ADS 503 - Team 7

Summer Purschke, Jacqueline Urenda, Oscar Gil

06/12/2022

```r
# 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)
```

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

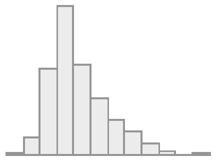
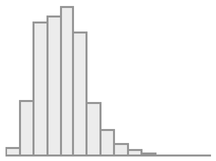
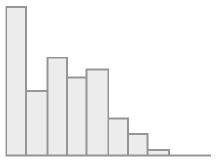
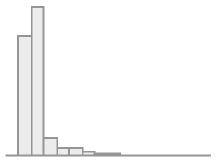
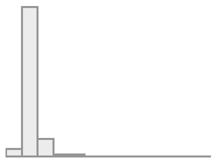
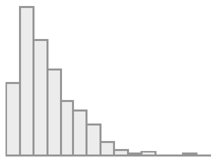
```r
wine <- read.csv(
  url("https://raw.githubusercontent.com/OscarG-DataSci/ADS503/main/winequality-red.csv")
                      , header = TRUE)
```

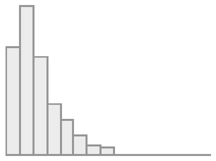
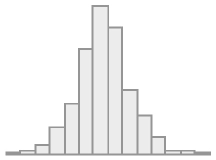
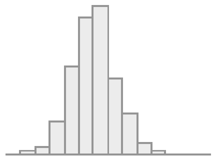
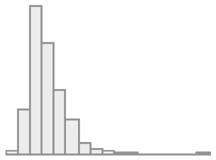
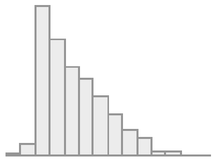
Data Summary

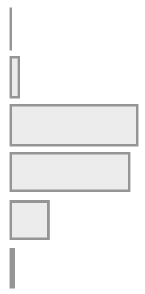
```r
dfSummary(wine,
          plain.ascii  = FALSE,
          style        = "grid",
          graph.magnif = 0.75,
          valid.col    = FALSE,
          tmp.img.dir  = "/tmp")
```

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

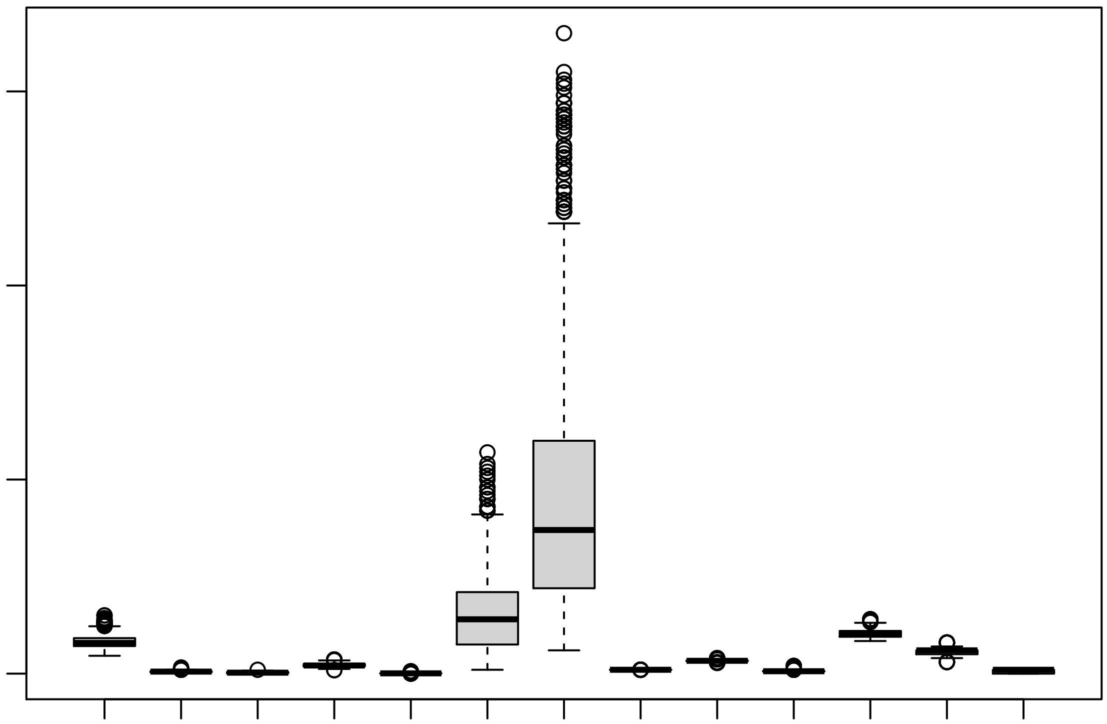
```
par(mar=c(1,1,1,1)) # to fix boxplot knit processing issues

# 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
```

