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.

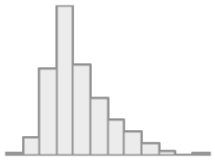
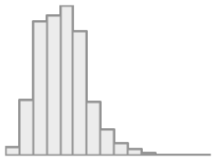
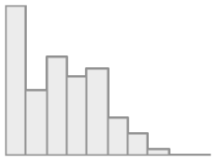
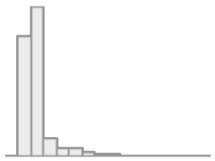
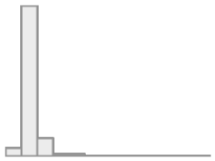
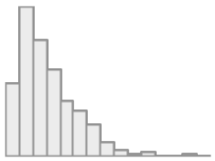
```
wine <- read.csv(
  url("https://raw.githubusercontent.com/OscarG-DataSci/ADS503/main/winequality-red.csv")
                      , header = TRUE)
```

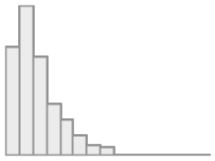
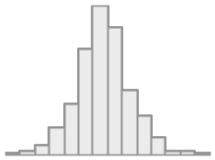
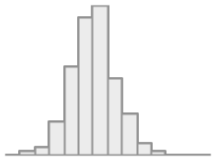
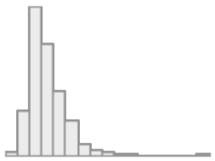
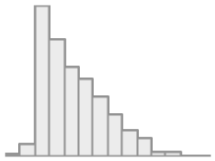
Data Summary

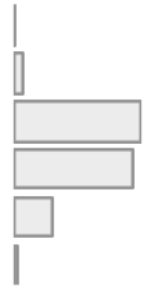
```
dfSummary(wine,
          plain.ascii  = FALSE,
          style        = "grid",
          graph.magnif = 0.75,
          valid.col    = FALSE,
          tmp.img.dir  = "/tmp")
```

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

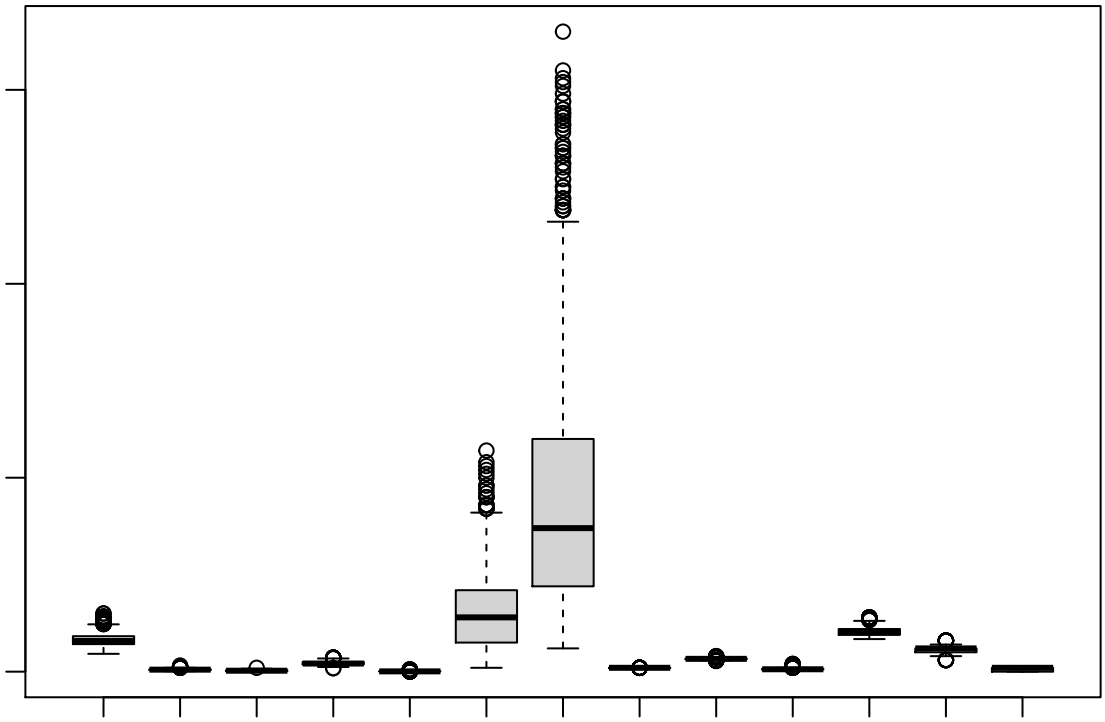
```
par(mar=c(1,1,1,1)) # to fix boxplot knit processing issues

# 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
```

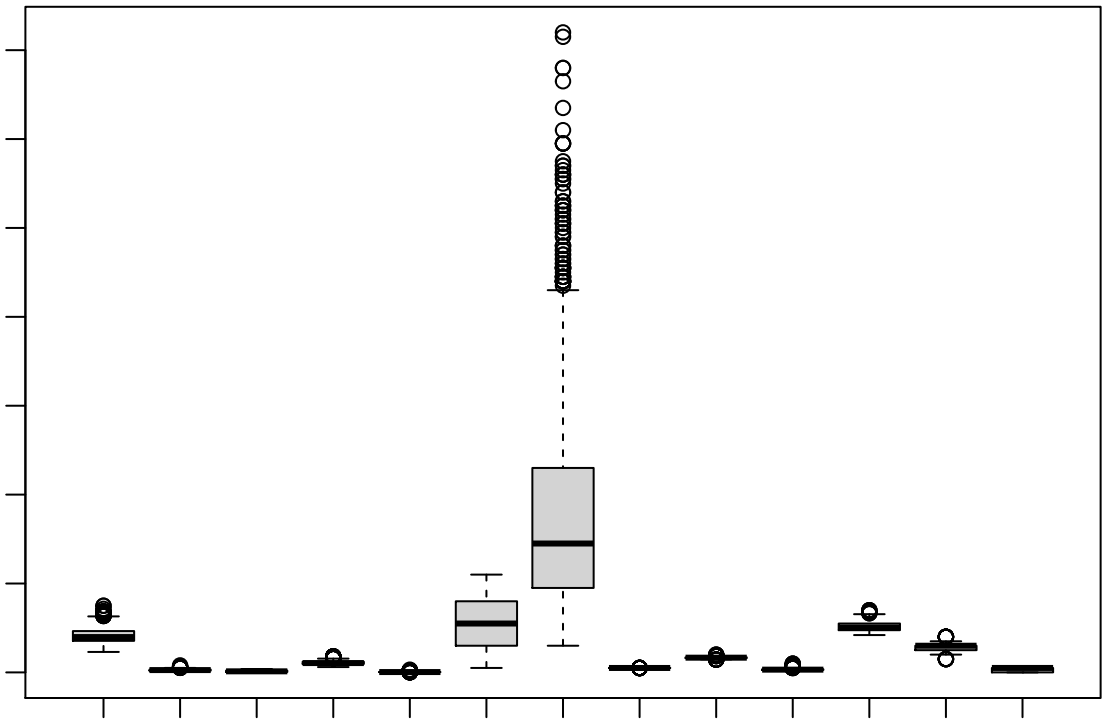
```
# Check for missing values in data set
wine %>% na.omit() %>% count() # there are no missing values
```

