ADS 503 - Team 7

Summer Purschke, Jacqueline Urenda, Oscar Gil

06/12/2022

```
# R Libraries
library(caret)
library(AppliedPredictiveModeling)
library(Hmisc)
library(dplyr)
library(tidyverse)
library(ggplot2)
library(corrplot)
library(MASS)
library(ISLR)
library(rpart)
library(partykit)
library(randomForestSRC)
library(earth)
library(MARSS)
library(e1071)
library(summarytools)
library(grid)
```

Load the Red Wine Quality data set from GitHub - data set copied from Kaggle and imported into GitHub.

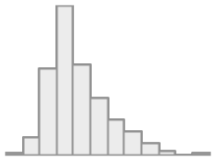
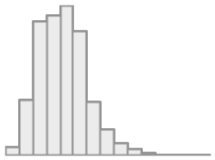
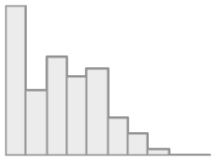
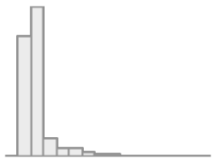
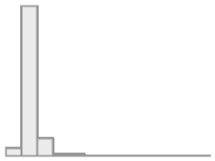
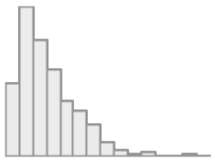
```
wine <- read.csv(
  url("https://raw.githubusercontent.com/OscarG-DataSci/ADS503/main/winequality-red.csv")
                      , header = TRUE)
```

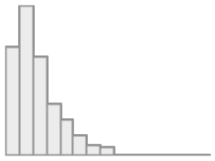
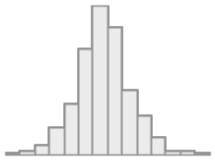
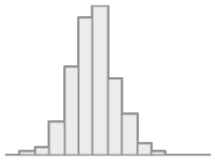
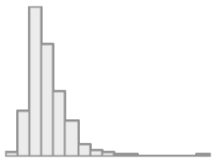
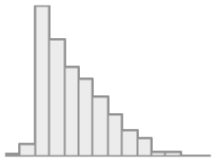
Data Summary

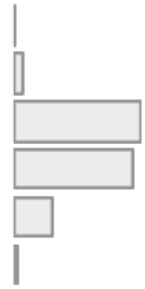
```
dfSummary(wine,
          plain.ascii  = FALSE,
          style        = "grid",
          graph.magnif = 0.75,
          valid.col    = FALSE,
          tmp.img.dir  = "/tmp")
```

Data Frame Summary

wine **Dimensions:** 1599 x 12
Duplicates: 240

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Missing
1	fixed.acidity [numeric]	Mean (sd) : 8.3 (1.7) min < med < max: 4.6 < 7.9 < 15.9 IQR (CV) : 2.1 (0.2)	96 distinct values		0 (0.0%)
2	volatile.acidity [numeric]	Mean (sd) : 0.5 (0.2) min < med < max: 0.1 < 0.5 < 1.6 IQR (CV) : 0.2 (0.3)	143 distinct values		0 (0.0%)
3	citric.acid [numeric]	Mean (sd) : 0.3 (0.2) min < med < max: 0 < 0.3 < 1 IQR (CV) : 0.3 (0.7)	80 distinct values		0 (0.0%)
4	residual.sugar [numeric]	Mean (sd) : 2.5 (1.4) min < med < max: 0.9 < 2.2 < 15.5 IQR (CV) : 0.7 (0.6)	91 distinct values		0 (0.0%)
5	chlorides [numeric]	Mean (sd) : 0.1 (0) min < med < max: 0 < 0.1 < 0.6 IQR (CV) : 0 (0.5)	153 distinct values		0 (0.0%)
6	free.sulfur.dioxide [numeric]	Mean (sd) : 15.9 (10.5) min < med < max: 1 < 14 < 72 IQR (CV) : 14 (0.7)	60 distinct values		0 (0.0%)

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Missing
7	total.sulfur.dioxide [numeric]	Mean (sd) : 46.5 (32.9) min < med < max: 6 < 38 < 289 IQR (CV) : 40 (0.7)	144 distinct values		0 (0.0%)
8	density [numeric]	Mean (sd) : 1 (0) min < med < max: 1 < 1 < 1 IQR (CV) : 0 (0)	436 distinct values		0 (0.0%)
9	pH [numeric]	Mean (sd) : 3.3 (0.2) min < med < max: 2.7 < 3.3 < 4 IQR (CV) : 0.2 (0)	89 distinct values		0 (0.0%)
10	sulphates [numeric]	Mean (sd) : 0.7 (0.2) min < med < max: 0.3 < 0.6 < 2 IQR (CV) : 0.2 (0.3)	96 distinct values		0 (0.0%)
11	alcohol [numeric]	Mean (sd) : 10.4 (1.1) min < med < max: 8.4 < 10.2 < 14.9 IQR (CV) : 1.6 (0.1)	65 distinct values		0 (0.0%)

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Missing
12	quality [integer]	Mean (sd) : 5.6 (0.8) min < med < max: 3 < 6 < 8 IQR (CV) : 1 (0.1)	3 : 10 (0.6%) 4 : 53 (3.3%) 5 : 681 (42.6%) 6 : 638 (39.9%) 7 : 199 (12.4%) 8 : 18 (1.1%)		0 (0.0%)

Pre-processing

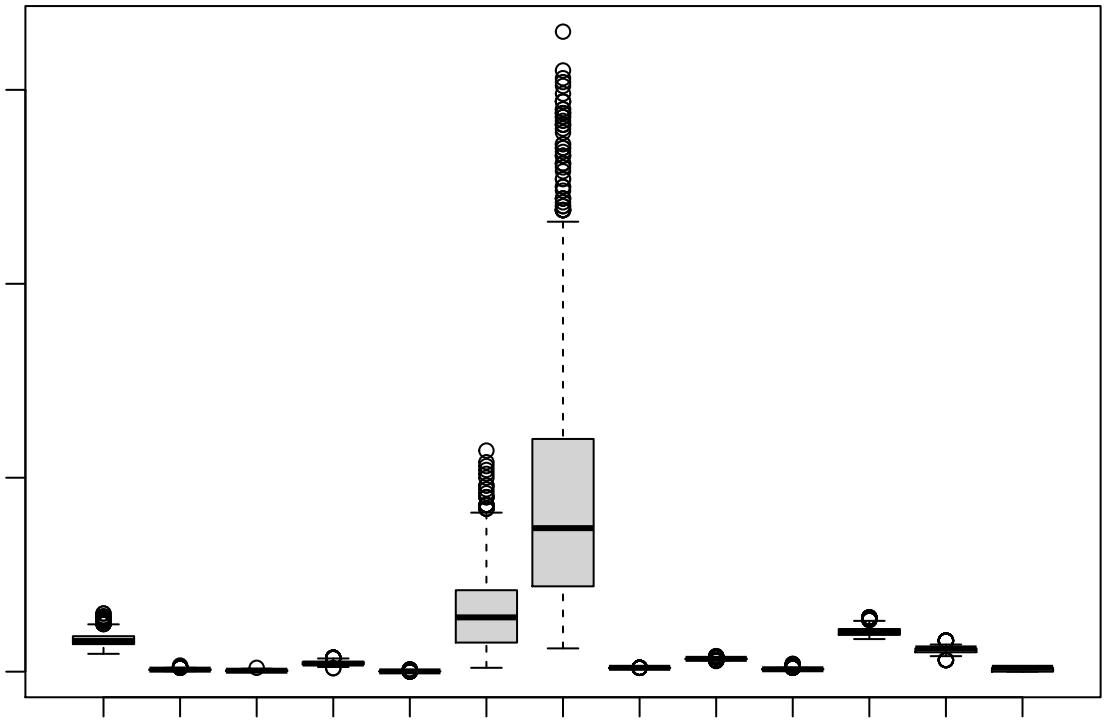
```
par(mar=c(1,1,1,1)) # to fix boxplot knit processing issues

# Create new variable, for quality values, split by half (0, 1)
wine$quality_target <- ifelse( wine$quality <= 5, 0, 1)

# Mean of new variable is at 0.5347 (close enough to 50% to maintain balance)
summary(wine$quality_target)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.5347  1.0000  1.0000
```

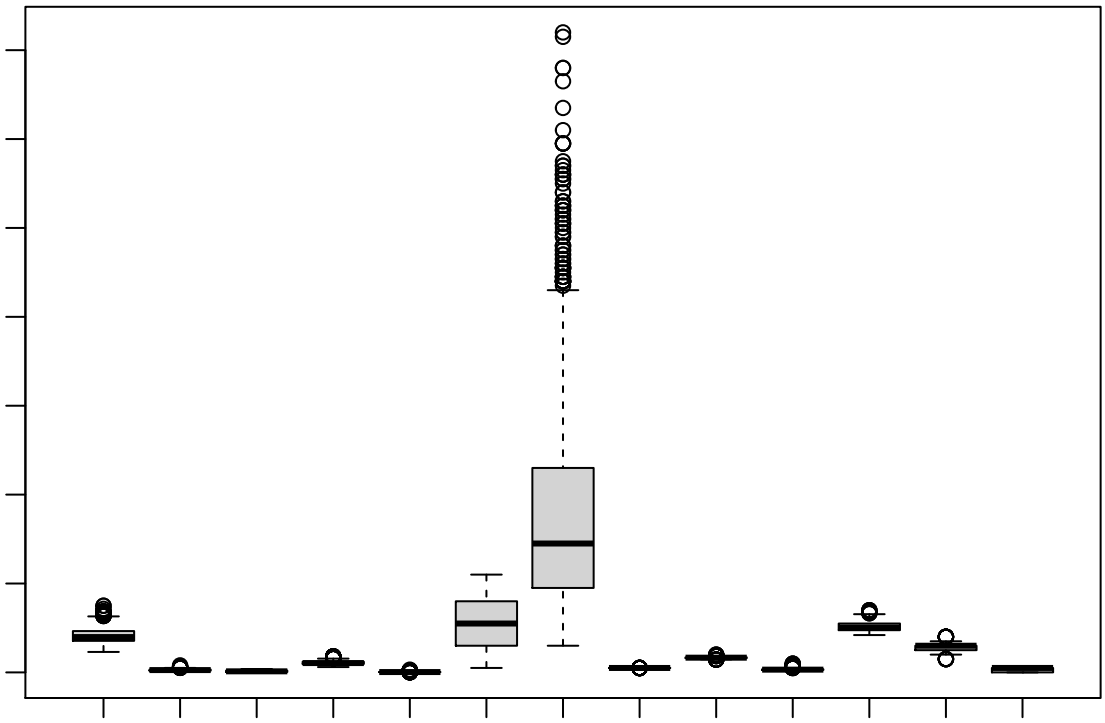
```
# Check for missing values in data set
wine %>% na.omit() %>% count() # there are no missing values
```

```
##      n
## 1 1599
```