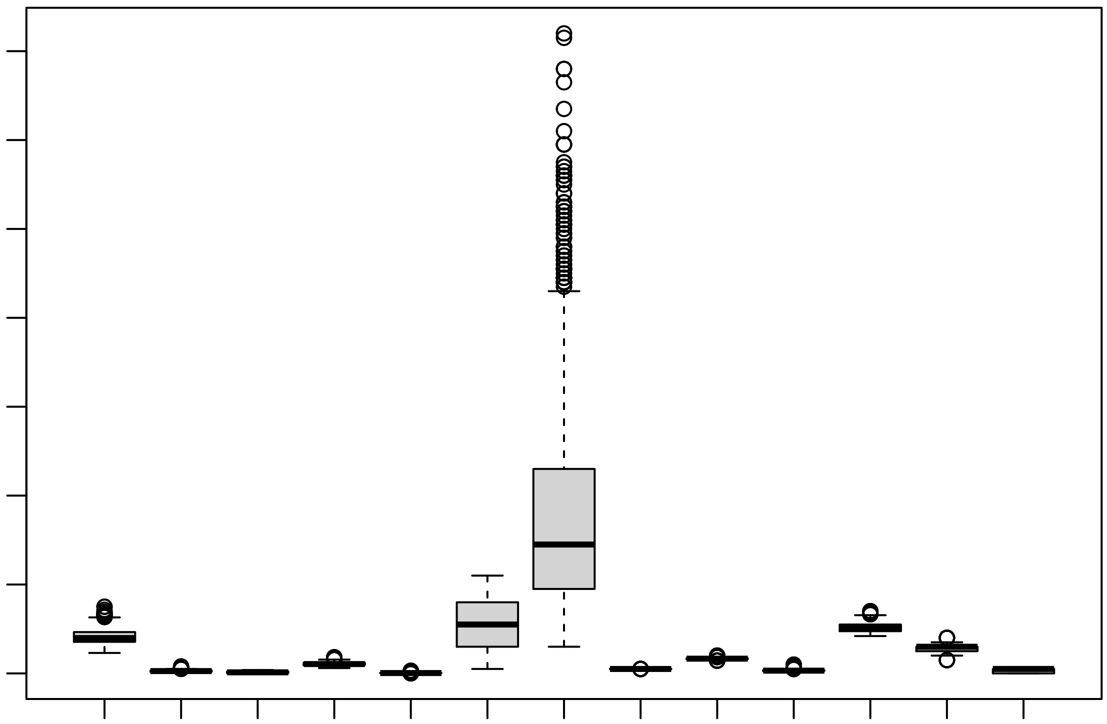
```
# Check for missing values in data set
wine %>% na.omit() %>% count() # there are no missing values
```

