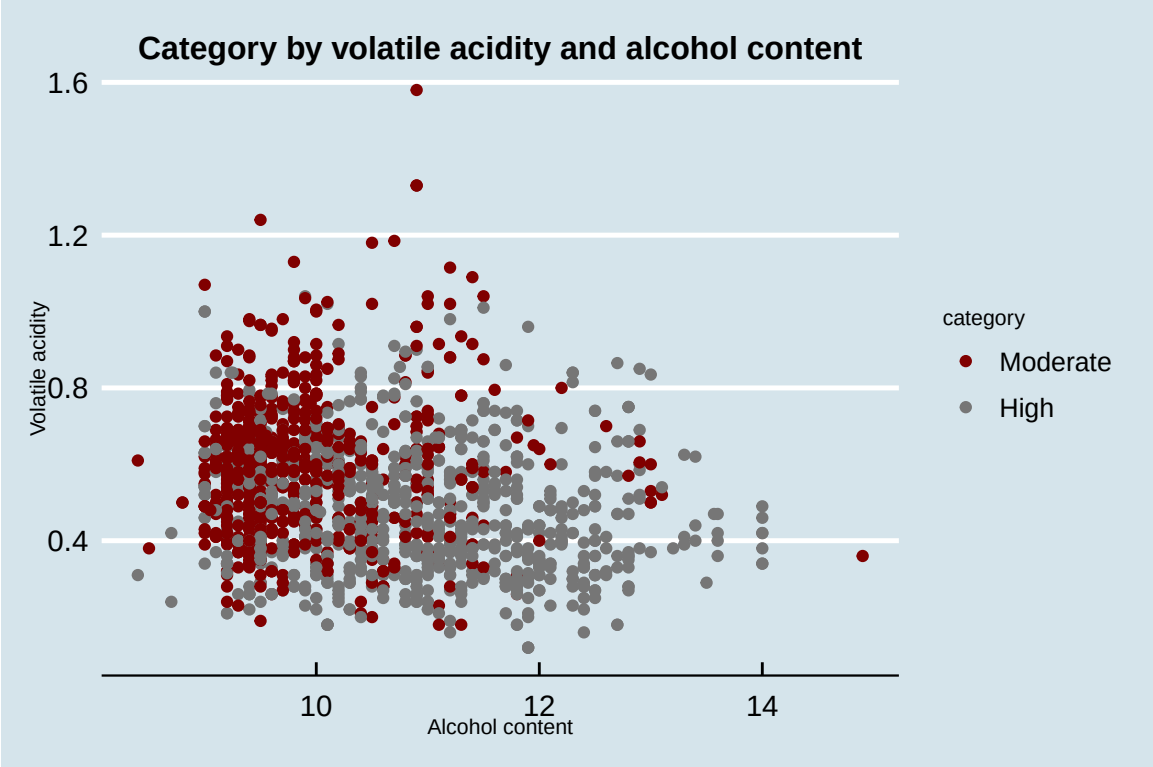
```
sum(category_wineplot$category == "Moderate")
```

```
## [1] 744
```

```
sum(category_wineplot$category == "High")
```

```
## [1] 855
```

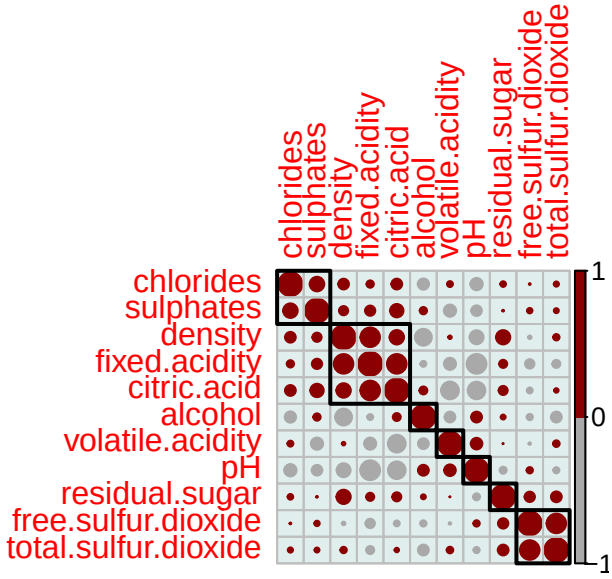
Finally, the multivariate plot below seemed to confirm that, roughly, high quality wines tended to have low concentrations of volatile acidity and higher alcohol content.



However, the models employed afterwards on this project helped elucidate whether these variables played a particularly important role in wine quality prediction and also whether other variables were significant.

Data cleaning

To promote better model fitting, variables with a moderate to high correlation were eliminated. This is due to the fact that highly correlated variables may impart the same information to the model, adding noise instead of incremental information. First, a correlation matrix was created (correlation plot below), then a cutoff was set to find variables that were relatively highly correlated. Variable correlation was observed in the plot below.



