WINE QUALITY PREDICTION

Giwa Ibironke

5/14/2020

Introduction

Different kinds of wine has different taste and winequality. The winequality of a wine can be attributed to various factors. These factors will be analysed in this work. The aim of this work is to explore data, get insight and build a model for wine winequality prediction.

Exploratory data analysis

Having acquired all necessary data for this work, the next thing to do is some cleaning and exploratory analysis. we will be exploring all the determinants of wine winequality to find trends and possible correlations. From this section, we will get insight on the different features in the dataset. This will help in tuning and building our model for better performance.

The dataset

We'll set the working directory and load the dataset from the directory into a variable named winequality

```
#setwd('C:\\Users\\Ronke\\Downloads')
winequality <- read.csv("wine.csv")
```

This dataset contains 4898 records and 12 variables:

- winequality: This is our target variable.
- Volatile acidity
- Citric acid
- Residual sugar
- Chlorides
- Free Sulfur dioxides
- Total sulfur dioxide
- Density
- pH
- Sulphates
- Alcohol
- Fixed acidity

Data preparation and data cleaning

Next we find get the summary and internal structure of our dataset.

```
str(winequality)
```

```
## 'data.frame':    4898 obs. of  12 variables:
##  $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
##  $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
```

```
##  $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
##  $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
##  $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
##  $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
##  $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
##  $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ winequality         : int  6 6 6 6 6 6 6 6 6 6 ...
```

```
summary(winequality)
```

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
##  Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4        Mean   :0.9940
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0        Max.   :1.0390
##        pH          sulphates         alcohol       winequality
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :3.000
##  1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.180   Median :0.4700   Median :10.40   Median :6.000
##  Mean   :3.188   Mean   :0.4898   Mean   :10.51   Mean   :5.878
##  3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000
```

We take a look at the dataset and the different variables. This shows all variables are numeric. However, because I want to perform a classification model, I will change my target variable to factor.

```
winequality$winequality<-as.factor(winequality$winequality)
```

Now we will check for missing values.

```
anyNA(winequality)
```

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

```
#complete.cases(winequality)
```

Let us now have a look at the first rows of our dataset. This clearly shows us that there are no missing data as all output were FALSE. And complete cases are all TRUE.

```
head(winequality)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.0             0.27        0.36           20.7     0.045
## 2           6.3             0.30        0.34            1.6     0.049
## 3           8.1             0.28        0.40            6.9     0.050
## 4           7.2             0.23        0.32            8.5     0.058
## 5           7.2             0.23        0.32            8.5     0.058
```

```
## 6           8.1             0.28        0.40            6.9     0.050
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  45                  170  1.0010 3.00      0.45     8.8
## 2                  14                  132  0.9940 3.30      0.49     9.5
## 3                  30                   97  0.9951 3.26      0.44    10.1
## 4                  47                  186  0.9956 3.19      0.40     9.9
## 5                  47                  186  0.9956 3.19      0.40     9.9
## 6                  30                   97  0.9951 3.26      0.44    10.1
##   winequality
## 1           6
## 2           6
## 3           6
## 4           6
## 5           6
## 6           6
```

Data Exploration

In this section we will explore all the features in our dataset to get more insight. We will be plotting graphs using these features. We will be doing univariate and bivariate analysis as the case as required and we will be using graphs that show the information properly.\First we will to do a correlation plot. To do this we need just the numeric variables, so we will separate this from the entire dataset. We will remove the last column as this is the only column that is a factor in the dataset.

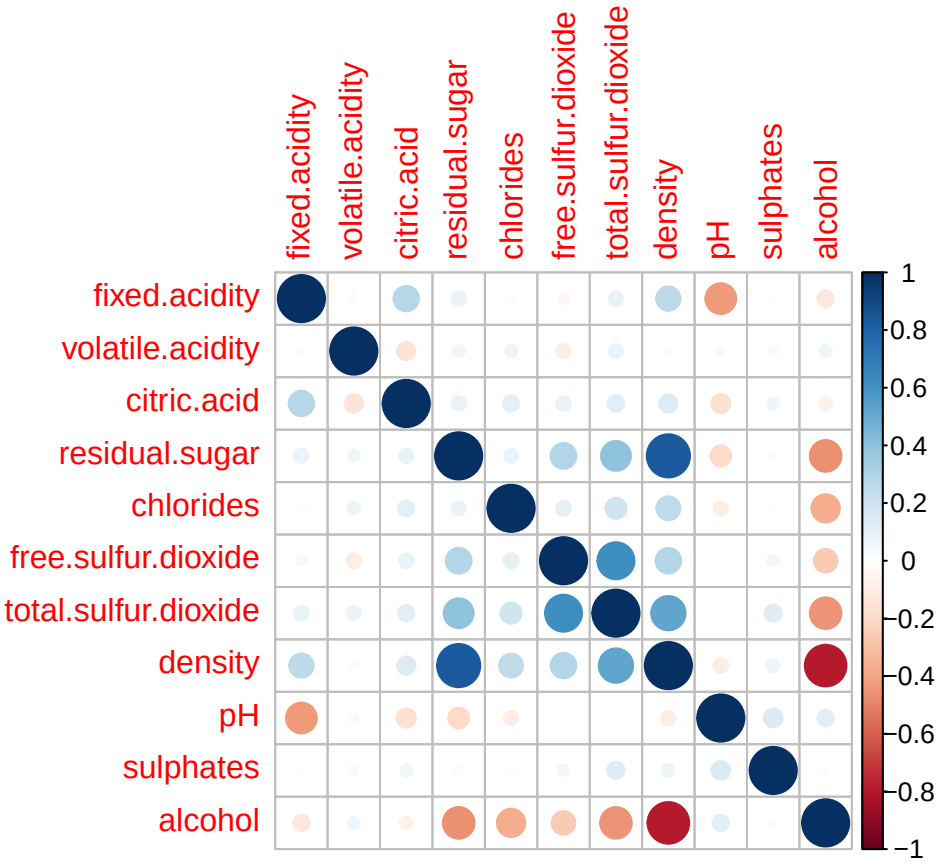
```
numcols<-winequality[,c('fixed.acidity','volatile.acidity','citric.acid','residual.sugar','chlorides','
```

Showing correlation between variables

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrplot(cor(numcols))
```



