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.

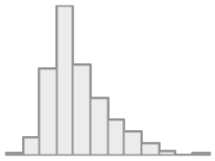
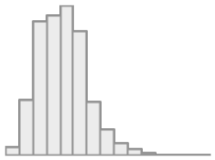
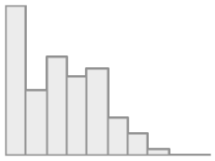
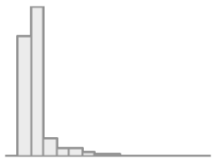
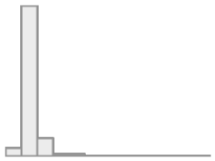
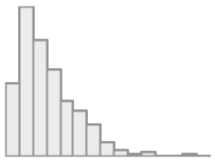
```
wine <- read.csv(
  url("https://raw.githubusercontent.com/OscarG-DataSci/ADS503/main/winequality-red.csv")
                      , header = TRUE)
```

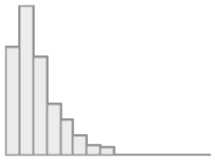
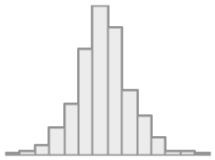
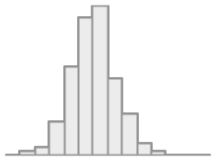
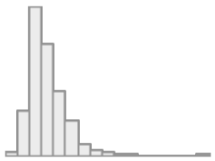
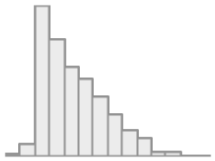
Data Summary

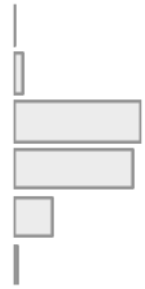
```
dfSummary(wine,
          plain.ascii  = FALSE,
          style        = "grid",
          graph.magnif = 0.75,
          valid.col    = FALSE,
          tmp.img.dir  = "/tmp")
```

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

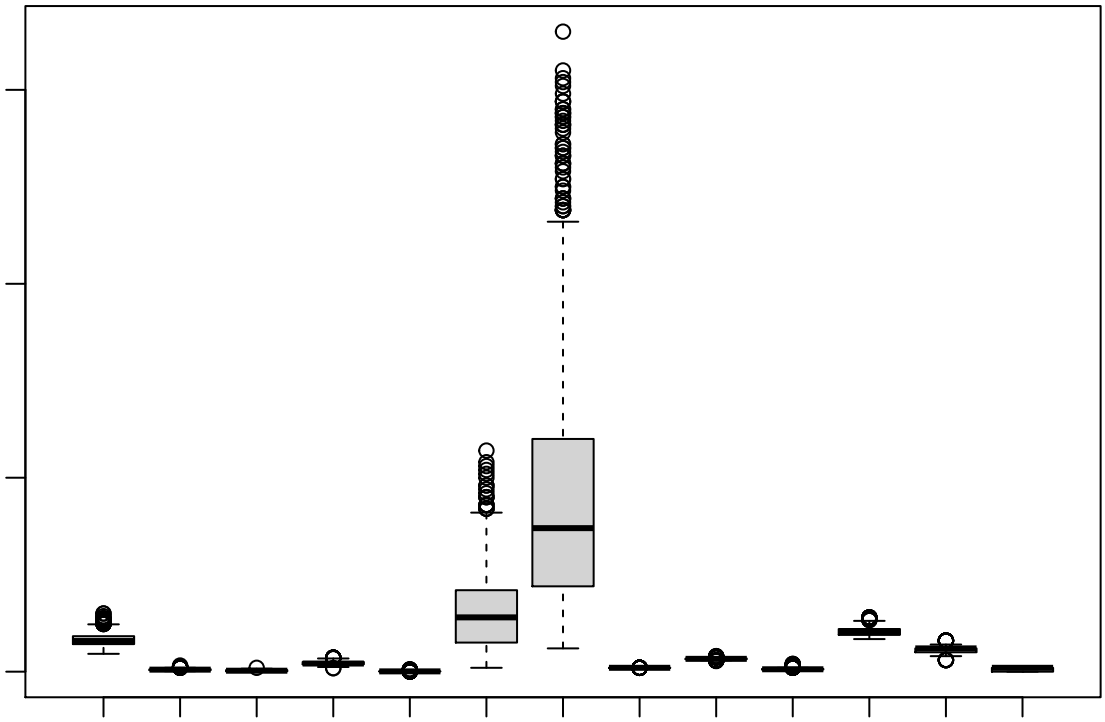
```
par(mar=c(1,1,1,1)) # to fix boxplot knit processing issues

# 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
```

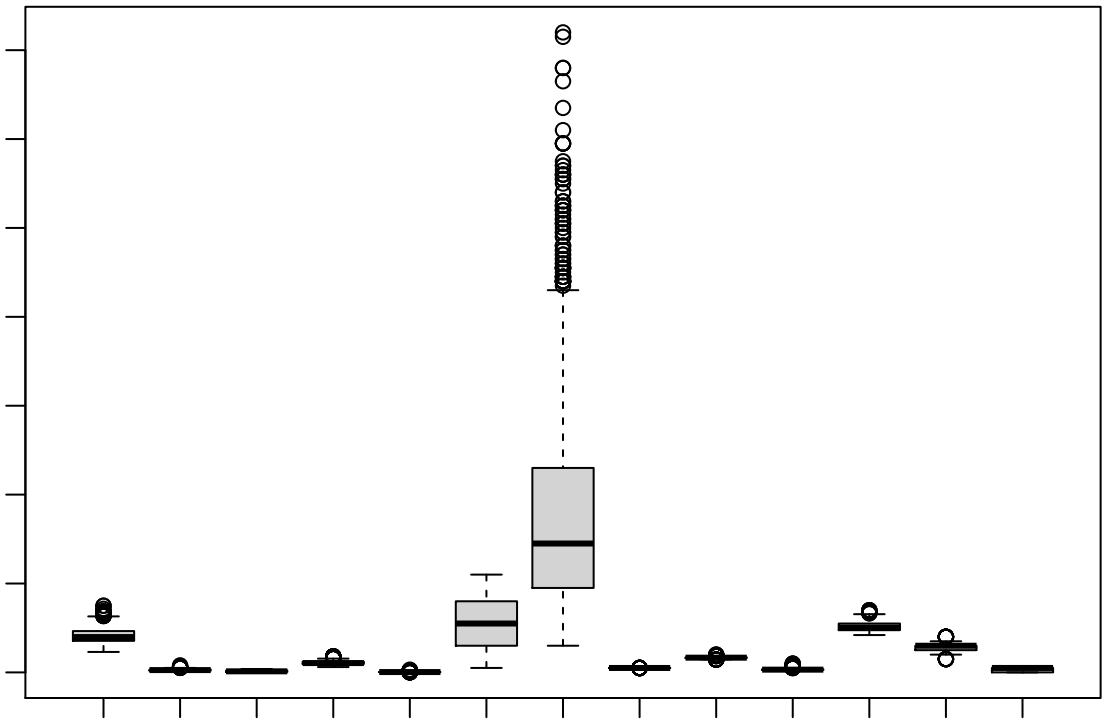
```
# Check for missing values in data set
wine %>% na.omit() %>% count() # there are no missing values
```