The variables eliminated were fixed acidity, citric acid and total sulfur dioxide (shown below).

```
correlated <- findCorrelation(correlationMatrix, cutoff = 0.5)  # the cutoff was set
print(correlated)
```

```
## [1] 1 3 7
```

```
correlationMatrix[,c(1,3,7)]  # identified highly correlated attributes by their index
```

```
##                       fixed.acidity citric.acid total.sulfur.dioxide
## fixed.acidity            1.00000000  0.67170343          -0.11318144
## volatile.acidity        -0.25613089 -0.55249568           0.07647000
## citric.acid              0.67170343  1.00000000           0.03553302
## residual.sugar           0.11477672  0.14357716           0.20302788
## chlorides                0.09370519  0.20382291           0.04740047
## free.sulfur.dioxide     -0.15379419 -0.06097813           0.66766645
## total.sulfur.dioxide    -0.11318144  0.03553302           1.00000000
## density                  0.66804729  0.36494718           0.07126948
## pH                      -0.68297819 -0.54190414          -0.06649456
## sulphates                0.18300566  0.31277004           0.04294684
## alcohol                 -0.06166827  0.10990325          -0.20565394
```

```
wine_clean <- select(wine, -c(1,3,7))  # correlated variables removal
```

Regression models fitting

The evaluation metric chosen was the root mean squared error (RMSE). A commonly used metric (loss function) for regression models that indicates the absolute fit of the model to the data, that is how close the observed data points are to the model's predicted values. RMSE is a negatively-oriented score, which means that lower values are better, and it expresses average model prediction error in units of the variable of interest. The following code chunk shows a function to compute the RMSE of the models.

```
RMSE <- function(actual, predicted){
  sqrt(mean((actual - predicted)^2))
}
```

Data partition to obtain train and test sets for the regression models was performed.

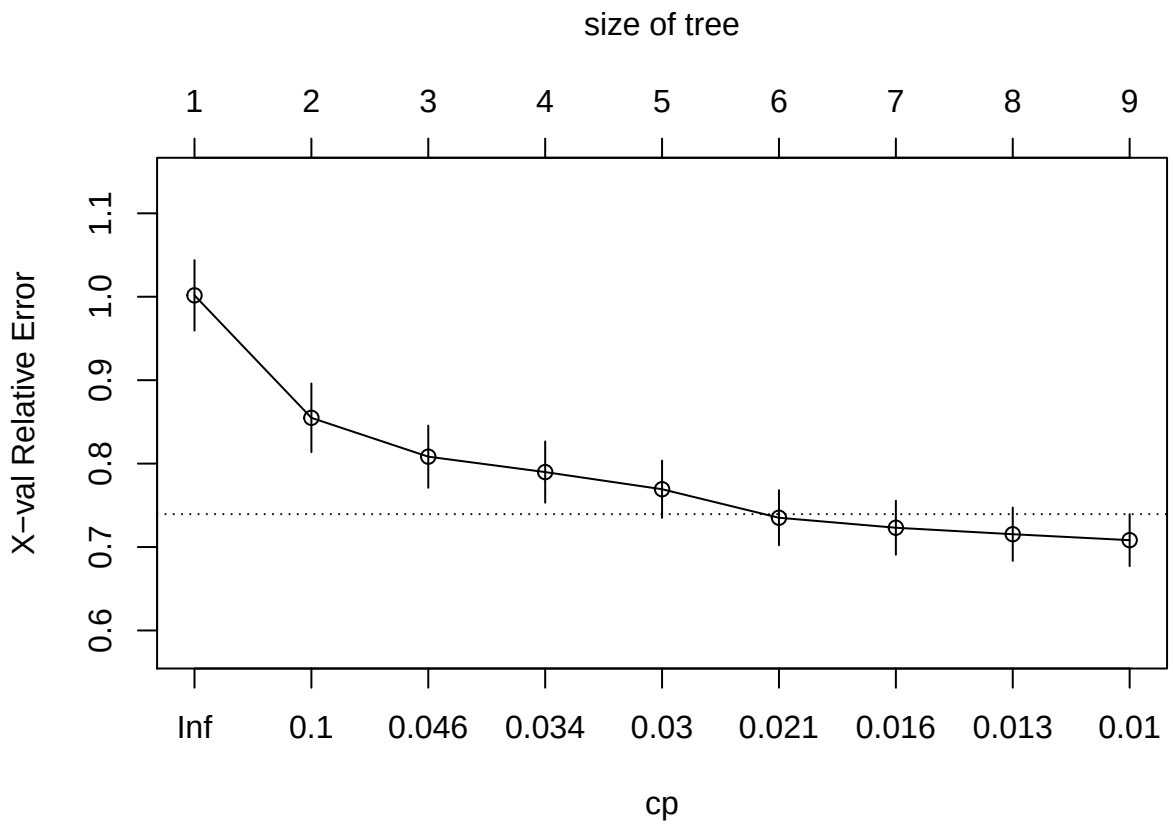
```
set.seed(6, sample.kind="Rounding")
test_index <- createDataPartition(y = wine_clean$quality, times = 1, p = 0.2,
                                  list = FALSE)  # data partition

train_reg <- wine_clean[-test_index,] # train and test sets
test_reg <- wine_clean[test_index,]
```

Regression tree

Regression/classification tree is a supervised learning algorithm built through a process known as binary recursive partitioning, which is an iterative process that splits the data into partitions or branches. The algorithm allocates the data into two first partitions and then selects the splits that minimize the sum of the squared deviations from the mean in the two separate partitions. This splitting rule is then applied to each of the new branches until each node reaches a specified minimum node size and becomes a terminal node. It can be used for both classification and regression problems. In this case it was implemented with the *rpart* function.

```
reg_tree <- rpart(quality ~ .,       # regression tree fit
                  data = train_reg,
                  method = "anova")

reg_tree$cptable  # results
```

```
##           CP nsplit rel error    xerror       xstd
## 1 0.16878600      0 1.0000000 1.0016569 0.04224172
## 2 0.05978193      1 0.8312140 0.8548689 0.04118198
## 3 0.03579611      2 0.7714321 0.8083280 0.03727314
## 4 0.03308295      3 0.7356360 0.7898516 0.03671808
## 5 0.02712585      4 0.7025530 0.7693030 0.03433291
## 6 0.01623942      5 0.6754272 0.7351118 0.03301872
## 7 0.01505158      6 0.6591877 0.7231328 0.03239876
## 8 0.01096141      7 0.6441362 0.7153794 0.03195004
## 9 0.01000000      8 0.6331747 0.7082491 0.03117807
```