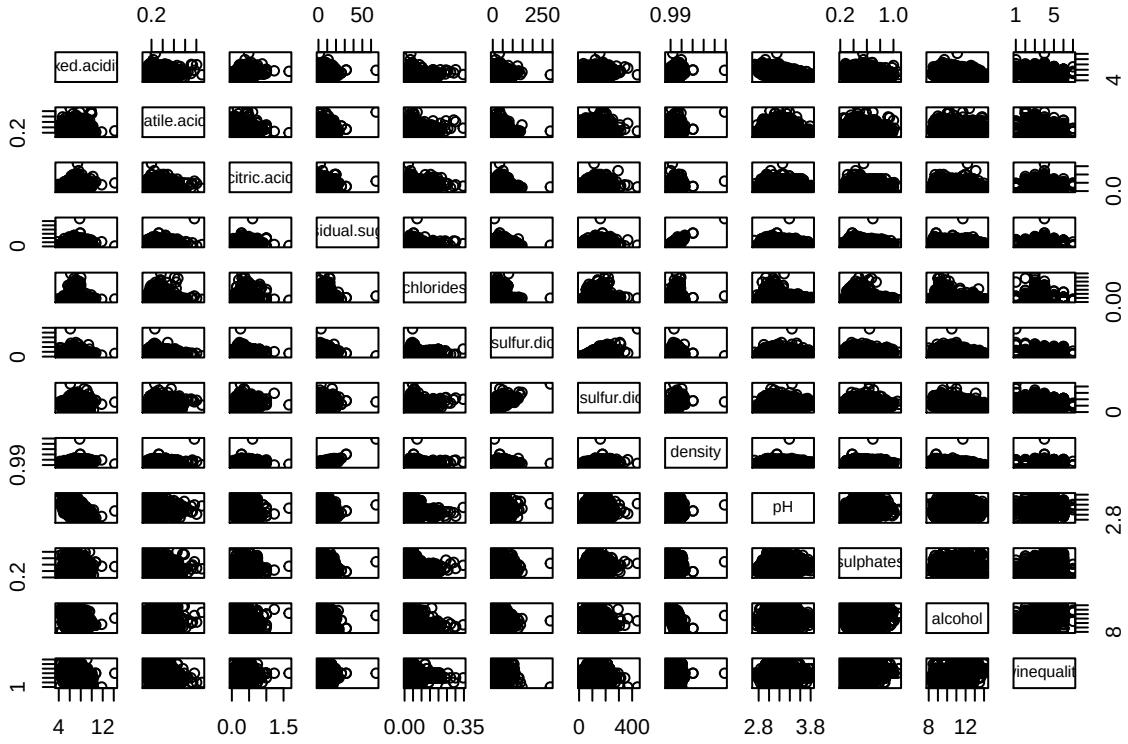
This shows the correlation of the variables and how they perform with the other variable. Now let us look at the figures behind the above plot.

```
cor(numcols)
```

```
##                       fixed.acidity volatile.acidity citric.acid residual.sugar
## fixed.acidity            1.00000000      -0.02269729  0.28918070     0.08902070
## volatile.acidity        -0.02269729       1.00000000 -0.14947181     0.06428606
## citric.acid              0.28918070      -0.14947181  1.00000000     0.09421162
## residual.sugar           0.08902070       0.06428606  0.09421162     1.00000000
## chlorides                0.02308564       0.07051157  0.11436445     0.08868454
## free.sulfur.dioxide     -0.04939586      -0.09701194  0.09407722     0.29909835
## total.sulfur.dioxide     0.09106976       0.08926050  0.12113080     0.40143931
## density                  0.26533101       0.02711385  0.14950257     0.83896645
## pH                      -0.42585829      -0.03191537 -0.16374821    -0.19413345
## sulphates               -0.01714299      -0.03572815  0.06233094    -0.02666437
## alcohol                 -0.12088112       0.06771794 -0.07572873    -0.45063122
##                        chlorides free.sulfur.dioxide total.sulfur.dioxide
## fixed.acidity         0.02308564       -0.0493958591          0.091069756
## volatile.acidity      0.07051157       -0.0970119393          0.089260504
## citric.acid           0.11436445        0.0940772210          0.121130798
## residual.sugar        0.08868454        0.2990983537          0.401439311
## chlorides             1.00000000        0.1013923521          0.198910300
## free.sulfur.dioxide   0.10139235        1.0000000000          0.615500965
## total.sulfur.dioxide  0.19891030        0.6155009650          1.000000000
## density               0.25721132        0.2942104109          0.529881324
## pH                   -0.09043946       -0.0006177961          0.002320972
```

```
## sulphates              0.01676288        0.0592172458          0.134562367
## alcohol               -0.36018871       -0.2501039415         -0.448892102
##                           density            pH   sulphates     alcohol
## fixed.acidity          0.26533101 -0.4258582910 -0.01714299 -0.12088112
## volatile.acidity       0.02711385 -0.0319153683 -0.03572815  0.06771794
## citric.acid            0.14950257 -0.1637482114  0.06233094 -0.07572873
## residual.sugar         0.83896645 -0.1941334540 -0.02666437 -0.45063122
## chlorides              0.25721132 -0.0904394560  0.01676288 -0.36018871
## free.sulfur.dioxide    0.29421041 -0.0006177961  0.05921725 -0.25010394
## total.sulfur.dioxide   0.52988132  0.0023209718  0.13456237 -0.44889210
## density                1.00000000 -0.0935914935  0.07449315 -0.78013762
## pH                    -0.09359149  1.0000000000  0.15595150  0.12143210
## sulphates              0.07449315  0.1559514973  1.00000000 -0.01743277
## alcohol               -0.78013762  0.1214320987 -0.01743277  1.00000000
```

```
pairs(winequality)
```

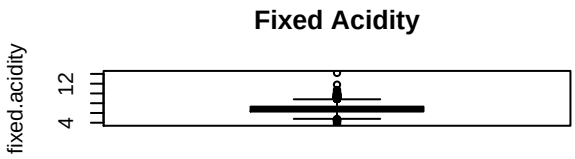


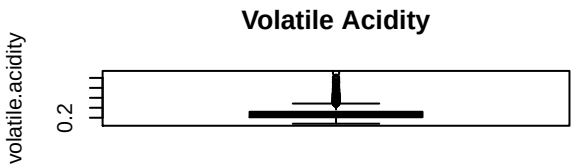
We see what the correlation of the variables and their relationship with the target variable.

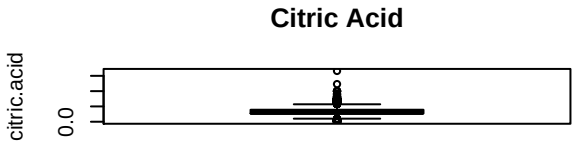
We will now plot graphs to show these relationships.

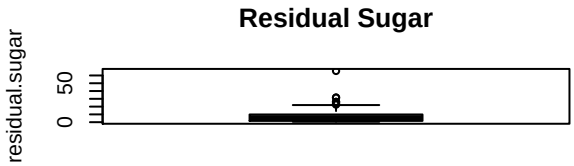
I will do a boxplot of my variables since they are numeric to see the distribution of data of each variable

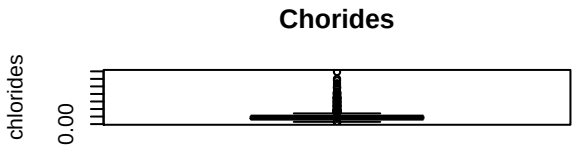
```
par(mfrow=(c(3,2)))
boxplot(winequality$fixed.acidity, ylab = "fixed.acidity", main = "Fixed Acidity")
boxplot(winequality$volatile.acidity, ylab = "volatile.acidity", main = "Volatile Acidity")
boxplot(winequality$citric.acid, ylab = "citric.acid", main = "Citric Acid")

boxplot(winequality$residual.sugar, ylab = "residual.sugar", main = "Residual Sugar")
boxplot(winequality$chlorides, ylab = "chlorides", main = "Chorides")
boxplot(winequality$free.sulfur.dioxide, ylab = "free.sulfur.dioxide", main = "Free Sulfur Dioxide")
```