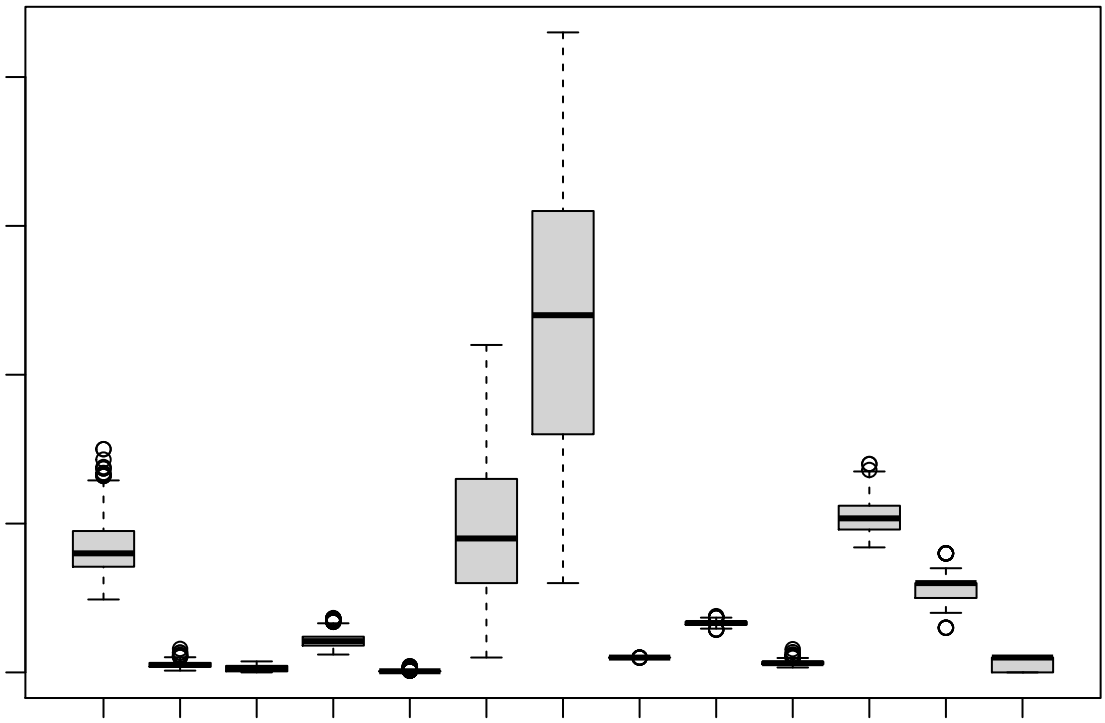
```
# Removing outliers for residual sugar:
Q <- quantile(wine$residual.sugar, probs=c(.25, .75), na.rm = FALSE)
iqr_rs <- IQR(wine$residual.sugar)
up_rs <-  Q[2]+1.5*iqr_rs # Upper Range
low_rs <- Q[1]-1.5*iqr_rs # Lower Range
eliminated_rs <- subset(wine, wine$residual.sugar > (Q[1] - 1.5*iqr_rs) & wine$residual.sugar < (Q[2]+1
boxplot(eliminated_rs)
```



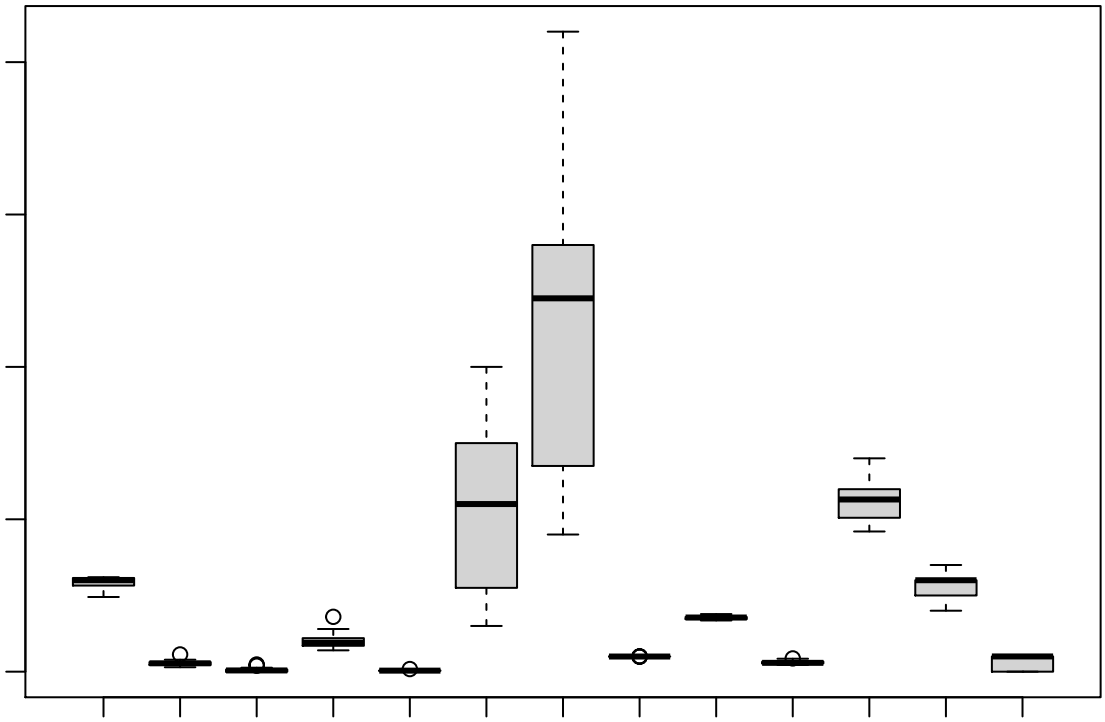
```r
#Removing outliers for free.sulfur.dioxide:
Q2 <- quantile(wine$free.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_fs <- IQR(eliminated_rs$free.sulfur.dioxide)
up_fs <-  Q2[2]+1.5*iqr_fs # Upper Range
low_fs <- Q2[1]-1.5*iqr_fs # Lower Range
eliminated_fs <- subset(eliminated_rs, eliminated_rs$free.sulfur.dioxide > (Q[1] - 1.5*iqr_fs) & elimina
boxplot(eliminated_fs)
```



```r
#Removing outliers for total.sulfur.dioxide:
Q3 <- quantile(wine$total.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_ts <- IQR(eliminated_fs$total.sulfur.dioxide)
up_ts <-  Q3[2]+1.5*iqr_ts # Upper Range
low_ts <- Q3[1]-1.5*iqr_ts # Lower Range
eliminated_ts <- subset(eliminated_fs, eliminated_fs$total.sulfur.dioxide > (Q[1] - 1.5*iqr_ts) & elimi
boxplot(eliminated_ts)
```



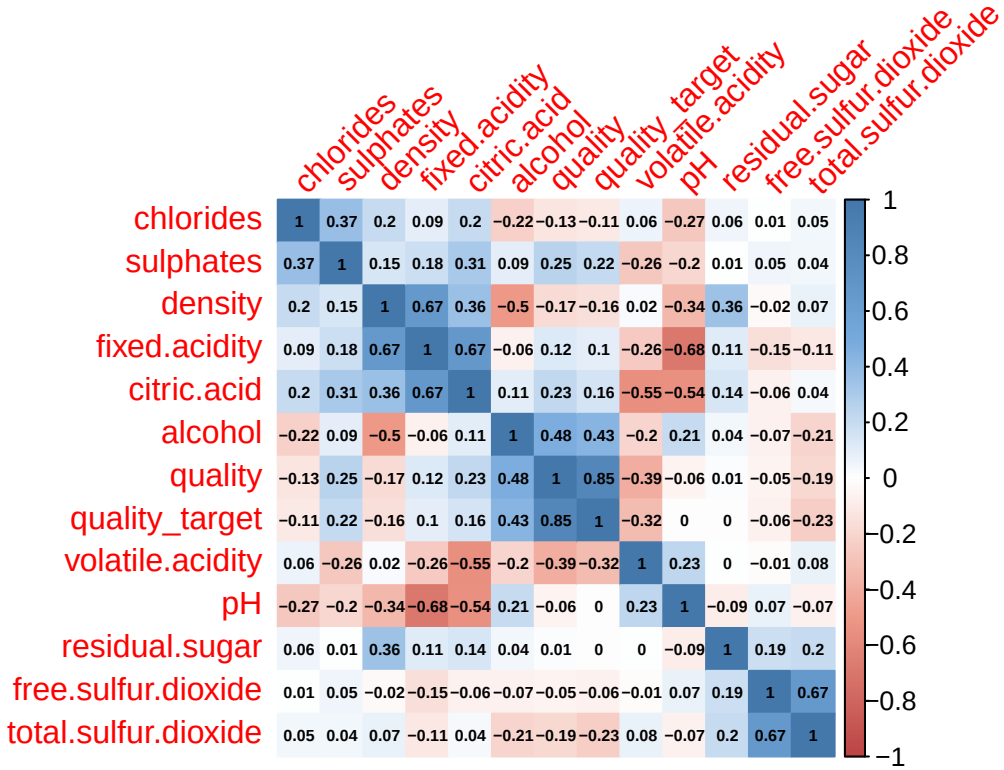
```r
#Removing outliers for fixed.acidity:
Q4 <- quantile(wine$fixed.acidity, probs=c(.25, .75), na.rm = FALSE)
iqr_fa <- IQR(eliminated_ts$fixed.acidity)
up_fa <-  Q[2]+1.5*iqr_fa # Upper Range
low_fa <- Q[1]-1.5*iqr_fa # Lower Range
eliminated_fa <- subset(eliminated_ts, eliminated_ts$fixed.acidity > (Q[1] - 1.5*iqr_fa) & eliminated_t
boxplot(eliminated_fa)
```