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

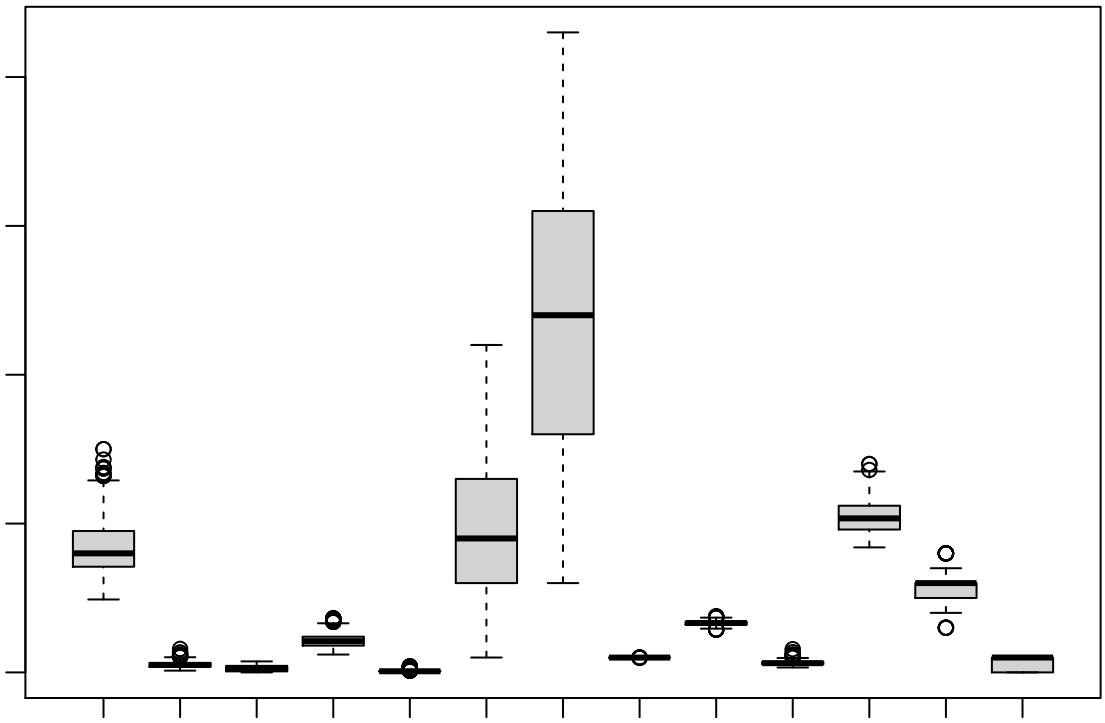
```
# Removing outliers for residual sugar:
Q <- quantile(wine$residual.sugar, probs=c(.25, .75), na.rm = FALSE)
iqr_rs <- IQR(wine$residual.sugar)
up_rs <-  Q[2]+1.5*iqr_rs # Upper Range
low_rs <- Q[1]-1.5*iqr_rs # Lower Range
eliminated_rs <- subset(wine, wine$residual.sugar > (Q[1] - 1.5*iqr_rs) &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



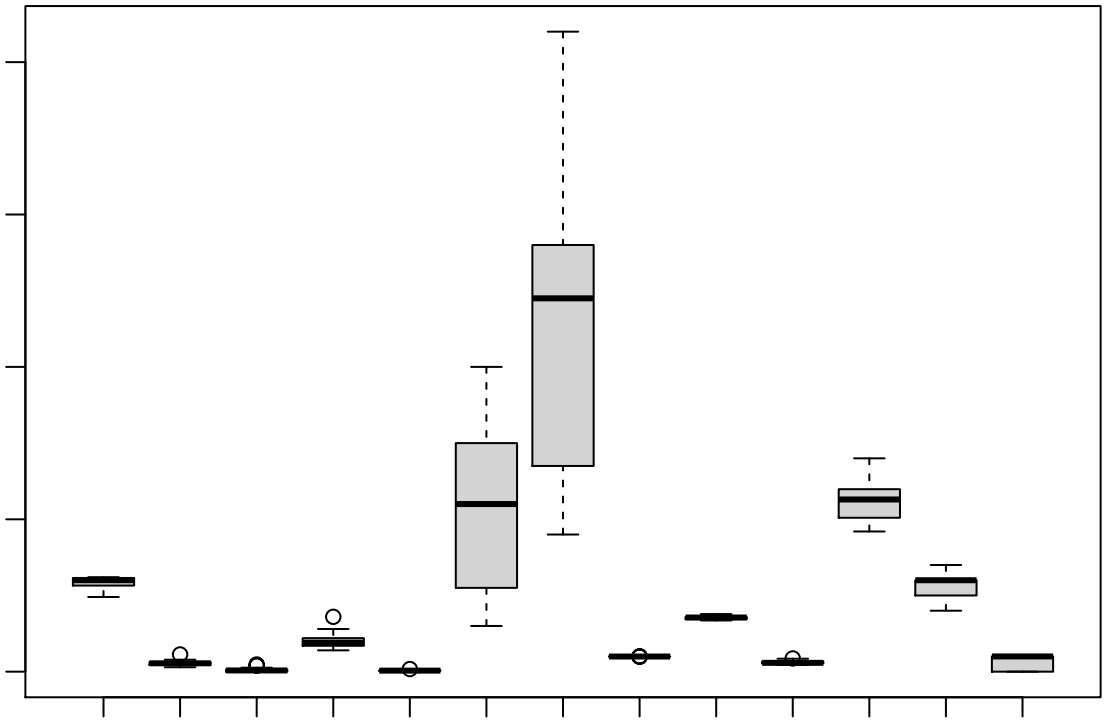
```r
#Removing outliers for free.sulfur.dioxide:
Q2 <- quantile(wine$free.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_fs <- IQR(eliminated_rs$free.sulfur.dioxide)
up_fs <-  Q2[2]+1.5*iqr_fs # Upper Range
low_fs <- Q2[1]-1.5*iqr_fs # Lower Range
eliminated_fs <- subset(eliminated_rs, eliminated_rs$free.sulfur.dioxide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