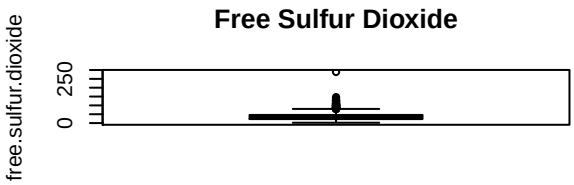




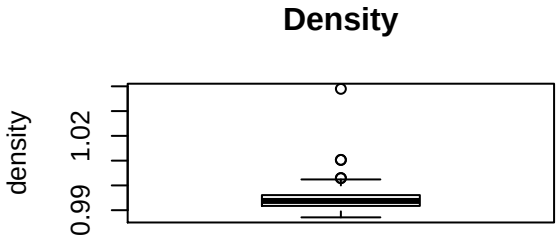


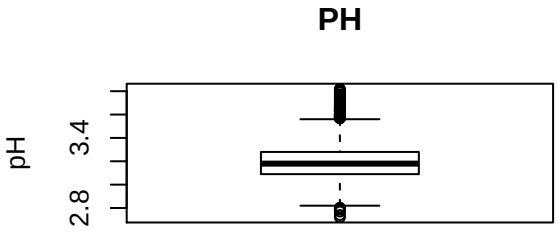


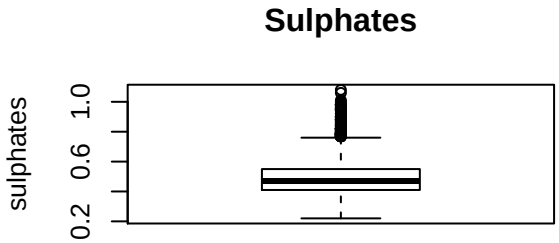


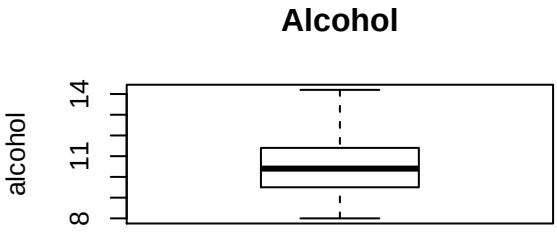



```
par(mfrow=(c(2,2)))
boxplot(winequality$density, ylab = "density", main = "Density")
boxplot(winequality$pH, ylab = "pH", main = "PH")
boxplot(winequality$sulphates, ylab = "sulphates", main = "Sulphates")
boxplot(winequality$alcohol, ylab = "alcohol", main = "Alcohol")
```



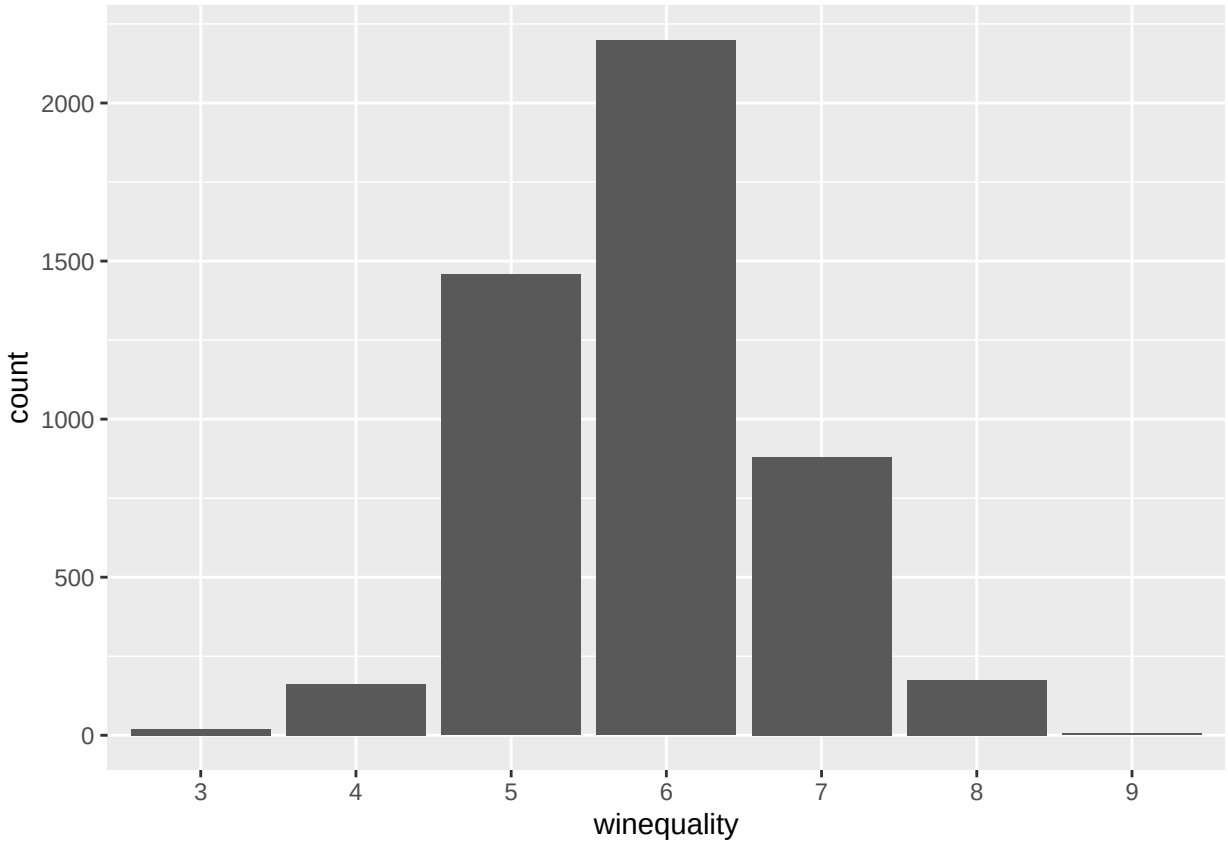




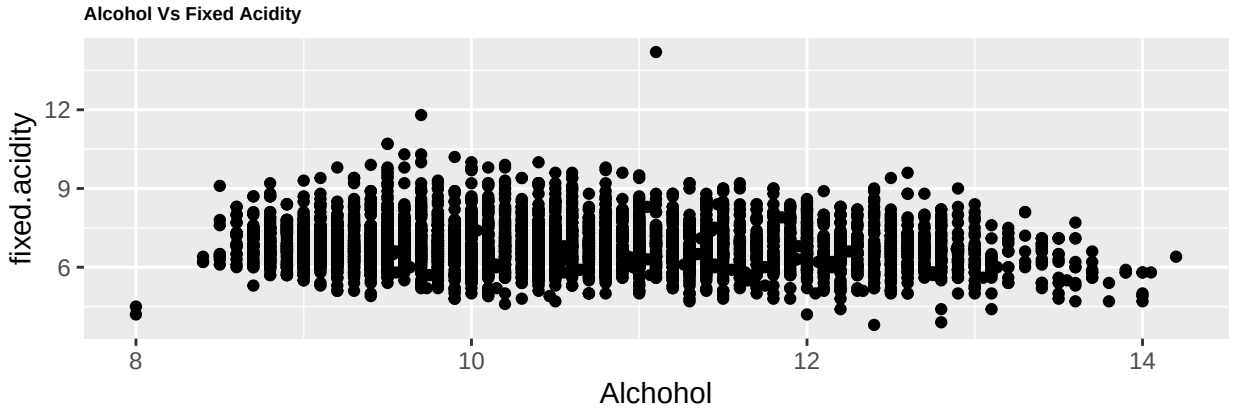


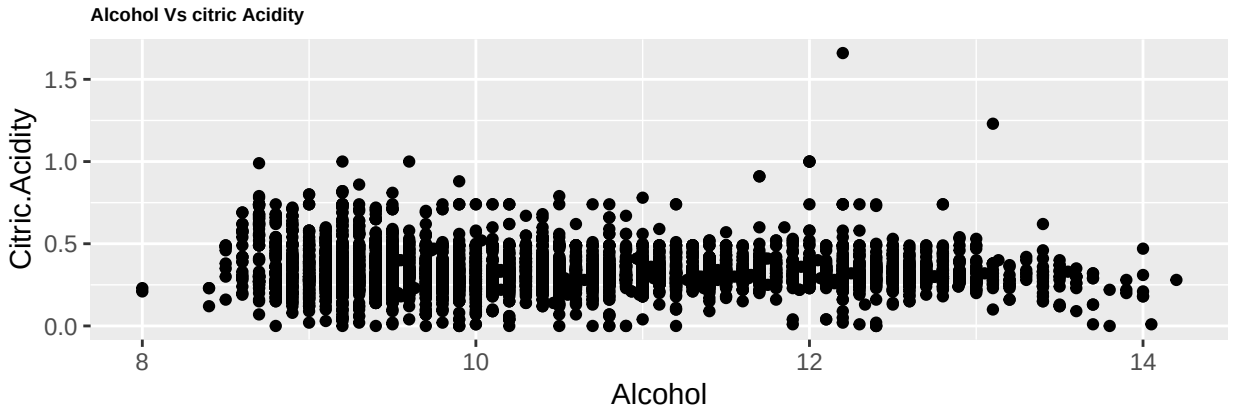
We will use a bar chart to show the response variable because it is categorical.

```
library(ggplot2)
ggplot(winequality,aes(winequality)) + geom_bar()
```