The plot above shows that the model automatically applied a range of cost complexity and performed cross validation. However, more specific cross validation was needed to ensure the use of optimal parameters (the cross validation process for this model was purposely made extensive to demonstrate the skills obtained in the course, the following techniques applied to the rest of the models were more concise), so further model improvement was achieved with hyper parameter tuning through a grid search (shown in the code chunk below).

```
tuning_grid <- expand.grid(minsplit = seq(3, 20, 1), maxdepth = seq(5, 15, 1))
nrow(tuning_grid)  # number of combinations in tuning grid
```

```
## [1] 198
```

```
model_comb <- list()
for (i in 1:nrow(tuning_grid)) {  # for loop to automate model training combinations
  minsplit <- tuning_grid$minsplit[i]
  maxdepth <- tuning_grid$maxdepth[i]

  model_comb[[i]] <- rpart(quality ~ .,
    train_reg,
    method  = "anova",
    control = list(minsplit = minsplit, maxdepth = maxdepth)
  )
}
```

```
opt_cp <- function(x) {            # function to obtain the optimal cp
  min    <- which.min(x$cptable[, "xerror"])
  cp <- x$cptable[min, "CP"]
}


min_error <- function(x) {         # function to obtain the minimum error
  min    <- which.min(x$cptable[, "xerror"])
  xerror <- x$cptable[min, "xerror"]
}



tuning <- tuning_grid %>%   # grid search results
       mutate(cp = purrr::map_dbl(model_comb, opt_cp),
       error = purrr::map_dbl(model_comb, min_error)) %>%
       arrange(error) %>%
       top_n(-5, wt = error)
```

Optimal parameters were computed.

```
tuning[1,] # optimal parameters
```

```
##   minsplit maxdepth   cp    error
## 1       11       13 0.01 0.673917
```

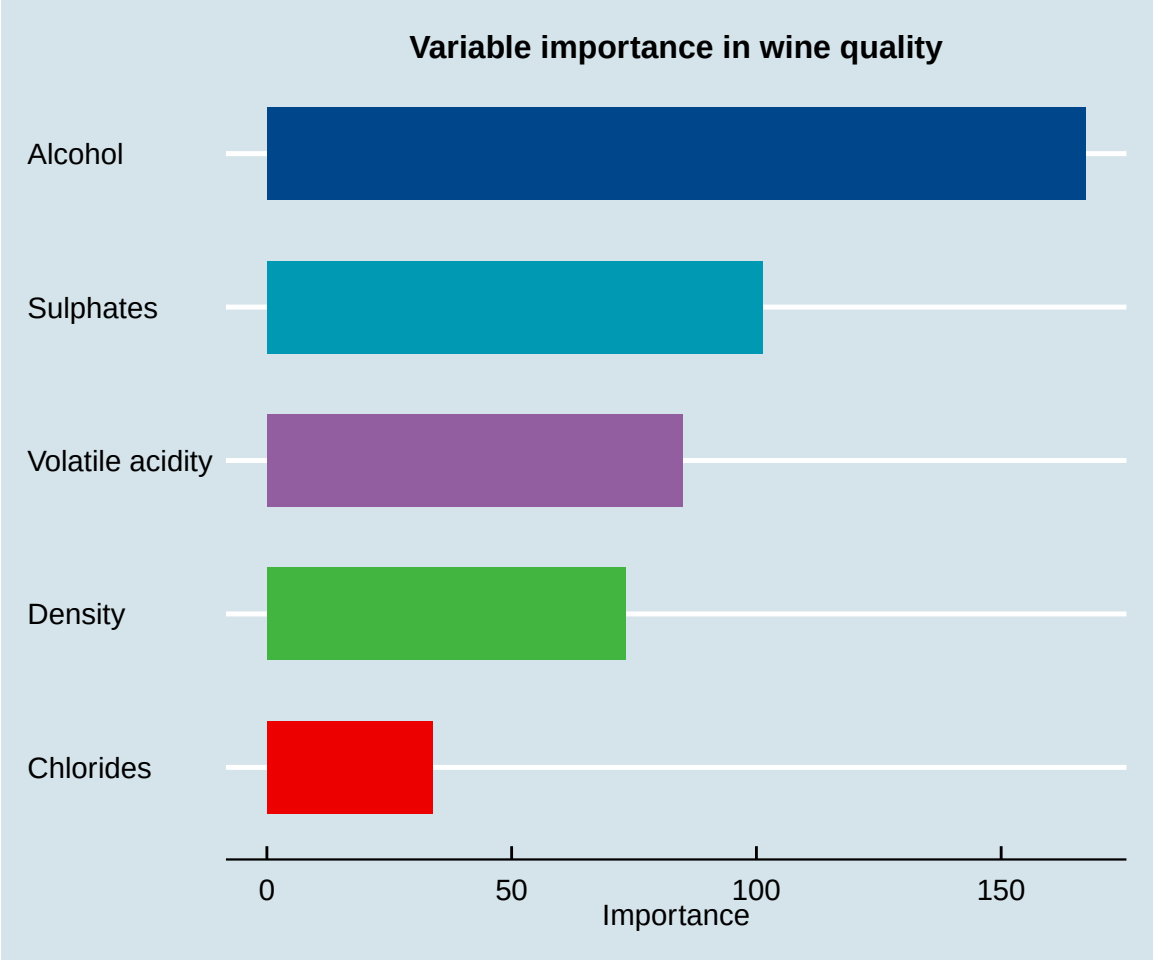
```
tuning[1,3] # optimal cp
```

```
## [1] 0.01
```

