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.

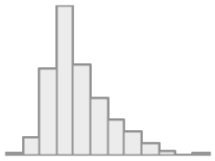
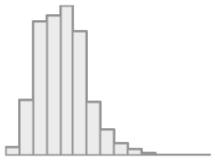
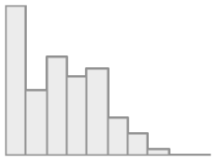
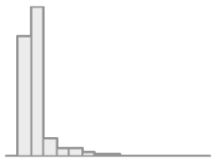
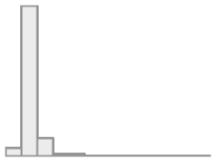
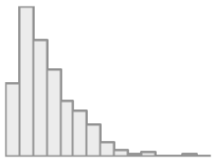
```
wine <- read.csv(
  url("https://raw.githubusercontent.com/OscarG-DataSci/ADS503/main/winequality-red.csv")
                      , header = TRUE)
```

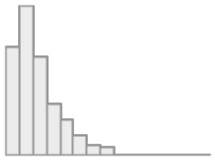
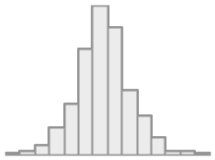
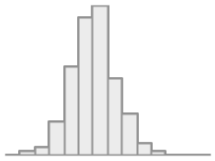
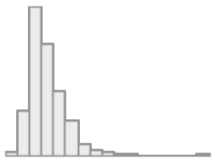
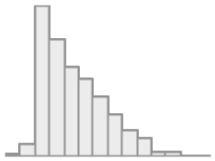
Data Summary

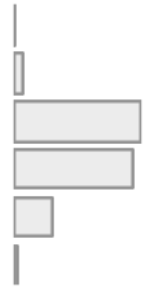
```
dfSummary(wine,
          plain.ascii  = FALSE,
          style        = "grid",
          graph.magnif = 0.75,
          valid.col    = FALSE,
          tmp.img.dir  = "/tmp")
```

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

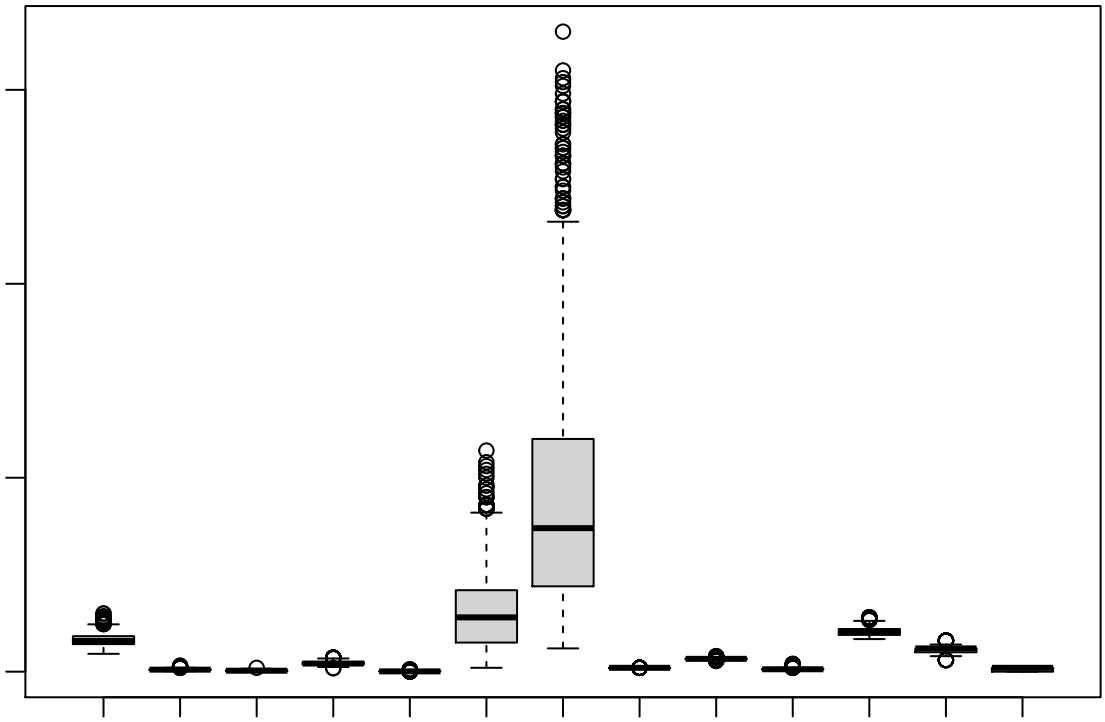
```
par(mar=c(1,1,1,1)) # to fix boxplot knit processing issues

# 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
```

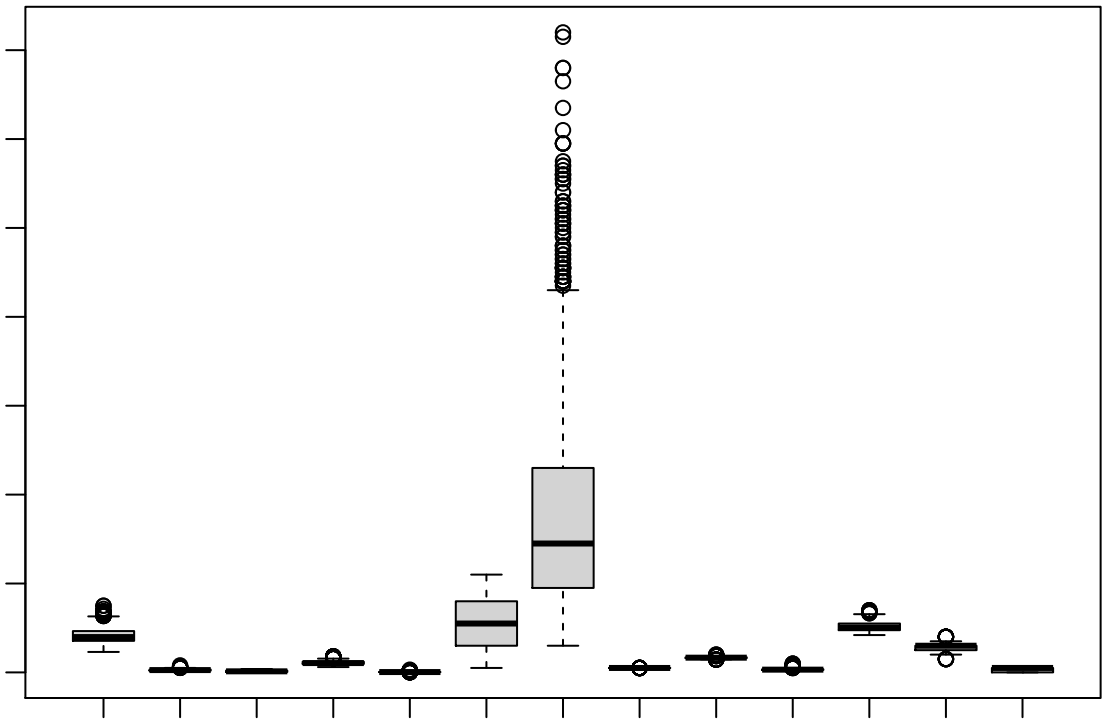
```
# Check for missing values in data set
wine %>% na.omit() %>% count() # there are no missing values
```