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

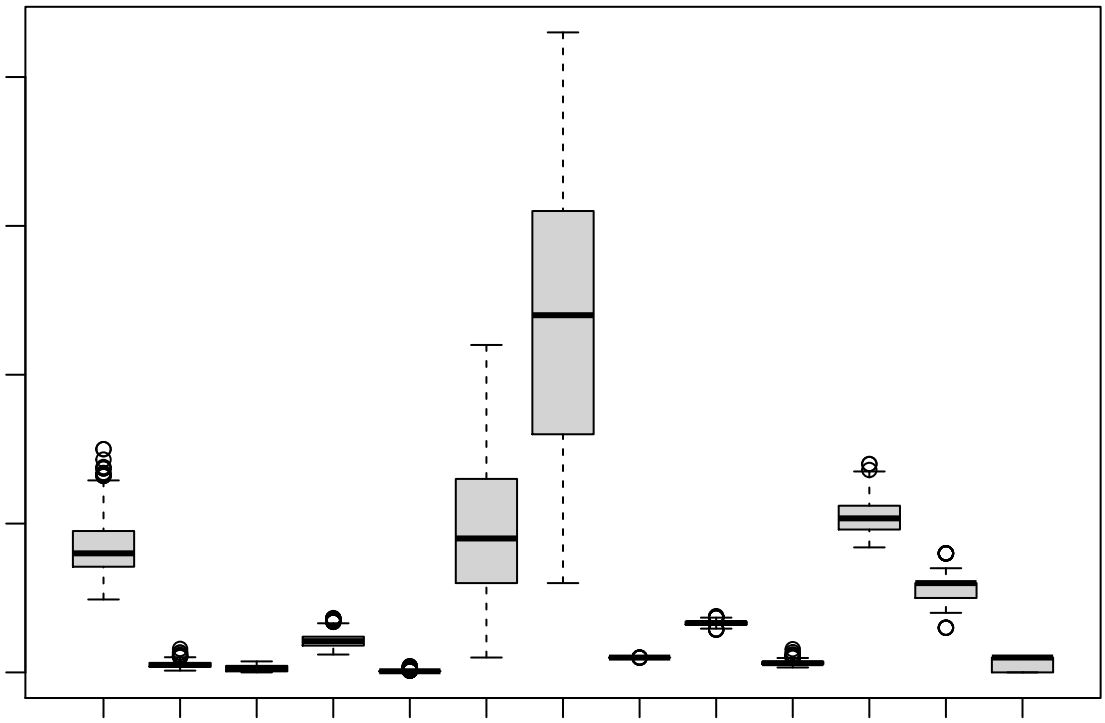
```
# Removing outliers for residual sugar:
Q <- quantile(wine$residual.sugar, probs=c(.25, .75), na.rm = FALSE)
iqr_rs <- IQR(wine$residual.sugar)
up_rs <-  Q[2]+1.5*iqr_rs # Upper Range
low_rs <- Q[1]-1.5*iqr_rs # Lower Range
eliminated_rs <- subset(wine, wine$residual.sugar > (Q[1] - 1.5*iqr_rs) &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



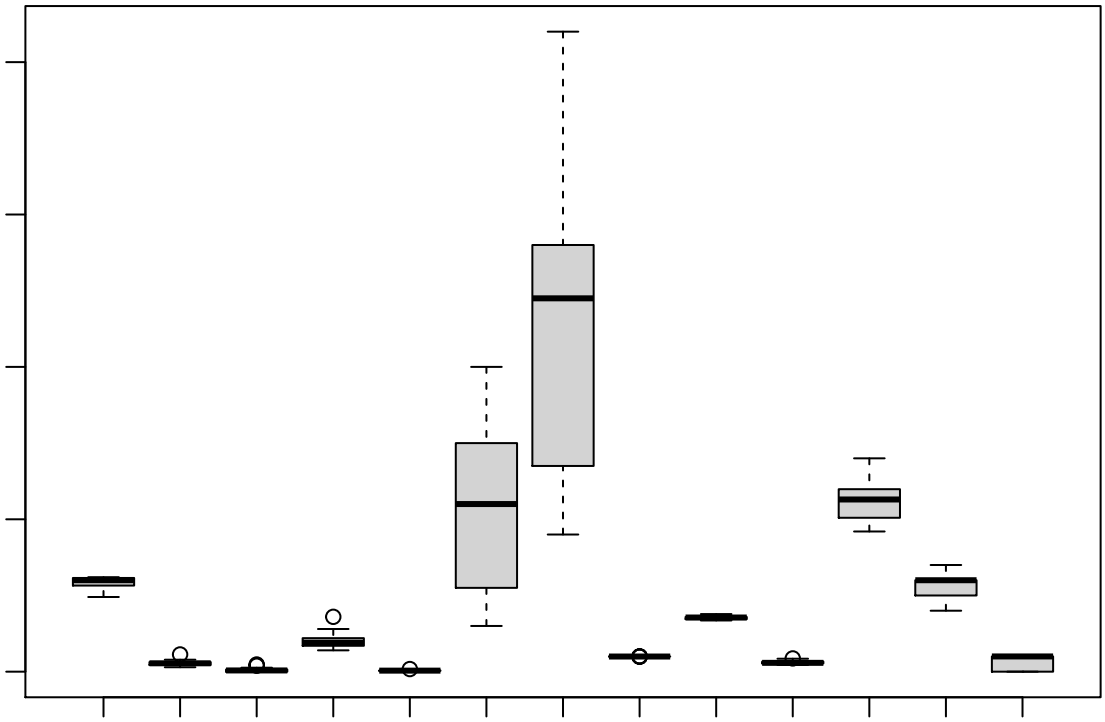
```r
#Removing outliers for free.sulfur.dioxide:
Q2 <- quantile(wine$free.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_fs <- IQR(eliminated_rs$free.sulfur.dioxide)
up_fs <-  Q2[2]+1.5*iqr_fs # Upper Range
low_fs <- Q2[1]-1.5*iqr_fs # Lower Range
eliminated_fs <- subset(eliminated_rs, eliminated_rs$free.sulfur.dioxide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



