# MSDS680 - Week 4 - Decision Trees

Benjamin Siebold

Regis University

MSDS680 - Week 4 - Decision Trees

## Introduction

In this project the wine dataset from UCI ML repository will be taken and using decision trees and random forests, the quality of the wine will be predicted based off different features. The wine dataset is made up of 1599 rows and 11 features, with six different rankings assigned to different wines. The source data for this analysis can be referenced here: https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

## Methodology

### Set Up

The first step in the process is to set up the environment for analysis and load in the data. This will entail loading in the necessary packages for analysis, setting the seedalong with loading in the dataset from the source.

```
library(DataExplorer)  #Used for initial investigation
library(caret)  #Used for performance of models
library(randomForest)  #Used to apply randomForest model
library(rpart)  #Used for Decision Tree models
library(rpart.plot)  #Used for plotting trees
library(plyr)  #Used for data splits and counts


set.seed(486)


wine_data <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-qu
    sep = ";", header = T)
```

### Data Investigation

With the data now loaded, an initial investigation is necessary. A correlation plot, as well as an investigation to make sure there is no missing data. The columns will also be renamed to make them shorter and moer easily read in a plot.

```
head(wine_data, 10)
```

```
##    fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1            7.4             0.70        0.00            1.9     0.076
## 2            7.8             0.88        0.00            2.6     0.098
## 3            7.8             0.76        0.04            2.3     0.092
## 4           11.2             0.28        0.56            1.9     0.075
## 5            7.4             0.70        0.00            1.9     0.076
## 6            7.4             0.66        0.00            1.8     0.075
## 7            7.9             0.60        0.06            1.6     0.069
## 8            7.3             0.65        0.00            1.2     0.065
## 9            7.8             0.58        0.02            2.0     0.073
## 10           7.5             0.50        0.36            6.1     0.071
##    free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                   11                  34  0.9978 3.51      0.56     9.4
## 2                   25                  67  0.9968 3.20      0.68     9.8
## 3                   15                  54  0.9970 3.26      0.65     9.8
## 4                   17                  60  0.9980 3.16      0.58     9.8
## 5                   11                  34  0.9978 3.51      0.56     9.4
## 6                   13                  40  0.9978 3.51      0.56     9.4
## 7                   15                  59  0.9964 3.30      0.46     9.4
## 8                   15                  21  0.9946 3.39      0.47    10.0
```

```
## 9                    9                   18  0.9968 3.36       0.57      9.5
## 10                  17                  102  0.9978 3.35       0.80     10.5
##    quality
## 1        5
## 2        5
## 3        5
## 4        6
## 5        5
## 6        5
## 7        5
## 8        7
## 9        7
## 10       5
```