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

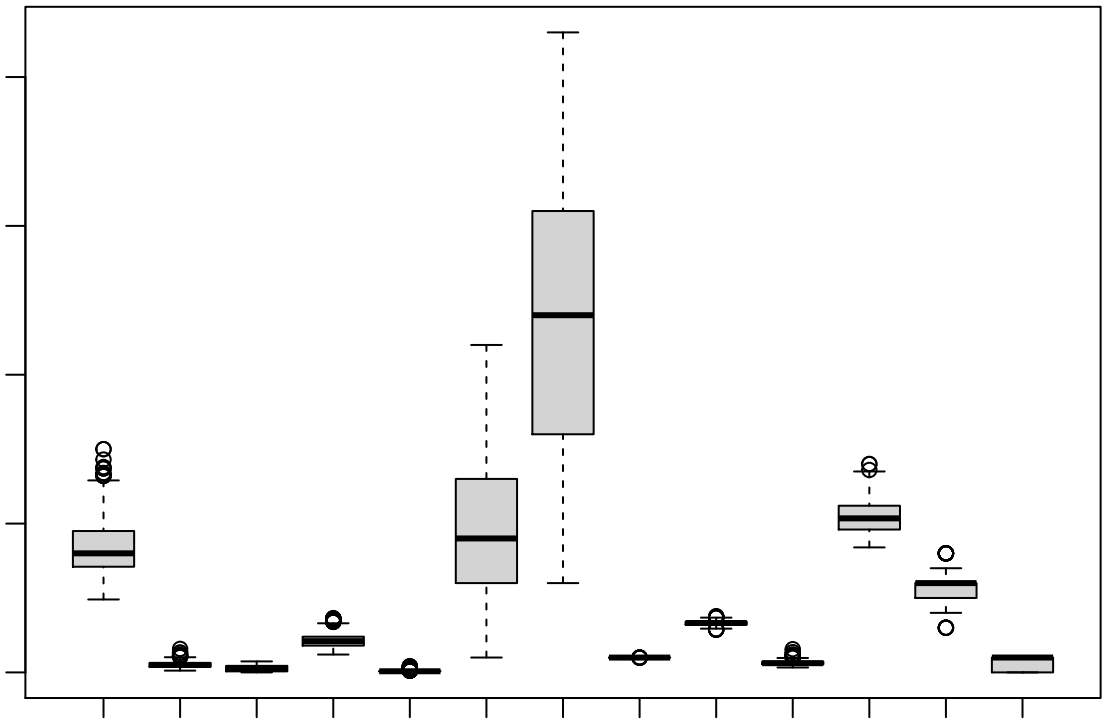
```
# Removing outliers for residual sugar:
Q <- quantile(wine$residual.sugar, probs=c(.25, .75), na.rm = FALSE)
iqr_rs <- IQR(wine$residual.sugar)
up_rs <-  Q[2]+1.5*iqr_rs # Upper Range
low_rs <- Q[1]-1.5*iqr_rs # Lower Range
eliminated_rs <- subset(wine, wine$residual.sugar > (Q[1] - 1.5*iqr_rs) &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



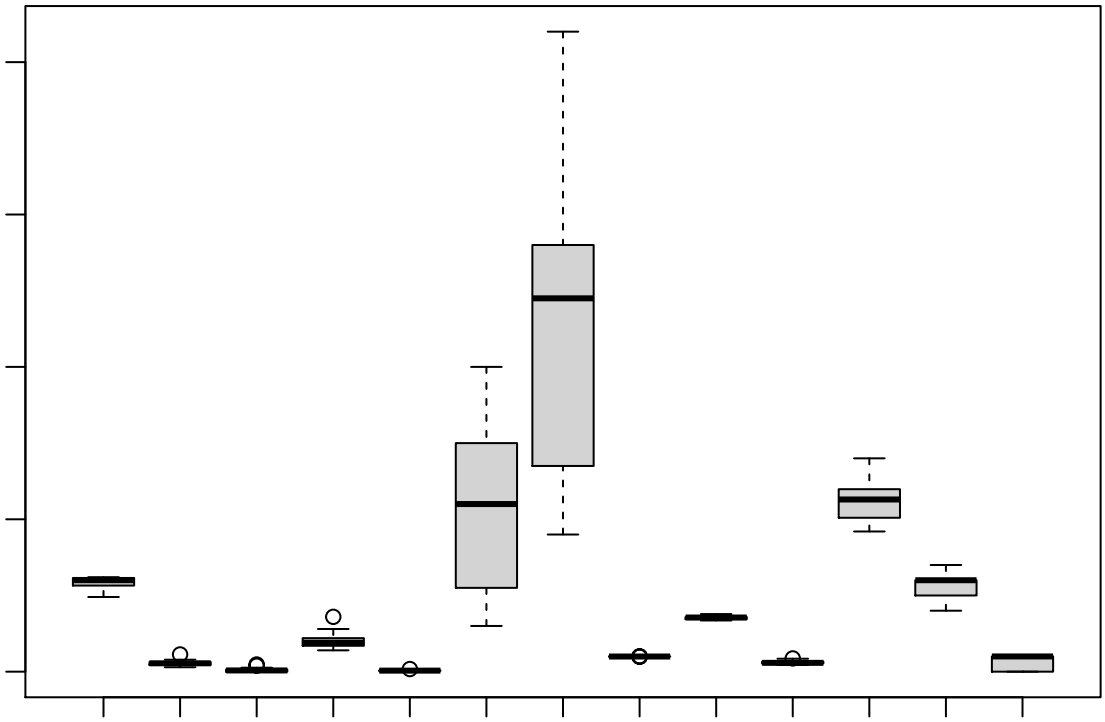
```r
#Removing outliers for free.sulfur.dioxide:
Q2 <- quantile(wine$free.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_fs <- IQR(eliminated_rs$free.sulfur.dioxide)
up_fs <-  Q2[2]+1.5*iqr_fs # Upper Range
low_fs <- Q2[1]-1.5*iqr_fs # Lower Range
eliminated_fs <- subset(eliminated_rs, eliminated_rs$free.sulfur.dioxide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