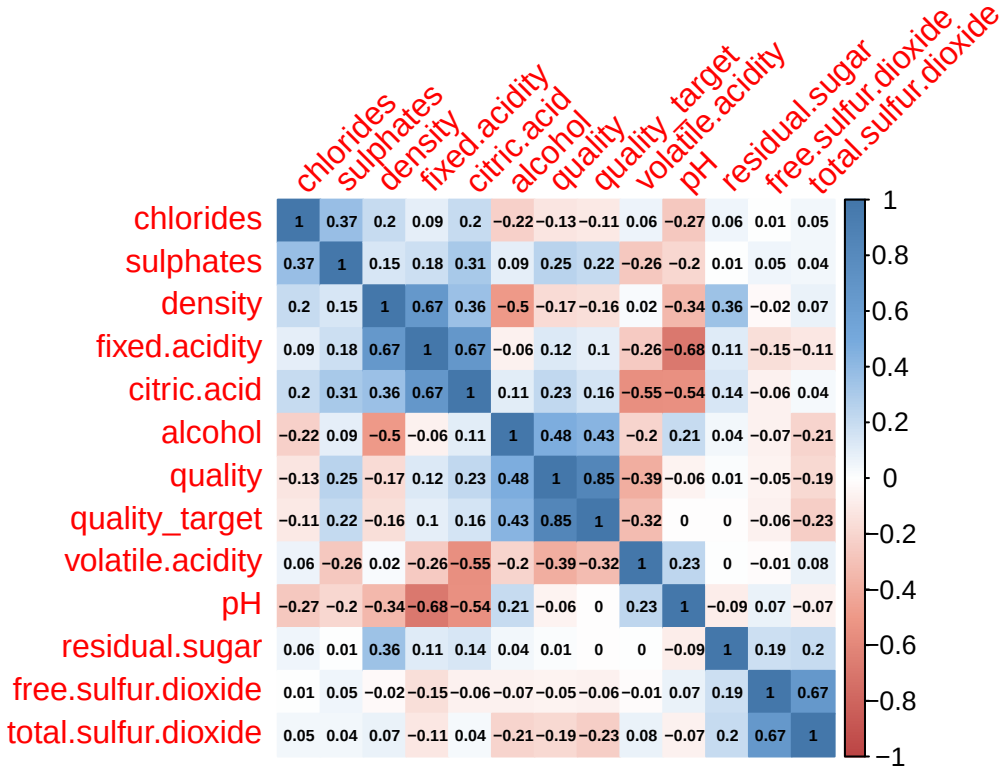
```r
#Removing outliers for fixed.acidity:
Q4 <- quantile(wine$fixed.acidity, probs=c(.25, .75), na.rm = FALSE)
iqr_fa <- IQR(eliminated_ts$fixed.acidity)
up_fa <-  Q[2]+1.5*iqr_fa # Upper Range
low_fa <- Q[1]-1.5*iqr_fa # Lower Range
eliminated_fa <- subset(eliminated_ts, eliminated_ts$fixed.acidity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



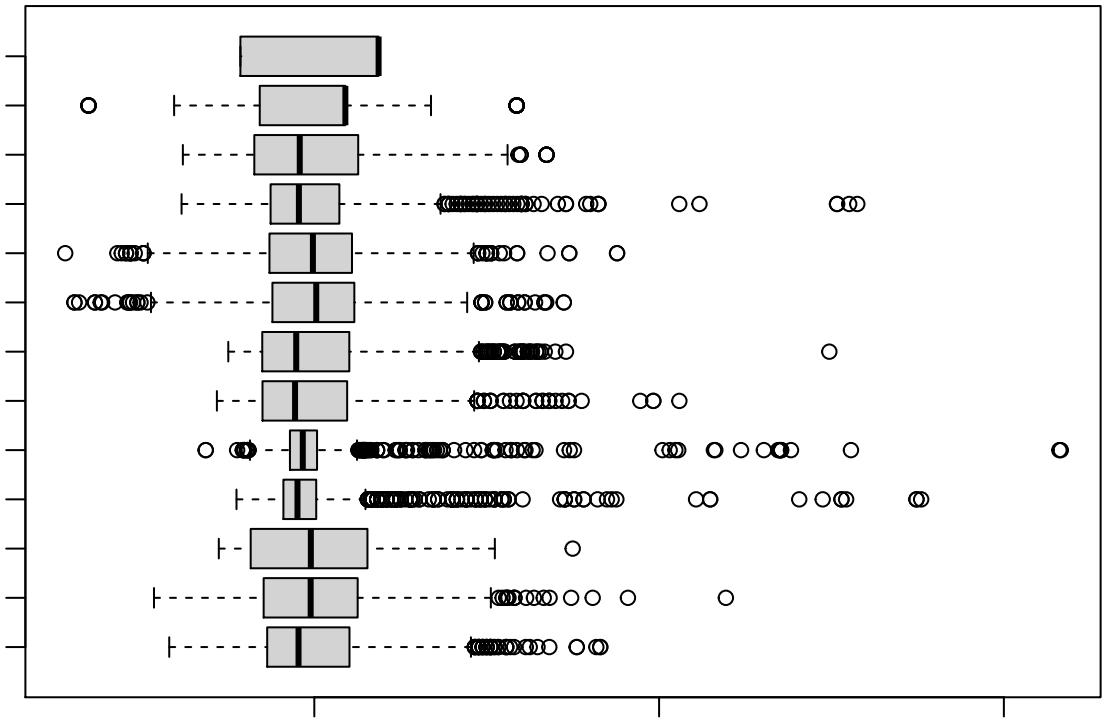
```r
new_wine_data <- eliminated_fa

# 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 = train_transformed)

summary(lmodel1)
```

```
##
## Call:
## lm(formula = quality_target ~ volatile.acidity + sulphates +
##     alcohol, data = train_transformed)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.01994 -0.70808 -0.02156  0.77597  2.05849
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -3.395e-15  2.409e-02   0.000        1
## volatile.acidity -1.937e-01  2.519e-02  -7.691 2.91e-14 ***
## sulphates         1.402e-01  2.484e-02   5.646 2.03e-08 ***
```