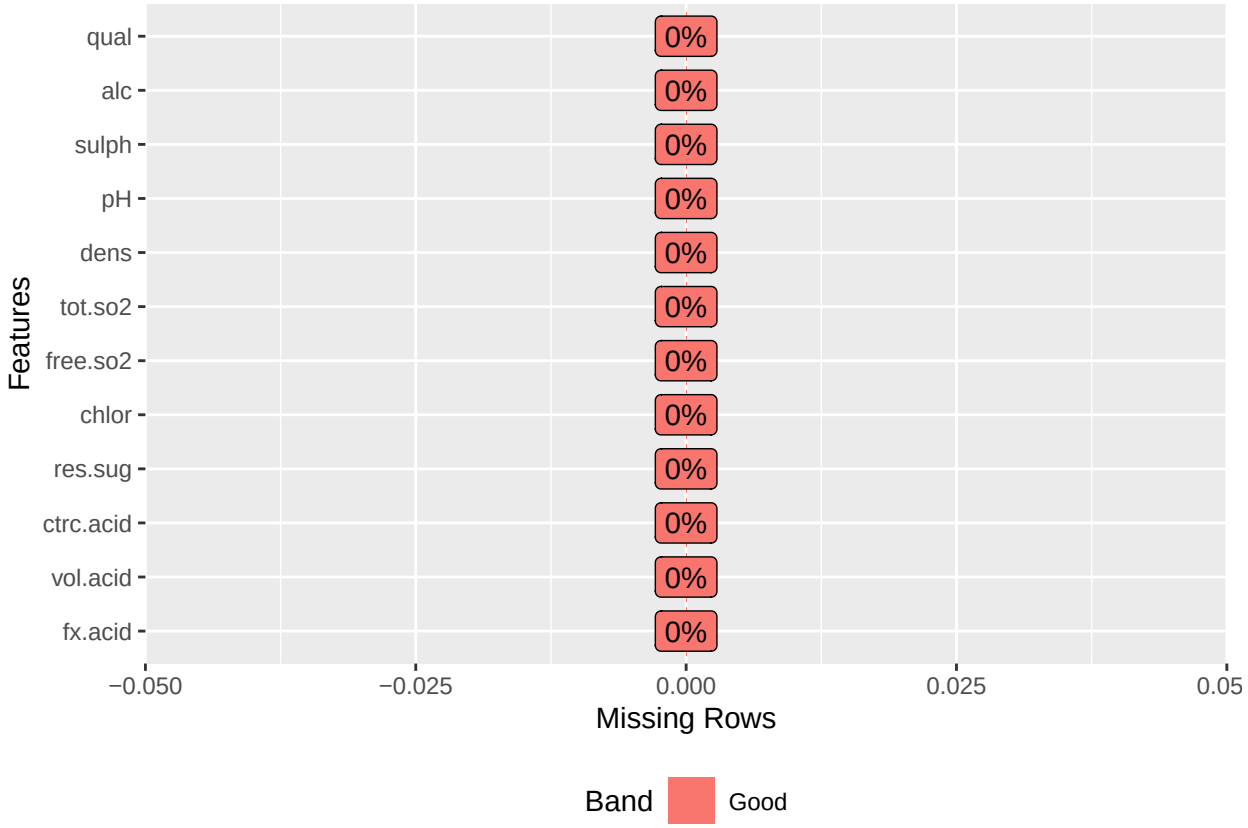
```
col_names <- c("fx.acid", "vol.acid", "ctrc.acid", "res.sug",
    "chlor", "free.so2", "tot.so2", "dens", "pH", "sulph", "alc",
    "qual")
names(wine_data) <- col_names


summary(wine_data)
```

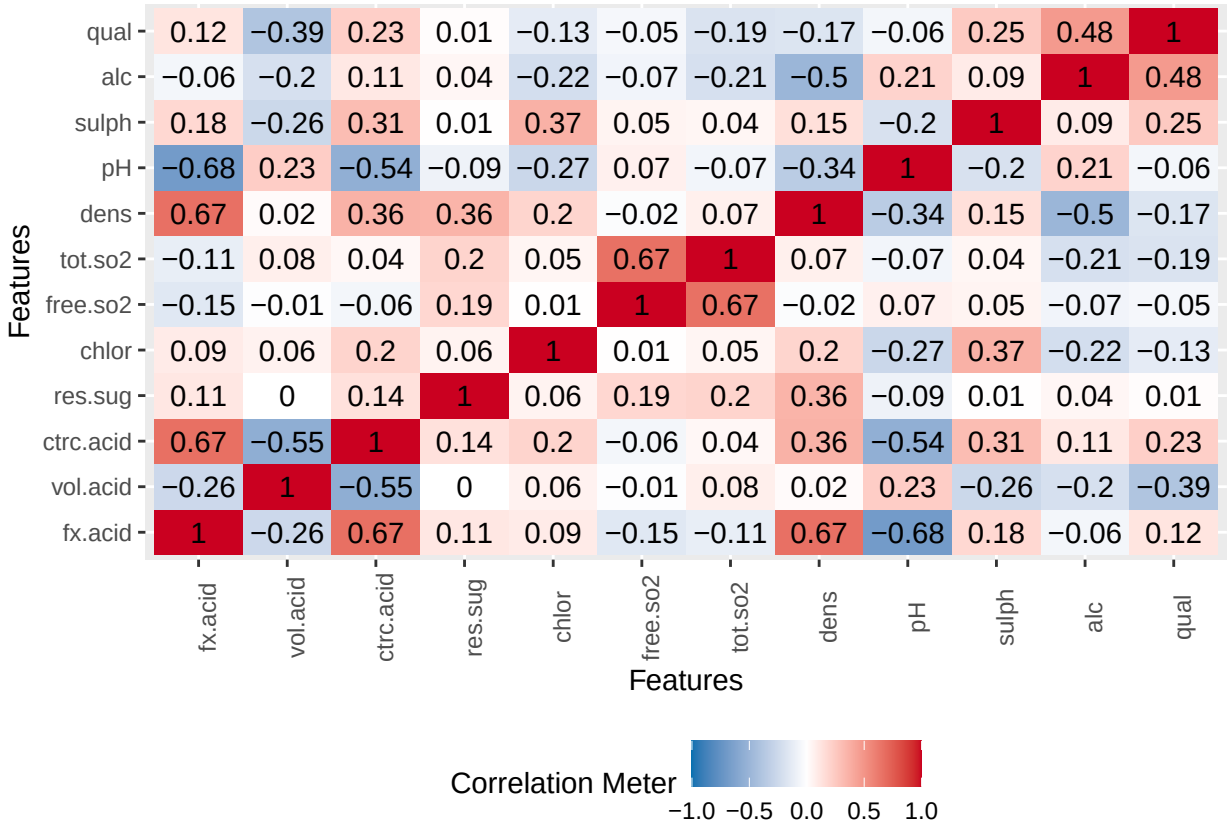
```
##     fx.acid         vol.acid        ctrc.acid        res.sug
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5278   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
```

```
##      chlor           free.so2       tot.so2          dens
##  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            sulph             alc             qual
##  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
```

```
plot_missing(wine_data)
```




```
plot_correlation(wine_data)
```




```
count(wine_data, vars = "qual")
```

```
##   qual freq
## 1    3   10
## 2    4   53
## 3    5  681
## 4    6  638
## 5    7  199
## 6    8   18
```