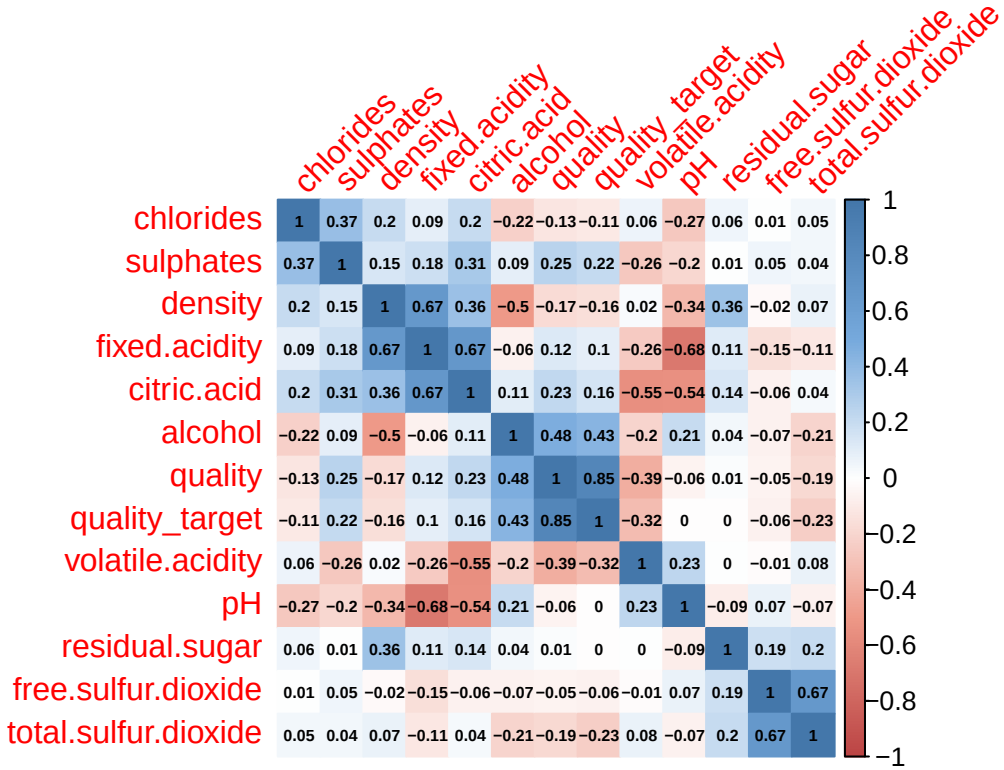
```r
#Removing outliers for fixed.acidity:
Q4 <- quantile(wine$fixed.acidity, probs=c(.25, .75), na.rm = FALSE)
iqr_fa <- IQR(eliminated_ts$fixed.acidity)
up_fa <-  Q[2]+1.5*iqr_fa # Upper Range
low_fa <- Q[1]-1.5*iqr_fa # Lower Range
eliminated_fa <- subset(eliminated_ts, eliminated_ts$fixed.acidity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



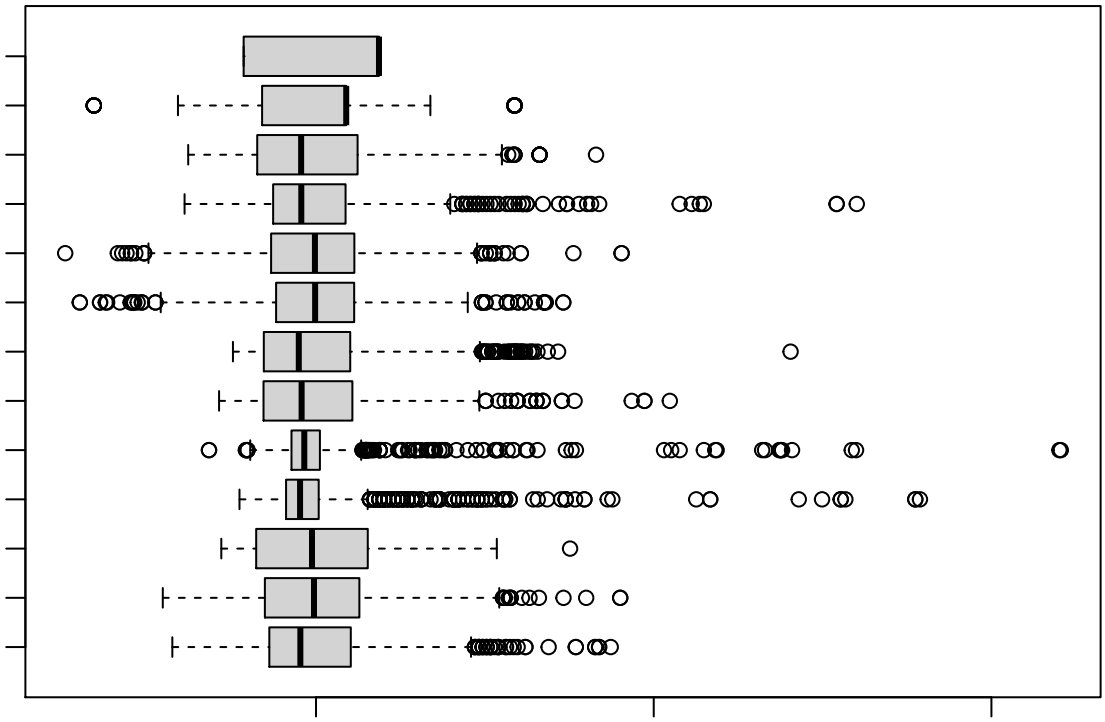
```r
new_wine_data <- eliminated_fa