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

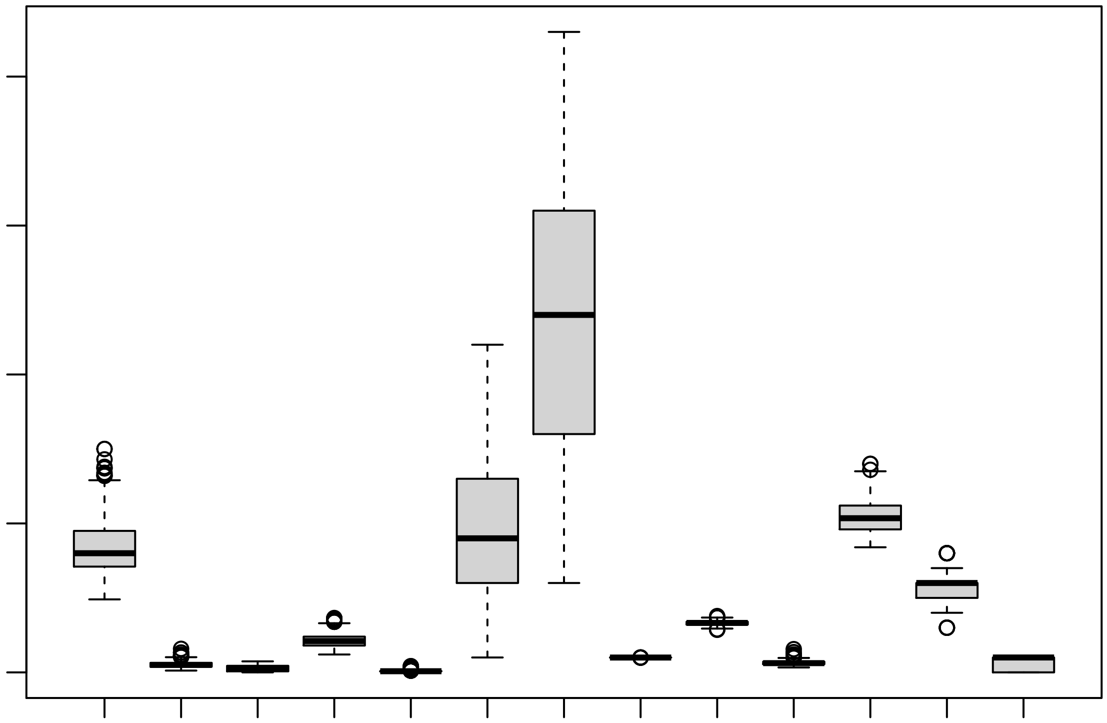
```
# Removing outliers for residual sugar:
Q <- quantile(wine$residual.sugar, probs=c(.25, .75), na.rm = FALSE)
iqr_rs <- IQR(wine$residual.sugar)
up_rs <-  Q[2]+1.5*iqr_rs # Upper Range
low_rs <- Q[1]-1.5*iqr_rs # Lower Range
eliminated_rs <- subset(wine, wine$residual.sugar > (Q[1] - 1.5*iqr_rs) &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



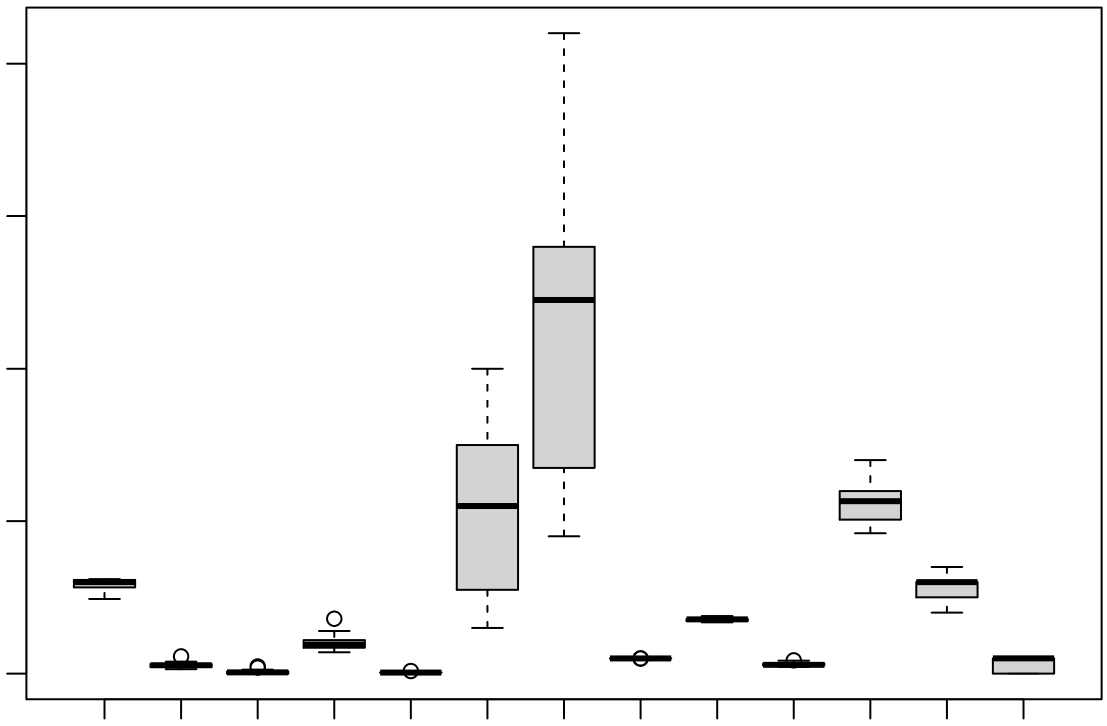
```
#Removing outliers for free.sulfur.dioxide:
Q2 <- quantile(wine$free.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_fs <- IQR(eliminated_rs$free.sulfur.dioxide)
up_fs <-  Q2[2]+1.5*iqr_fs # Upper Range
low_fs <- Q2[1]-1.5*iqr_fs # Lower Range
eliminated_fs <- subset(eliminated_rs, eliminated_rs$free.sulfur.dioxide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



```
#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)
```