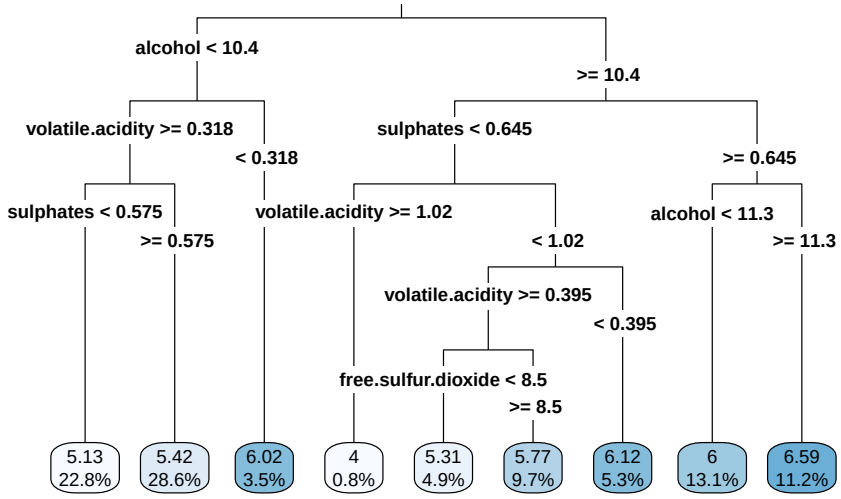
Trees are usually pruned to counter overfitting. Overfitting happens when a model memorizes its training data so well that it is learning noise on top of the signal. Machine learning is a problem of trade-offs, the classic issue is overfitting versus underfitting. In this case, pruning should reduce the size of the tree without reducing predictive accuracy. The tuning parameter chosen to optimize the model's performance was the cost complexity (trade-off with the value that is delivered by complexity), and the optimal value obtained on the previous cross validation was used.

```
pruned_tree <- prune(reg_tree, cp = 0.01)
```

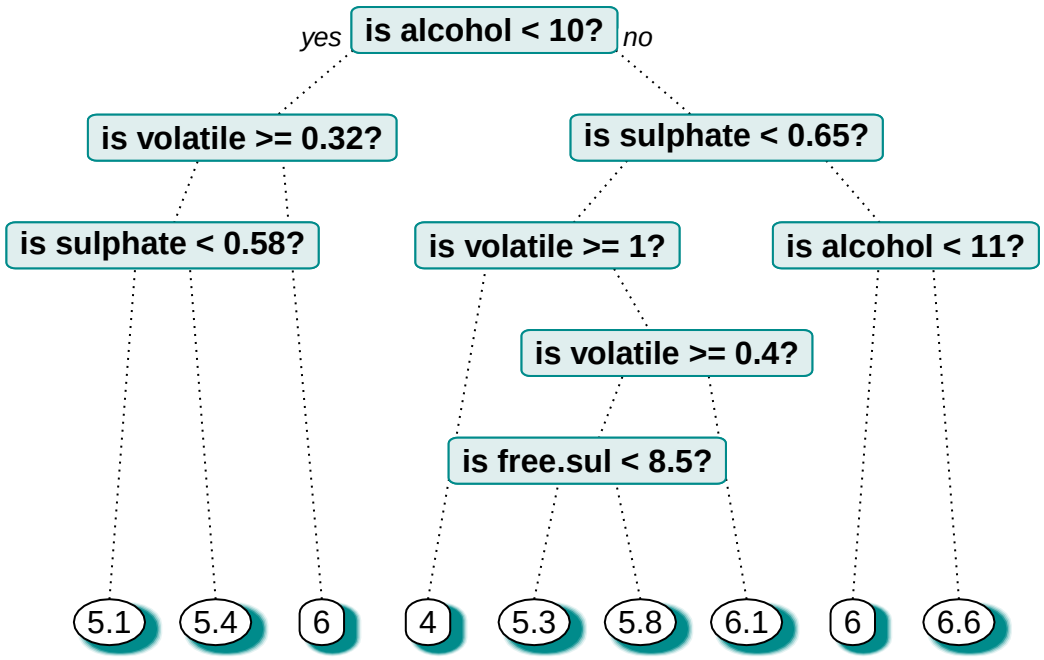
Variable importance was plotted as shown below (top five most important variables), and it was confirmed that alcohol content and volatile acidity have a significant effect on predicting wine quality, along with sulphates, density and chlorides.



A tree plot was generated below from the final pruned tree using the *rpart.plot* package.



The pruned tree plot above was adjusted and simplified with the *prp* function.