# 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
## -3.03401 -0.69442 -0.04274  0.73427  2.06407
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -4.524e-15  2.396e-02   0.000        1
## volatile.acidity -1.964e-01  2.522e-02  -7.788 1.40e-14 ***
## sulphates         1.289e-01  2.479e-02   5.198 2.34e-07 ***
```

```
## alcohol           3.955e-01  2.453e-02  16.126  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8574 on 1277 degrees of freedom
## Multiple R-squared:  0.2666, Adjusted R-squared:  0.2649
## F-statistic: 154.7 on 3 and 1277 DF,  p-value: < 2.2e-16
```

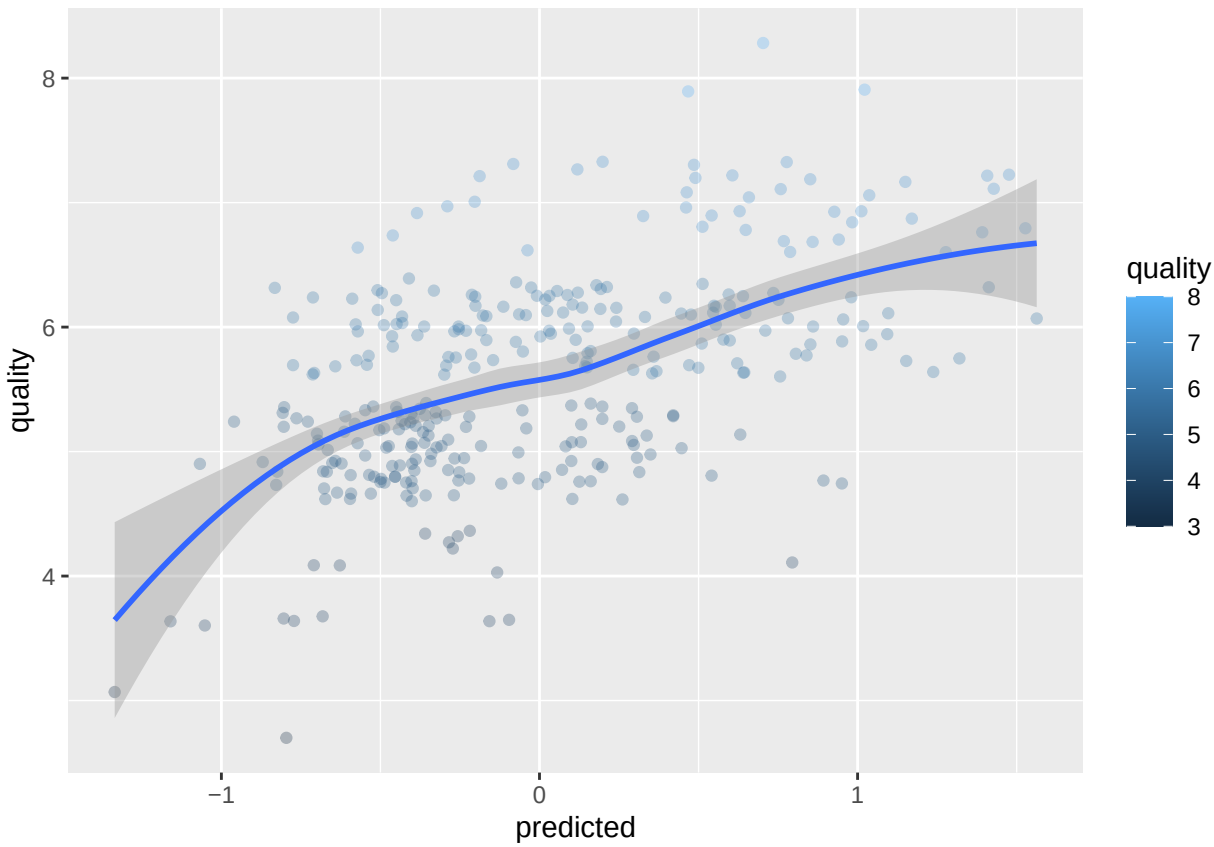
```r
# Model using "quality" as target variable
lmodel2 <- lm(quality~ volatile.acidity + sulphates + alcohol, data = train_transformed)

summary(lmodel2)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + sulphates + alcohol,
##     data = train_transformed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3574 -0.4662 -0.0793  0.5930  2.7763
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.542e-15  2.274e-02   0.000        1
## volatile.acidity -2.337e-01  2.394e-02  -9.760  < 2e-16 ***
## sulphates         1.440e-01  2.353e-02   6.117 1.26e-09 ***
## alcohol           4.381e-01  2.328e-02  18.815  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.814 on 1277 degrees of freedom
## Multiple R-squared:  0.339,  Adjusted R-squared:  0.3375
## F-statistic: 218.3 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.   :-1.3354   Min.   :-2.95538   Min.   :0.2847
##  1st Qu.:-0.4413   1st Qu.:-2.04038   1st Qu.:1.1578
##  Median :-0.1168   Median :-1.71700   Median :1.4834
##  Mean   : 0.0000   Mean   :-1.59933   Mean   :1.5993
##  3rd Qu.: 0.4143   3rd Qu.:-1.18453   3rd Qu.:2.0131
##  Max.   : 1.5631   Max.   :-0.04073   Max.   :3.1669
```

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

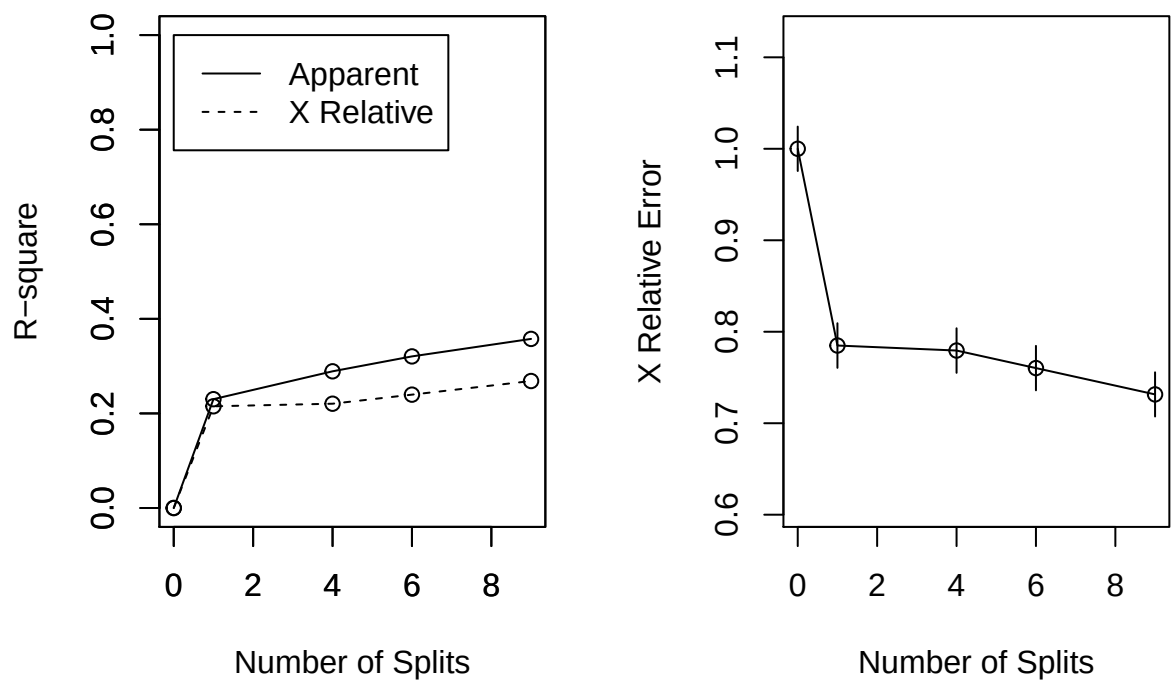