```
## alcohol           3.871e-01  2.457e-02  15.753  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8622 on 1277 degrees of freedom
## Multiple R-squared:  0.2584, Adjusted R-squared:  0.2567
## F-statistic: 148.3 on 3 and 1277 DF,  p-value: < 2.2e-16
```

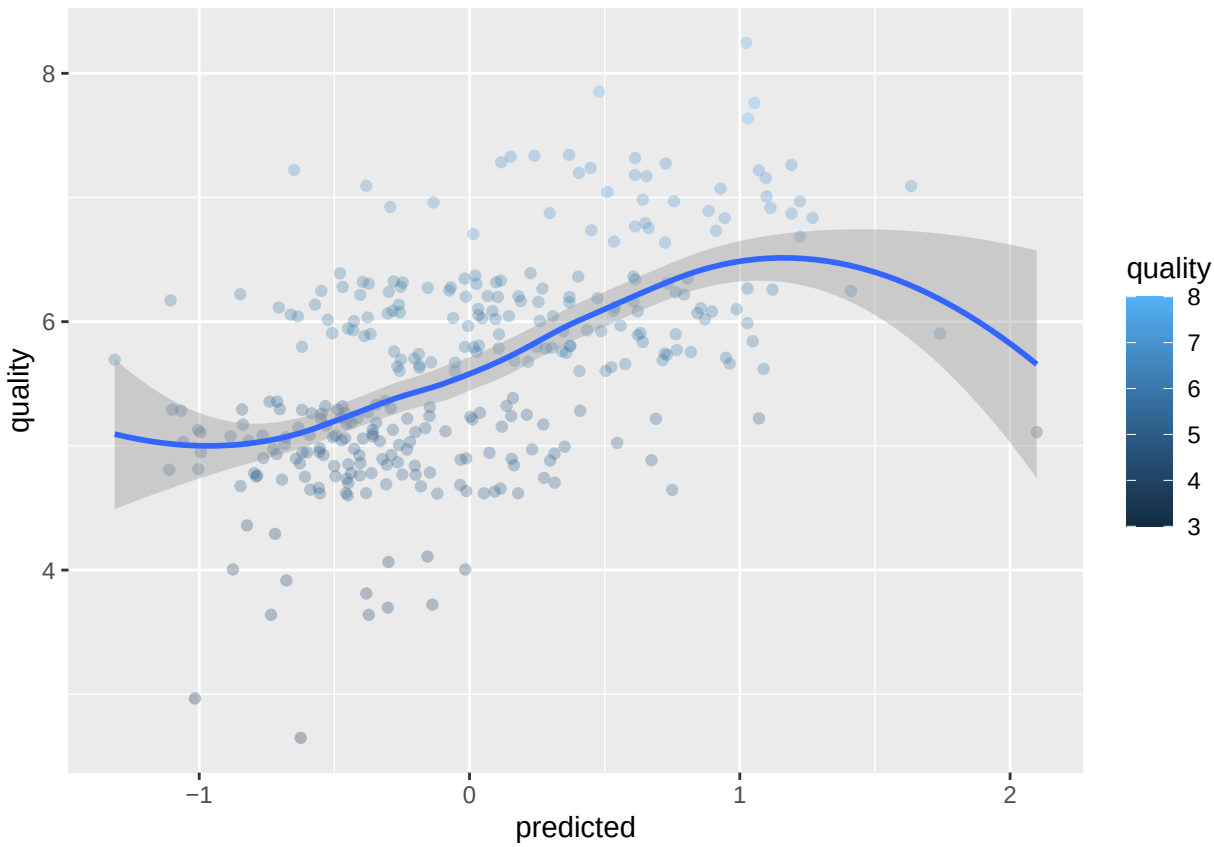
```r
# Model using "quality" as target variable
lmodel2 <- lm(quality~ volatile.acidity + sulphates + alcohol, data = train_transformed)

summary(lmodel2)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + sulphates + alcohol,
##     data = train_transformed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3710 -0.4761 -0.0750  0.5986  2.7090
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.314e-16  2.280e-02   0.000        1
## volatile.acidity -2.709e-01  2.384e-02 -11.362   <2e-16 ***
## sulphates         1.384e-01  2.351e-02   5.888    5e-09 ***
## alcohol           4.150e-01  2.326e-02  17.844   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.816 on 1277 degrees of freedom
## Multiple R-squared:  0.3358, Adjusted R-squared:  0.3342
## F-statistic: 215.2 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 = test_transformed)) -> df

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

```
##       fit                lwr               upr
##  Min.   :-1.31279   Min.   :-2.9185   Min.   :0.293
##  1st Qu.:-0.45010   1st Qu.:-2.0521   1st Qu.:1.152
##  Median :-0.07221   Median :-1.6756   Median :1.531
##  Mean   : 0.00000   Mean   :-1.6032   Mean   :1.603
##  3rd Qu.: 0.40626   3rd Qu.:-1.1957   3rd Qu.:2.008
##  Max.   : 2.09848   Max.   : 0.4861   Max.   :3.711
```