Above it can be seen there is no missing data. The correlation matrix shows the quality of wine is most heavily impacted by it's alcohol level, and it's volatile.acidity. More interestingly is the count of wine rankings. Although there are 6 different rankings of quality, only .6% of the wine were of quality 3, and only 1.1% of the wines were of quality

9. The majority of the wines (82%) were ranked 5 or 6. This brings up the question of whether or not the rankings should stay as is, or be reajusted to a new quality system. The below were run three times, normal, with ranks 3,4 and 7,8 combined, and finally with two rankings ("High"/"Low").

Of the three the 2 grouped model performed the most accurate, but it didn't seem to accomplish the goal. Wine of quality 8 would have to be much better than that of 6, and there was plenty of wine ranked 7. For this reason, the decision was made to use the quality grouped into 4 rankings instead of 6.

### Data Cleaning and Preparation

With the decision made to use four rankings, the wine quality needs to be combined into smaller rankings and other steps to prepare for model application will be completed. This includes assigning the quality column to a factor, and splitting the data into training and test data.

```
wine_data$qual[wine_data$qual == 3] <- 4
wine_data$qual[wine_data$qual == 8] <- 7
count(wine_data, vars = "qual")
```

```
##   qual freq
## 1    4   63
## 2    5  681
## 3    6  638
## 4    7  217
```

```
wine_data$qual <- as.factor(wine_data$qual)
idx = createDataPartition(wine_data$qual, p = 0.7, list = F)
```

```
wine_train = wine_data[idx, ]
wine_test = wine_data[-idx, ]
prop.table(table(wine_train$qual))
```

```
##
##          4          5          6          7
## 0.04014273 0.42551293 0.39875112 0.13559322
```

```
prop.table(table(wine_test$qual))
```

```
##
##         4         5         6         7
## 0.0376569 0.4267782 0.3995816 0.1359833
```