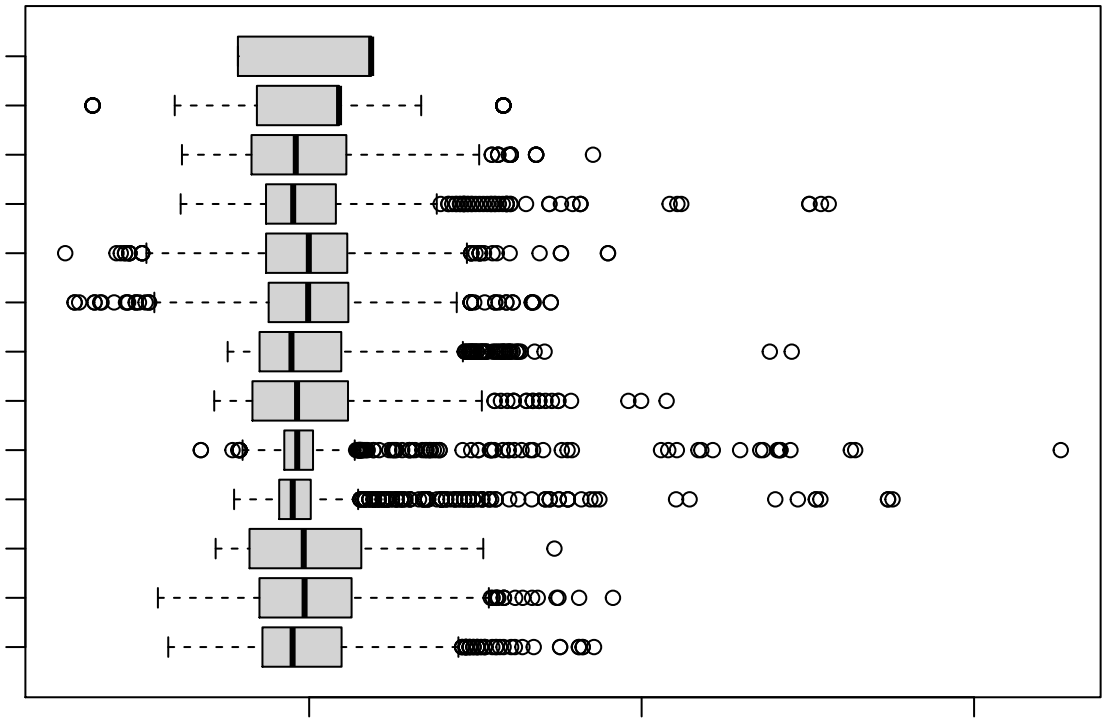
```r
new_wine_data <- eliminated_fa

# Removing outliers reduced dimension of data set from 1599 observations to 48
# team opted not to use new_wine_data and keep outlier data
dim(new_wine_data)
```

```
## [1] 48 13
```

```r
# Correlation Matrix
cor <- cor(wine)

# Colors for Correlation Matrix
colors <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))

corrplot(cor, order="hclust", method = "color", addCoef.col = "black"
         , tl.srt = 45, number.cex = 0.47, col=colors(200))
```




```r
# Cutoff Correlation features
cutoffCorr <- findCorrelation(cor, cutoff = .8)
cutoffCorrFeatures <- wine[, -cutoffCorr]

# Train and Test split
wine_split <- createDataPartition(wine$quality, p = .8, list = FALSE)
wine_train <- wine[ wine_split,]
wine_test  <- wine[-wine_split,]

# Transform Train Data
train_trans <- preProcess(wine_train, method = c("center", "scale"))
train_transformed <- predict(train_trans, wine_train)

# Transform Test Data
test_trans <- preProcess(wine_test, method = c("center", "scale"))
test_transformed <- predict(test_trans, wine_test)

# Boxplot of transformed train data
boxplot(train_transformed, horizontal = TRUE, las = 2, cex.axis = .65, cex.lab = 7)
```



Logistic Regression Model

```
# Cutoff Correlation string to copy + paste into feature area of model
subset(cutoffCorrFeatures, select = -c(quality_target)) %>%
      colnames() %>%
      paste0(collapse = " + ")
```

```
## [1] "fixed.acidity + volatile.acidity + citric.acid + residual.sugar + chlorides + free.sulfur.dioxi
```

```
set.seed(4)

# Model using "quality_target" as target variable
lmodel1 <- lm(quality_target~ volatile.acidity + sulphates + alcohol, data = wine_train)

summary(lmodel1)
```

```
##
## Call:
## lm(formula = quality_target ~ volatile.acidity + sulphates +
##     alcohol, data = wine_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.48724 -0.35555 -0.01397  0.38826  1.04649
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.22211    0.14472  -8.445  < 2e-16 ***
## volatile.acidity -0.61008    0.07184  -8.492  < 2e-16 ***
## sulphates         0.38756    0.07284   5.321 1.22e-07 ***
```

```
## alcohol           0.17473    0.01171  14.921  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4309 on 1277 degrees of freedom
## Multiple R-squared:  0.2562, Adjusted R-squared:  0.2544
## F-statistic: 146.6 on 3 and 1277 DF,  p-value: < 2.2e-16
```

```r
# Model using "quality" as target variable
lmodel2 <- lm(quality~ volatile.acidity + sulphates + alcohol, data = wine_train)

summary(lmodel2)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + sulphates + alcohol,
##     data = wine_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.71140 -0.37940 -0.05416  0.46348  2.18647
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.60361    0.22291  11.680  < 2e-16 ***
## volatile.acidity -1.23024    0.11066 -11.117  < 2e-16 ***
## sulphates         0.66850    0.11219   5.959 3.28e-09 ***
## alcohol           0.31052    0.01804  17.216  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6636 on 1277 degrees of freedom
## Multiple R-squared:  0.3292, Adjusted R-squared:  0.3277
## F-statistic: 208.9 on 3 and 1277 DF,  p-value: < 2.2e-16
```

```r
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(lmodel2, newdata = wine_test)) -> df

# Summary of predicted interval
predict(lmodel2, newdata = wine_test, interval = "prediction") %>%
  summary()
```

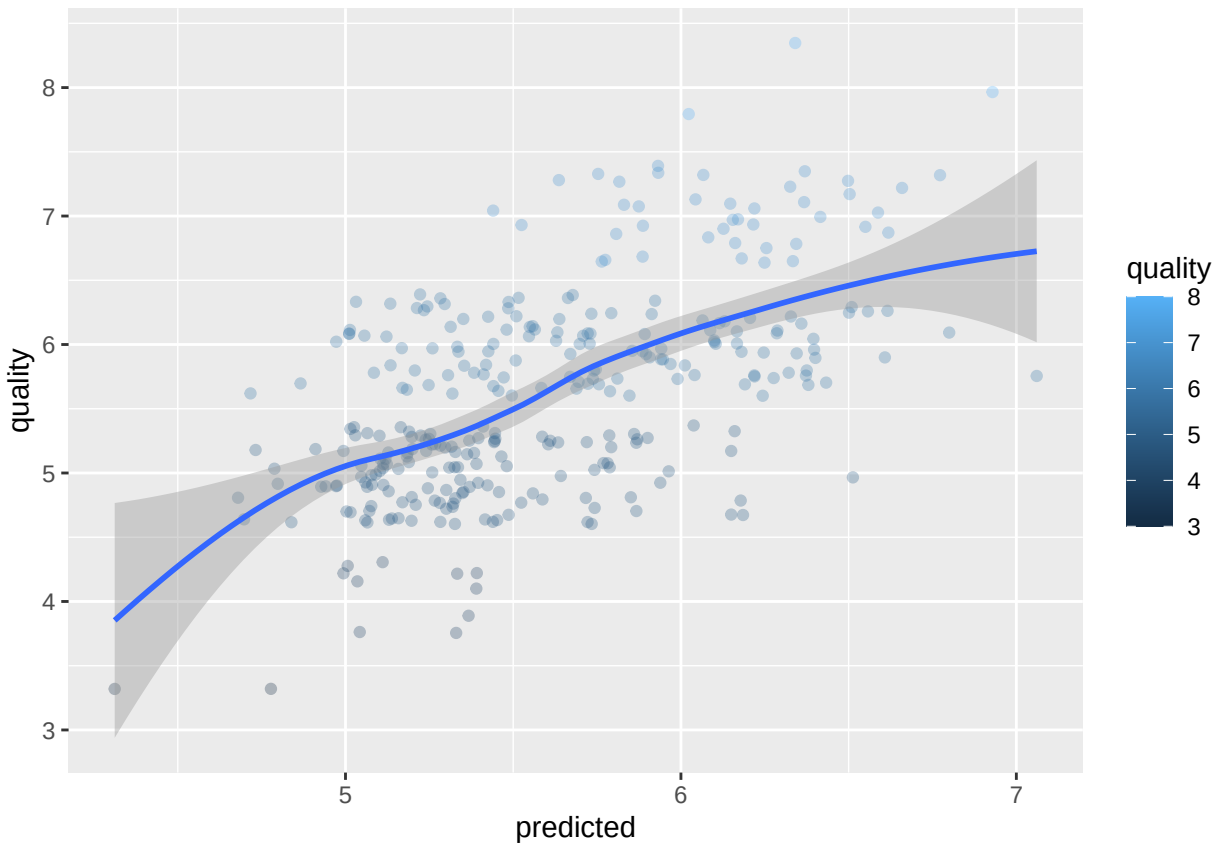
```
##       fit             lwr             upr
##  Min.   :4.312   Min.   :2.990   Min.   :5.634
##  1st Qu.:5.227   1st Qu.:3.923   1st Qu.:6.530
##  Median :5.502   Median :4.198   Median :6.806
##  Mean   :5.609   Mean   :4.305   Mean   :6.913
##  3rd Qu.:5.941   3rd Qu.:4.638   3rd Qu.:7.244
##  Max.   :7.061   Max.   :5.752   Max.   :8.369
```

```r
# Confusion Matrix
confusionMatrix(table(df$quality, wine_test$quality))
```