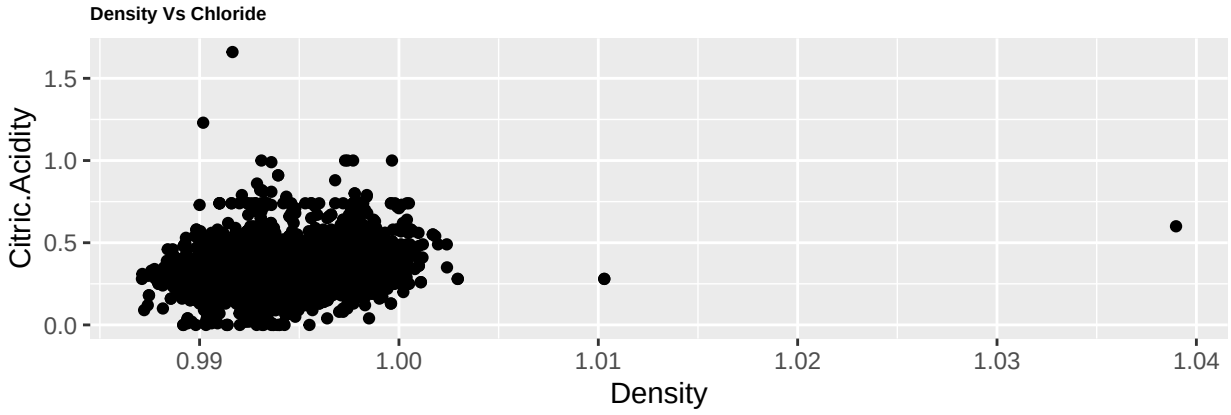


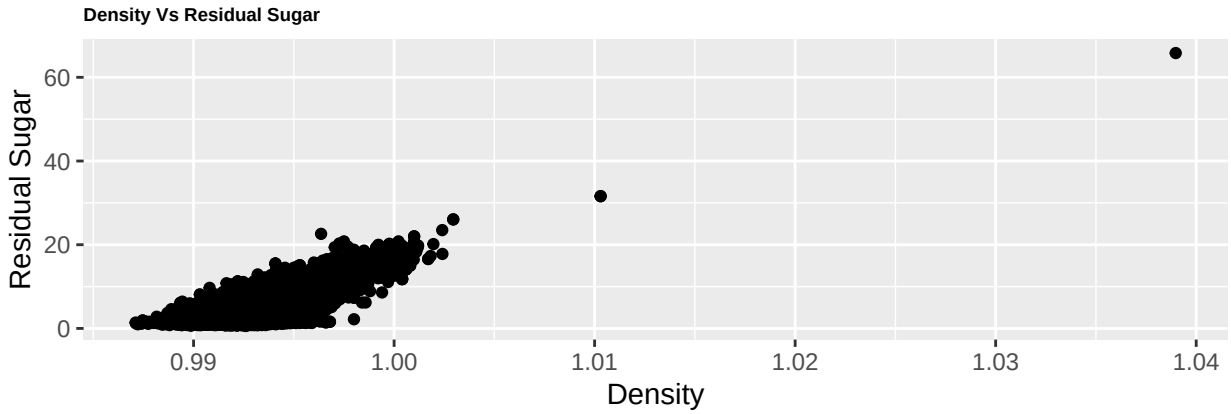

```
library(gridExtra)

g1<-ggplot(winequality,aes(x=alcohol,y=fixed.acidity)) + geom_point() +xlab('Alchohol') +ylab('fixed.ac

g2<-ggplot(winequality,aes(x=alcohol, y=citric.acid)) +geom_point() + xlab('Alcohol') + ylab("Citric.Ac
grid.arrange(g1,g2)
```






```
g3<-ggplot(winequality,aes(x=density, y=citric.acid)) +geom_point() + xlab('Density') + ylab("Citric.Ac

g4<-ggplot(winequality,aes(x=density, y=residual.sugar)) +geom_point() + xlab('Density') + ylab("Residua

grid.arrange(g3,g4)
```





From the above plots we see that there is no linear relationship between the variables. This however does not mean that they are not determinants of wine quality.

Random forest

I will be modelling a Random Forest classification algorithm using the caret library. I'll split the dataset into training and testing. 70% for train and the remaining 30% for test. I'll also show no of nodes in the tree

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(party)
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
## Loading required package: modeltools
## Loading required package: stats4
## Loading required package: strucchange
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
## Loading required package: sandwich
```

```
set.seed(123)


index <- sample(nrow(winequality), nrow(winequality)*0.7)
train.set <- winequality[index,]
test.set <- winequality[-index,]

RFmodel<- randomForest(winequality~., data=train.set)
RFmodel
```

```
##
## Call:
##  randomForest(formula = winequality ~ ., data = train.set)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 32.96%
## Confusion matrix:
##   3  4   5    6   7  8 9 class.error
## 3 0  0   3    8   0  0 0   1.0000000
## 4 0 27  59   38   1  0 0   0.7840000
## 5 0  3 676  318   8  0 0   0.3273632
## 6 0  2 220 1222  97  2 0   0.2080363
## 7 0  1  14  277 320  3 0   0.4796748
## 8 0  0   1   42  28 53 0   0.5725806
## 9 0  0   1    1   3  0 0   1.0000000
```

```
plot(RFmodel)
attributes(RFmodel)
```

```
## $names
##  [1] "call"            "type"            "predicted"       "err.rate"
##  [5] "confusion"       "votes"           "oob.times"       "classes"
##  [9] "importance"      "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"          "y"
## [17] "test"            "inbag"           "terms"
##
## $class
## [1] "randomForest.formula" "randomForest"
```

```
RFmodel$importance
```

```
##                       MeanDecreaseGini
## fixed.acidity                 173.4680
## volatile.acidity              229.6680
## citric.acid                   188.6046
## residual.sugar                205.3406
## chlorides                     198.9890
## free.sulfur.dioxide           219.4148
## total.sulfur.dioxide          213.9441
## density                       241.5263
## pH                            196.4370
## sulphates                     187.5436
## alcohol                       263.9027
```