Regression tree's RMSE was computed and saved.

```
regtree_pred <- predict(pruned_tree, test_reg) # model evaluation through RMSE
regtree_rmse <- RMSE(test_reg$quality, regtree_pred)
print(regtree_rmse)
```

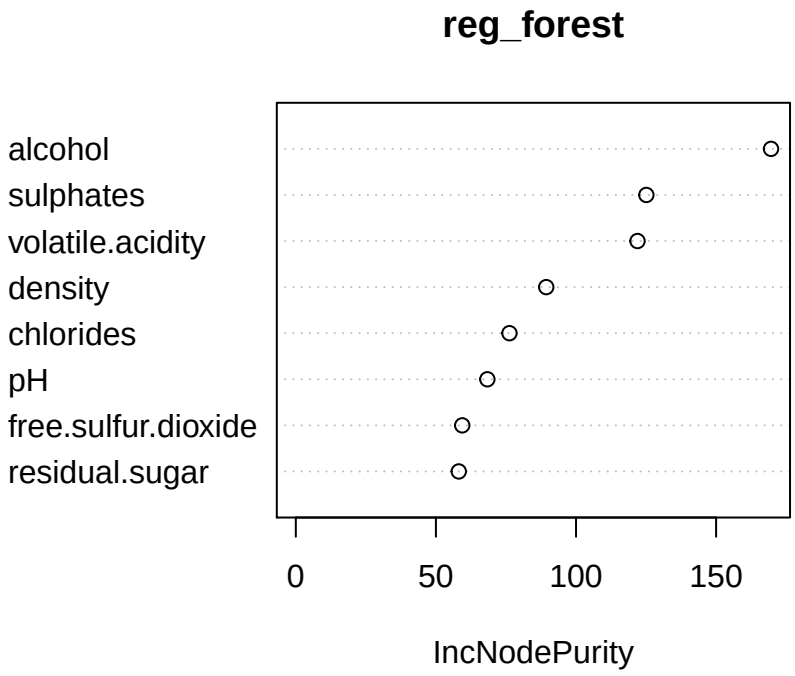
```
## [1] 0.7038633
```

Random Forest regression

To mitigate the decision tree model's possible overfitting and attempt to improve the RMSE the random forest algorithm was used. Random forest is a supervised learning algorithm that builds an ensemble of regression or classification trees, usually trained with the bagging method. The general idea of the bagging method is that a combination of learning models increases the overall result, in this case, the multiple trees obtained are merged to get a more accurate and stable prediction. It can be used for both classification and regression problems. First the *randomForest* package was used.

```
reg_forest <- randomForest(quality ~ .,  # random forest regression fit
                           data = train_reg,
                           type = "regression",
                           proximity = TRUE)
```

Variable importance was plotted below once more, this time using an automatic function from the *randomForest* package.



The train function from the *caret* package was used afterwards for this regression model along with the *Rborist* package and method to perform cross validation. In this case the relevant tuning parameters were *predFixed* which is the number of trial predictors for a split (mtry) and *minNode* which is the minimum number of distinct row references to split a node.

```
regforest_cv <- train(quality ~ .,  # cross validation
      method = "Rborist",
      tuneGrid = data.frame(predFixed = 3, minNode = c(3, 50)),
      data = train_reg)
```

Random forest regression's RMSE was computed and saved.

```
regforest_pred <- predict(regforest_cv, test_reg) # model evaluation through RMSE
rf_rmse <- RMSE(test_reg$quality, regforest_pred)
print(rf_rmse)
```

```
## [1] 0.5931275
```

Classification models fitting

As per the explanation discussed before, the cutoff to transform the data set's integer output variable into a categorical variable was set.

```
# Dichotomization of the integer output variable into moderate quality and high quality

category <- cut(wine_clean$quality,
           breaks = as.character(c(-Inf, 5, Inf)),  # setting the cutoff
           labels = c("Moderate", "High"),
           include.lowest = TRUE)
```

The new data set was created and data partition to obtain train and test sets for the classification models was performed.

```
class_set <- wine_clean %>% mutate(category = category) %>% select(-9)
head(class_set)  # the new category column was added
```

```
##   volatile.acidity residual.sugar chlorides free.sulfur.dioxide density   pH
## 1             0.70            1.9     0.076                  11  0.9978 3.51
## 2             0.88            2.6     0.098                  25  0.9968 3.20
## 3             0.76            2.3     0.092                  15  0.9970 3.26
## 4             0.28            1.9     0.075                  17  0.9980 3.16
## 5             0.70            1.9     0.076                  11  0.9978 3.51
## 6             0.66            1.8     0.075                  13  0.9978 3.51
##   sulphates alcohol category
## 1      0.56     9.4 Moderate
## 2      0.68     9.8 Moderate
## 3      0.65     9.8 Moderate
## 4      0.58     9.8     High
## 5      0.56     9.4 Moderate
## 6      0.56     9.4 Moderate
```

```
set.seed(6, sample.kind="Rounding")
test_index <- createDataPartition(y = class_set$category, times = 1, p = 0.2,
                                  list = FALSE)  # data partition

train_class <- class_set[-test_index,]  # train and test sets
test_class <- class_set[test_index,]
```

The *caret* package train function was also used for the classification models, and the appropriate model control arguments were provided to perform repeated cross validation. The ROC metric was used to create a curve plot for these models.

```
# Computations control for cross validation with the train function
models_control <- trainControl(method = "repeatedcv",
                               number = 10,
                               repeats = 3,
                               classProbs = TRUE,
                               savePredictions = TRUE,
                               summaryFunction = twoClassSummary)
```

Naive Bayes classification

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems which simplifies the probability calculation for each hypothesis to make it more tractable. Rather than attempting to calculate the values of each attribute value, they are assumed to be conditionally independent. This is a very strong assumption that is most unlikely in real data. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold. First, it converts the data set into a frequency table, then it creates a likelihood table by finding specific probabilities, and finally it uses the Naive Bayes equation to calculate posterior probability for each class (in this case moderate or high quality wine). The class with the highest posterior probability is the outcome of prediction.