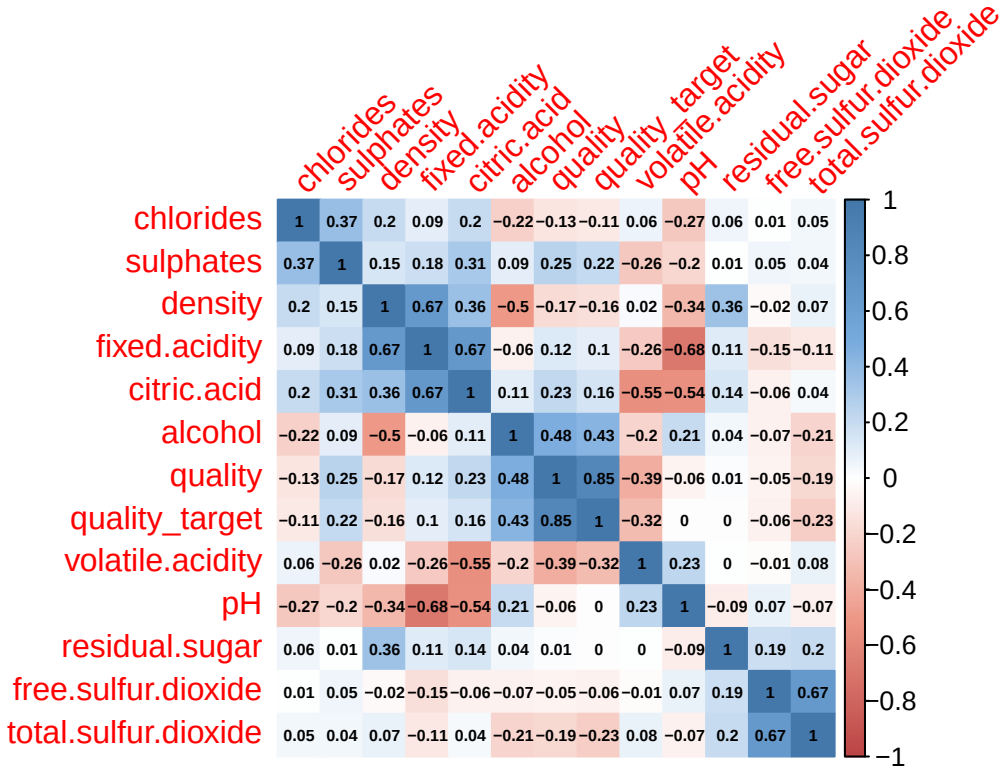
```r
#Removing outliers for fixed.acidity:
Q4 <- quantile(wine$fixed.acidity, probs=c(.25, .75), na.rm = FALSE)
iqr_fa <- IQR(eliminated_ts$fixed.acidity)
up_fa <-  Q[2]+1.5*iqr_fa # Upper Range
low_fa <- Q[1]-1.5*iqr_fa # Lower Range
eliminated_fa <- subset(eliminated_ts, eliminated_ts$fixed.acidity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



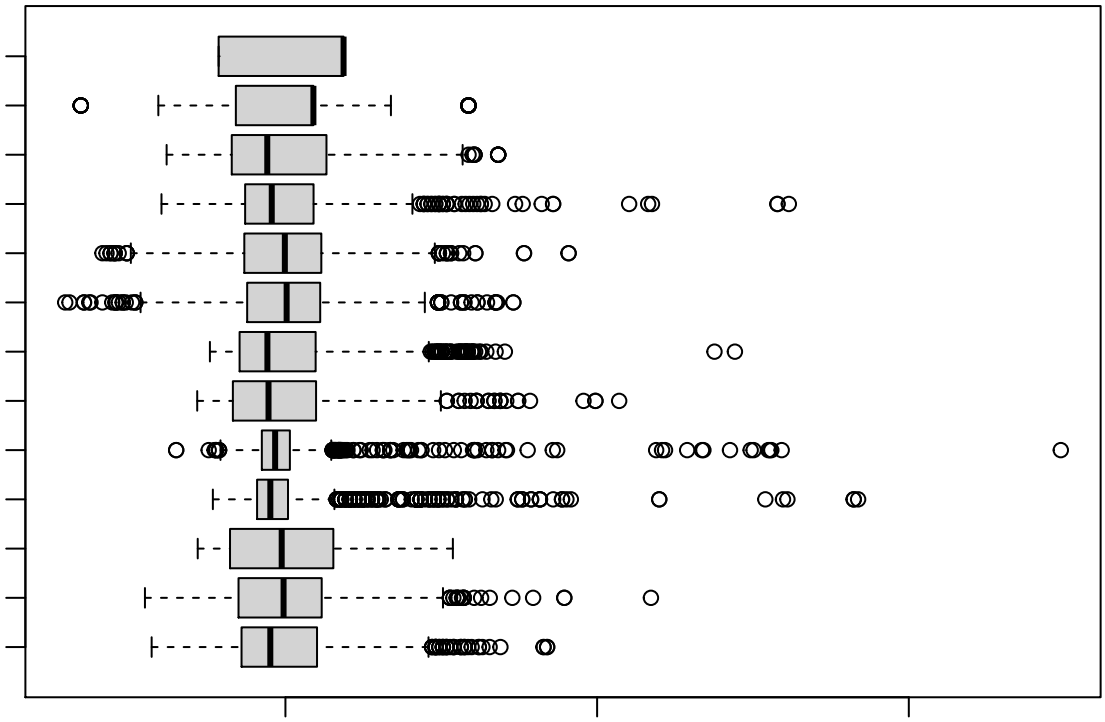
```r
new_wine_data <- eliminated_fa

# 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
## -1.0595 -0.3536  0.0053  0.3861  1.0246
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.28359    0.14282  -8.987  < 2e-16 ***
## volatile.acidity -0.55996    0.07064  -7.927 4.87e-15 ***
## sulphates         0.46111    0.07666   6.015 2.35e-09 ***
```