```
p1 <- predict(RFmodel, test.set)

head(p1)
```

```
##  6  8  9 14 16 22
##  6  6  6  7  6  6
## Levels: 3 4 5 6 7 8 9
```

```
head(test.set$winequality)
```

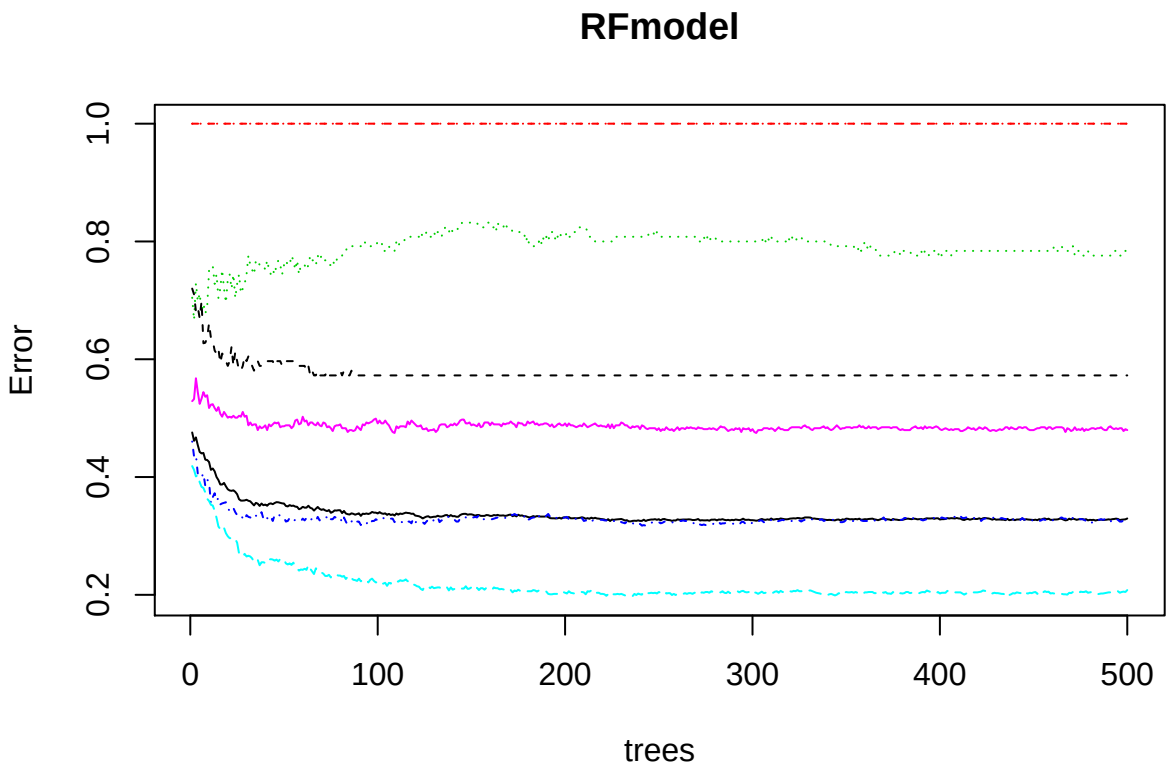
```
## [1] 6 6 6 7 7 7
## Levels: 3 4 5 6 7 8 9
```

```
confusionMatrix(p1, test.set$winequality)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   0   0   0   0   0   0
##          4   0   7   2   0   0   0   0
##          5   6  18 304  89   4   1   0
##          6   3  12 143 527 130  21   0
##          7   0   1   3  39 131  13   0
##          8   0   0   0   0   0  16   0
##          9   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6701
##                  95% CI : (0.6454, 0.6941)
##     No Information Rate : 0.4456
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4803
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
```

```
## Sensitivity          0.000000 0.184211   0.6726   0.8046  0.49434  0.31373
## Specificity          1.000000 0.998603   0.8841   0.6209  0.95353  1.00000
## Pos Pred Value            NaN 0.777778   0.7204   0.6304  0.70053  1.00000
## Neg Pred Value       0.993878 0.978782   0.8588   0.7981  0.89556  0.97593
## Prevalence           0.006122 0.025850   0.3075   0.4456  0.18027  0.03469
## Detection Rate       0.000000 0.004762   0.2068   0.3585  0.08912  0.01088
## Detection Prevalence 0.000000 0.006122   0.2871   0.5687  0.12721  0.01088
## Balanced Accuracy    0.500000 0.591407   0.7783   0.7127  0.72393  0.65686
##                      Class: 9
## Sensitivity                NA
## Specificity                 1
## Pos Pred Value             NA
## Neg Pred Value             NA
## Prevalence                  0
## Detection Rate              0
## Detection Prevalence        0
## Balanced Accuracy          NA
```

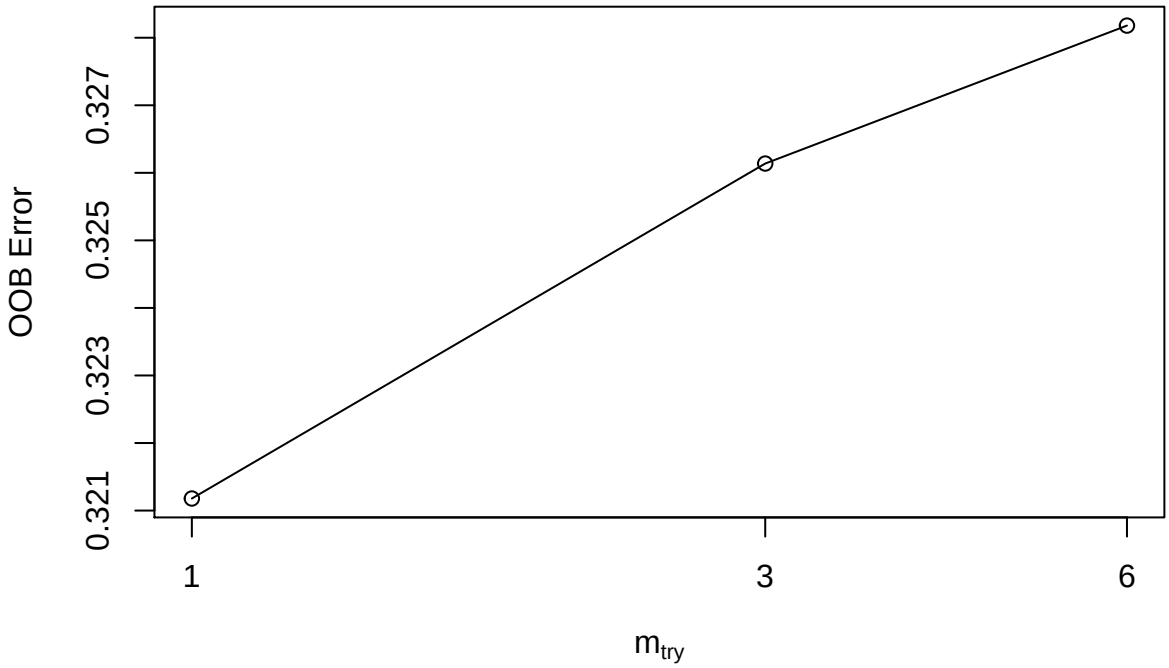
```
plot(RFmodel)
```




```
check<-tuneRF(train.set[, -12], train.set[, 12],
              stepFactor = 0.5,
              plot = TRUE,
              ntreeTry = 450,
              trace = TRUE,
              improve = 0.05)
```

```
## mtry = 3  OOB error = 32.61%
## Searching left ...
## mtry = 6     OOB error = 32.82%
## -0.006261181 0.05
```

```
## Searching right ...
## mtry = 1     OOB error = 32.12%
## 0.01520572 0.05
```