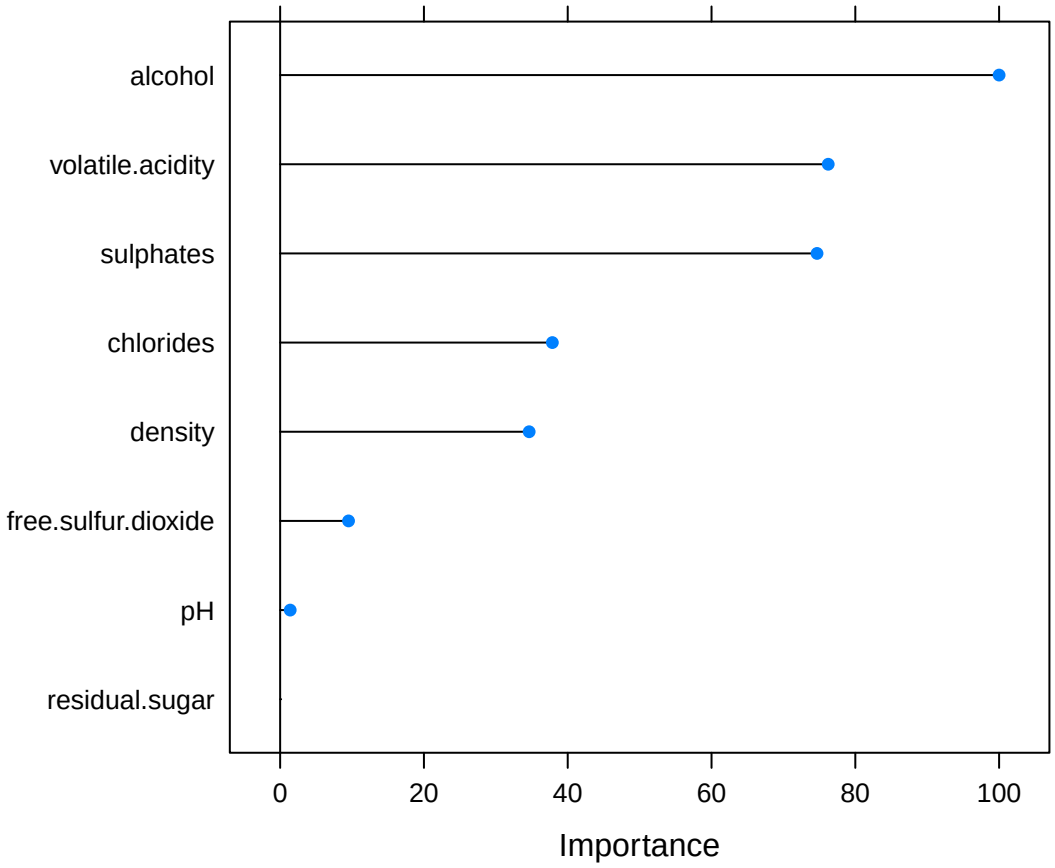
```
bayes_class <- train(category~.,  # Naive Bayes fit
             data = train_class,
             method = "nb",
             metric = "ROC",
             trControl = models_control) # cross validation

bayes_class$results  # results
```

```
##   usekernel fL adjust       ROC      Sens      Spec      ROCSD     SensSD
## 1     FALSE  0      1 0.7893404 0.7069680 0.7513640 0.04509699 0.06225096
## 2      TRUE  0      1 0.8155602 0.7886723 0.7270105 0.03616804 0.05799122
##       SpecSD
## 1 0.04931342
## 2 0.05013343
```

```
bayes_class$bestTune
```

```
##   fL usekernel adjust
## 2  0      TRUE      1
```

Variable importance was plotted and it was observed to be very similar to that of the regression models.



The evaluation metric of choice for the classification models was the accuracy. Model accuracy in terms of classification models can be defined as the ratio of correctly classified samples to the total number of samples. Naive Bayes' confusion matrix and accuracy results were computed and saved.

```
nb_accuracy <- confusionMatrix(predict(bayes_class, test_class),
                               test_class$category)$overall["Accuracy"]
print(nb_accuracy)    # model evaluation through accuracy
```