```
## alcohol           0.17393    0.01167  14.905  < 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:  0.256,  Adjusted R-squared:  0.2543
## F-statistic: 146.5 on 3 and 1277 DF,  p-value: < 2.2e-16
```

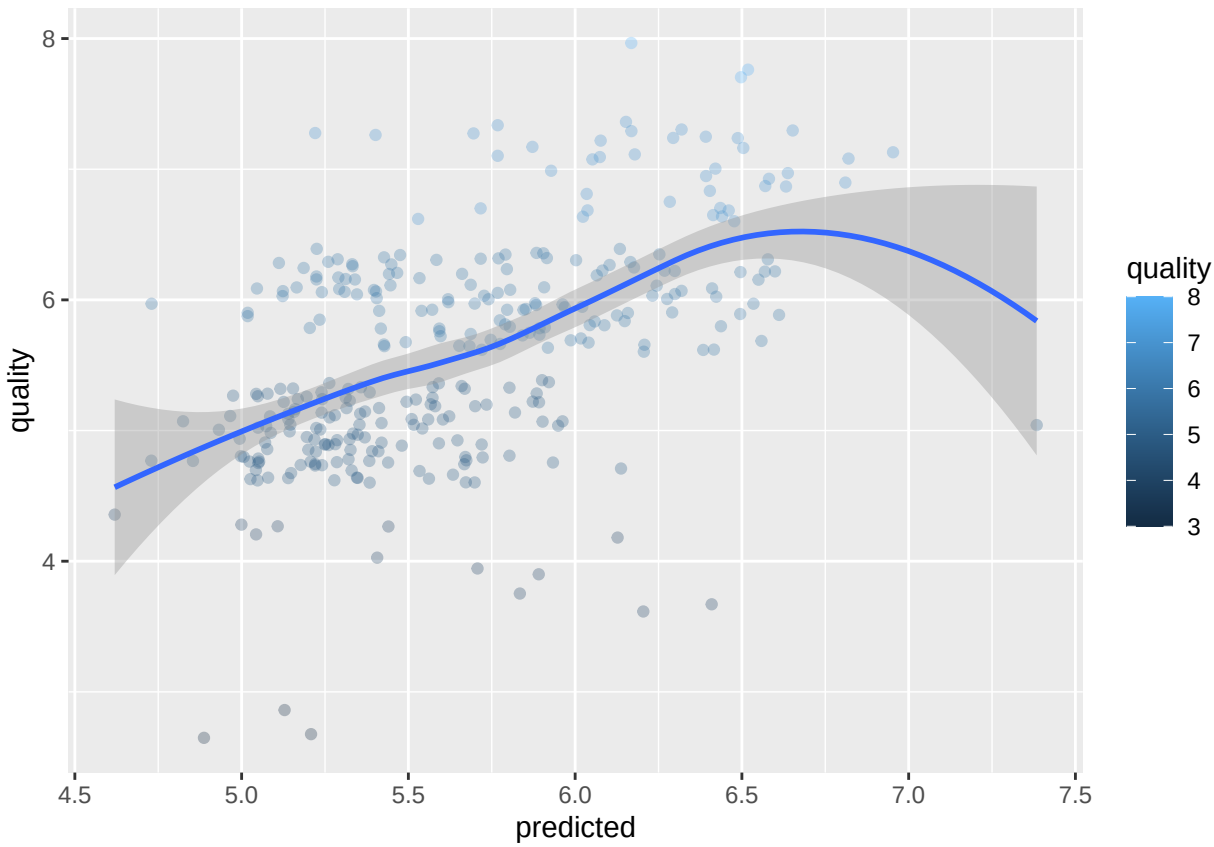
```r
# Model using "quality" as target variable
lmodel2 <- lm(quality~ volatile.acidity + sulphates + alcohol, data = wine_train)

summary(lmodel2)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + sulphates + alcohol,
##     data = wine_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.74152 -0.38150 -0.06589  0.48377  2.17397
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.44803    0.21704  11.279  < 2e-16 ***
## volatile.acidity -1.13840    0.10735 -10.604  < 2e-16 ***
## sulphates         0.80327    0.11650   6.895 8.46e-12 ***
## alcohol           0.31349    0.01773  17.678  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6548 on 1277 degrees of freedom
## Multiple R-squared:  0.3382, Adjusted R-squared:  0.3366
## F-statistic: 217.5 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.   :4.619   Min.   :3.329   Min.   :5.910
##  1st Qu.:5.261   1st Qu.:3.976   1st Qu.:6.548
##  Median :5.592   Median :4.307   Median :6.878
##  Mean   :5.660   Mean   :4.373   Mean   :6.946
##  3rd Qu.:5.999   3rd Qu.:4.713   3rd Qu.:7.284
##  Max.   :7.384   Max.   :6.090   Max.   :8.678
```

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

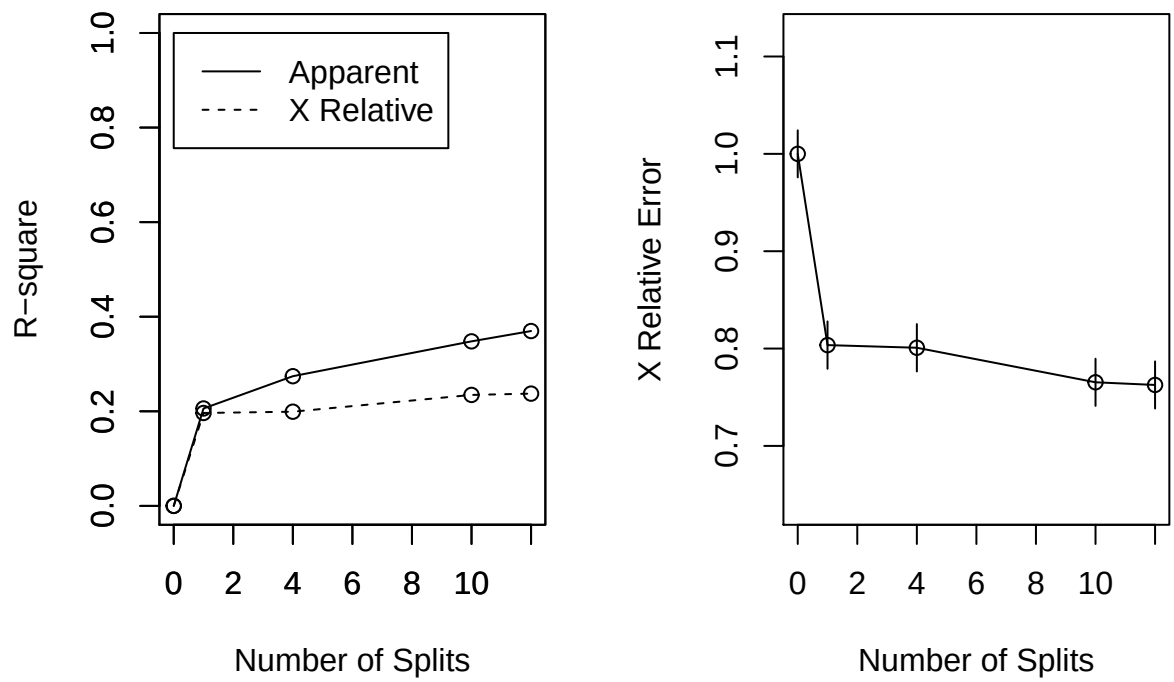