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

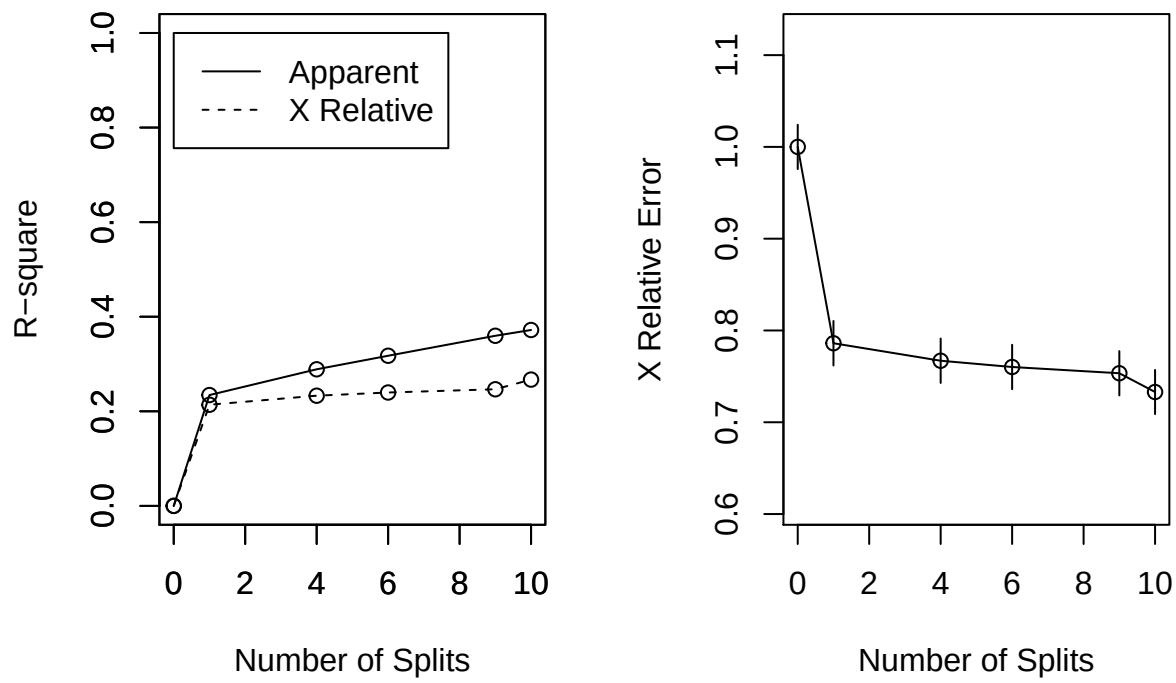



```
# The scatter plot supports the summary of the predicted interval, in the ranges of the fit,
# lower, and upper ranges. The R-squared value of 0.3283 of the model, indicates that this
# information can be predicted 33% of the time, with the data available, for the variance
# of the information.
```

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        free.sulfur.dioxide
## [4] sulphates            total.sulfur.dioxide volatile.acidity
##
## Root node error: 734/1281 = 0.57299
##
## n= 1281
##
##         CP nsplit rel error  xerror     xstd
## 1 0.234332      0   1.00000 1.00000 0.024120
## 2 0.018165      1   0.76567 0.78610 0.024261
## 3 0.014305      4   0.71117 0.76703 0.024202
## 4 0.014078      6   0.68256 0.76022 0.024178
## 5 0.012262      9   0.64033 0.75341 0.024152
## 6 0.010000     10   0.62807 0.73297 0.024067
```

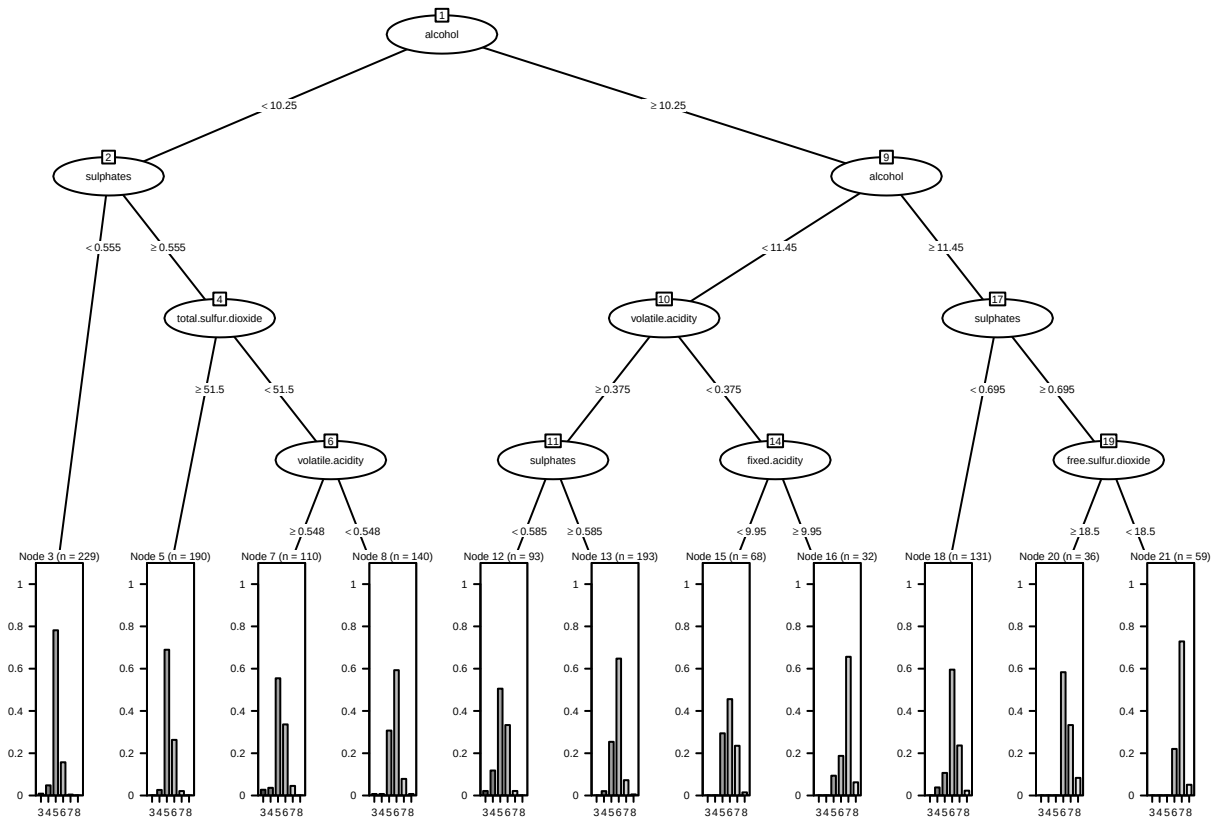