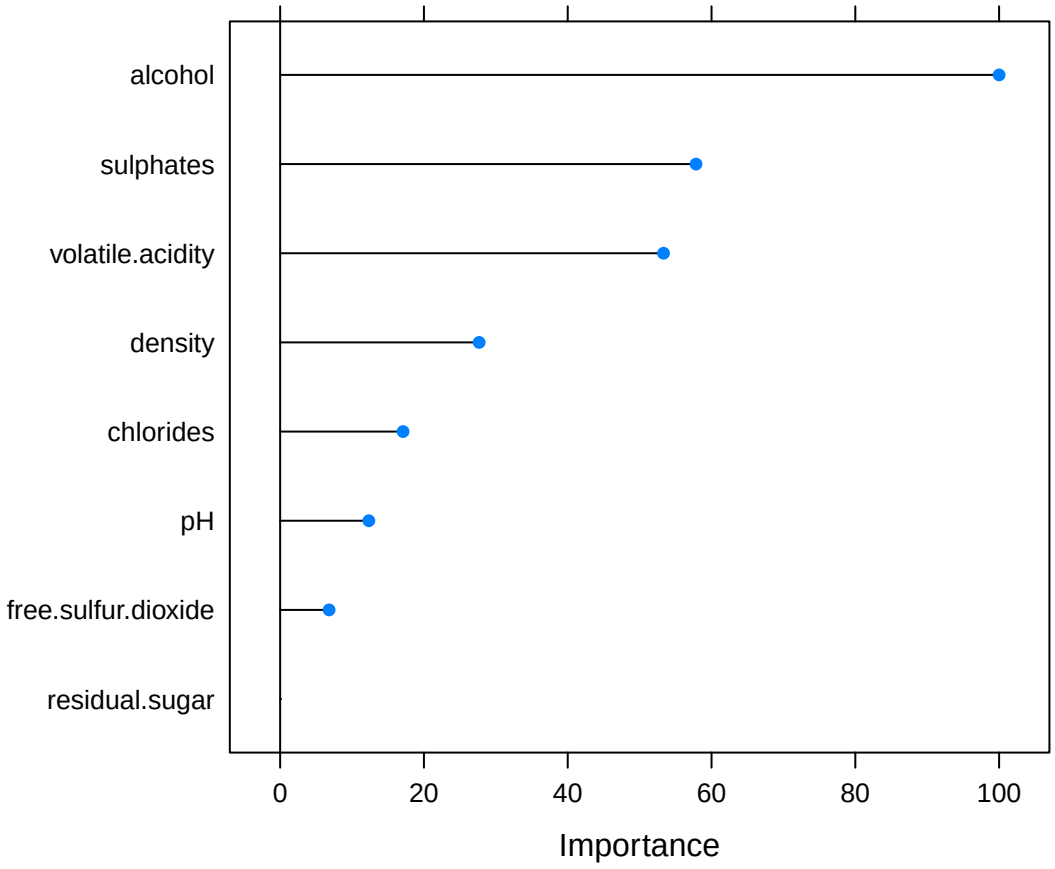
```
## Accuracy
## 0.740625
```

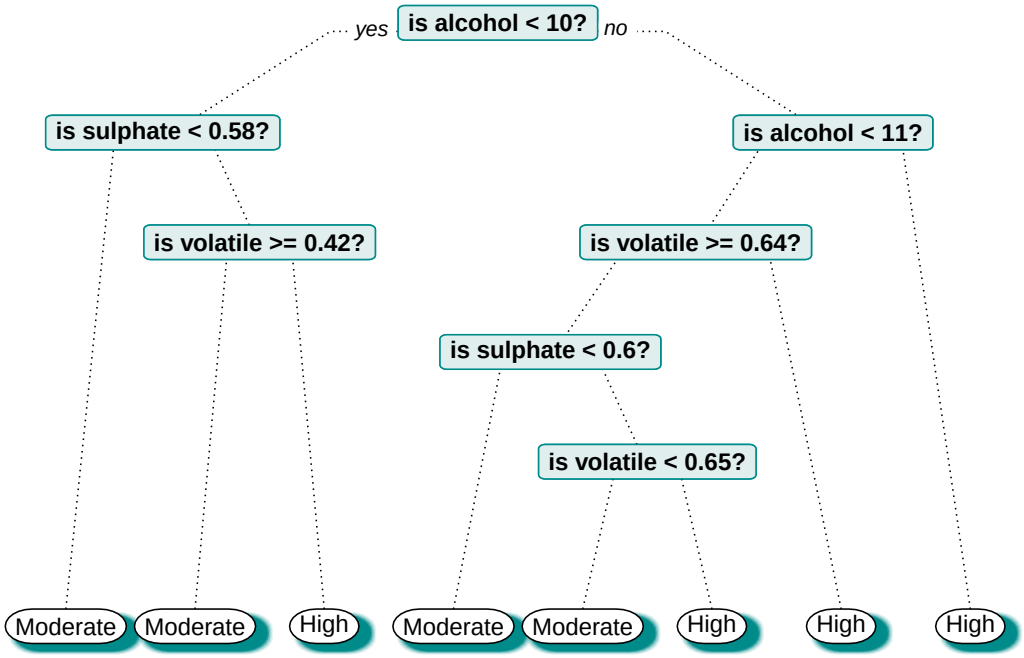
Random forest classification

This time the random forest algorithm was used for classification and the relevant parameter for cross validation was mtry, which refers to the number of variables available for splitting at each tree node.

```
forest_class <- train(category~ .,  # random forest classification fit
                      data = train_class,
                      method = "rf",
                      metric = "ROC",
                      trControl = models_control,
                      tunegrid = expand.grid(mtry = c(1:15))) # cross validation
```

A further look at this model was taken by plotting its variable importance below.





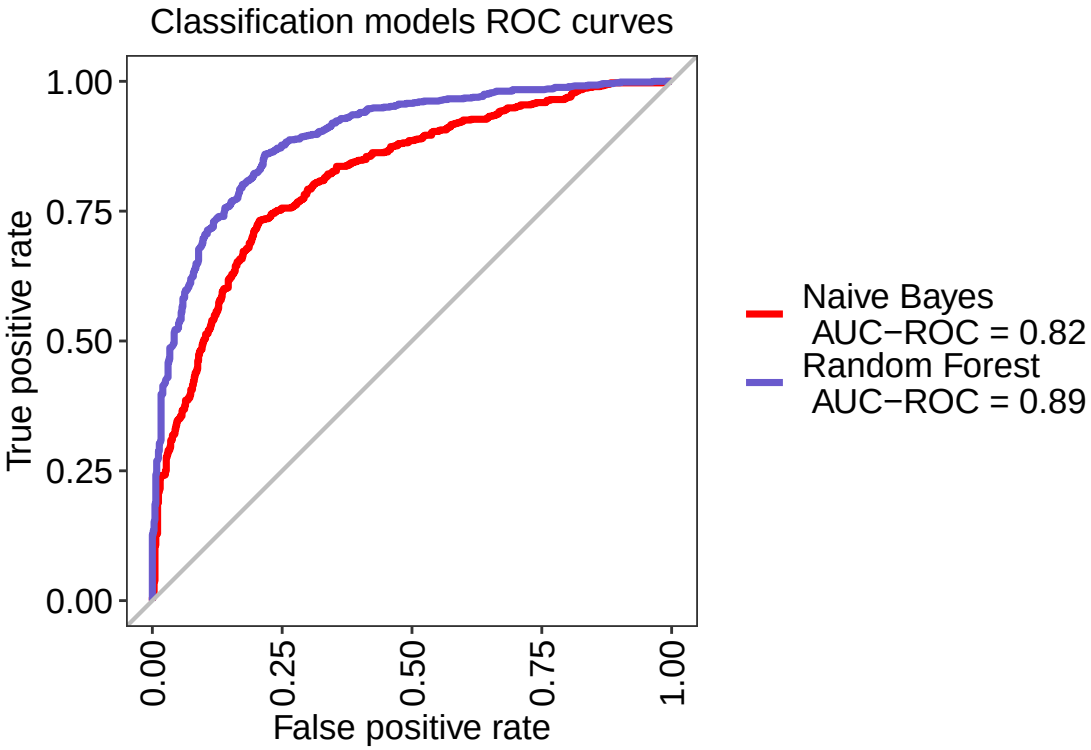
A single classification tree was plotted above.

