# Wine quality- decision tree/ random forest

Brittani Wilson

May 31, 2019

```
require(caret)
require(rpart)
require(rpart.plot)
require(randomForest)
set.seed(1)


str(wine)
```

```
## '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 0.073 0.071 ...
##  $ 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 ...
```

First we change the predictor variable "quality"" to a factor. Before the data is split it is important to note that there are no "na" values. The integer ratings in data frame range for 3 to 8 so in the histogram ratings at level 3 and 4 represent ratings 5 and 6 in the data frame. Now we can split the data using the decision tree classification method, the wine is classified into six levels based on its properties. The samples are randomized so it is important to set the seed and then we split the data into 80% for training and 20% for testing respectively to create two randomized samples of the data.

In this model "no" will always be to the right and "yes" will always be to the left with each branch representing a decision for splitting the data into a new classification. The decision tree split the data into only 3 of the 6 available classifications: 5, 6 and 7 as indicated by the "unused" 3,4,and 8 in the key. We can see the furthest branches show that the prediction made a good amount of errors, and when tested on unseen data the predictions were only 57% accurate, which isn't very good.

```
table(wine$quality)
```

```
##
##   3   4   5   6   7   8
##  10  53 681 638 199  18
```

```
names(wine)
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
```

```
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
```

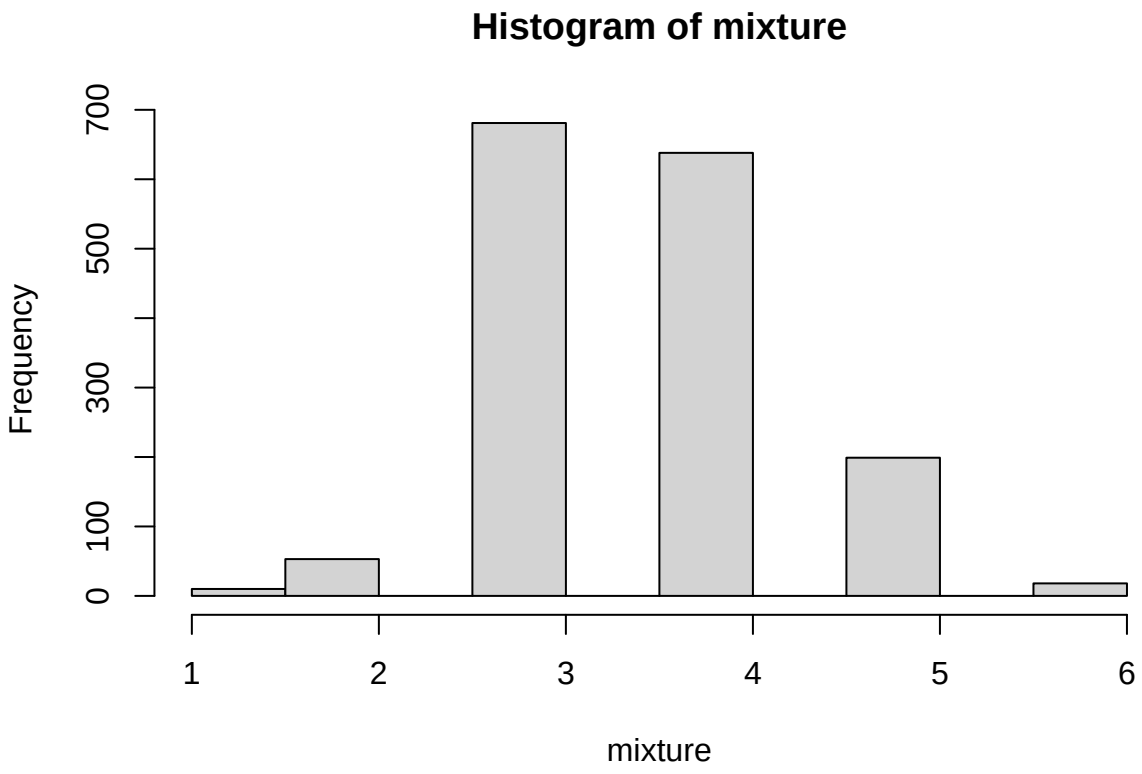
```
sum(is.na(wine))
```

```
## [1] 0
```

```
wine$quality <- as.factor(wine$quality)
str(wine$quality)
```

```
##  Factor w/ 6 levels "3","4","5","6",..: 3 3 3 4 3 3 3 5 5 3 ...
```

```
mixture <- as.numeric(wine$quality)
hist(mixture)
```




```
.8 * 1599
```

```
## [1] 1279.2
```