### Model Application

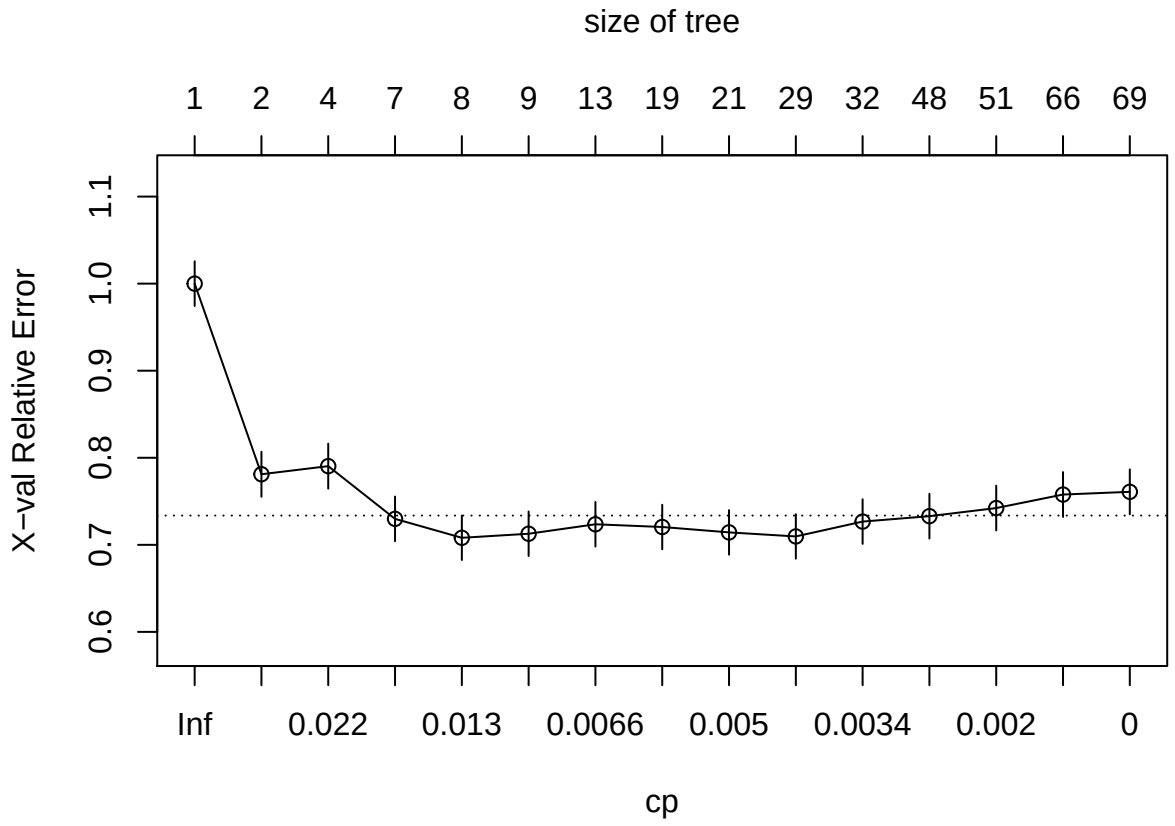
With the data cleaned and split, the next step in the analysis is to build and visualize the tree, apply the model to test data, and analyze the results.

```
wine_tree <- rpart(qual ~ ., data = wine_train, cp = 0)
printcp(wine_tree)
```