```
# The scatter plot supports the summary of the predicted interval, in the ranges of the fit,
# lower, and upper ranges. The R-squared value of 0.3283 of the model, indicates that this
# information can be predicted 33% of the time, with the data available, for the variance
# of the information.
```

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            fixed.acidity
## [4] pH                   sulphates            total.sulfur.dioxide
## [7] volatile.acidity
##
## Root node error: 733/1281 = 0.57221
##
## n= 1281
##
##         CP nsplit rel error  xerror     xstd
## 1 0.206003      0   1.00000 1.00000 0.024158
## 2 0.022738      1   0.79400 0.80355 0.024335
## 3 0.012278      4   0.72578 0.80082 0.024329
## 4 0.010914     10   0.65211 0.76535 0.024225
## 5 0.010000     12   0.63029 0.76262 0.024216
```

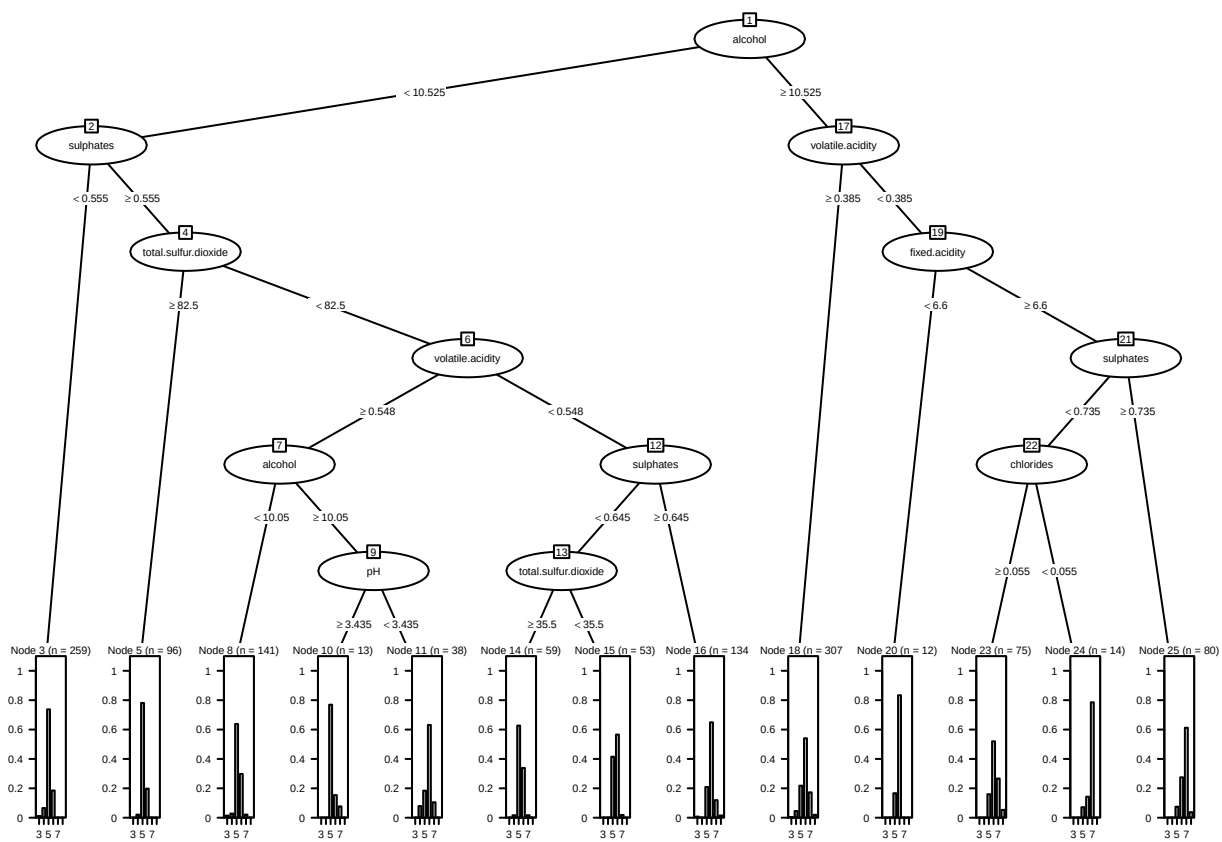



```r
# Results
rpartTree2
```

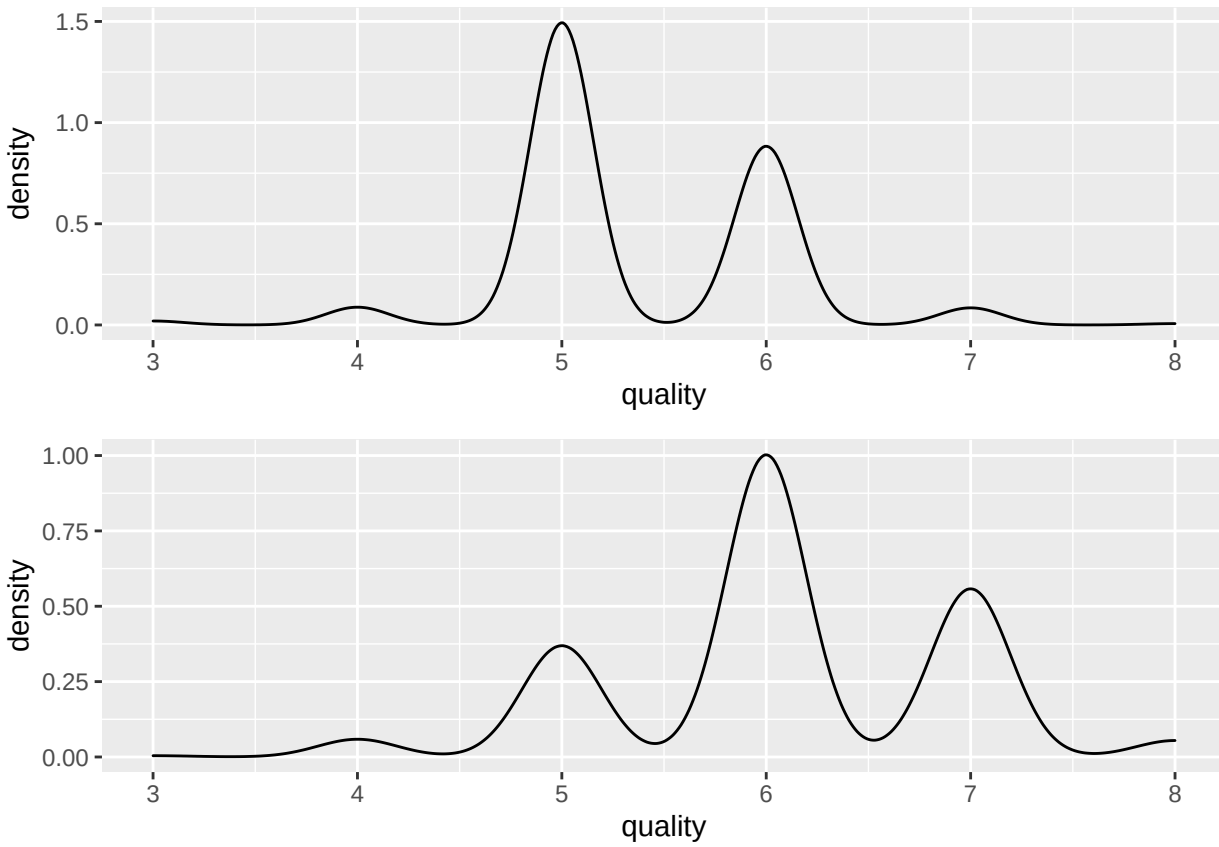
```
##
## Model formula:
## quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol
##
## Fitted party:
## [1] root
## |   [2] alcohol < 10.525
## |   |   [3] sulphates < 0.555: 5 (n = 259, err = 26.3%)
```

```
## |   |   [4] sulphates >= 0.555
## |   |   |   [5] total.sulfur.dioxide >= 82.5: 5 (n = 96, err = 21.9%)
## |   |   |   [6] total.sulfur.dioxide < 82.5
## |   |   |   |   [7] volatile.acidity >= 0.5475
## |   |   |   |   |   [8] alcohol < 10.05: 5 (n = 141, err = 36.2%)
## |   |   |   |   |   [9] alcohol >= 10.05
## |   |   |   |   |   |   [10] pH >= 3.435: 5 (n = 13, err = 23.1%)
## |   |   |   |   |   |   [11] pH < 3.435: 6 (n = 38, err = 36.8%)
## |   |   |   |   [12] volatile.acidity < 0.5475
## |   |   |   |   |   [13] sulphates < 0.645
## |   |   |   |   |   |   [14] total.sulfur.dioxide >= 35.5: 5 (n = 59, err = 37.3%)
## |   |   |   |   |   |   [15] total.sulfur.dioxide < 35.5: 6 (n = 53, err = 43.4%)
## |   |   |   |   |   [16] sulphates >= 0.645: 6 (n = 134, err = 35.1%)
## |   [17] alcohol >= 10.525
## |   |   [18] volatile.acidity >= 0.385: 6 (n = 307, err = 45.9%)
## |   |   [19] volatile.acidity < 0.385
## |   |   |   [20] fixed.acidity < 6.6: 6 (n = 12, err = 16.7%)
## |   |   |   [21] fixed.acidity >= 6.6
## |   |   |   |   [22] sulphates < 0.735
## |   |   |   |   |   [23] chlorides >= 0.0555: 6 (n = 75, err = 48.0%)
## |   |   |   |   |   [24] chlorides < 0.0555: 7 (n = 14, err = 21.4%)
## |   |   |   |   [25] sulphates >= 0.735: 7 (n = 80, err = 38.8%)
##
## Number of inner nodes:    12
## Number of terminal nodes: 13
```

```
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

```r
set.seed(4)

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

print(rf)
```

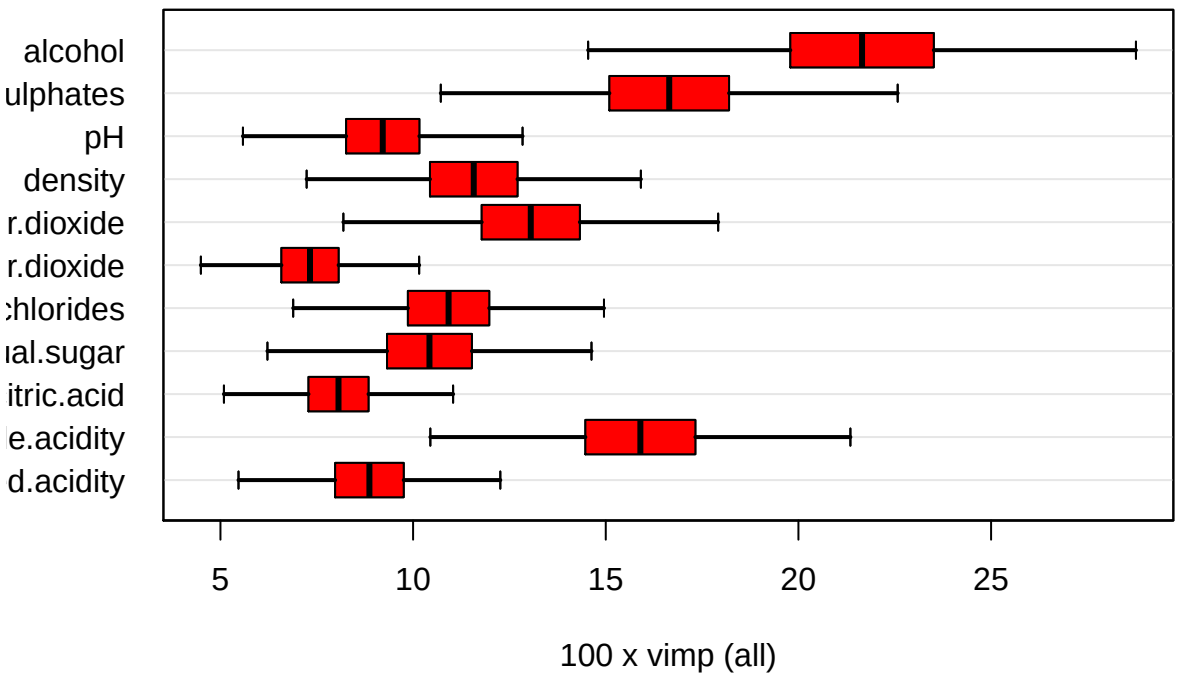
```
##                          Sample size: 1281
##            Frequency of class labels: 7, 41, 548, 511, 159, 15
##                      Number of trees: 500
##            Forest terminal node size: 1
```