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

```
##
##       3   4   5   6   7   8
##   3   2   0   0   0   0   0
##   4   0  10   0   0   0   0
##   5   0   0 136   0   0   0
##   6   0   0   0 127   0   0
##   7   0   0   0   0  40   0
##   8   0   0   0   0   0   3
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9885, 1)
##     No Information Rate : 0.4277
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Specificity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Pos Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Neg Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Prevalence           0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Detection Rate       0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Detection Prevalence 0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Balanced Accuracy    1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
```

```r
# Scatter plot of predicted
ggplot(df, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```

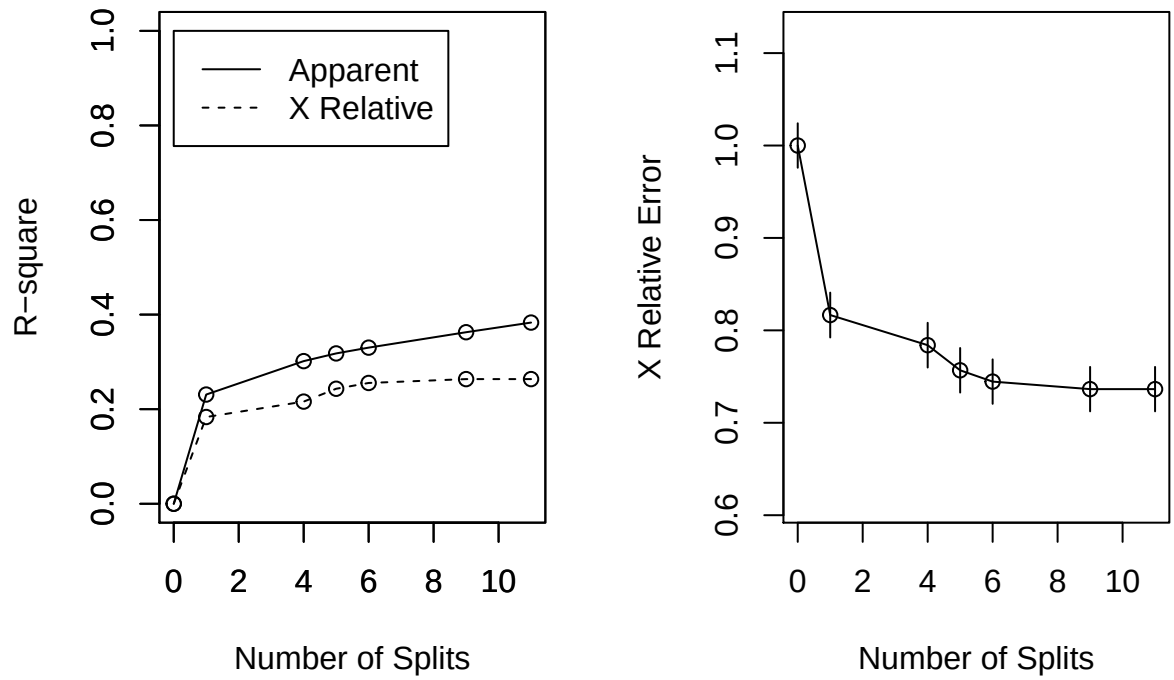



```
# The scatter plot supports the summary of the predicted interval, in the ranges of the fit,
# lower, and upper ranges. The R-squared value of 0.3283 of the model, indicates that this
# information can be predicted 33% of the time, with the data available, for the variance
# of the information.
```

CART

```
set.seed(4)
# Subset both train and test sets, to exclude "quality_target"
# Using non-transformed versions of train and test, to get actual values in the nodes
subset(wine_train, select = -c(quality_target)) -> rf_wine_train
subset(wine_test, select = -c(quality_target)) -> rf_wine_test

# Convert target variable to factor to ensure proper interpretation by model
rf_wine_train$quality <- as.factor(rf_wine_train$quality)

# Begin model...
rPartTree <- rpart(quality ~ ., data = rf_wine_train)

rpartTree2 <- as.party(rPartTree)

# R-Squared plot
par(mfrow=c(1,2))
rsq.rpart(rPartTree)
```

```
##
## Classification tree:
## rpart(formula = quality ~ ., data = rf_wine_train)
##
## Variables actually used in tree construction:
## [1] alcohol              fixed.acidity        pH
## [4] sulphates            total.sulfur.dioxide
##
## Root node error: 736/1281 = 0.57455
##
## n= 1281
##
##         CP nsplit rel error  xerror     xstd
## 1 0.230978      0   1.00000 1.00000 0.024043
## 2 0.023551      1   0.76902 0.81658 0.024268
## 3 0.016304      4   0.69837 0.78397 0.024195
## 4 0.012228      5   0.68207 0.75679 0.024107
## 5 0.010870      6   0.66984 0.74457 0.024060
## 6 0.010190      9   0.63723 0.73641 0.024025
## 7 0.010000     11   0.61685 0.73641 0.024025
```

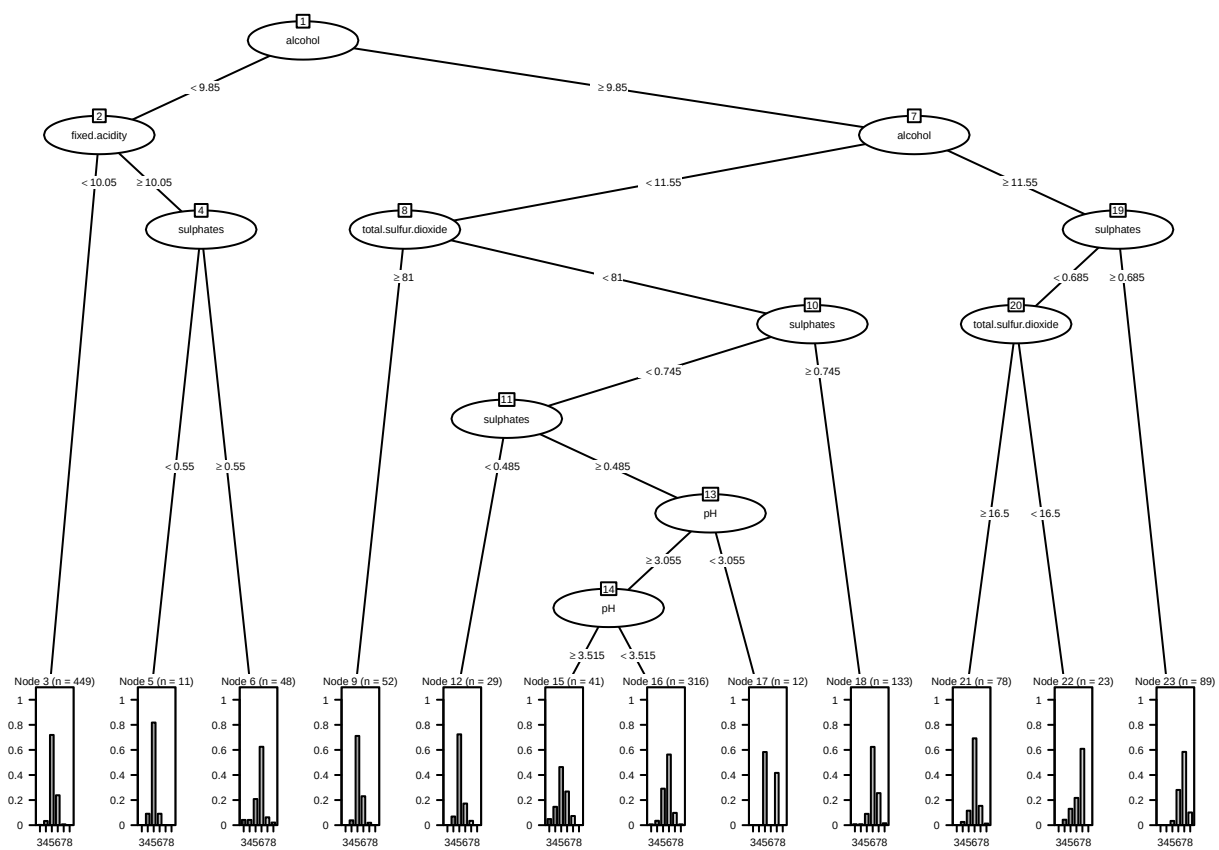



```r
# Results
rpartTree2
```

```
##
## Model formula:
## quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol
##
## Fitted party:
## [1] root
## |   [2] alcohol < 9.85
```

```
## |   |   [3] fixed.acidity < 10.05: 5 (n = 449, err = 28.1%)
## |   |   [4] fixed.acidity >= 10.05
## |   |   |   [5] sulphates < 0.55: 5 (n = 11, err = 18.2%)
## |   |   |   [6] sulphates >= 0.55: 6 (n = 48, err = 37.5%)
## |   [7] alcohol >= 9.85
## |   |   [8] alcohol < 11.55
## |   |   |   [9] total.sulfur.dioxide >= 81: 5 (n = 52, err = 28.8%)
## |   |   |   [10] total.sulfur.dioxide < 81
## |   |   |   |   [11] sulphates < 0.745
## |   |   |   |   |   [12] sulphates < 0.485: 5 (n = 29, err = 27.6%)
## |   |   |   |   |   [13] sulphates >= 0.485
## |   |   |   |   |   |   [14] pH >= 3.055
## |   |   |   |   |   |   |   [15] pH >= 3.515: 5 (n = 41, err = 53.7%)
## |   |   |   |   |   |   |   [16] pH < 3.515: 6 (n = 316, err = 43.7%)
## |   |   |   |   |   |   [17] pH < 3.055: 5 (n = 12, err = 41.7%)
## |   |   |   |   [18] sulphates >= 0.745: 6 (n = 133, err = 37.6%)
## |   |   [19] alcohol >= 11.55
## |   |   |   [20] sulphates < 0.685
## |   |   |   |   [21] total.sulfur.dioxide >= 16.5: 6 (n = 78, err = 30.8%)
## |   |   |   |   [22] total.sulfur.dioxide < 16.5: 7 (n = 23, err = 39.1%)
## |   |   |   [23] sulphates >= 0.685: 7 (n = 89, err = 41.6%)
##
## Number of inner nodes:    11
## Number of terminal nodes: 12
```