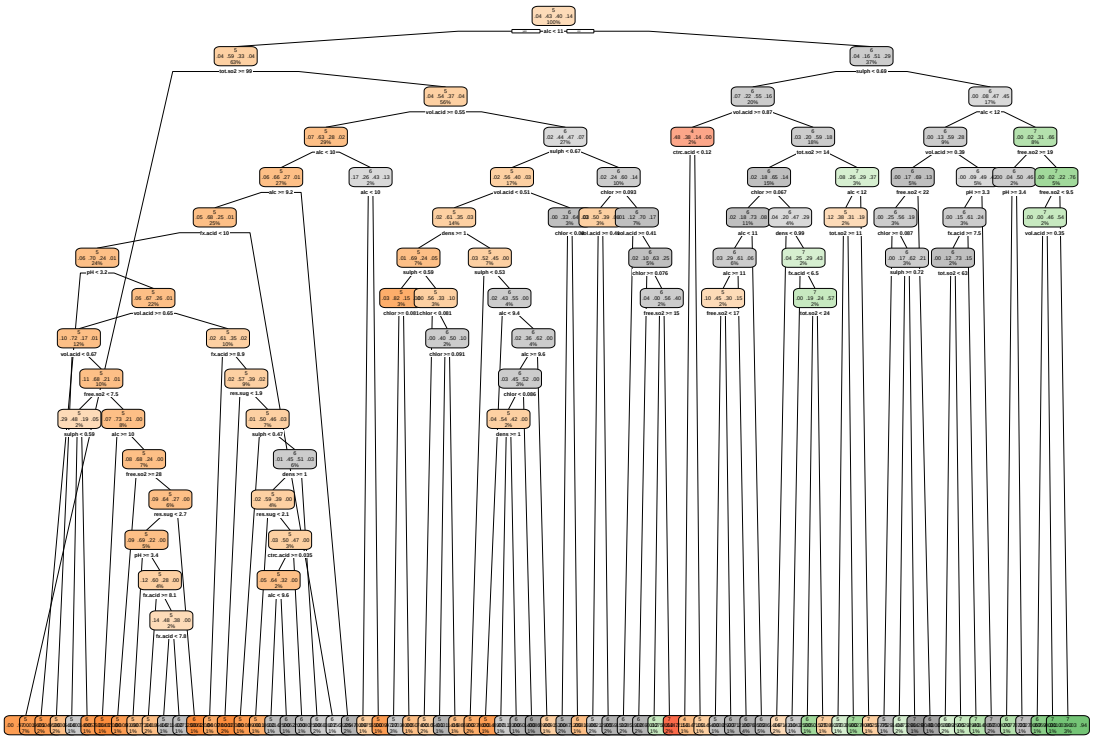
```
##
## Classification tree:
## rpart(formula = qual ~ ., data = wine_train, cp = 0)
##
## Variables actually used in tree construction:
##  [1] alc       chlor     ctrc.acid dens      free.so2  fx.acid   pH
```

```
##  [8] res.sug   sulph     tot.so2   vol.acid
##
## Root node error: 644/1121 = 0.57449
##
## n= 1121
##
##           CP nsplit rel error  xerror     xstd
## 1  0.2298137      0   1.00000 1.00000 0.025705
## 2  0.0232919      1   0.77019 0.78106 0.025858
## 3  0.0212215      3   0.72360 0.79037 0.025885
## 4  0.0155280      6   0.65994 0.72981 0.025654
## 5  0.0108696      7   0.64441 0.70807 0.025539
## 6  0.0069876      8   0.63354 0.71273 0.025565
## 7  0.0062112     12   0.60559 0.72360 0.025623
## 8  0.0054348     18   0.56832 0.72050 0.025607
## 9  0.0046584     20   0.55745 0.71429 0.025574
## 10 0.0036232     28   0.51708 0.70963 0.025548
## 11 0.0031056     31   0.50621 0.72671 0.025638
## 12 0.0025880     47   0.45031 0.73292 0.025669
## 13 0.0015528     50   0.44255 0.74224 0.025712
## 14 0.0005176     65   0.41770 0.75776 0.025776
## 15 0.0000000     68   0.41615 0.76087 0.025788
```

```
plotcp(wine_tree)
```




```
rpart.plot(wine_tree, branch = 0.3)
```



The tree was built with every features available when setting the cp=0, which creates one, a hard to interpret decision tree, and two may have a problem of overfitting.

```
wine_pred <- predict(wine_tree, wine_test, type = "class")
confusionMatrix(wine_pred, wine_test$qual)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   4   5   6   7
##          4   0   1   3   0
##          5  12 120  43   9
##          6   5  72 113  23
##          7   1  11  32  33
##
## Overall Statistics
##
##                Accuracy : 0.5565
##                  95% CI : (0.5107, 0.6016)
##     No Information Rate : 0.4268
##     P-Value [Acc > NIR] : 8.112e-09
##
##                   Kappa : 0.302
##
##  Mcnemar's Test P-Value : 0.003014
##
## Statistics by Class:
##
```

```
##                      Class: 4 Class: 5 Class: 6 Class: 7
## Sensitivity          0.000000   0.5882   0.5916  0.50769
## Specificity          0.991304   0.7664   0.6516  0.89346
## Pos Pred Value       0.000000   0.6522   0.5305  0.42857
## Neg Pred Value       0.962025   0.7143   0.7057  0.92020
## Prevalence           0.037657   0.4268   0.3996  0.13598
## Detection Rate       0.000000   0.2510   0.2364  0.06904
## Detection Prevalence 0.008368   0.3849   0.4456  0.16109
## Balanced Accuracy    0.495652   0.6773   0.6216  0.70058
```