```
RFmodel<- randomForest(winequality~., data=train.set, mtry = 1,ntree=450, importance = TRUE)
RFmodel
```

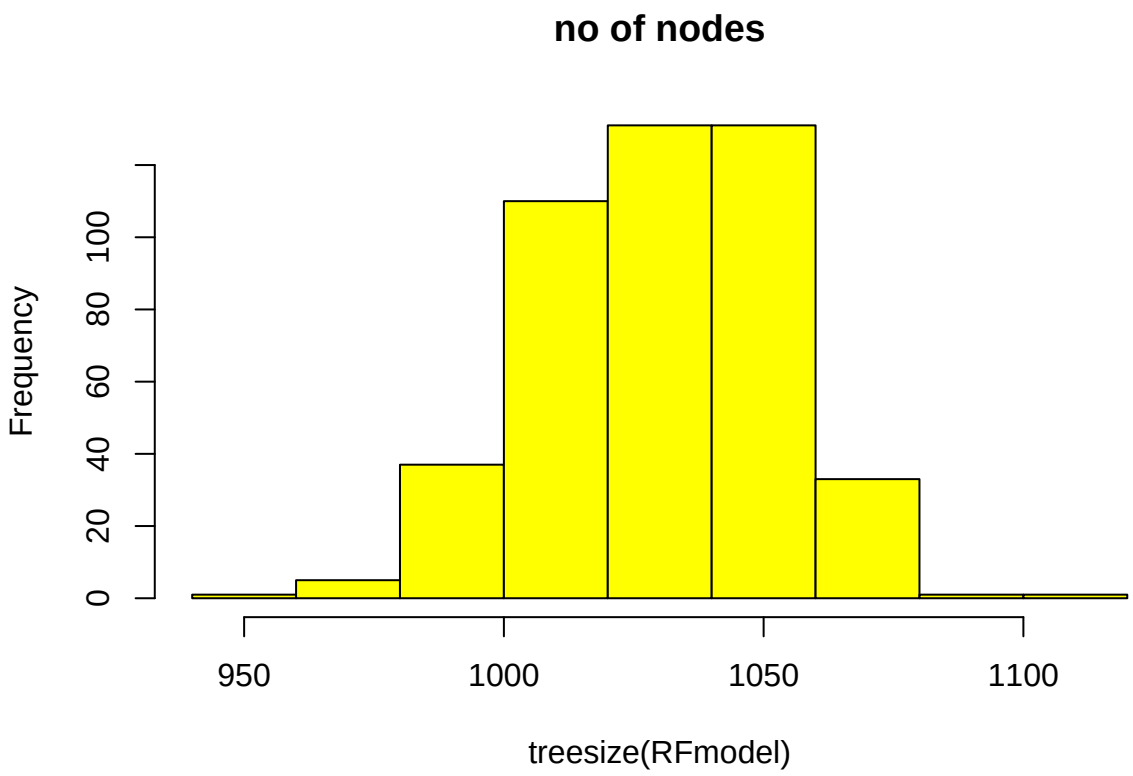
```
##
## Call:
##  randomForest(formula = winequality ~ ., data = train.set, mtry = 1,      ntree = 450, importance = 
##                Type of random forest: classification
##                      Number of trees: 450
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 32.79%
## Confusion matrix:
##   3  4   5    6   7  8 9 class.error
## 3 0  0   2    9   0  0 0   1.0000000
## 4 0 18  64   43   0  0 0   0.8560000
## 5 0  4 660  338   3  0 0   0.3432836
## 6 0  2 196 1267  78  0 0   0.1788723
## 7 0  0  10  296 306  3 0   0.5024390
## 8 0  0   1   44  26 53 0   0.5725806
## 9 0  0   0    2   3  0 0   1.0000000
```

```
p3 <- predict(RFmodel, test.set)
cm <- confusionMatrix(p3, test.set$winequality)
cm
```

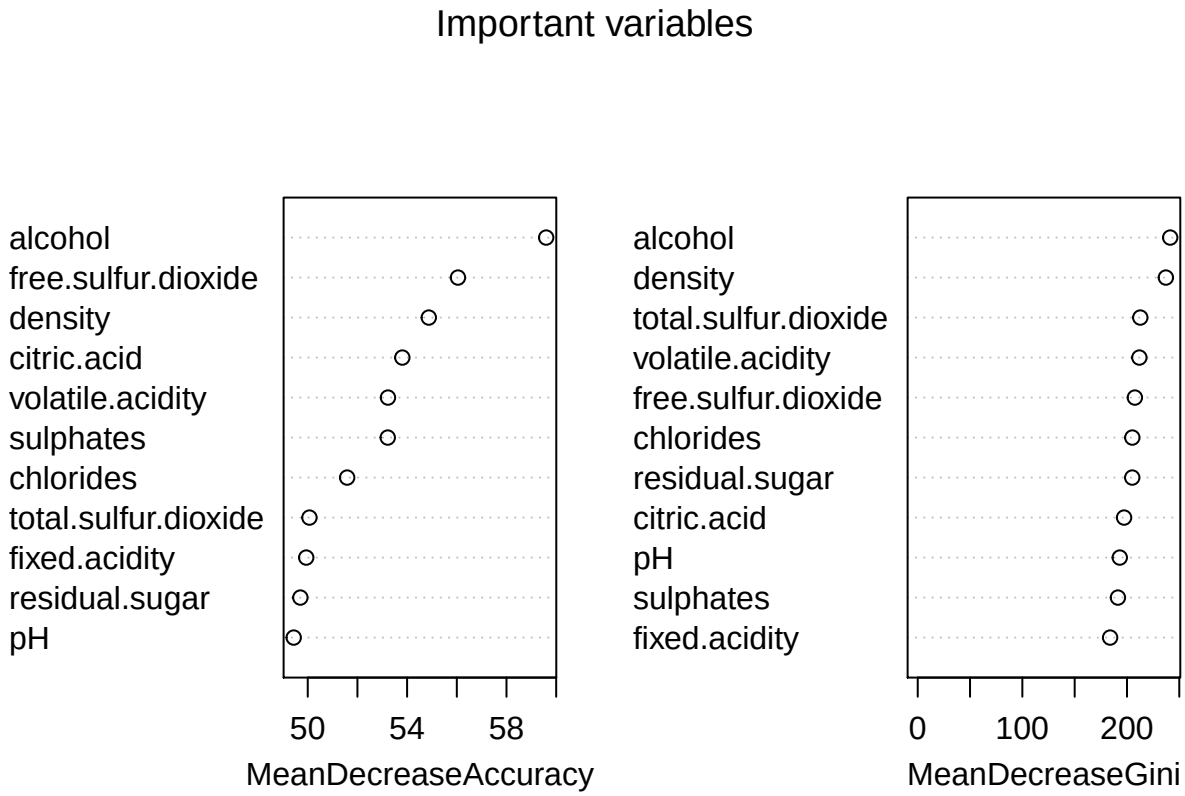
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   0   0   0   0   0   0
```

```
##          4   0   7   1   0   0   0   0
##          5   5  19 305  79   2   0   0
##          6   4  11 144 543 139  22   0
##          7   0   1   2  33 124  13   0
##          8   0   0   0   0   0  16   0
##          9   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6769
##                  95% CI : (0.6523, 0.7007)
##     No Information Rate : 0.4456
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4878
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000 0.184211   0.6748   0.8290  0.46792  0.31373
## Specificity          1.000000 0.999302   0.8969   0.6074  0.95934  1.00000
## Pos Pred Value            NaN 0.875000   0.7439   0.6292  0.71676  1.00000
## Neg Pred Value       0.993878 0.978796   0.8613   0.8155  0.89129  0.97593
## Prevalence           0.006122 0.025850   0.3075   0.4456  0.18027  0.03469
## Detection Rate       0.000000 0.004762   0.2075   0.3694  0.08435  0.01088
## Detection Prevalence 0.000000 0.005442   0.2789   0.5871  0.11769  0.01088
## Balanced Accuracy    0.500000 0.591756   0.7858   0.7182  0.71363  0.65686
##                      Class: 9
## Sensitivity                NA
## Specificity                 1
## Pos Pred Value             NA
## Neg Pred Value             NA
## Prevalence                  0
## Detection Rate              0
## Detection Prevalence        0
## Balanced Accuracy          NA
```

```
hist(treesize(RFmodel),
     main = "no of nodes",
     col = "yellow")
```




```
varImpPlot(RFmodel, main = "Important variables", sort = T)
```




```
importance(RFmodel)
```

```
##                               3         4         5         6         7         8
```

```
## fixed.acidity        -1.01443875 15.31155 36.87411 35.61326 35.31009 32.41229
## volatile.acidity      0.78552508 22.37290 41.67154 41.67475 37.02501 35.08865
## citric.acid          -0.35628549 19.48146 37.43467 37.47747 38.10254 33.49770
## residual.sugar        1.24397425 14.31061 34.99165 36.24251 35.90089 30.67317
## chlorides             0.90571455 14.93023 41.06008 31.70917 42.92747 34.33973
## free.sulfur.dioxide   3.72117662 25.68128 40.61367 39.34138 35.21007 30.74350
## total.sulfur.dioxide  2.56993216 18.50102 37.73728 30.14341 37.69019 31.64633
## density               0.06114839 12.50728 37.46808 33.56866 41.91316 34.17815
## pH                    1.72143355 13.60044 35.98587 33.21513 38.58560 31.77549
## sulphates             2.53383439 14.57126 34.77960 37.93103 39.19782 32.68775
## alcohol              -1.20090422 14.75375 48.77098 31.56607 45.15739 42.80924
##                                9 MeanDecreaseAccuracy MeanDecreaseGini
## fixed.acidity          1.4173668             49.93911         183.8837
## volatile.acidity       1.0011130             53.22918         211.8339
## citric.acid            0.2773738             53.80763         197.2347
## residual.sugar         2.2485951             49.70305         205.0443
## chlorides              2.0089486             51.58618         205.1048
## free.sulfur.dioxide   -1.3465687             56.04674         207.5305
## total.sulfur.dioxide   0.0000000             50.06869         212.7326
## density                2.2485951             54.87267         237.1655
## pH                     1.0011130             49.43491         193.1180
## sulphates             -0.2626330             53.22062         191.3398
## alcohol                1.8643010             59.59754         241.3094
```