```
plot(rpartTree2, gp = gpar(fontsize=4))
```



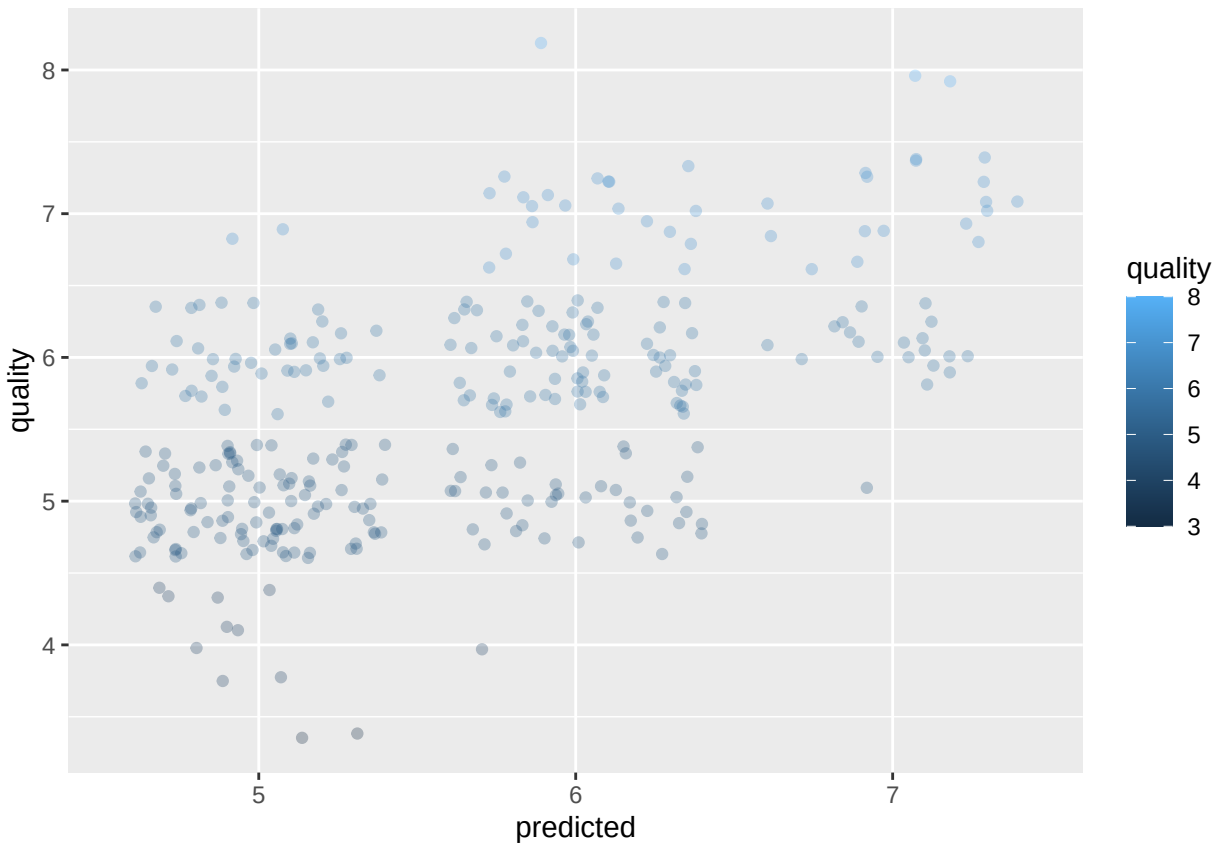
```r
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(rpartTree2, newdata = wine_test)) -> df2

# Summary of predicted values
predict(rpartTree2, newdata = wine_test, interval = "prediction") %>%
  summary()
```

```
##   3   4   5   6   7   8
##   0   0 151 128  39   0
```

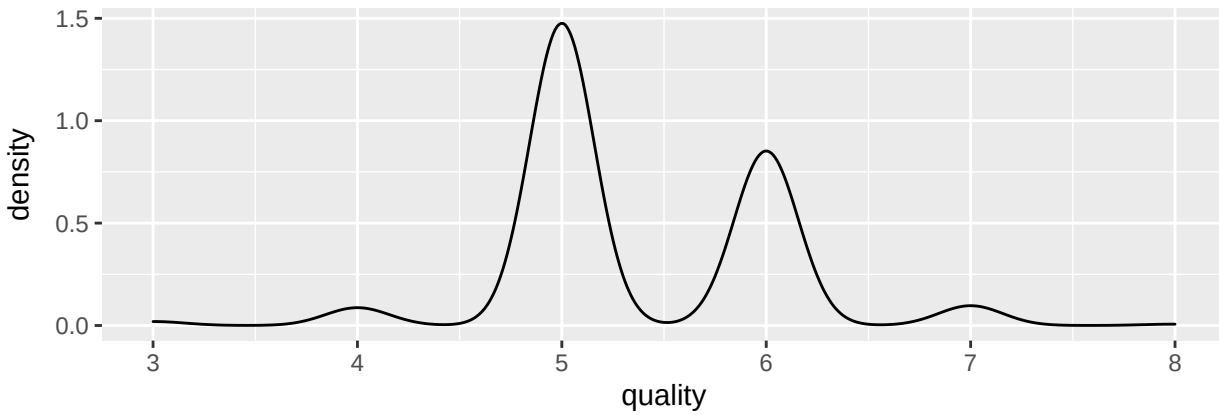
```r
# Confusion Matrix
confusionMatrix(table(df2$quality, wine_test$quality))
```

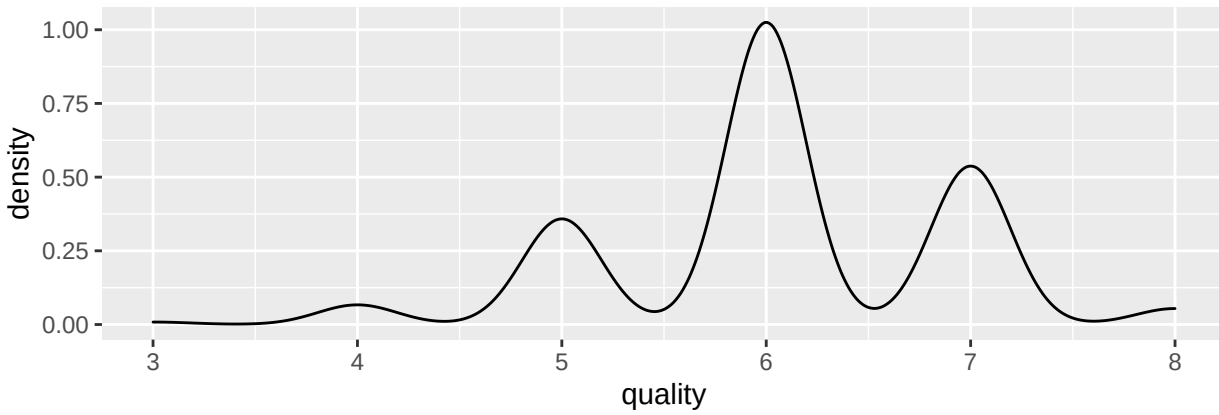
```
## Confusion Matrix and Statistics
##
##
##       3   4   5   6   7   8
##   3   2   0   0   0   0   0
##   4   0  10   0   0   0   0
##   5   0   0 136   0   0   0
##   6   0   0   0 127   0   0
##   7   0   0   0   0  40   0
##   8   0   0   0   0   0   3
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9885, 1)
##     No Information Rate : 0.4277
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Specificity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Pos Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Neg Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Prevalence           0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Detection Rate       0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Detection Prevalence 0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Balanced Accuracy    1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
```

```r
# Scatter plot of predicted
ggplot(df2, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```




```
# Root Node Left vs Right, Quality Density Comparisons
grid.newpage()
filter(wine_train, alcohol < 10.525) %>%
  dplyr::select(quality, alcohol) %>%
  ggplot(aes(x = quality)) + geom_density() -> RootNodeLeft


filter(wine_train, alcohol >= 10.525) %>%
  dplyr::select(quality, alcohol) %>%
  ggplot(aes(x = quality)) + geom_density() -> RootNodeRight

grid.draw(rbind(ggplotGrob(RootNodeLeft), ggplotGrob(RootNodeRight), size = "last"))
```





Random Forest

```
set.seed(4)

rf <- rfsrc(quality ~ ., data = rf_wine_train)

print(rf)
```