With an accuracy score of .5565 the tree does not seem to be an accurate source of determining wine quality. To hopefully alleviate this, the tree will next be pruned and the above steps reproduced to see if results get better. In order to prune the tree, the minimum xerror value will be taken from the splits above, and by finding the CP value correlated with the minimum xerror value, the splits occuring after the CP will be removed, pruning the tree.

```
min(wine_tree$cptable[, "xerror"])
```

```
## [1] 0.7080745
```

```
which.min(wine_tree$cptable[, "xerror"])
```

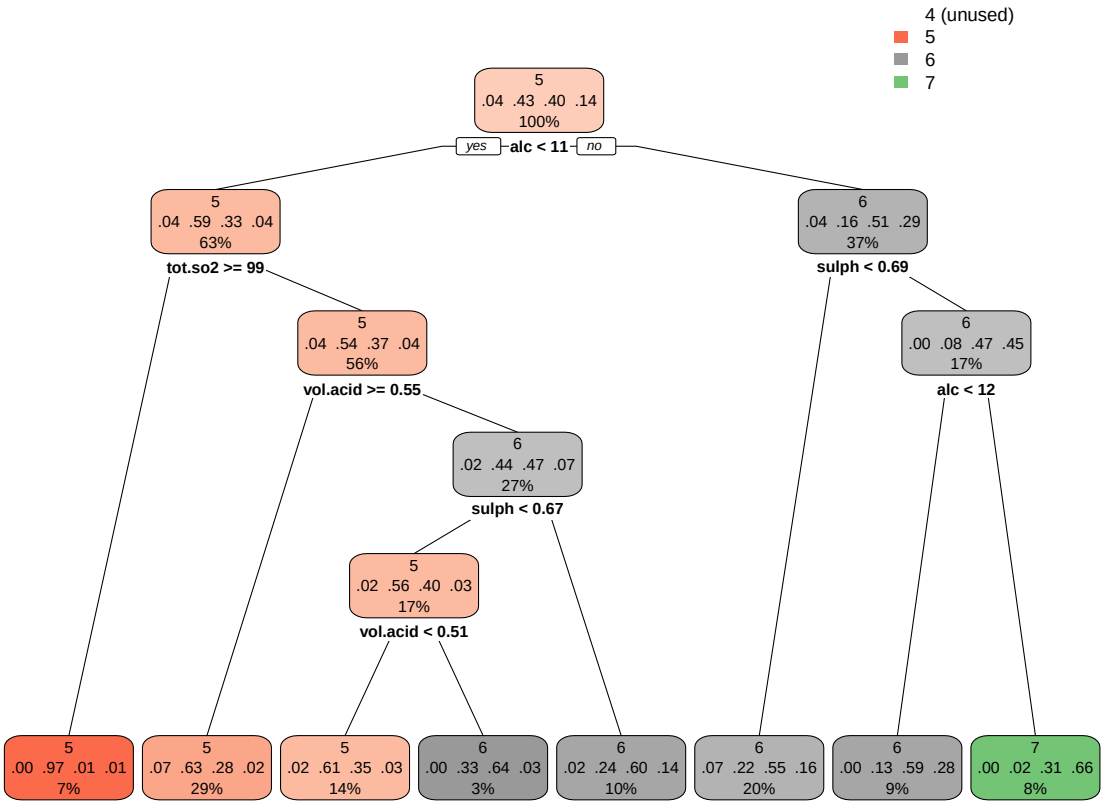
```
## 5
## 5
```

```
wine_cp <- wine_tree$cptable[which.min(wine_tree$cptable[, "xerror"]),
    "CP"]
```

```
pruned_wine <- prune(wine_tree, cp = wine_cp)
printcp(pruned_wine)
```

```
##
## Classification tree:
## rpart(formula = qual ~ ., data = wine_train, cp = 0)
##
## Variables actually used in tree construction:
## [1] alc      sulph    tot.so2  vol.acid
##
## Root node error: 644/1121 = 0.57449
##
## n= 1121
##
##         CP nsplit rel error  xerror     xstd
## 1 0.229814      0   1.00000 1.00000 0.025705
## 2 0.023292      1   0.77019 0.78106 0.025858
## 3 0.021222      3   0.72360 0.79037 0.025885
## 4 0.015528      6   0.65994 0.72981 0.025654
## 5 0.010870      7   0.64441 0.70807 0.025539
```

```
rpart.plot(pruned_wine, branch = 0.3)
```




```
pruned_pred <- predict(pruned_wine, wine_test, type = "class")

confusionMatrix(pruned_pred, wine_test$qual)
```

```
## Confusion Matrix and Statistics

##

##           Reference

## Prediction   4   5   6   7

##          4   0   0   0   0

##          5  11 142  62   4

##          6   7  61 120  43

##          7   0   1   9  18

##

## Overall Statistics

##
```