```
# The scatter plot supports the summary of the predicted interval, in the ranges of the fit,
# lower, and upper ranges. The R-squared value of 0.3283 of the model, indicates that this
# information can be predicted 33% of the time, with the data available, for the variance
# of the information.
```

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              chlorides            density
## [4] sulphates            total.sulfur.dioxide volatile.acidity
##
## Root node error: 730/1281 = 0.56987
##
## n= 1281
##
##         CP nsplit rel error  xerror     xstd
## 1 0.230137      0   1.00000 1.00000 0.024274
## 2 0.019635      1   0.76986 0.78493 0.024378
## 3 0.015753      4   0.71096 0.77945 0.024361
## 4 0.012329      6   0.67945 0.76027 0.024295
## 5 0.010000      9   0.64247 0.73151 0.024173
```

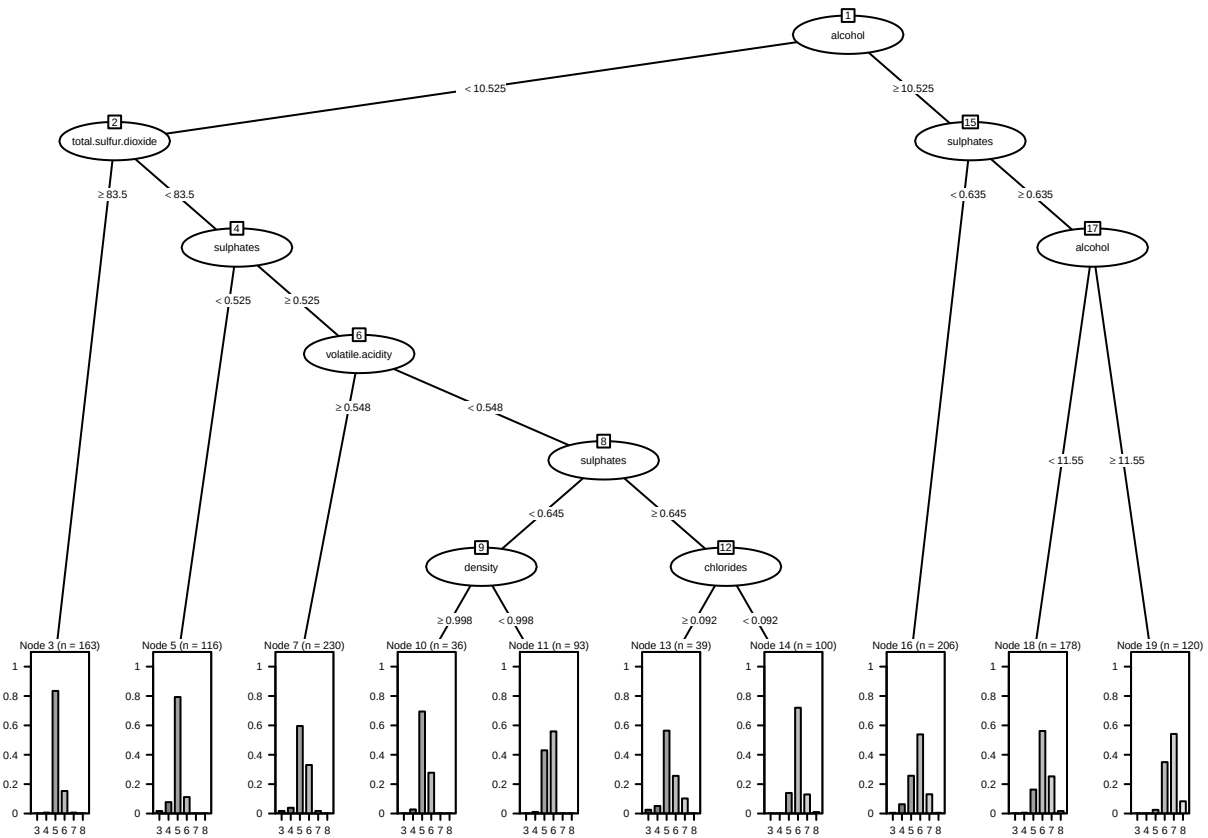



```r
# Results
rpartTree2
```

```
##
## 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 < 10.525
## |   |   [3] total.sulfur.dioxide >= 83.5: 5 (n = 163, err = 16.6%)
## |   |   [4] total.sulfur.dioxide < 83.5
```