```r
# Results
rpartTree2
```

```
##
## 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 < 10.25
## |   |   [3] sulphates < 0.555: 5 (n = 229, err = 21.8%)
```

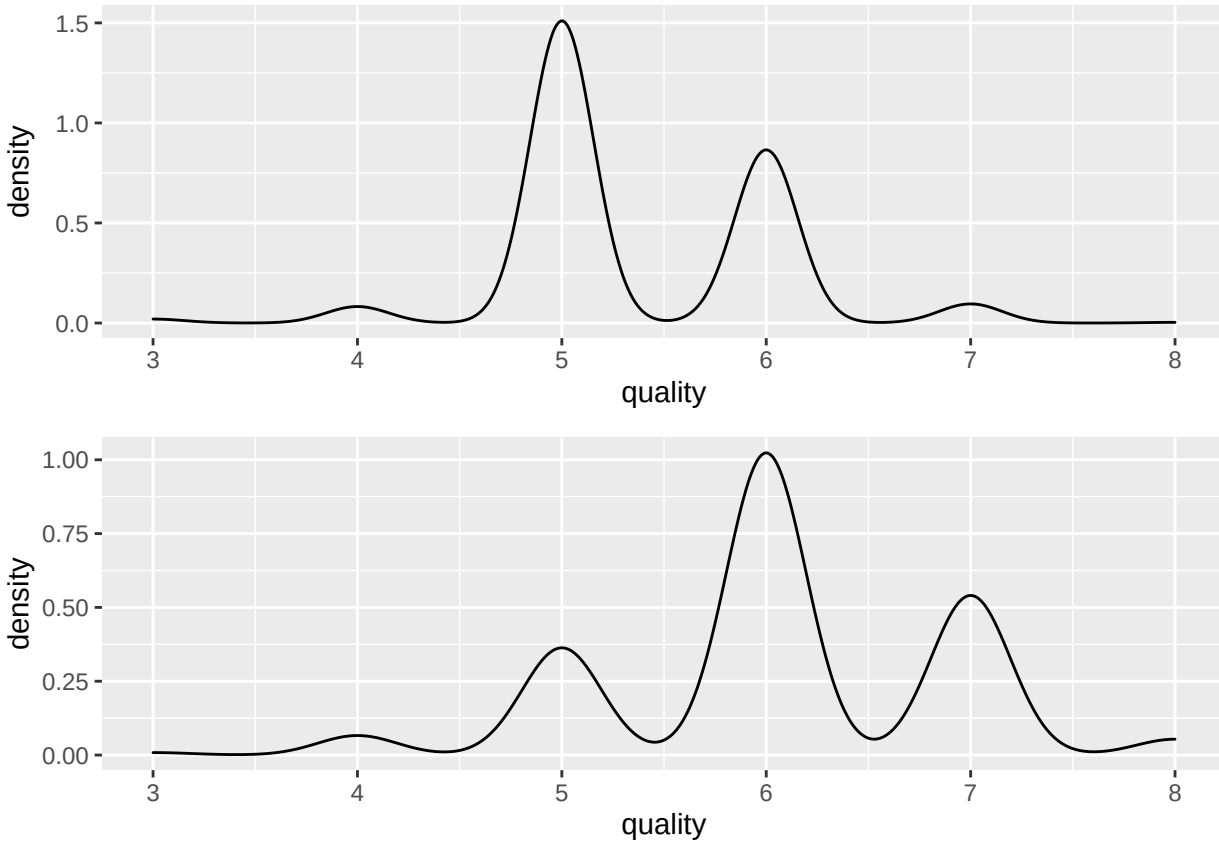
```
## |   |   [4] sulphates >= 0.555
## |   |   |   [5] total.sulfur.dioxide >= 51.5: 5 (n = 190, err = 31.1%)
## |   |   |   [6] total.sulfur.dioxide < 51.5
## |   |   |   |   [7] volatile.acidity >= 0.5475: 5 (n = 110, err = 44.5%)
## |   |   |   |   [8] volatile.acidity < 0.5475: 6 (n = 140, err = 40.7%)
## |   [9] alcohol >= 10.25
## |   |   [10] alcohol < 11.45
## |   |   |   [11] volatile.acidity >= 0.375
## |   |   |   |   [12] sulphates < 0.585: 5 (n = 93, err = 49.5%)
## |   |   |   |   [13] sulphates >= 0.585: 6 (n = 193, err = 35.2%)
## |   |   |   [14] volatile.acidity < 0.375
## |   |   |   |   [15] fixed.acidity < 9.95: 6 (n = 68, err = 54.4%)
## |   |   |   |   [16] fixed.acidity >= 9.95: 7 (n = 32, err = 34.4%)
## |   |   [17] alcohol >= 11.45
## |   |   |   [18] sulphates < 0.695: 6 (n = 131, err = 40.5%)
## |   |   |   [19] sulphates >= 0.695
## |   |   |   |   [20] free.sulfur.dioxide >= 18.5: 6 (n = 36, err = 41.7%)
## |   |   |   |   [21] free.sulfur.dioxide < 18.5: 7 (n = 59, err = 27.1%)
##
## Number of inner nodes:    10
## Number of terminal nodes: 11
```

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




```r
# 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)
```

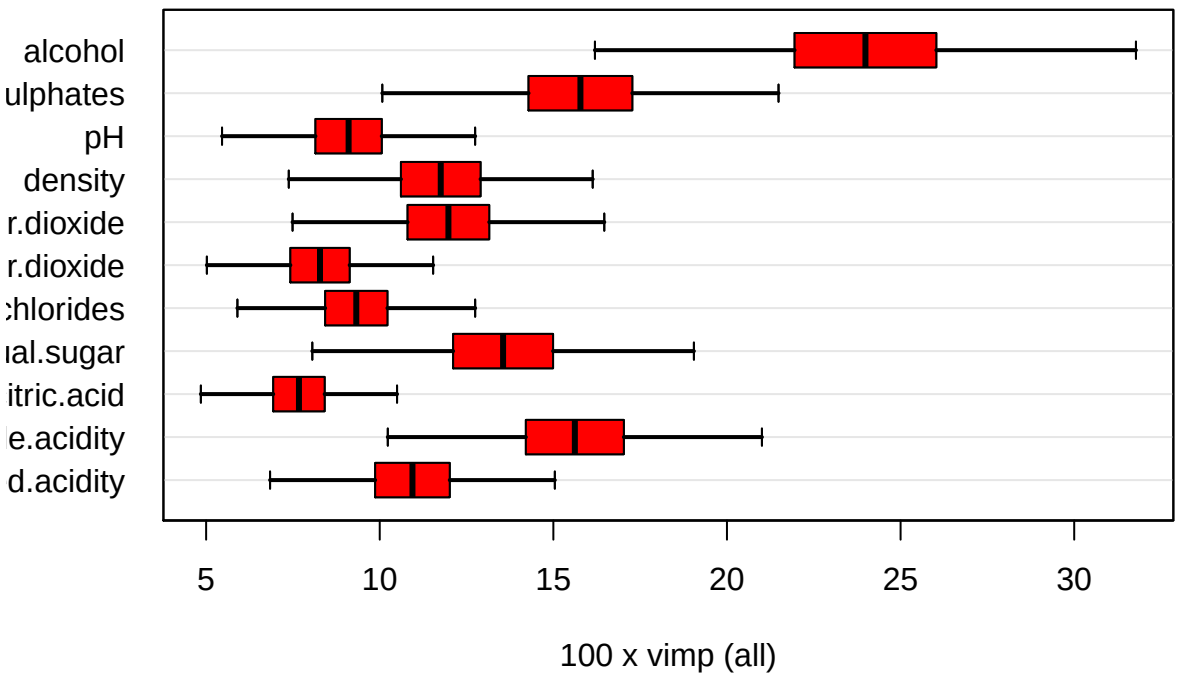
```
##                          Sample size: 1281
##            Frequency of class labels: 8, 41, 547, 511, 160, 14
##                      Number of trees: 500
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 251.55
## No. of variables tried at each split: 4
##               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: 0.06888955
##         (OOB) Normalized Brier score: 0.49600479
##                              (OOB) AUC: 0.81308972
##    (OOB) Requested performance error: 0.28883685, 1, 1, 0.19561243, 0.26027397, 0.43125, 0.85714286
##
## Confusion matrix:
##
##           predicted
##   observed 3 4   5   6  7 8 class.error
##          3 0 1   5   2  0 0      1.0000
##          4 1 0  25  14  1 0      1.0000
##          5 1 2 440 100  4 0      0.1956
##          6 0 1 110 376 24 0      0.2642
##          7 0 0   4  64 91 1      0.4312
##          8 0 0   0   8  4 2      0.8571
##
##       (OOB) Misclassification rate: 0.2903981
```

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

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