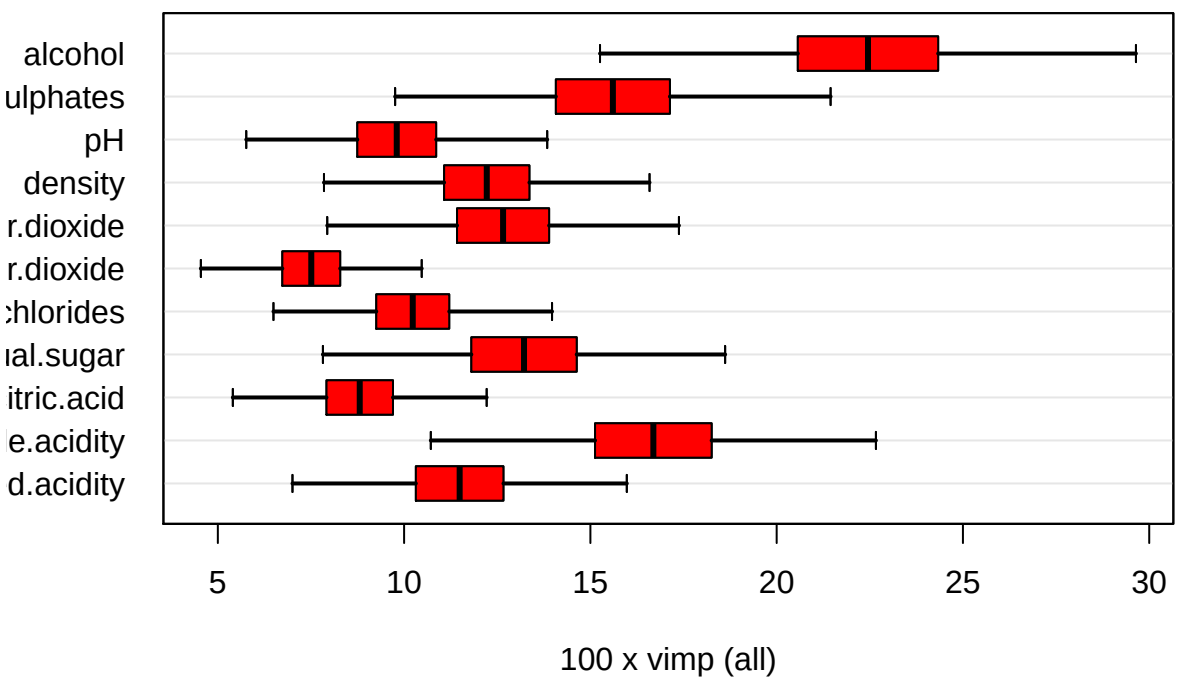
```
##                          Sample size: 1281
##            Frequency of class labels: 8, 43, 545, 511, 159, 15
##                      Number of trees: 500
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 251.7
## No. of variables tried at each split: 4
##               Total no. of variables: 11
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 810
##                             Analysis: RF-C
##                               Family: class
##                      Splitting rule: gini
##                    (OOB) Brier score: 0.06847406
##         (OOB) Normalized Brier score: 0.49301322
##                            (OOB) AUC: 0.81432297
##    (OOB) Requested performance error: 0.28649493, 1, 1, 0.19633028, 0.23091977, 0.47798742, 1
##
```

```
## Confusion matrix:
##
##           predicted
##   observed 3 4   5   6  7 8 class.error
##          3 0 1   6   1  0 0      1.0000
##          4 0 0  27  15  1 0      1.0000
##          5 1 0 440  98  6 0      0.1927
##          6 0 1  94 393 23 0      0.2309
##          7 0 0   9  68 82 0      0.4843
##          8 0 0   0   8  7 0      1.0000
##
##       (OOB) Misclassification rate: 0.2857143
```

```r
# Variable Importance
vi <- subsample(rf, verbose = FALSE)

extract.subsample(vi)$var.jk.sel.Z
```

```
##                          lower      mean     upper       pvalue signif
## fixed.acidity         8.076378 11.490446 14.904514 2.105061e-11   TRUE
## volatile.acidity     12.144486 16.690021 21.235557 3.089421e-13   TRUE
## citric.acid           6.216803  8.808917 11.401032 1.363119e-11   TRUE
## residual.sugar        9.110773 13.218684 17.326595 1.423488e-10   TRUE
## chlorides             7.387026 10.231847 13.076668 8.989784e-13   TRUE
## free.sulfur.dioxide   5.252383  7.507856  9.763328 3.418536e-11   TRUE
## total.sulfur.dioxide  9.065001 12.656476 16.247950 2.475441e-12   TRUE
## density               8.893376 12.218259 15.543142 2.956732e-13   TRUE
## pH                    6.728840  9.802123 12.875407 2.036139e-10   TRUE
## sulphates            11.156529 15.603397 20.050265 3.051573e-12   TRUE
## alcohol              16.978421 22.451380 27.924339 4.483200e-16   TRUE
```

```r
# Variable Importance Plot
plot(vi)
```



```r
# Confusion Matrix
# https://www.rdocumentation.org/packages/randomForestSRC/versions/3.1.0/topics/predict.rfsrc
randomForestSRC::predict.rfsrc(rf, rf_wine_test)
```

```
##   Sample size of test (predict) data: 318
##                 Number of grow trees: 500
##   Average no. of grow terminal nodes: 251.7
##          Total no. of grow variables: 11
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 810
##                             Analysis: RF-C
##                               Family: class
##                          Brier score: 0.07220666
##               Normalized Brier score: 0.51988793
##                                  AUC: 0.83610494
##          Requested performance error: 0.32075472, 1, 1, 0.20588235, 0.34645669, 0.4, 0.66666667
##
## Confusion matrix:
##
##           predicted
##   observed 3 4   5  6  7 8 class.error
##          3 0 0   2  0  0 0      1.0000
##          4 0 0   9  1  0 0      1.0000
##          5 0 0 108 28  0 0      0.2059
##          6 0 0  31 83 13 0      0.3465
##          7 0 0   0 16 24 0      0.4000
##          8 0 0   0  2  0 1      0.6667
##
##            Misclassification error: 0.3207547
```

Partial Least Squares

```r
tctrl <- trainControl(method = "repeatedcv", repeats = 5, number =10)

set.seed(4)
pls_wine <- train(quality~ volatile.acidity + chlorides + total.sulfur.dioxide +
               sulphates + alcohol, data = wine_train,
                  method = "pls",
                  preProc = c("center", "scale", "BoxCox"),
                  tunelength =20,
                  trControl = tctrl)

pls_wine
```

```
## Partial Least Squares
##
## 1281 samples
##    5 predictor
##
## Pre-processing: centered (5), scaled (5), Box-Cox transformation (5)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 1153, 1153, 1153, 1154, 1153, 1152, ...
```

```
## Resampling results across tuning parameters:
##
##   ncomp  RMSE       Rsquared   MAE
##   1      0.6526777  0.3528246  0.5038449
##   2      0.6521761  0.3536735  0.5041600
##   3      0.6522194  0.3535854  0.5034022
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 2.
```

```r
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(pls_wine, newdata = wine_test)) -> df3

# Summary of predicted interval
predict(pls_wine, newdata = wine_test, interval = "prediction") %>%
  summary()
```

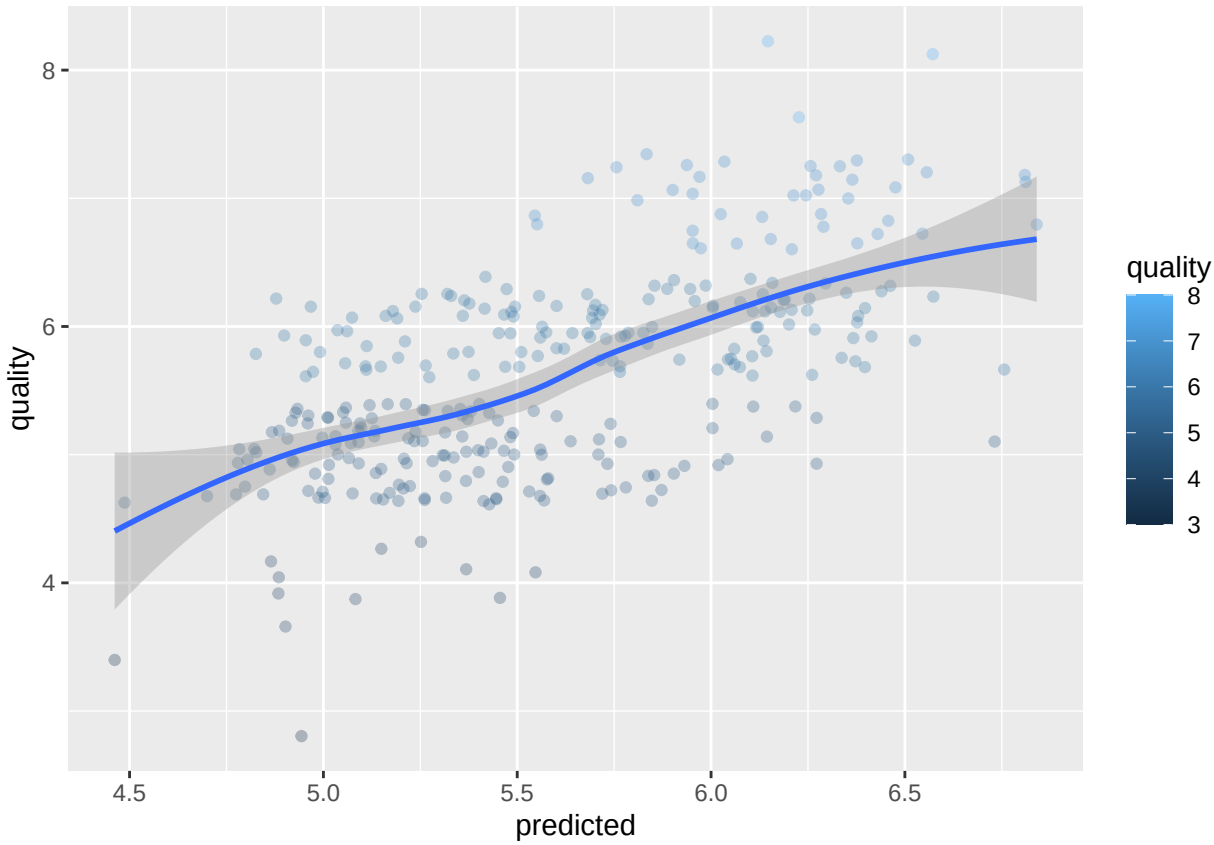
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.462   5.173   5.537   5.596   6.019   6.840
```

```r
# Confusion Matrix
confusionMatrix(table(df3$quality, wine_test$quality))
```

```
## Confusion Matrix and Statistics
##
##
##       3   4   5   6   7   8
##   3   2   0   0   0   0   0
##   4   0  10   0   0   0   0
##   5   0   0 136   0   0   0
##   6   0   0   0 127   0   0
##   7   0   0   0   0  40   0
##   8   0   0   0   0   0   3
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9885, 1)
##     No Information Rate : 0.4277
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Specificity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Pos Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Neg Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Prevalence           0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Detection Rate       0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
```

```
## Detection Prevalence 0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Balanced Accuracy    1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
```

```
# Scatter plot of predicted
ggplot(df3, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```