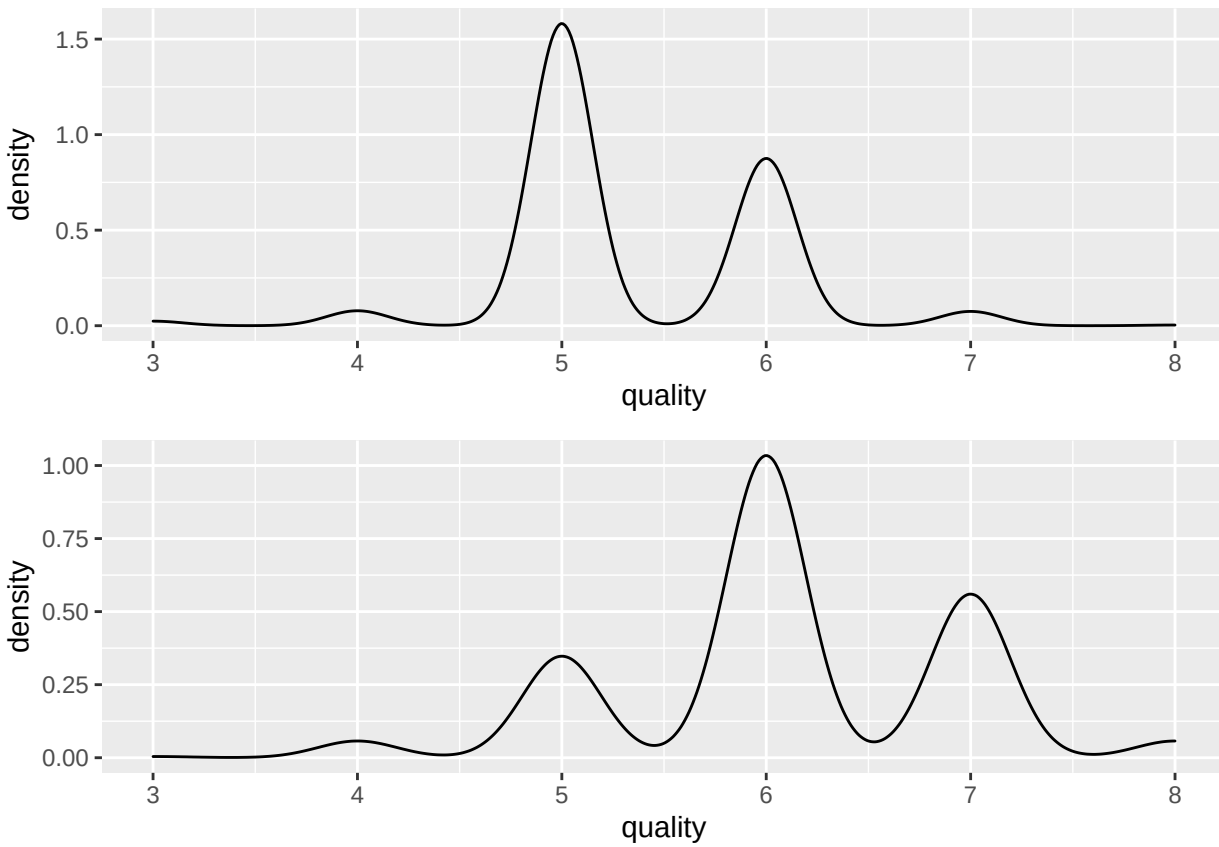
```
## |   |   |   [5] sulphates < 0.525: 5 (n = 116, err = 20.7%)
## |   |   |   [6] sulphates >= 0.525
## |   |   |   |   [7] volatile.acidity >= 0.5475: 5 (n = 230, err = 40.4%)
## |   |   |   |   [8] volatile.acidity < 0.5475
## |   |   |   |   |   [9] sulphates < 0.645
## |   |   |   |   |   |   [10] density >= 0.9977: 5 (n = 36, err = 30.6%)
## |   |   |   |   |   |   [11] density < 0.9977: 6 (n = 93, err = 44.1%)
## |   |   |   |   |   [12] sulphates >= 0.645
## |   |   |   |   |   |   [13] chlorides >= 0.0925: 5 (n = 39, err = 43.6%)
## |   |   |   |   |   |   [14] chlorides < 0.0925: 6 (n = 100, err = 28.0%)
## |   [15] alcohol >= 10.525
## |   |   [16] sulphates < 0.635: 6 (n = 206, err = 46.1%)
## |   |   [17] sulphates >= 0.635
## |   |   |   [18] alcohol < 11.55: 6 (n = 178, err = 43.8%)
## |   |   |   [19] alcohol >= 11.55: 7 (n = 120, err = 45.8%)
##
## Number of inner nodes:     9
## Number of terminal nodes: 10
```

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




```
# 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
```

```r
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

```r
set.seed(4)

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

print(rf)
```

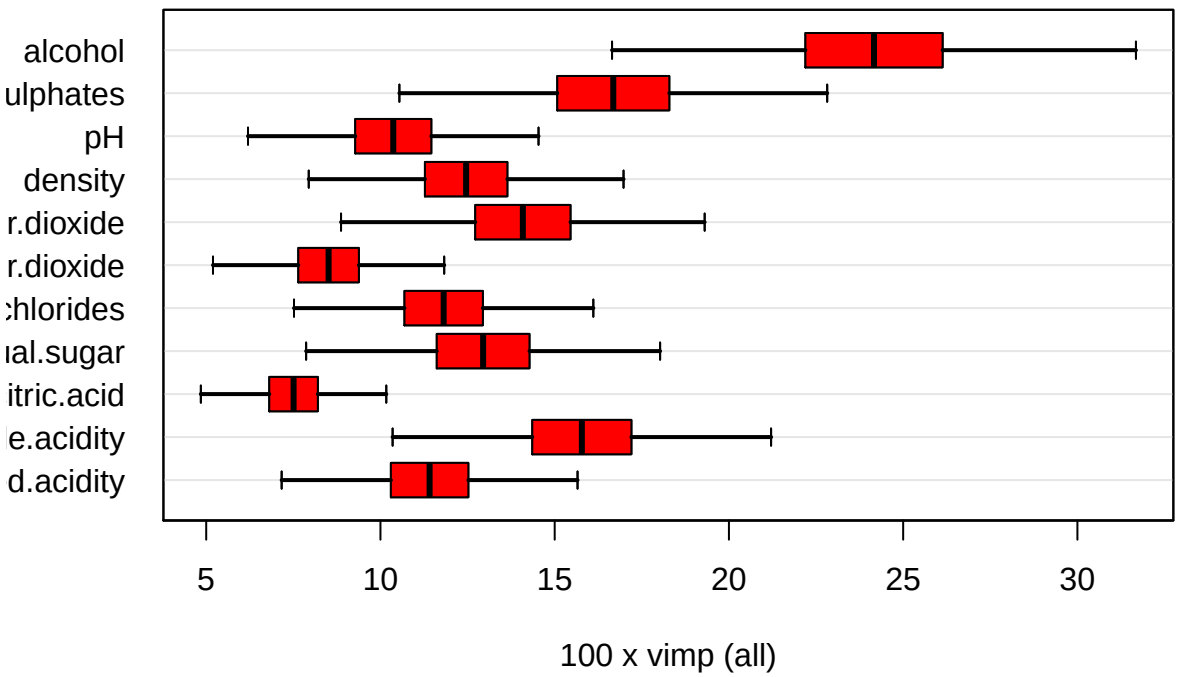
```
##                         Sample size: 1281
##            Frequency of class labels: 8, 37, 551, 511, 159, 15
##                      Number of trees: 500
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 247.234
## No. of variables tried at each split: 4
##               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: 0.0667632
##         (OOB) Normalized Brier score: 0.48069502
##                             (OOB) AUC: 0.82205375
##    (OOB) Requested performance error: 0.29274005, 1, 1, 0.19419238, 0.28571429, 0.40251572, 0.866666
##
## Confusion matrix:
##
##           predicted
##   observed 3 4   5   6  7 8 class.error
##          3 0 1   5   2  0 0      1.0000
##          4 0 0  25  12  0 0      1.0000
##          5 0 1 444 100  6 0      0.1942
##          6 0 0 117 363 31 0      0.2896
##          7 0 0   7  58 93 1      0.4151
##          8 0 0   0   4  9 2      0.8667
##
##       (OOB) Misclassification rate: 0.2958626
```

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

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