```
##                          lower      mean     upper       pvalue signif
## fixed.acidity         7.822284 10.942251 14.062217 3.122965e-12   TRUE
## volatile.acidity     11.519656 15.619108 19.718560 4.084881e-14   TRUE
## citric.acid           5.523488  7.673461  9.823434 1.323452e-12   TRUE
## residual.sugar        9.372322 13.552866 17.733411 1.049014e-10   TRUE
## chlorides             6.719966  9.324841 11.929716 1.139834e-12   TRUE
## free.sulfur.dioxide   5.800015  8.279406 10.758796 2.976994e-11   TRUE
## total.sulfur.dioxide  8.562416 11.977919 15.393422 3.133360e-12   TRUE
## density               8.426709 11.756688 15.086667 2.261907e-12   TRUE
## pH                    6.330221  9.103185 11.876148 6.204633e-11   TRUE
## sulphates            11.438171 15.779618 20.121064 5.250561e-13   TRUE
## alcohol              18.060354 23.988110 29.915867 1.082823e-15   TRUE
```

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




```r
# Predict
# 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: 251.55
##           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: 0.07125171
##                Normalized Brier score: 0.51301232
##                                   AUC: 0.83566831
##          Requested performance error: 0.31446541, 1, 1, 0.20149254, 0.27559055, 0.51282051, 1
##
## Confusion matrix:
##
##           predicted
##   observed 3 4   5  6  7 8 class.error
##          3 0 0   2  0  0 0      1.0000
##          4 0 0  11  1  0 0      1.0000
##          5 0 0 107 23  4 0      0.2015
##          6 0 0  24 92 11 0      0.2756
##          7 0 0   3 15 19 2      0.5128
##          8 0 0   0  3  1 0      1.0000
##
##            Misclassification error: 0.3144654
```

Partial Least Squares

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

set.seed(4)
pls_wine <- train(quality~ volatile.acidity + chlorides + total.sulfur.dioxide +
               sulphates + alcohol, data = train_transformed,
                  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)
## 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      0.8092217  0.3487899  0.6321922
##   2      0.8084946  0.3500058  0.6313616
##   3      0.8085600  0.3499713  0.6312280
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 2.
```

Mars Tuning

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

mars_wine
```

```
## Selected 15 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 14 (additive model)
## GCV 0.6320845    RSS 773.4543    GRSq 0.3684089    RSq 0.3957388
```

```
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##             data=train_transformed)
##
##                                  coefficients
## (Intercept)                        -2.5033210
## h(2.67395-volatile.acidity)         0.1991889
```

```
## h(volatile.acidity-2.67395)       -0.6668072
## h(chlorides- -0.974217)            3.1754977
## h(-0.0840505-chlorides)            2.9589079
## h(chlorides- -0.0840505)          -3.2562975
## h(total.sulfur.dioxide-1.4629)    -0.3625056
## h(2.54092-total.sulfur.dioxide)    0.0681306
## h(total.sulfur.dioxide-2.54092)    0.7874583
## h(sulphates-0.891645)              5.1631418
## h(0.950322-sulphates)             -0.4029818
## h(sulphates-0.950322)             -5.9799150
## h(sulphates-1.77179)               0.8950301
## h(alcohol-0.730876)                0.2508004
## h(1.86099-alcohol)                -0.2972706
##
## Selected 15 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 14 (additive model)
## GCV 0.6320845    RSS 773.4543    GRSq 0.3684089    RSq 0.3957388
```

```r
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 =train_transformed,
                       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      0.8772197  0.2325876  0.6968713
##   1        3      0.8377194  0.3005966  0.6561111
##   1        4      0.8098598  0.3462154  0.6318368
##   1        5      0.8120630  0.3466304  0.6309034
##   1        6      0.8192711  0.3390515  0.6325261
##   1        7      0.8256888  0.3340875  0.6344632
##   1        8      0.8249701  0.3333771  0.6350216
##   1        9      0.8220832  0.3374982  0.6336243
##   1       10      0.8233786  0.3352357  0.6334661
##   1       11      0.8245091  0.3336547  0.6344863
##   1       12      0.8274942  0.3319896  0.6358688
```