Mars Tuning

```
mars_wine <- earth(quality~ volatile.acidity + chlorides + total.sulfur.dioxide +
               sulphates + alcohol, data =wine_train)

mars_wine
```

```
## Selected 12 of 16 terms, and 5 of 5 predictors
## Termination condition: Reached nk 21
## Importance: alcohol, volatile.acidity, sulphates, total.sulfur.dioxide, ...
## Number of terms at each degree of interaction: 1 11 (additive model)
## GCV 0.415055    RSS 512.7644    GRSq 0.3668827    RSq 0.3884591
```

```
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##             data=wine_train)
##
##                              coefficients
## (Intercept)                    -12.176470
```

```
## h(volatile.acidity-0.28)       -1.014973
## h(chlorides-0.041)             68.493464
## h(chlorides-0.062)            -15.171908
## h(0.387-chlorides)             54.564648
## h(chlorides-0.387)            -58.435648
## h(total.sulfur.dioxide-28)     -0.003646
## h(total.sulfur.dioxide-130)     0.011603
## h(0.82-sulphates)              -1.833070
## h(alcohol-11)                   0.163438
## h(13.5667-alcohol)             -0.219391
## h(alcohol-13.5667)             -1.478732
##
## Selected 12 of 16 terms, and 5 of 5 predictors
## Termination condition: Reached nk 21
## Importance: alcohol, volatile.acidity, sulphates, total.sulfur.dioxide, ...
## Number of terms at each degree of interaction: 1 11 (additive model)
## GCV 0.415055    RSS 512.7644    GRSq 0.3668827    RSq 0.3884591
```

```
preProc_Arguments = c("center", "scale")
marsGrid_wine = expand.grid(.degree=1:2, .nprune=2:38)

set.seed(4)

marsModel_wine = train(quality~ volatile.acidity + chlorides + total.sulfur.dioxide +
                       sulphates + alcohol, data =wine_train,
                       method="earth",
                       preProc=preProc_Arguments,
                       tuneGrid=marsGrid_wine)

marsModel_wine
```

```
## Multivariate Adaptive Regression Spline
##
## 1281 samples
##    5 predictor
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1281, 1281, 1281, 1281, 1281, 1281, ...
## Resampling results across tuning parameters:
##
##   degree  nprune  RMSE       Rsquared   MAE
##   1        2      0.7234413  0.2198385  0.5700694
##   1        3      0.6876489  0.2941272  0.5263963
##   1        4      0.6665053  0.3362394  0.5126019
##   1        5      0.6658819  0.3375292  0.5113499
##   1        6      0.6668338  0.3365171  0.5113402
##   1        7      0.6672370  0.3360041  0.5109459
##   1        8      0.6668844  0.3366998  0.5101416
##   1        9      0.6674243  0.3362649  0.5097657
##   1       10      0.6686726  0.3343613  0.5103793
##   1       11      0.6692458  0.3336017  0.5104716
##   1       12      0.6695281  0.3332090  0.5108972
##   1       13      0.6703990  0.3317790  0.5115922
##   1       14      0.6711876  0.3304337  0.5121725
```

```
##   1        15       0.6711159  0.3305835  0.5120943
##   1        16       0.6738762  0.3267381  0.5130379
##   1        17       0.6738762  0.3267381  0.5130379
##   1        18       0.6738762  0.3267381  0.5130379
##   1        19       0.6738762  0.3267381  0.5130379
##   1        20       0.6738762  0.3267381  0.5130379
##   1        21       0.6738762  0.3267381  0.5130379
##   1        22       0.6738762  0.3267381  0.5130379
##   1        23       0.6738762  0.3267381  0.5130379
##   1        24       0.6738762  0.3267381  0.5130379
##   1        25       0.6738762  0.3267381  0.5130379
##   1        26       0.6738762  0.3267381  0.5130379
##   1        27       0.6738762  0.3267381  0.5130379
##   1        28       0.6738762  0.3267381  0.5130379
##   1        29       0.6738762  0.3267381  0.5130379
##   1        30       0.6738762  0.3267381  0.5130379
##   1        31       0.6738762  0.3267381  0.5130379
##   1        32       0.6738762  0.3267381  0.5130379
##   1        33       0.6738762  0.3267381  0.5130379
##   1        34       0.6738762  0.3267381  0.5130379
##   1        35       0.6738762  0.3267381  0.5130379
##   1        36       0.6738762  0.3267381  0.5130379
##   1        37       0.6738762  0.3267381  0.5130379
##   1        38       0.6738762  0.3267381  0.5130379
##   2         2       0.7234083  0.2196138  0.5707769
##   2         3       0.6911889  0.2874159  0.5315924
##   2         4       0.6762783  0.3175039  0.5183509
##   2         5       0.6716680  0.3294075  0.5133977
##   2         6       0.6685308  0.3364889  0.5100397
##   2         7       0.6722990  0.3308373  0.5118466
##   2         8       0.6729484  0.3308414  0.5119695
##   2         9       0.6724712  0.3327993  0.5113852
##   2        10       0.6728268  0.3326572  0.5118917
##   2        11       0.6735647  0.3315483  0.5124138
##   2        12       0.6763713  0.3275009  0.5141396
##   2        13       0.6767380  0.3268138  0.5140346
##   2        14       0.6767016  0.3279846  0.5144852
##   2        15       0.6767649  0.3278388  0.5145884
##   2        16       0.6772042  0.3280601  0.5148153
##   2        17       0.6772278  0.3281488  0.5148012
##   2        18       0.6776391  0.3275331  0.5147954
##   2        19       0.6778781  0.3271154  0.5150502
##   2        20       0.6778781  0.3271154  0.5150502
##   2        21       0.6778781  0.3271154  0.5150502
##   2        22       0.6778781  0.3271154  0.5150502
##   2        23       0.6778781  0.3271154  0.5150502
##   2        24       0.6778781  0.3271154  0.5150502
##   2        25       0.6778781  0.3271154  0.5150502
##   2        26       0.6778781  0.3271154  0.5150502
##   2        27       0.6778781  0.3271154  0.5150502
##   2        28       0.6778781  0.3271154  0.5150502
##   2        29       0.6778781  0.3271154  0.5150502
##   2        30       0.6778781  0.3271154  0.5150502
##   2        31       0.6778781  0.3271154  0.5150502
```

```
##   2       32      0.6778781  0.3271154  0.5150502
##   2       33      0.6778781  0.3271154  0.5150502
##   2       34      0.6778781  0.3271154  0.5150502
##   2       35      0.6778781  0.3271154  0.5150502
##   2       36      0.6778781  0.3271154  0.5150502
##   2       37      0.6778781  0.3271154  0.5150502
##   2       38      0.6778781  0.3271154  0.5150502
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nprune = 5 and degree = 1.
```

```r
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(marsModel_wine, newdata = wine_test)) -> df4

# Summary of predicted interval
predict(marsModel_wine, newdata = wine_test, interval = "prediction") %>%
  summary()
```

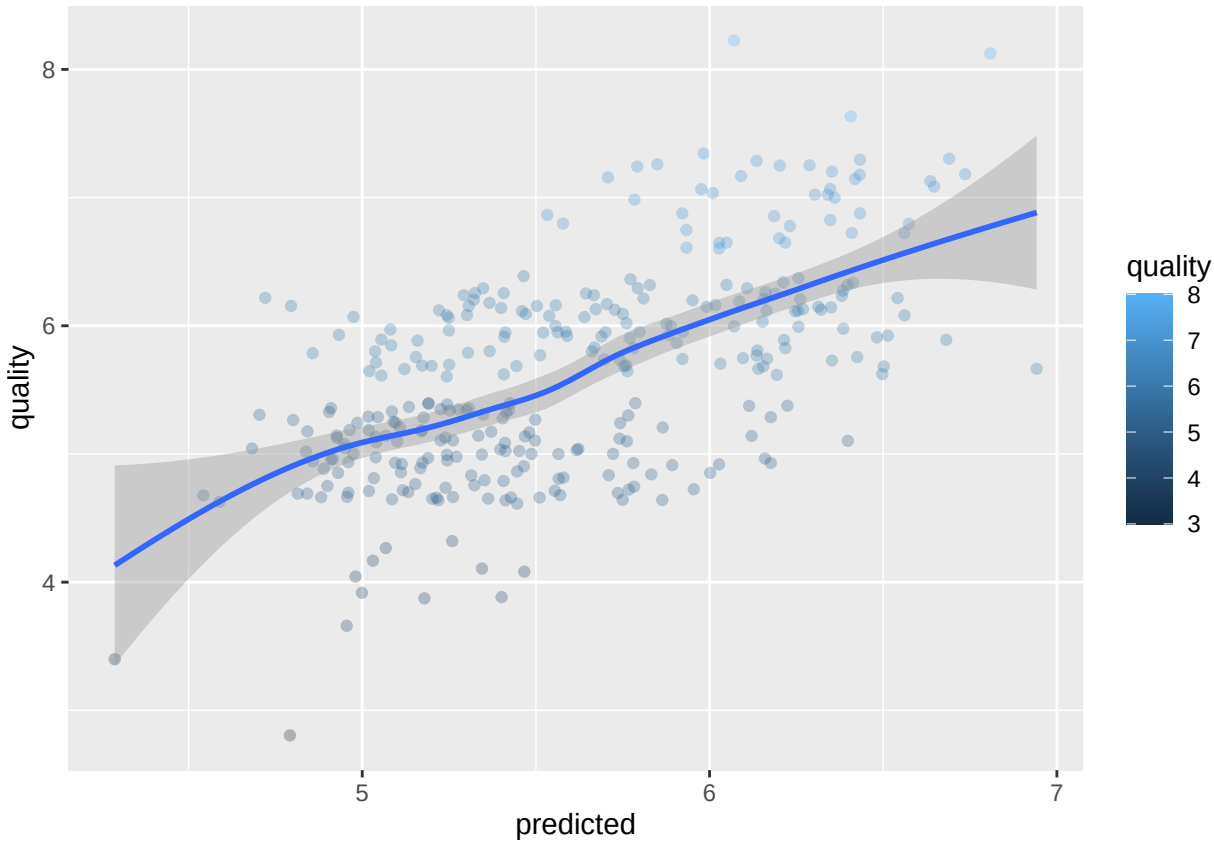
```
##        y
##  Min.   :4.287
##  1st Qu.:5.190
##  Median :5.527
##  Mean   :5.605
##  3rd Qu.:6.030
##  Max.   :6.942
```

```r
# Confusion Matrix
confusionMatrix(table(df4$quality, wine_test$quality))
```

```
## Confusion Matrix and Statistics
##
##
##       3   4   5   6   7   8
##   3   2   0   0   0   0   0
##   4   0  10   0   0   0   0
##   5   0   0 136   0   0   0
##   6   0   0   0 127   0   0
##   7   0   0   0   0  40   0
##   8   0   0   0   0   0   3
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9885, 1)
##     No Information Rate : 0.4277
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                       Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
```

```
## Sensitivity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Specificity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Pos Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Neg Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Prevalence           0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Detection Rate       0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Detection Prevalence 0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Balanced Accuracy    1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
```

```
# Scatter plot of predicted
ggplot(df4, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```



KNN Neighbors