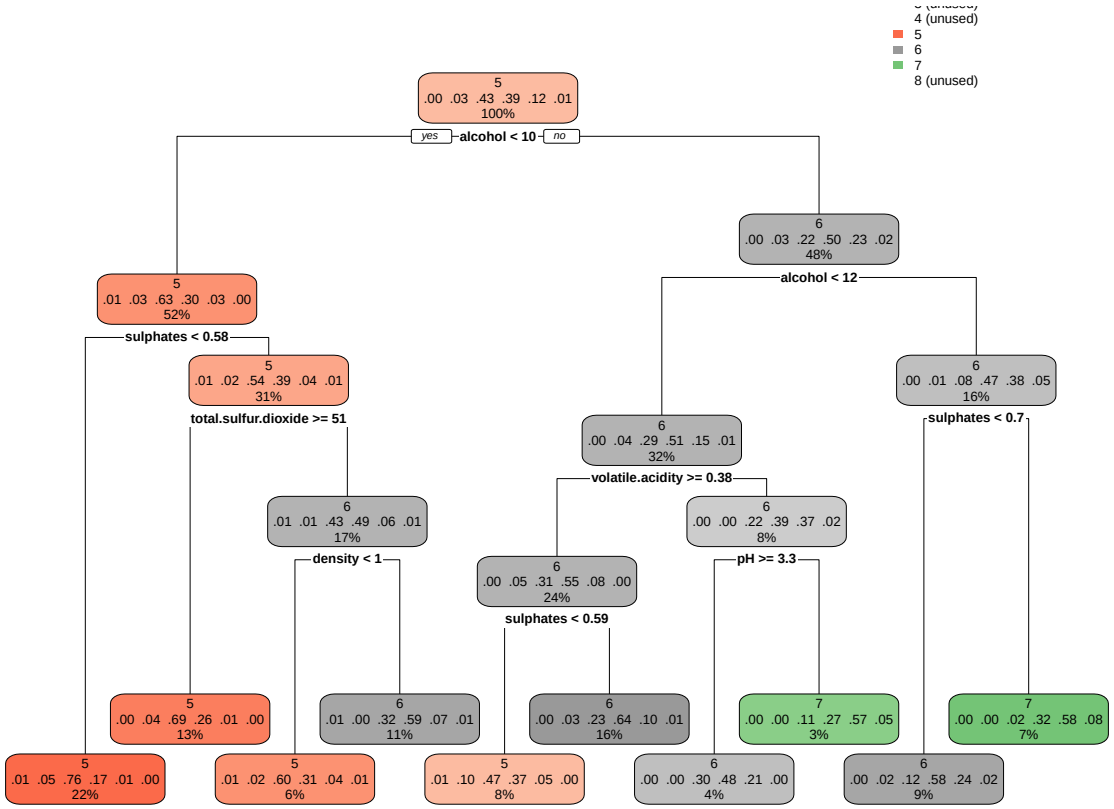
```
s <- sample(1599, 1279)
wine_train <- wine[s, ]
wine_test <- wine[-s, ]
dim(wine_train)
```

```
## [1] 1279   12
```