```
##   1        13       0.8286668  0.3303074  0.6368753
##   1        14       0.8334899  0.3277271  0.6383487
##   1        15       0.8337138  0.3276716  0.6383209
##   1        16       0.8337138  0.3276716  0.6383209
##   1        17       0.8337138  0.3276716  0.6383209
##   1        18       0.8337138  0.3276716  0.6383209
##   1        19       0.8337138  0.3276716  0.6383209
##   1        20       0.8337138  0.3276716  0.6383209
##   1        21       0.8337138  0.3276716  0.6383209
##   1        22       0.8337138  0.3276716  0.6383209
##   1        23       0.8337138  0.3276716  0.6383209
##   1        24       0.8337138  0.3276716  0.6383209
##   1        25       0.8337138  0.3276716  0.6383209
##   1        26       0.8337138  0.3276716  0.6383209
##   1        27       0.8337138  0.3276716  0.6383209
##   1        28       0.8337138  0.3276716  0.6383209
##   1        29       0.8337138  0.3276716  0.6383209
##   1        30       0.8337138  0.3276716  0.6383209
##   1        31       0.8337138  0.3276716  0.6383209
##   1        32       0.8337138  0.3276716  0.6383209
##   1        33       0.8337138  0.3276716  0.6383209
##   1        34       0.8337138  0.3276716  0.6383209
##   1        35       0.8337138  0.3276716  0.6383209
##   1        36       0.8337138  0.3276716  0.6383209
##   1        37       0.8337138  0.3276716  0.6383209
##   1        38       0.8337138  0.3276716  0.6383209
##   2         2       0.8798762  0.2279516  0.6983886
##   2         3       0.8400993  0.2973075  0.6617128
##   2         4       0.8135537  0.3413551  0.6336905
##   2         5       0.8072307  0.3520153  0.6286168
##   2         6       0.8064809  0.3533716  0.6261324
##   2         7       0.8035546  0.3583334  0.6231224
##   2         8       0.8057935  0.3573503  0.6232250
##   2         9       0.8090957  0.3519144  0.6260984
##   2        10       0.8144111  0.3474315  0.6286740
##   2        11       0.8188760  0.3415895  0.6321054
##   2        12       0.8246988  0.3375539  0.6335220
##   2        13       0.8250409  0.3376475  0.6339500
##   2        14       0.8254534  0.3371904  0.6340676
##   2        15       0.8247307  0.3384005  0.6335939
##   2        16       0.8253053  0.3378067  0.6338244
##   2        17       0.8263724  0.3364452  0.6344826
##   2        18       0.8263724  0.3364452  0.6344826
##   2        19       0.8263724  0.3364452  0.6344826
##   2        20       0.8263724  0.3364452  0.6344826
##   2        21       0.8263724  0.3364452  0.6344826
##   2        22       0.8263724  0.3364452  0.6344826
##   2        23       0.8263724  0.3364452  0.6344826
##   2        24       0.8263724  0.3364452  0.6344826
##   2        25       0.8263724  0.3364452  0.6344826
##   2        26       0.8263724  0.3364452  0.6344826
##   2        27       0.8263724  0.3364452  0.6344826
##   2        28       0.8263724  0.3364452  0.6344826
##   2        29       0.8263724  0.3364452  0.6344826
```

```
##   2       30      0.8263724  0.3364452  0.6344826
##   2       31      0.8263724  0.3364452  0.6344826
##   2       32      0.8263724  0.3364452  0.6344826
##   2       33      0.8263724  0.3364452  0.6344826
##   2       34      0.8263724  0.3364452  0.6344826
##   2       35      0.8263724  0.3364452  0.6344826
##   2       36      0.8263724  0.3364452  0.6344826
##   2       37      0.8263724  0.3364452  0.6344826
##   2       38      0.8263724  0.3364452  0.6344826
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nprune = 7 and degree = 2.
```

KNN Neighbors

```r
set.seed(4)

knn_wine <- train(quality~ volatile.acidity + chlorides + total.sulfur.dioxide +
               sulphates + alcohol, data =train_transformed,
               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  0.9502984  0.2991612  0.5486131
##    2  0.8844451  0.3015216  0.6061430
##    3  0.8625020  0.3075591  0.6185690
##    4  0.8344760  0.3294039  0.6208217
##    5  0.8178180  0.3484426  0.6151643
##    6  0.8107219  0.3563340  0.6138202
##    7  0.8028777  0.3650241  0.6093655
##    8  0.8010857  0.3660416  0.6128788
##    9  0.7963709  0.3719588  0.6106432
##   10  0.7984982  0.3686980  0.6136400
##   11  0.7959547  0.3714267  0.6135750
##   12  0.7927222  0.3761906  0.6112068
##   13  0.7915539  0.3770617  0.6120356
##   14  0.7924647  0.3752828  0.6138970
##   15  0.7918687  0.3769313  0.6134381
```

```
##   16  0.7944776  0.3722746  0.6161107
##   17  0.7964967  0.3692669  0.6175261
##   18  0.7961766  0.3694947  0.6173256
##   19  0.7966202  0.3682022  0.6182923
##   20  0.7941559  0.3722440  0.6164731
##   21  0.7950309  0.3707509  0.6176403
##   22  0.7947911  0.3710394  0.6178171
##   23  0.7944354  0.3714830  0.6180363
##   24  0.7935100  0.3731100  0.6174050
##   25  0.7939866  0.3720788  0.6185562
##   26  0.7937022  0.3728013  0.6187629
##   27  0.7931459  0.3740204  0.6182039
##   28  0.7930188  0.3742001  0.6177695
##   29  0.7930460  0.3741266  0.6174238
##   30  0.7915465  0.3765013  0.6161953
##   31  0.7913394  0.3767309  0.6167973
##   32  0.7921529  0.3753747  0.6177052
##   33  0.7930361  0.3742365  0.6189554
##   34  0.7939773  0.3725938  0.6194651
##   35  0.7940947  0.3727952  0.6195313
##   36  0.7936816  0.3736685  0.6187028
##   37  0.7946011  0.3722985  0.6197477
##   38  0.7954886  0.3710285  0.6206054
##   39  0.7945168  0.3728000  0.6206926
##   40  0.7952116  0.3717022  0.6220040
##   41  0.7948058  0.3725069  0.6213238
##   42  0.7952223  0.3719802  0.6216250
##   43  0.7959278  0.3708367  0.6222182
##   44  0.7964459  0.3700022  0.6233678
##   45  0.7963313  0.3703754  0.6232731
##   46  0.7959668  0.3710433  0.6240970
##   47  0.7956386  0.3717463  0.6242420
##   48  0.7947695  0.3733183  0.6239088
##   49  0.7944822  0.3740503  0.6236121
##   50  0.7949465  0.3733624  0.6241523
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 31.
```

SVM

```r
set.seed(4)

subset(train_transformed, select = -c(quality_target, quality)) -> predictors
train_transformed$quality -> quality

svmTune <- train(predictors, quality,
                 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
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1153, 1153, 1153, 1154, 1153, 1152, ...
## Resampling results across tuning parameters:
##
##   C     RMSE       Rsquared   MAE
##   0.25  0.7933984  0.3755164  0.5910630
##   0.50  0.7802254  0.3948300  0.5759678
##   1.00  0.7742886  0.4041230  0.5688540
##   2.00  0.7745598  0.4059322  0.5662985
##   4.00  0.7804442  0.4040349  0.5674192
##
## Tuning parameter 'sigma' was held constant at a value of 0.1039459
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were sigma = 0.1039459 and C = 1.
```