The mtry is the no of variables it will use for each tree. For a classification model, the default is sq.root(p) where p is the number of variables. So therefore mtry is 1. It will also use the default ntree which is 500.The important variables in the model was showmn. we pridicted using the test data and saved in it in a variable called p1. We evaluated the model usimg confusion matrix. With this, some misclssifaction in classes were known and also the conifidence level. Every correct classification in each classes was also given. Sensitivity ans specificy was a. We plotted a graph which showed us the no. of the mtry when OOB error was at the lowest. And plotted the model which was able to show us that the model does not improve further after the no. of tree is 450. So we refitted the model using mtry value when oob is at its lowest, and changed no of trees to 450.We also showed the distribution of no. of nodes in the 450 trees that we used,there are about 90 trees that contain 790-810 nodes in them. And others distributed between 740-860.THe MeanDecreaseAccuracy graph test how worse the model will perform without each variable. Free.sulphur.dioxide and alcohol because of their high values have maximum importance in terms of contributing to the accuracy.While MeanDecreaseGini graph measures how pure the nodes are at the end of the tree without each variable.If free.sulfur.dioxide & residual.sugar is remonved, the MDG will reduce to about 28. We Checked the important variables and sort them and finally we used varUsed to shows the preditor variables used in the Random forest. It can be seen from the above, the variable that occured in the least time is 'citric.acid' and that because it has the least importnat value, the one with the higest occurence is the 'residual.sugar' and that's because it is the most important variable in the graph. Accuracy of this model is 67%. The Pos ped value and Neg pred value showing NAs in level 3 and level 9 shows the data is imbalance.

Support Vector Machine

In this subsection, we will be tuning and testing our model with support vector machine algorithmWe'll make a table to show the predicted and actual value and calculate the missclassification error.

Tuning for SVM with radial kernel

```
library(ggplot2)
library (e1071)

smodel <- svm(winequality~., data = winequality)
summary (smodel)
```

```
##
## Call:
## svm(formula = winequality ~ ., data = winequality)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  4344
##
##  ( 1883 1249 849 175 163 20 5 )
##
##
## Number of Classes:  7
##
## Levels:
##  3 4 5 6 7 8 9
```

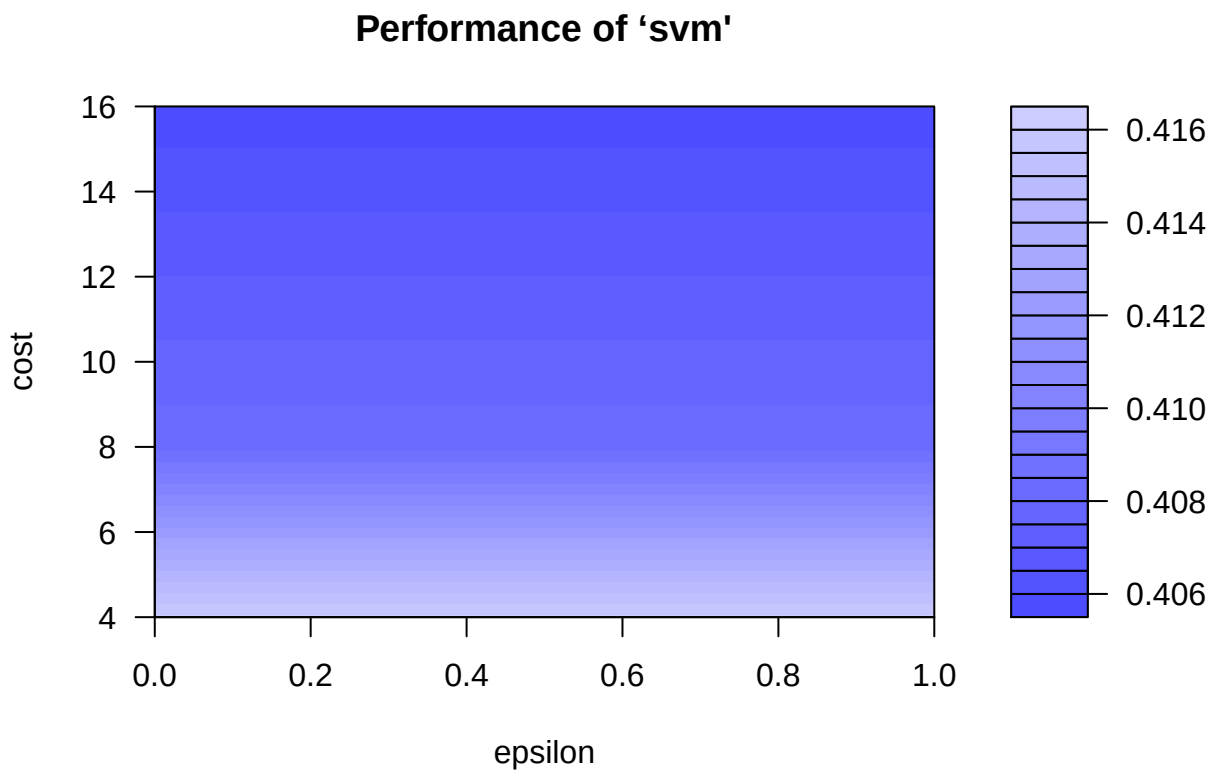
```
p1 <- predict(smodel, winequality)

tab <- table(Predicted = p1, Acual = winequality$winequality)
tab
```

```
##          Acual
## Predicted    3    4    5    6    7    8    9
##         3    1    0    0    0    0    0    0
##         4    0   20    1    0    0    0    0
##         5    7   96  930  333   21    1    0
##         6   12   45  522 1793  584  119    3
##         7    0    2    4   72  275   55    2
##         8    0    0    0    0    0    0    0
##         9    0    0    0    0    0    0    0
```

```
1-sum(diag(tab))/sum(tab)
```

```
## [1] 0.383626
```

```
#tuning the model
set.seed(123)
nsmodel <- tune(svm, winequality~., data = winequality,
     ranges=list(epsilon = seq(0,1,0.1), cost = 2^(2:4)))
plot(nsmodel)
```




```
summary(nsmodel)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  epsilon cost
##        0   16
##
## - best performance: 0.4056734
##
## - Detailed performance results:
##    epsilon cost     error dispersion
## 1      0.0    4 0.4160832 0.02770908
## 2      0.1    4 0.4160832 0.02770908
## 3      0.2    4 0.4160832 0.02770908
## 4      0.3    4 0.4160832 0.02770908
## 5      0.4    4 0.4160832 0.02770908
## 6      0.5    4 0.4160832 0.02770908
## 7      0.6    4 0.4160832 0.02770908
## 8      0.7    4 0.4160832 0.02770908
## 9      0.8    4 0.4160832 0.02770908
## 10     0.9    4 0.4160832 0.02770908
## 11     1.0    4 0.4160832 0.02770908
## 12     0.0    8 0.4083269 0.02596750
## 13     0.1    8 0.4083269 0.02596750
## 14     0.2    8 0.4083269 0.02596750
## 15     0.3    8 0.4083269 0.02596750
```

```
## 16     0.4    8 0.4083269 0.02596750
## 17     0.5    8 0.4083269 0.02596750
## 18     0.6    8 0.4083269 0.02596750
## 19     0.7    8 0.4083269 0.02596750
## 20     0.8    8 0.4083269 0.02596750
## 21     0.9    8 0.4083269 0.02596750
## 22     1.0    8 0.4083269 0.02596750
## 23     0.0   16 0.4056734 0.02169375
## 24     0.1   16 0.4056734 0.02169375
## 25     0.2   16 0.4056734 0.02169375
## 26     0.3   16 0.4056734 0.02169375
## 27     0.4   16 0.4056734 0.02169375
## 28     0.5   16 0.4056734 0.02169375
## 29     0.6   16 0.4056734 0.02169375
## 30     0.7   16 0.4056734 0.02169375
## 31     0.8   16 0.4056734 0.02169375
## 32     0.9   16 0.4056734 0.02169375
## 33     1.0   16 0.4056734 0.02169375
```

```
fsmodel <- nsmodel$best.model
summary (fsmodel)
```

```
##
## Call:
## best.tune(method = svm, train.x = winequality ~ ., data = winequality,
##     ranges = list(epsilon = seq(0, 1, 0.1), cost = 2^(2:4)))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  16
##
## Number of Support Vectors:  4128
##
##  ( 1808 1166 800 170 159 20 5 )
##
##
## Number of Classes:  7
##
## Levels:
##  3 4 5 6 7 8 9
```

```
p2 <- predict(fsmodel, winequality)

tab <- table(Predicted = p2, Actual = winequality$winequality)
tab
```

```
##           Actual
## Predicted    3    4    5    6    7    8    9
##         3   16    0    0    0    0    0    0
##         4    0   93    3    2    0    0    0
##         5    1   45 1099  230   10    1    0
##         6    3   25  342 1892  367   56    1
##         7    0    0   13   73  500   44    0
##         8    0    0    0    1    3   74    0
```

```
##         9    0    0    0    0    0    0    4
```

```
1-sum(diag(tab))/sum(tab)
```

```
## [1] 0.2490813
```