```
##        Average no. of terminal nodes: 253.422
## 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: 0.06943054
##         (OOB) Normalized Brier score: 0.49989991
##                            (OOB) AUC: 0.8082197
##    (OOB) Requested performance error: 0.30444965, 1, 1, 0.20072993, 0.2778865, 0.48427673, 0.8666666
##
## Confusion matrix:
##
##           predicted
##   observed 3 4   5   6  7 8 class.error
##          3 0 0   4   3  0 0      1.0000
##          4 0 0  31  10  0 0      1.0000
##          5 0 0 440 104  4 0      0.1971
##          6 0 2 104 367 38 0      0.2818
##          7 0 0   9  67 83 0      0.4780
##          8 0 0   0   7  6 2      0.8667
##
##       (OOB) Misclassification rate: 0.303669
```

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

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

```
##                          lower      mean     upper       pvalue signif
## fixed.acidity         6.281015  8.866242 11.451469 8.971488e-12   TRUE
## volatile.acidity     11.750633 15.898089 20.045546 2.890216e-14   TRUE
## citric.acid           5.801079  8.064119 10.327159 1.433068e-12   TRUE
## residual.sugar        7.224972 10.424628 13.624285 8.533405e-11   TRUE
## chlorides             7.852130 10.920594 13.989059 1.524279e-12   TRUE
## free.sulfur.dioxide   5.168766  7.324416  9.480067 1.373645e-11   TRUE
## total.sulfur.dioxide  9.353308 13.053928 16.754548 2.359727e-12   TRUE
## density               8.273913 11.573673 14.873433 3.111819e-12   TRUE
## pH                    6.451411  9.212314 11.973218 3.079709e-11   TRUE
## sulphates            12.135730 16.647134 21.158537 2.374570e-13   TRUE
## alcohol              16.242647 21.651380 27.060113 2.150723e-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: 253.422
##           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: 0.07068781
##                Normalized Brier score: 0.50895221
##                                   AUC: 0.814764
##          Requested performance error: 0.29874214, 1, 1, 0.17293233, 0.25984252, 0.525, 1
##
## Confusion matrix:
##
##           predicted
##   observed 3 4   5  6  7 8 class.error
##          3 0 0   3  0  0 0      1.0000
##          4 1 0   7  4  0 0      1.0000
##          5 0 1 111 20  1 0      0.1654
##          6 0 0  27 94  6 0      0.2598
##          7 0 0   4 16 19 1      0.5250
##          8 0 0   0  2  1 0      1.0000
##
##            Misclassification error: 0.2955975
```

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      0.6478811  0.3522576  0.5069455
##   2      0.6470840  0.3536482  0.5065996
##   3      0.6470244  0.3538341  0.5060809
##
## 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 15 of 16 terms, and 5 of 5 predictors
## Termination condition: Reached nk 21
## Importance: alcohol, sulphates, volatile.acidity, total.sulfur.dioxide, ...
## Number of terms at each degree of interaction: 1 14 (additive model)
## GCV 0.4100085    RSS 501.7096    GRSq 0.3661884    RSq 0.3936143
```

```
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##             data=wine_train)
##
##                              coefficients
## (Intercept)                     29.475710
## h(1.01-volatile.acidity)         0.852686
```

```
## h(volatile.acidity-1.01)       -1.821456
## h(chlorides-0.042)             67.980049
## h(chlorides-0.061)            -17.972067
## h(0.152-chlorides)             52.272594
## h(chlorides-0.152)            -50.371827
## h(total.sulfur.dioxide-9)      -0.243157
## h(total.sulfur.dioxide-94)     -0.008360
## h(131-total.sulfur.dioxide)    -0.241856
## h(total.sulfur.dioxide-131)     0.260480
## h(0.76-sulphates)              -2.185146
## h(alcohol-11.1)                 0.300588
## h(12.3-alcohol)                -0.218249
## h(alcohol-12.3)                -0.389894
##
## Selected 15 of 16 terms, and 5 of 5 predictors
## Termination condition: Reached nk 21
## Importance: alcohol, sulphates, volatile.acidity, total.sulfur.dioxide, ...
## Number of terms at each degree of interaction: 1 14 (additive model)
## GCV 0.4100085    RSS 501.7096    GRSq 0.3661884    RSq 0.3936143
```

```
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      0.7122760  0.2171101  0.5701033
##   1        3      0.6752567  0.2973558  0.5289540
##   1        4      0.6535081  0.3409243  0.5129189
##   1        5      0.6570769  0.3338440  0.5144791
##   1        6      0.6591071  0.3305375  0.5166057
##   1        7      0.6578213  0.3337205  0.5158201
##   1        8      0.6590069  0.3318615  0.5170450
##   1        9      0.6593878  0.3315075  0.5168354
##   1       10      0.6597737  0.3311809  0.5163084
##   1       11      0.6598222  0.3315357  0.5158173
##   1       12      0.6595859  0.3323695  0.5157959
```

```