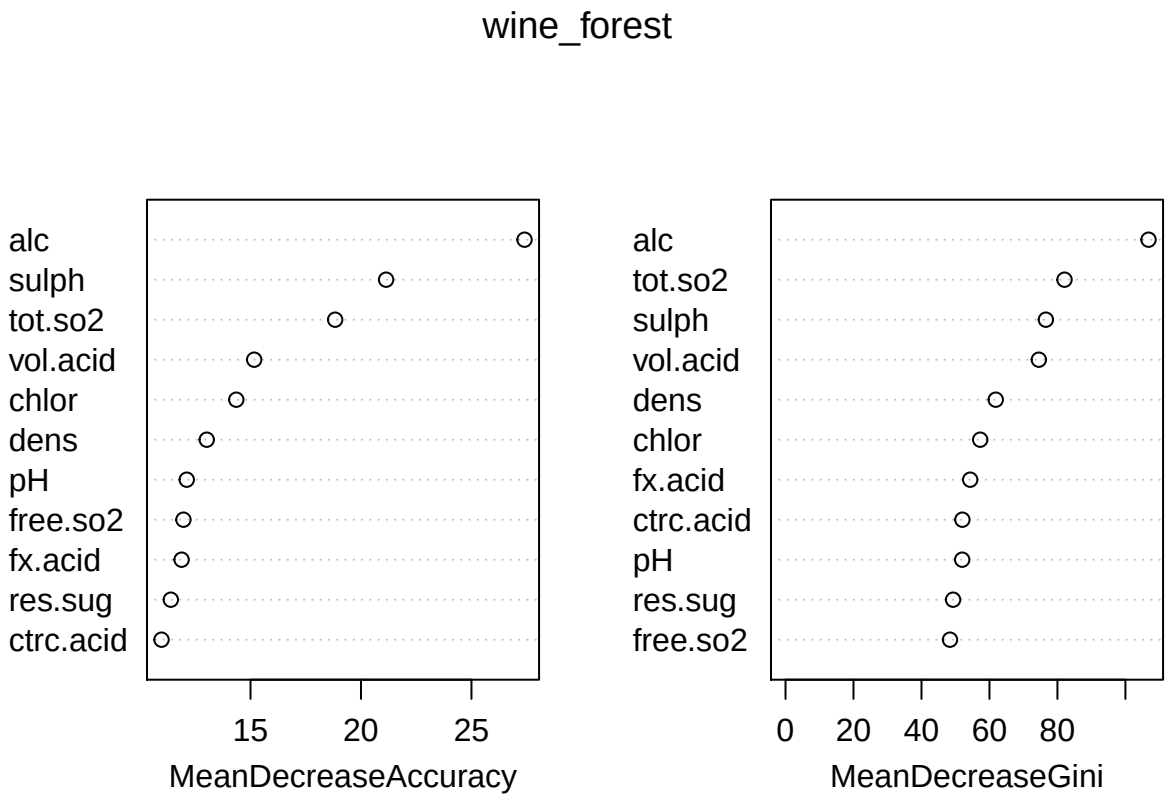
```
##                 Accuracy : 0.5858
##                   95% CI : (0.5402, 0.6303)
##     No Information Rate : 0.4268
##     P-Value [Acc > NIR] : 2.062e-12
##
##                    Kappa : 0.3135
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 4 Class: 5 Class: 6 Class: 7
## Sensitivity           0.00000   0.6961   0.6283  0.27692
## Specificity           1.00000   0.7190   0.6132  0.97579
## Pos Pred Value            NaN   0.6484   0.5195  0.64286
## Neg Pred Value        0.96234   0.7606   0.7126  0.89556
## Prevalence            0.03766   0.4268   0.3996  0.13598
## Detection Rate        0.00000   0.2971   0.2510  0.03766
## Detection Prevalence  0.00000   0.4582   0.4833  0.05858
## Balanced Accuracy     0.50000   0.7075   0.6208  0.62636
```