The parameters in the above model are costs and epsilon. cost captures the cost of constraint variation. If cost is too high, the model will be overfitted and underfitted if too low.Hence we shoud use a large range to cpature optimal cost value. Epsilon makes use of a sequeuce that starts with 0-1 with 0.1 increment. Both parametrs are what makes up combinations. Better results can be seen in the darker regions and it has lower misclassification error. The model is best when cost is 16 along epsilon 0-1. We predict with the best model and check for accuracy with the best model. The accuracy for this model is 75% The number of suport vectors was 4128. The sampling method was 10-fold cross n=validation.

Ordinal Logistic Regression

```
library(MASS)

set.seed(123)

logregmodel<-polr(winequality ~., data = train.set, method='logistic', Hess = TRUE)
summary(logregmodel)
```

```
## Call:
## polr(formula = winequality ~ ., data = train.set, Hess = TRUE,
##     method = "logistic")
##
## Coefficients:
##                          Value Std. Error    t value
## fixed.acidity         2.620e-02   0.045500    0.57584
## volatile.acidity     -5.376e+00   0.369242  -14.55932
## citric.acid          -2.875e-02   0.285919   -0.10056
## residual.sugar        1.397e-01   0.007977   17.51039
## chlorides            -1.572e-01   1.589308   -0.09893
## free.sulfur.dioxide   9.204e-03   0.002651    3.47174
## total.sulfur.dioxide -1.144e-03   0.001141   -1.00226
## density              -1.947e+02   0.547786 -355.35742
## pH                    9.479e-01   0.251022    3.77598
## sulphates             1.517e+00   0.289854    5.23321
## alcohol               7.383e-01   0.037216   19.83887
##
## Intercepts:
##     Value     Std. Error t value
## 3|4 -188.7702    0.5578  -338.4471
## 4|5 -186.0884    0.5551  -335.2605
## 5|6 -183.1405    0.5596  -327.2447
## 6|7 -180.5786    0.5696  -317.0098
## 7|8 -178.3680    0.5798  -307.6195
## 8|9 -175.0446    0.7260  -241.1103
##
## Residual Deviance: 7707.526
## AIC: 7741.526
```

```
predictedClass <- predict(logregmodel, test.set)
head(predictedClass)
```

```
## [1] 6 6 5 7 6 6
## Levels: 3 4 5 6 7 8 9
```

```
table(test.set$winequality, predictedClass)
```

```
##    predictedClass
##       3   4   5   6   7   8   9
##   3   0   0   6   3   0   0   0
##   4   0   0  24  13   1   0   0
##   5   0   1 210 238   3   0   0
##   6   0   0 108 511  36   0   0
##   7   0   0   6 213  46   0   0
##   8   0   0   0  32  19   0   0
##   9   0   0   0   0   0   0   0
```

```
mean(as.character(test.set$winequality) != as.character(predictedClass))
```

```
## [1] 0.4782313
```

Here we fit the model using the because our target variables are ordinal values, We fit using the train set and we test with the test set. We also tried to find the difference between predicted values and the original values in the test set. The accuracy here was 52%

Table 1: Accuracy of Models

Model	Accuracy	Kappa
SVM with Radial kernel	0.7	0.4
Random forest	0.6	0.4
Ordinal Logistic Regression	0.5	–

Evalution

The models above have show high level of accuarcy which each one showing For Rf, sensitivity is at it best in level 6, while specificty is as its best for level 3,8 & 9. The SVM model is at its best accuracy when the number of support vectors is 4128 and sharing like this 1808 1166 800 170 159 20 5 among the 7 classes. And the cost is i6. The kernel type is radial which works best for classification algorithm.The ordinal regression has the least of the accuracy. However across all the models, it is evident that level 6 & 7 are more accurately predicted and not much misclassification is seen in them. I didn't do cross validation because my dataset is large,instead i splitt into training and testing set.

Conclusion

We have seen that quality of wines in level 5, 6 & 7 are predicted more accurately than we have in other levels. This shows us that our dataset is imbalanced with the higher percentage falling into the level 5, 6, & 7 categories. Due to this, the model learns more from the data in this category which explains it accuracy. Hence I'll conclude the models fitted would perform well in being used on wines in these levels. SVM is the best model here going by the accuracy, and OLR being the least. This is not to say Ordinal Regression isn't fit, as a matter of fact, it classified classes in level 6 perfectly well. There's a chance to improve the model more than we already did by removing the variables that are not contributing to the model, and doing hyper parameter optimization even further. Although all features were used to fit the model, the most important features for this analysis are: pH, Volatile Acidity, Free Sulphur Dioxide, pH and sulphates.