##   1        13       0.6600013  0.3318148  0.5159406
##   1        14       0.6603922  0.3312989  0.5160279
##   1        15       0.6605024  0.3310738  0.5160779
##   1        16       0.6605024  0.3310738  0.5160779
##   1        17       0.6605024  0.3310738  0.5160779
##   1        18       0.6605024  0.3310738  0.5160779
##   1        19       0.6605024  0.3310738  0.5160779
##   1        20       0.6605024  0.3310738  0.5160779
##   1        21       0.6605024  0.3310738  0.5160779
##   1        22       0.6605024  0.3310738  0.5160779
##   1        23       0.6605024  0.3310738  0.5160779
##   1        24       0.6605024  0.3310738  0.5160779
##   1        25       0.6605024  0.3310738  0.5160779
##   1        26       0.6605024  0.3310738  0.5160779
##   1        27       0.6605024  0.3310738  0.5160779
##   1        28       0.6605024  0.3310738  0.5160779
##   1        29       0.6605024  0.3310738  0.5160779
##   1        30       0.6605024  0.3310738  0.5160779
##   1        31       0.6605024  0.3310738  0.5160779
##   1        32       0.6605024  0.3310738  0.5160779
##   1        33       0.6605024  0.3310738  0.5160779
##   1        34       0.6605024  0.3310738  0.5160779
##   1        35       0.6605024  0.3310738  0.5160779
##   1        36       0.6605024  0.3310738  0.5160779
##   1        37       0.6605024  0.3310738  0.5160779
##   1        38       0.6605024  0.3310738  0.5160779
##   2         2       0.7131117  0.2154654  0.5711957
##   2         3       0.6761504  0.2945847  0.5340899
##   2         4       0.6564135  0.3354539  0.5150608
##   2         5       0.6508974  0.3467644  0.5109214
##   2         6       0.6470358  0.3544765  0.5080996
##   2         7       0.6519450  0.3472954  0.5100834
##   2         8       0.6548159  0.3436376  0.5116912
##   2         9       0.6711602  0.3311970  0.5140466
##   2        10       0.6705270  0.3311357  0.5144146
##   2        11       0.6731133  0.3267571  0.5163083
##   2        12       0.6719547  0.3290581  0.5149547
##   2        13       0.6729162  0.3278180  0.5156340
##   2        14       0.6731796  0.3274458  0.5154864
##   2        15       0.6737603  0.3269853  0.5157601
##   2        16       0.6737658  0.3273973  0.5157295
##   2        17       0.6738441  0.3273854  0.5157636
##   2        18       0.6740397  0.3271363  0.5158683
##   2        19       0.6740397  0.3271363  0.5158683
##   2        20       0.6740397  0.3271363  0.5158683
##   2        21       0.6740397  0.3271363  0.5158683
##   2        22       0.6740397  0.3271363  0.5158683
##   2        23       0.6740397  0.3271363  0.5158683
##   2        24       0.6740397  0.3271363  0.5158683
##   2        25       0.6740397  0.3271363  0.5158683
##   2        26       0.6740397  0.3271363  0.5158683
##   2        27       0.6740397  0.3271363  0.5158683
##   2        28       0.6740397  0.3271363  0.5158683
##   2        29       0.6740397  0.3271363  0.5158683
```

```
##   2       30      0.6740397  0.3271363  0.5158683
##   2       31      0.6740397  0.3271363  0.5158683
##   2       32      0.6740397  0.3271363  0.5158683
##   2       33      0.6740397  0.3271363  0.5158683
##   2       34      0.6740397  0.3271363  0.5158683
##   2       35      0.6740397  0.3271363  0.5158683
##   2       36      0.6740397  0.3271363  0.5158683
##   2       37      0.6740397  0.3271363  0.5158683
##   2       38      0.6740397  0.3271363  0.5158683
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nprune = 6 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  0.7504817  0.3041580  0.4223146
##    2  0.6983824  0.3237618  0.4784021
##    3  0.6717419  0.3442214  0.4839236
##    4  0.6702850  0.3371634  0.4963434
##    5  0.6627934  0.3448450  0.4970748
##    6  0.6600855  0.3448719  0.4996920
##    7  0.6540051  0.3521529  0.4974915
##    8  0.6519397  0.3536865  0.4984599
##    9  0.6483577  0.3578728  0.4971793
##   10  0.6458723  0.3620134  0.4964883
##   11  0.6435687  0.3655187  0.4969729
##   12  0.6429914  0.3656200  0.4982548
##   13  0.6397289  0.3711190  0.4955805
##   14  0.6411480  0.3678476  0.4973248
##   15  0.6393988  0.3708496  0.4964588
```

```
##   16  0.6378593  0.3734042  0.4951719
##   17  0.6369138  0.3749404  0.4959561
##   18  0.6362271  0.3765123  0.4952884
##   19  0.6370582  0.3749337  0.4962376
##   20  0.6372569  0.3747771  0.4971556
##   21  0.6382742  0.3731375  0.4986083
##   22  0.6380964  0.3734679  0.4979368
##   23  0.6377412  0.3742550  0.4977067
##   24  0.6387696  0.3722746  0.4985198
##   25  0.6368451  0.3757202  0.4976829
##   26  0.6364275  0.3762866  0.4984160
##   27  0.6346862  0.3795795  0.4973558
##   28  0.6347058  0.3795358  0.4974800
##   29  0.6360878  0.3770104  0.4993224
##   30  0.6357780  0.3778721  0.4999014
##   31  0.6367139  0.3762620  0.5005842
##   32  0.6369187  0.3757665  0.5002004
##   33  0.6371917  0.3752852  0.5006800
##   34  0.6377712  0.3740873  0.5007256
##   35  0.6379469  0.3737611  0.5014462
##   36  0.6378240  0.3742385  0.5015524
##   37  0.6374811  0.3749206  0.5011665
##   38  0.6375055  0.3749360  0.5014616
##   39  0.6371916  0.3757942  0.5006347
##   40  0.6368473  0.3765730  0.5004336
##   41  0.6365946  0.3770658  0.5003279
##   42  0.6365554  0.3772882  0.5006586
##   43  0.6367931  0.3768451  0.5006802
##   44  0.6380737  0.3744045  0.5014973
##   45  0.6381101  0.3745029  0.5011388
##   46  0.6379189  0.3750057  0.5009169
##   47  0.6379198  0.3751492  0.5009907
##   48  0.6375034  0.3761452  0.5005620
##   49  0.6374270  0.3763801  0.5003075
##   50  0.6377491  0.3759204  0.5014729
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 27.
```

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  0.6424186  0.3663702  0.4834089
##   0.50  0.6360205  0.3784258  0.4756782
##   1.00  0.6303977  0.3892811  0.4703437
##   2.00  0.6281033  0.3952003  0.4644550
##   4.00  0.6342263  0.3896250  0.4665780
##
## Tuning parameter 'sigma' was held constant at a value of 0.09155044
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were sigma = 0.09155044 and C = 2.
```