The pruned tree accomplished a few things. Looking at the tree it is readable and it can be seen the features were cut off at four features being used, instead of 14. This, coupled with the accuracy score of the pruned tree being .5858 shows pruning the tree was successful. It not only improved the real accuracy of the model, but also by removing unnecessary features will improve the ability to reproduce along with the simplicity of predicting the quality of wine.

### Random Forest Application

Although the pruned tree slightly increased the models accuracy, it still is not nearly close to reliable enough to use. Fortunately Random Forests usually have better performance than single trees. The last model that will be applied to the wine dataset is a random forest model. Along with the model predicting test data, a plot of variable importance will be produced to assist with interpreting the forest.

```
wine_forest <- randomForest(qual ~ ., wine_train, ntree = 100,
    importance = T)
varImpPlot(wine_forest)
```




```
forest_pred <- predict(wine_forest, wine_test, type = "class")
confusionMatrix(forest_pred, wine_test$qual)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   4   5   6   7
##          4   1   0   1   0
##          5  13 164  35   2
##          6   4  39 148  31
##          7   0   1   7  32
##
## Overall Statistics
##
##                Accuracy : 0.7218
##                  95% CI : (0.6792, 0.7615)
##     No Information Rate : 0.4268
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5452
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 4 Class: 5 Class: 6 Class: 7
## Sensitivity          0.055556   0.8039   0.7749  0.49231
## Specificity          0.997826   0.8175   0.7422  0.98063
## Pos Pred Value       0.500000   0.7664   0.6667  0.80000
## Neg Pred Value       0.964286   0.8485   0.8320  0.92466
## Prevalence           0.037657   0.4268   0.3996  0.13598
```

```
## Detection Rate       0.002092   0.3431   0.3096  0.06695
## Detection Prevalence 0.004184   0.4477   0.4644  0.08368
## Balanced Accuracy    0.526691   0.8107   0.7585  0.73647
```

Fortunately the random forest did provide a much more accurate model with an accuracy score of .7218. Unfortunately this is still not a very performant algorithm, and would not be suggested to use on predicting the wine scores, as it struggled with both the wine ranked 4 and 7 quite a bit. The variable importance plots align quite well with the pruned tree, showing in the RF model alcohol was significantly more impactful in predicting the quality than any other feature, and the logarithmic shape of the accuracy plot shows the bottom half of the features are not very helpful.

**Conclusion**

Overall, the analysis above shows the potential use of trees, pruning, and random forests; along with the limitations of using any of the models with data. Having a deeper dataset with more data from the outer classes potentially could've greatly increased all three models accuracy, and especially the random forest models accuracy. It can be seen that using a random forest opposed to a singular tree can easily increase the accuracy of the model by 10-20% if not more, and in most cases would prove to be used over a singular decision tree. This concludes the Random Forest analysis in R.

## References

Lantz, B. (2015). Machine Learning with R : Expert Techniques for Predictive Modeling to Solve All Your Data Analysis Problems: Vol. Second edition. Packt Publishing.

Yu-Wei, C. (2015). Machine Learning with R Cookbook. Packt Publishing.