Random forest classification's confusion matrix and accuracy were computed and saved.

```
rf_accuracy <- confusionMatrix(predict(forest_class, test_class),
                test_class$category)$overall["Accuracy"]
print(rf_accuracy)  # model evaluation through accuracy
```

```
## Accuracy
## 0.784375
```

An ROC curve of the two classification models was plotted below. ROC curves demonstrate the tradeoff between sensitivity and specificity, they also allow to study model's accuracy and compare multiple models. This approach could be useful and relevant in the context of the wine industry and wine quality determination where there is a great benefit from a true positive and not too high cost of a false positive. In this case, by comparing the two models it was observed the random forest model presented a higher AUC value, which indicates how much the model is capable of distinguishing between classes.



3. Results

Model	RMSE
Regression tree	0.7038633
Random Forest Regression	0.5931275

Model	Accuracy
Naive Bayes	0.740625
Random Forest Classification	0.784375

The models fitted to the data set on this project successfully predicted wine quality ratings in the test set. As expected the random forest regression was more effective than the regression tree algorithm, presenting a lower RMSE value. This might be explained by the fact that random forests tend to perform better because their trees are diverse, each one is learned on a random sample, and at each node, a random set of features are considered for splitting, which results in a smoother decision boundary. The random forest classification also slightly outperformed the Naive Bayes model presenting a higher accuracy, this might be due to the fact that the Naive Bayes algorithm performs better in cases where the input variables are categorical as opposed to numerical. This is because for numerical variables a normal distribution is assumed, which is a strong assumption. The variable importance for prediction was practically the same in all models, alcohol content was the most significant predictor followed by volatile acidity and sulphates.

(Lantz and Brett, 2013) (Schumacker, 2014) (Irrizary, 2019)

4. Conclusion

Red wine quality was successfully predicted with both regression and categorical approaches and the evaluation metrics utilized are acceptable for this project's specific task, however, more sophisticated evaluation metrics could also be used to determine the model's prediction effectiveness. Further model improvement and also fitting other models should not be discarded, as machine learning is the art of trial and error of finding the best features. In this project a real-world application of data science and machine learning was presented. Data driven sensory quality prediction is relevant not only in the wine industry but also in the food science domain in general, plus these techniques are also being applied to resolve other matters, such as food shelf life prediction, food market trends analysis, processes automatization and standardization, among others. As previously stated, data analysis is not meant to replace the human labor element of this discipline but to complement it to promote understanding of the link between physicochemical characterization and products' quality while also minimizing subjective error. There is a promising outlook for this data science application as it is increasingly being implemented and improved with diverse powerful approaches such as data mining, neural networks and support vector machines.

References

Jantis J.E. 1992. The Role of Sensory Analysis in Quality Control. ASTM publications, Philadelphia.

Cortez, Paulo & Teixeira, Juliana & Cerdeira, António & Almeida, Fernando & Matos, Telmo & Reis, José. (2009). Using Data Mining for Wine Quality Assessment. 66-79.

Marks D. 2019. A critique of wine ratings. American association of wine economists. No. 239

Sanchez-Palomo E., Izquierdo Cañas P.M., Delgado J.A., Gonzalez Viñas, M.A. 2018. Sensory Characterization of Wines Obtained by Blending Cencibel Grapes and Minority Grape Varieties Cultivated in La Mancha Region. Hindawi Journal of Food Quality. Wiley.

Villamor R.R. 2012. The impact of wine components on the chemical and sensory properties of wines. School of food science. Washington state university.

Yildirim H, Yucel U, Elmaci Y, Ova G and Altug T. 2005. Interpretation of organic wine's flavour profile by multivariate statistical analysis. Department of food engineering, Ege University, Turkey.

Lantz B. 2013. Machine Learning with R. Packt Publishing.

Schumacker R.E. 2014. Learning statistics using R. Sage.

Irizarry R.A. 2019. Introduction to data science. CRC Press.

Appendix

```
# Environment
print("Operating System:")
```

```
## [1] "Operating System:"
```

```
version
```

```
##                _
## platform       x86_64-apple-darwin17.0
## arch           x86_64
## os             darwin17.0
## system         x86_64, darwin17.0
## status
## major          4
## minor          0.3
## year           2020
## month          10
## day            10
## svn rev        79318
## language       R
## version.string R version 4.0.3 (2020-10-10)
## nickname       Bunny-Wunnies Freak Out
```