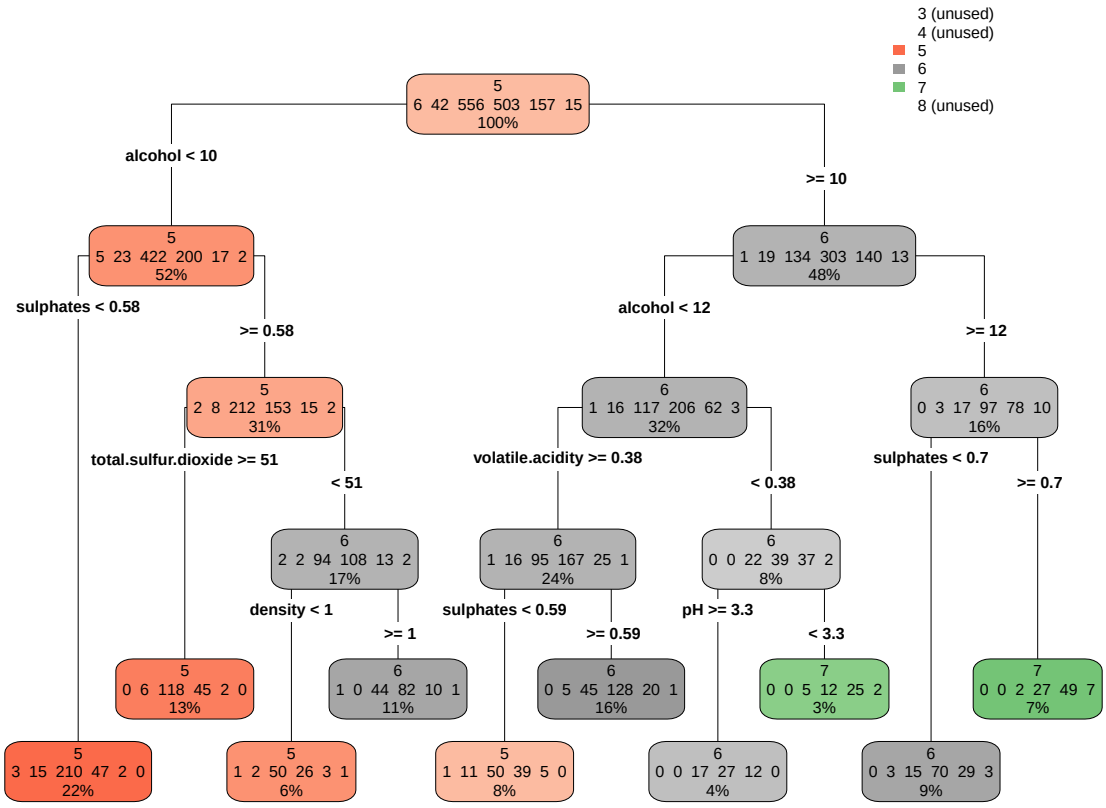
```
dim(wine_test)
```

```
## [1] 320  12
```

```
tm <- rpart(quality~., wine_train, method = "class")
rpart.plot(tm, tweak = .9)
```




```
rpart.plot(tm, type = 4, extra = 101, tweak = .9)
```




```
pred <- predict(tm, wine_test, type = "class")
table(wine_test$quality, pred)
```

```
##    pred
##      3  4  5  6  7  8
##   3  0  0  3  1  0  0
##   4  0  0  9  2  0  0
##   5  0  0 90 34  1  0
##   6  0  0 48 73 14  0
##   7  0  0  6 17 19  0
##   8  0  0  0  1  2  0
```

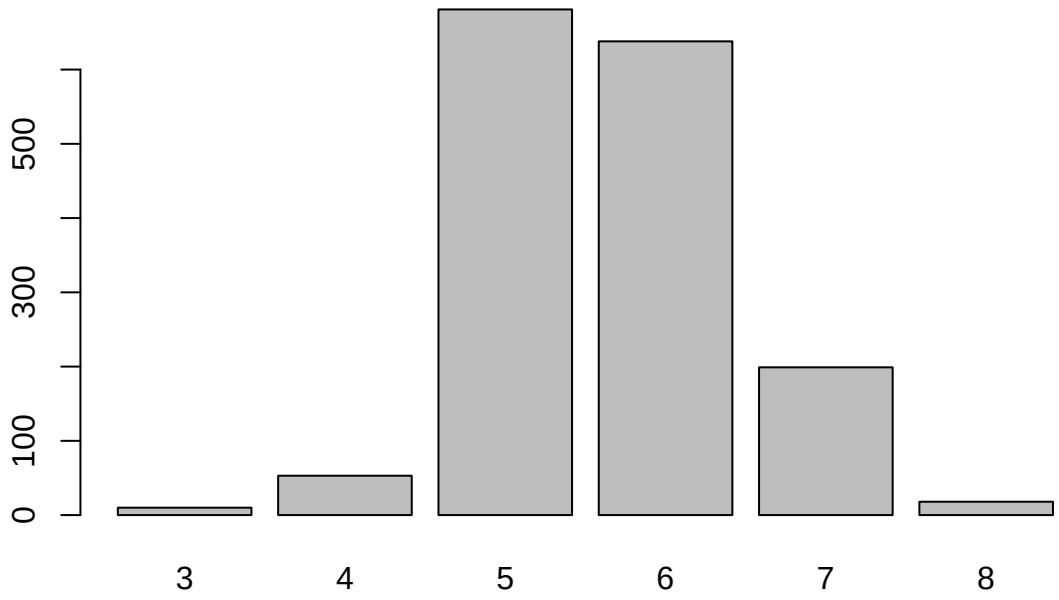
```
confusionMatrix(table(pred, wine_test$quality))
```

```
## Confusion Matrix and Statistics
##
##
## pred  3  4  5  6  7  8
##    3  0  0  0  0  0  0
##    4  0  0  0  0  0  0
##    5  3  9 90 48  6  0
##    6  1  2 34 73 17  1
##    7  0  0  1 14 19  2
##    8  0  0  0  0  0  0
##
## Overall Statistics
##
```

```
##                Accuracy : 0.5688
##                  95% CI : (0.5125, 0.6237)
##     No Information Rate : 0.4219
##     P-Value [Acc > NIR] : 8.847e-08
##
##                   Kappa : 0.3112
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity            0.0000  0.00000   0.7200   0.5407  0.45238 0.000000
## Specificity            1.0000  1.00000   0.6615   0.7027  0.93885 1.000000
## Pos Pred Value            NaN      NaN   0.5769   0.5703  0.52778      NaN
## Neg Pred Value         0.9875  0.96562   0.7866   0.6771  0.91901 0.990625
## Prevalence             0.0125  0.03438   0.3906   0.4219  0.13125 0.009375
## Detection Rate         0.0000  0.00000   0.2812   0.2281  0.05937 0.000000
## Detection Prevalence   0.0000  0.00000   0.4875   0.4000  0.11250 0.000000
## Balanced Accuracy      0.5000  0.50000   0.6908   0.6217  0.69561 0.500000
```