```
##                          lower      mean     upper       pvalue signif
## fixed.acidity         8.180121 11.410616 14.641111 2.212440e-12   TRUE
## volatile.acidity     11.647117 15.779193 19.911270 3.590176e-14   TRUE
## citric.acid           5.482045  7.506582  9.531118 1.835635e-13   TRUE
## residual.sugar        9.081405 12.946072 16.810740 2.591304e-11   TRUE
## chlorides             8.545886 11.814013 15.082139 6.946629e-13   TRUE
## free.sulfur.dioxide   5.987560  8.512102 11.036643 1.941270e-11   TRUE
## total.sulfur.dioxide 10.116592 14.086624 18.056657 1.770066e-12   TRUE
## density               9.022301 12.459023 15.895746 5.999366e-13   TRUE
## pH                    7.195548 10.367728 13.539909 7.478815e-11   TRUE
## sulphates            12.010919 16.681529 21.352138 1.277993e-12   TRUE
## alcohol              18.443184 24.163057 29.882931 6.175356e-17   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: 247.234
##           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: 0.08004649
##                Normalized Brier score: 0.57633473
##                                   AUC: 0.80679148
##          Requested performance error: 0.36163522, 1, 1, 0.23846154, 0.30708661, 0.6, 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  4  1 0      1.0000
##          5 0 0 99 31  0 0      0.2385
##          6 0 0 29 88 10 0      0.3071
##          7 0 0  4 20 16 0      0.6000
##          8 0 0  0  3  0 0      1.0000
##
##            Misclassification error: 0.3616352
```

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 = 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      0.6410968  0.3647402  0.4947294
##   2      0.6390156  0.3687722  0.4930808
##   3      0.6388538  0.3691179  0.4921412
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 3.
```

Mars Tuning

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

mars_wine
```

```
## Selected 11 of 16 terms, and 5 of 5 predictors
## Termination condition: Reached nk 21
## Importance: alcohol, sulphates, volatile.acidity, chlorides, ...
## Number of terms at each degree of interaction: 1 10 (additive model)
## GCV 0.3995695    RSS 495.2041    GRSq 0.3804081    RSq 0.3996191
```

```
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##             data=wine_train)
##
##                              coefficients
## (Intercept)                     6.2560668
## h(volatile.acidity-0.26)       -0.7896152
```

```
## h(0.152-chlorides)             2.8759318
## h(144-total.sulfur.dioxide)    0.0020873
## h(0.82-sulphates)             -1.9120561
## h(sulphates-0.82)             -0.4929467
## h(alcohol-11.3)               -3.4323569
## h(alcohol-11.4)                6.5386404
## h(alcohol-11.7)               -6.7169766
## h(alcohol-11.8)                3.5629306
## h(12.4-alcohol)               -0.2402167
##
## Selected 11 of 16 terms, and 5 of 5 predictors
## Termination condition: Reached nk 21
## Importance: alcohol, sulphates, volatile.acidity, chlorides, ...
## Number of terms at each degree of interaction: 1 10 (additive model)
## GCV 0.3995695    RSS 495.2041    GRSq 0.3804081    RSq 0.3996191
```

```
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      0.6958248  0.2461512  0.5467038
##   1        3      0.6648091  0.3129951  0.5082758
##   1        4      0.6486695  0.3467222  0.4987637
##   1        5      0.6489489  0.3466279  0.4992545
##   1        6      0.6505245  0.3440970  0.4996087
##   1        7      0.6501727  0.3449717  0.4995417
##   1        8      0.6495943  0.3465777  0.4989477
##   1        9      0.6495414  0.3468115  0.4985270
##   1       10      0.6507816  0.3450530  0.4999083
##   1       11      0.6505098  0.3460652  0.4997600
##   1       12      0.6520164  0.3438562  0.5003093
##   1       13      0.6539775  0.3411943  0.5015090
##   1       14      0.6552017  0.3392044  0.5023241
##   1       15      0.6549327  0.3396373  0.5020724
##   1       16      0.6549506  0.3396558  0.5021755
```