```
set.seed(4)

knn_wine <- train(quality~ volatile.acidity + chlorides + total.sulfur.dioxide +
                sulphates + alcohol, data =wine_train,
                method = "knn",
                preProc = c("center", "scale"),
                tuneGrid = data.frame(.k = 1:50),
                trControl = trainControl(method = "cv"))

knn_wine
```

```
## k-Nearest Neighbors
##
## 1281 samples
##    5 predictor
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1153, 1153, 1153, 1154, 1153, 1152, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1  0.7546527  0.3186801  0.4316729
##    2  0.7133784  0.3122928  0.4867106
##    3  0.6930651  0.3220558  0.4964064
##    4  0.6812281  0.3265331  0.5016008
##    5  0.6715061  0.3381102  0.5030667
##    6  0.6624739  0.3502293  0.4997296
##    7  0.6557274  0.3587050  0.4968164
##    8  0.6503647  0.3658804  0.4968629
##    9  0.6487984  0.3662867  0.4961581
##   10  0.6486688  0.3642824  0.4987591
##   11  0.6454132  0.3691115  0.4966535
##   12  0.6473606  0.3649919  0.5007087
##   13  0.6486722  0.3622100  0.5029631
##   14  0.6486259  0.3624485  0.5023383
##   15  0.6511986  0.3573020  0.5043849
##   16  0.6513267  0.3567453  0.5037143
##   17  0.6512601  0.3564462  0.5019815
##   18  0.6491198  0.3606198  0.5011056
##   19  0.6508770  0.3571356  0.5025093
##   20  0.6499922  0.3590977  0.5022235
##   21  0.6484430  0.3622590  0.5016461
##   22  0.6484380  0.3620425  0.5018443
##   23  0.6491924  0.3607441  0.5036764
##   24  0.6494199  0.3602642  0.5039036
##   25  0.6488873  0.3613303  0.5033547
##   26  0.6477122  0.3636785  0.5025664
##   27  0.6487738  0.3615960  0.5038678
##   28  0.6477562  0.3640820  0.5029529
##   29  0.6476376  0.3645493  0.5027909
##   30  0.6475762  0.3648880  0.5024314
##   31  0.6480928  0.3638368  0.5034399
##   32  0.6471761  0.3657069  0.5025370
##   33  0.6469131  0.3662738  0.5029386
##   34  0.6464623  0.3671240  0.5024577
##   35  0.6467044  0.3667156  0.5028359
##   36  0.6471165  0.3660257  0.5039599
##   37  0.6466947  0.3669011  0.5034105
##   38  0.6471261  0.3661636  0.5041958
##   39  0.6471787  0.3662349  0.5040420
##   40  0.6464795  0.3674693  0.5031270
##   41  0.6471078  0.3663165  0.5038061
##   42  0.6471290  0.3665087  0.5042624
##   43  0.6468191  0.3672130  0.5038617
```

```
##   44  0.6465630  0.3680413  0.5038448
##   45  0.6465756  0.3680413  0.5030909
##   46  0.6465459  0.3681856  0.5035092
##   47  0.6464026  0.3684926  0.5039392
##   48  0.6460595  0.3693877  0.5037035
##   49  0.6462672  0.3690032  0.5041684
##   50  0.6457945  0.3701385  0.5036881
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 11.
```

```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(knn_wine, newdata = wine_test)) -> df5

# Summary of predicted interval
predict(knn_wine, newdata = wine_test, interval = "prediction") %>%
  summary()
```

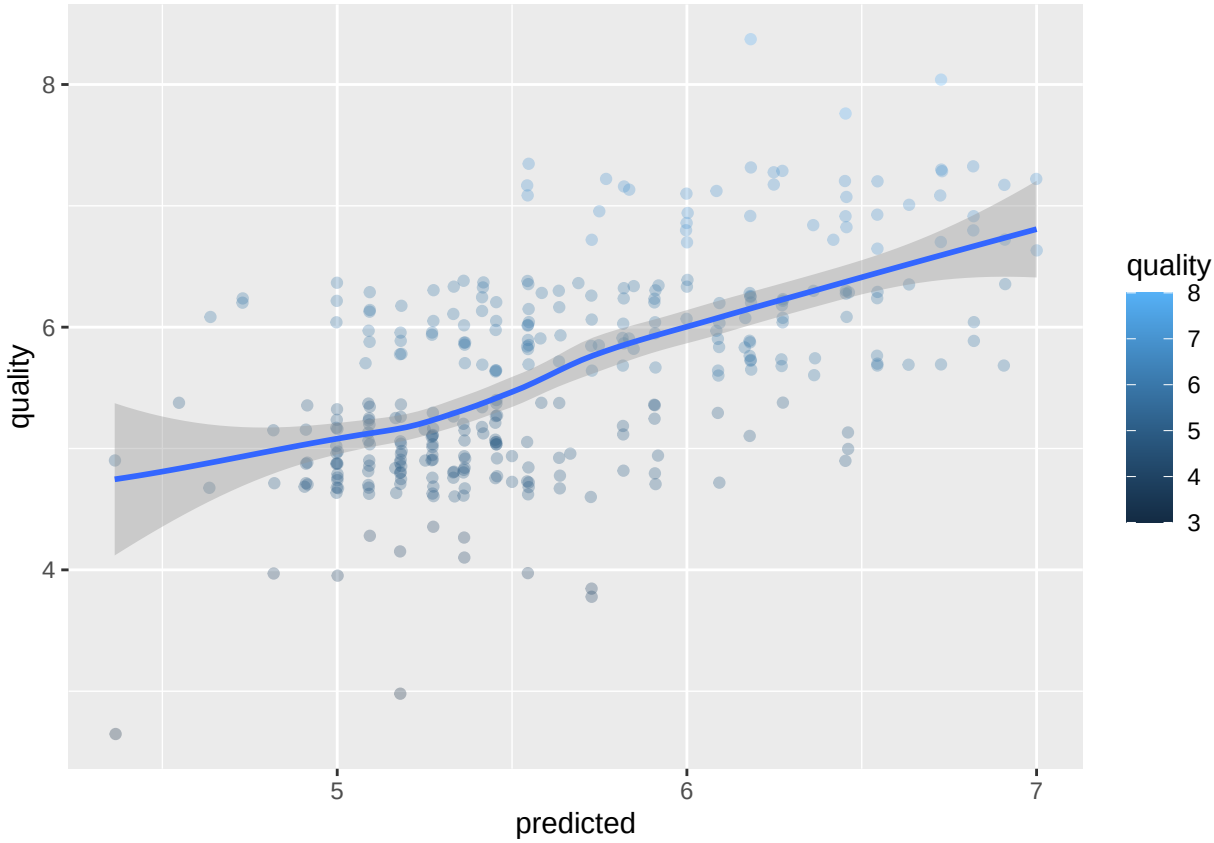
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.364   5.182   5.545   5.645   6.089   7.000
```

```
# Confusion Matrix
confusionMatrix(table(df5$quality, wine_test$quality))
```

```
## Confusion Matrix and Statistics
##
##
##       3   4   5   6   7   8
##   3   2   0   0   0   0   0
##   4   0  10   0   0   0   0
##   5   0   0 136   0   0   0
##   6   0   0   0 127   0   0
##   7   0   0   0   0  40   0
##   8   0   0   0   0   0   3
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9885, 1)
##     No Information Rate : 0.4277
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Specificity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Pos Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Neg Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Prevalence           0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
```

```
## Detection Rate       0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Detection Prevalence 0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Balanced Accuracy    1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
```

```
# Scatter plot of predicted
ggplot(df5, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```



SVM

```
set.seed(4)


svmTune <- train(quality ~ ., data = rf_wine_train, # using the subset data as used in random forest
                 method = "svmRadial",
                 preProc = c("center", "scale"),
                 tuneLength= 5,
                 trControl = trainControl(method = "cv"))
svmTune
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1281 samples
##   11 predictor
##    6 classes: '3', '4', '5', '6', '7', '8'
##
```

```
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1154, 1153, 1154, 1151, 1152, 1153, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.5995721  0.3205309
##   0.50  0.6183041  0.3651593
##   1.00  0.6284119  0.3858982
##   2.00  0.6362430  0.4023162
##   4.00  0.6393500  0.4131101
##
## Tuning parameter 'sigma' was held constant at a value of 0.09547498
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.09547498 and C = 4.
```

```r
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(svmTune, newdata = wine_test)) -> df6

# Summary of predicted interval
predict(svmTune, newdata = wine_test, interval = "prediction") %>%
  summary()
```

```
##   3   4   5   6   7   8
##   2   1 160 125  29   1
```

```r
# Confusion Matrix
confusionMatrix(table(df6$quality, wine_test$quality))
```

```
## Confusion Matrix and Statistics
##
##
##       3   4   5   6   7   8
##   3   2   0   0   0   0   0
##   4   0  10   0   0   0   0
##   5   0   0 136   0   0   0
##   6   0   0   0 127   0   0
##   7   0   0   0   0  40   0
##   8   0   0   0   0   0   3
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9885, 1)
##     No Information Rate : 0.4277
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                       Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
```

```
## Sensitivity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Specificity          1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Pos Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Neg Pred Value       1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
## Prevalence           0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Detection Rate       0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Detection Prevalence 0.006289  0.03145   0.4277   0.3994   0.1258 0.009434
## Balanced Accuracy    1.000000  1.00000   1.0000   1.0000   1.0000 1.000000
```

```r
# Scatter plot of predicted
ggplot(df6, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```