To help improve the power of the model we reduce the levels of classification from 6 to 3. Wines ranked at 7 and 8 become "good", 5 and 6 became "normal", and then 3 and 4 become "bad". After splitting keeping 80% for testing we can now test the remaining data, which shows 88% accuracy and is a lot better than the decision tree.

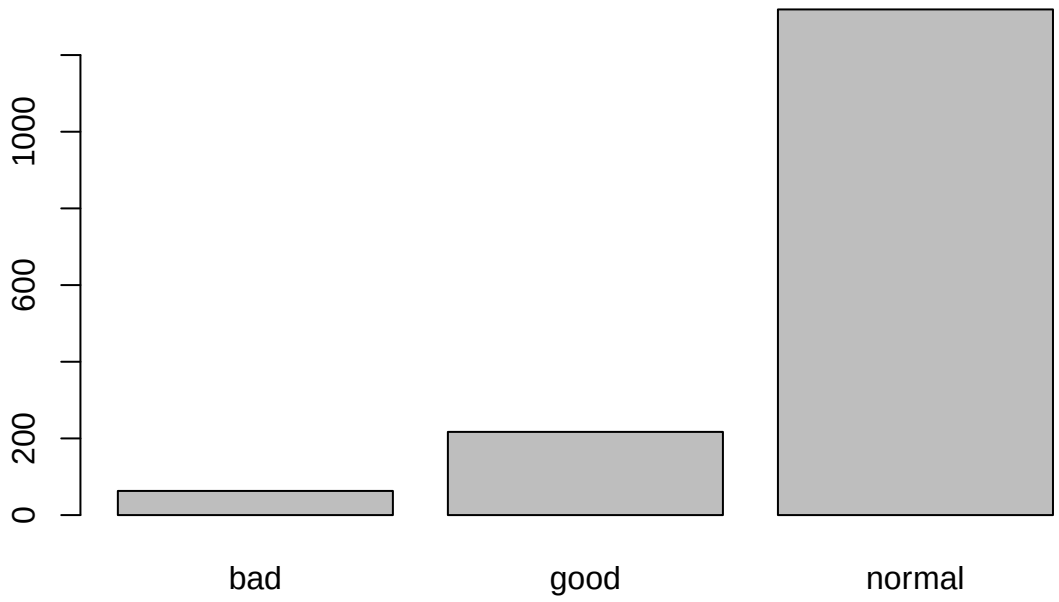
```
wine2 <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequali
set.seed(1)
barplot(table(wine2$quality))
```




```
wine2$taste <- ifelse(wine2$quality < 5, "bad", "good")
wine2$taste[wine2$quality == 5] <- "normal"
wine2$taste[wine2$quality == 6] <- "normal"
wine2$taste <- as.factor(wine2$taste)
str(wine2$taste)
```

```
##  Factor w/ 3 levels "bad","good","normal": 3 3 3 3 3 3 3 2 2 3 ...
```

```
barplot(table(wine2$taste))
```




```
table(wine2$taste)
```

```
##
##    bad   good normal
##     63    217   1319
```

```
samp <- sample(1599, 1279)
wine_train2 <- wine2[samp, ]
wine_test2 <- wine2[-samp, ]
dim(wine_train2)
```

```
## [1] 1279   13
```

```
dim(wine_test2)
```

```
## [1] 320  13
```

```
model <- randomForest(taste ~ . - quality, data = wine_train2)
model
```

```
##
## Call:
##  randomForest(formula = taste ~ . - quality, data = wine_train2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 12.28%
## Confusion matrix:
```

```
##        bad good normal class.error
## bad      0    1     47  1.00000000
## good     0   88     84  0.48837209
## normal   0   25   1034  0.02360718
```

```
prediction <- predict(model, newdata = wine_test2)
table(prediction, wine_test2$taste)
```

```
##
## prediction bad good normal
##     bad      0    0      0
##     good     0   25      5
##     normal  15   20    255
```

```
(0 + 25 + 255) / nrow(wine_test2)
```

```
## [1] 0.875
```