```
##   1        17       0.6549506  0.3396558  0.5021755
##   1        18       0.6549506  0.3396558  0.5021755
##   1        19       0.6549506  0.3396558  0.5021755
##   1        20       0.6549506  0.3396558  0.5021755
##   1        21       0.6549506  0.3396558  0.5021755
##   1        22       0.6549506  0.3396558  0.5021755
##   1        23       0.6549506  0.3396558  0.5021755
##   1        24       0.6549506  0.3396558  0.5021755
##   1        25       0.6549506  0.3396558  0.5021755
##   1        26       0.6549506  0.3396558  0.5021755
##   1        27       0.6549506  0.3396558  0.5021755
##   1        28       0.6549506  0.3396558  0.5021755
##   1        29       0.6549506  0.3396558  0.5021755
##   1        30       0.6549506  0.3396558  0.5021755
##   1        31       0.6549506  0.3396558  0.5021755
##   1        32       0.6549506  0.3396558  0.5021755
##   1        33       0.6549506  0.3396558  0.5021755
##   1        34       0.6549506  0.3396558  0.5021755
##   1        35       0.6549506  0.3396558  0.5021755
##   1        36       0.6549506  0.3396558  0.5021755
##   1        37       0.6549506  0.3396558  0.5021755
##   1        38       0.6549506  0.3396558  0.5021755
##   2         2       0.6954928  0.2468476  0.5463763
##   2         3       0.6660576  0.3103660  0.5137931
##   2         4       0.6454110  0.3522298  0.4936360
##   2         5       0.6422467  0.3598510  0.4931323
##   2         6       0.6437021  0.3580451  0.4932673
##   2         7       0.6444236  0.3580010  0.4944992
##   2         8       0.6468150  0.3544162  0.4953825
##   2         9       0.6483401  0.3524432  0.4964011
##   2        10       0.6507646  0.3491618  0.4975523
##   2        11       0.6523252  0.3471198  0.4982322
##   2        12       0.6540995  0.3445165  0.4984894
##   2        13       0.6542404  0.3445736  0.4985894
##   2        14       0.6550265  0.3443170  0.4985331
##   2        15       0.6560467  0.3430849  0.4987826
##   2        16       0.6562863  0.3426595  0.4989498
##   2        17       0.6560625  0.3428740  0.4988956
##   2        18       0.6564013  0.3423725  0.4989296
##   2        19       0.6564013  0.3423725  0.4989296
##   2        20       0.6564013  0.3423725  0.4989296
##   2        21       0.6564013  0.3423725  0.4989296
##   2        22       0.6564013  0.3423725  0.4989296
##   2        23       0.6564013  0.3423725  0.4989296
##   2        24       0.6564013  0.3423725  0.4989296
##   2        25       0.6564013  0.3423725  0.4989296
##   2        26       0.6564013  0.3423725  0.4989296
##   2        27       0.6564013  0.3423725  0.4989296
##   2        28       0.6564013  0.3423725  0.4989296
##   2        29       0.6564013  0.3423725  0.4989296
##   2        30       0.6564013  0.3423725  0.4989296
##   2        31       0.6564013  0.3423725  0.4989296
##   2        32       0.6564013  0.3423725  0.4989296
##   2        33       0.6564013  0.3423725  0.4989296
```

```
##   2       34      0.6564013  0.3423725  0.4989296
##   2       35      0.6564013  0.3423725  0.4989296
##   2       36      0.6564013  0.3423725  0.4989296
##   2       37      0.6564013  0.3423725  0.4989296
##   2       38      0.6564013  0.3423725  0.4989296
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nprune = 5 and degree = 2.
```

KNN Neighbors

```r
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  0.7417840  0.3247830  0.4177499
##    2  0.6974146  0.3323450  0.4696999
##    3  0.6731901  0.3448979  0.4791641
##    4  0.6665302  0.3467985  0.4836518
##    5  0.6533331  0.3600591  0.4824286
##    6  0.6457641  0.3687854  0.4794655
##    7  0.6430043  0.3710435  0.4799530
##    8  0.6416843  0.3717036  0.4821467
##    9  0.6339821  0.3832671  0.4796633
##   10  0.6354233  0.3804117  0.4819247
##   11  0.6338017  0.3827153  0.4821245
##   12  0.6360295  0.3780547  0.4844249
##   13  0.6366929  0.3757671  0.4856975
##   14  0.6349456  0.3781415  0.4862533
##   15  0.6338248  0.3798528  0.4858869
##   16  0.6351536  0.3769359  0.4873973
##   17  0.6335827  0.3795098  0.4862066
##   18  0.6325047  0.3814530  0.4848592
##   19  0.6325468  0.3809950  0.4861473
```

```
##   20  0.6307150  0.3843236  0.4845210
##   21  0.6333513  0.3793499  0.4872637
##   22  0.6344618  0.3769321  0.4892463
##   23  0.6341257  0.3778016  0.4894453
##   24  0.6334651  0.3789900  0.4900132
##   25  0.6320417  0.3818758  0.4891855
##   26  0.6325094  0.3809279  0.4899857
##   27  0.6319274  0.3823033  0.4905126
##   28  0.6316129  0.3832613  0.4907701
##   29  0.6310374  0.3848262  0.4909850
##   30  0.6309881  0.3848130  0.4908735
##   31  0.6298325  0.3870258  0.4899401
##   32  0.6298265  0.3871018  0.4907527
##   33  0.6301718  0.3862833  0.4908459
##   34  0.6292436  0.3880411  0.4897946
##   35  0.6292789  0.3881321  0.4899380
##   36  0.6291079  0.3885095  0.4893900
##   37  0.6307087  0.3854649  0.4905386
##   38  0.6307204  0.3856069  0.4900028
##   39  0.6301815  0.3866979  0.4898142
##   40  0.6298177  0.3875118  0.4901264
##   41  0.6286737  0.3897724  0.4892917
##   42  0.6280837  0.3911209  0.4889145
##   43  0.6275018  0.3922607  0.4888636
##   44  0.6277265  0.3918345  0.4891569
##   45  0.6278012  0.3918612  0.4889342
##   46  0.6281747  0.3912383  0.4892897
##   47  0.6278795  0.3918446  0.4896199
##   48  0.6273854  0.3931893  0.4891903
##   49  0.6279997  0.3919908  0.4901511
##   50  0.6286275  0.3907562  0.4906589
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 48.
```

SVM

```r
set.seed(4)

subset(wine_train, select = -c(quality_target, quality)) -> predictors
wine_train$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  0.6253012  0.3968055  0.4644470
##   0.50  0.6184829  0.4095823  0.4549595
##   1.00  0.6130196  0.4198850  0.4475702
##   2.00  0.6135949  0.4206505  0.4458400
##   4.00  0.6207280  0.4117663  0.4492851
##
## Tuning parameter 'sigma' was held constant at a value of 0.08844672
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were sigma = 0.08844672 and C = 1.
```