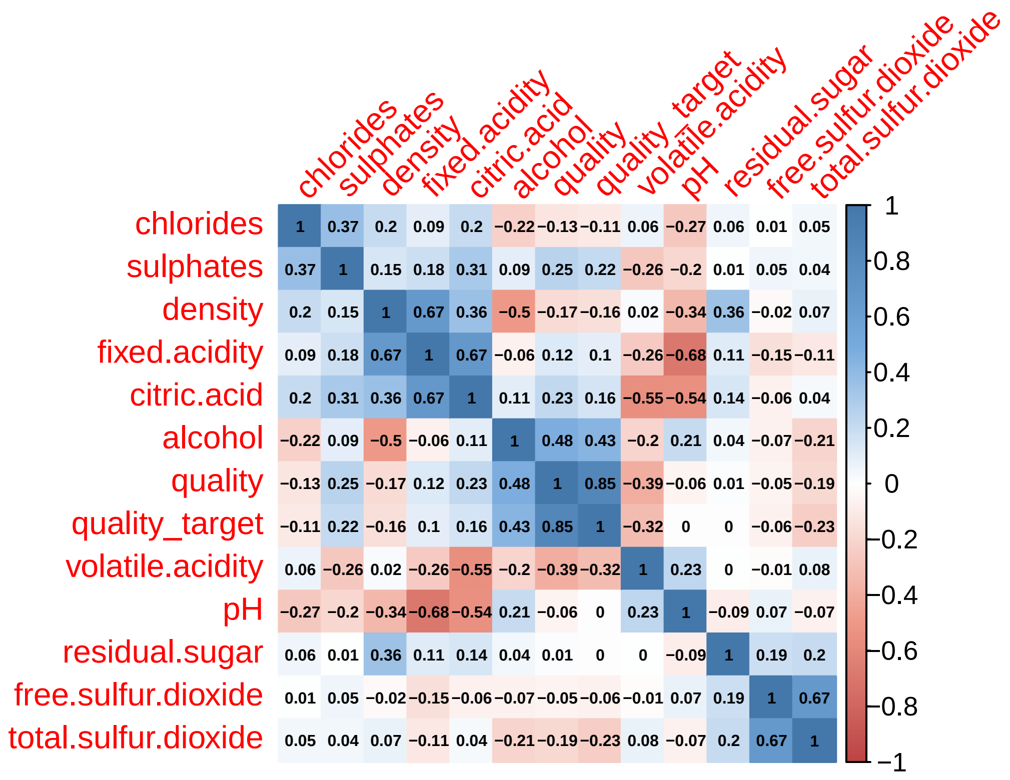
```
#Removing outliers for fixed.acidity:
Q4 <- quantile(wine$fixed.acidity, probs=c(.25, .75), na.rm = FALSE)
iqr_fa <- IQR(eliminated_ts$fixed.acidity)
up_fa <-  Q[2]+1.5*iqr_fa # Upper Range
low_fa <- Q[1]-1.5*iqr_fa # Lower Range
eliminated_fa <- subset(eliminated_ts, eliminated_ts$fixed.acidity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



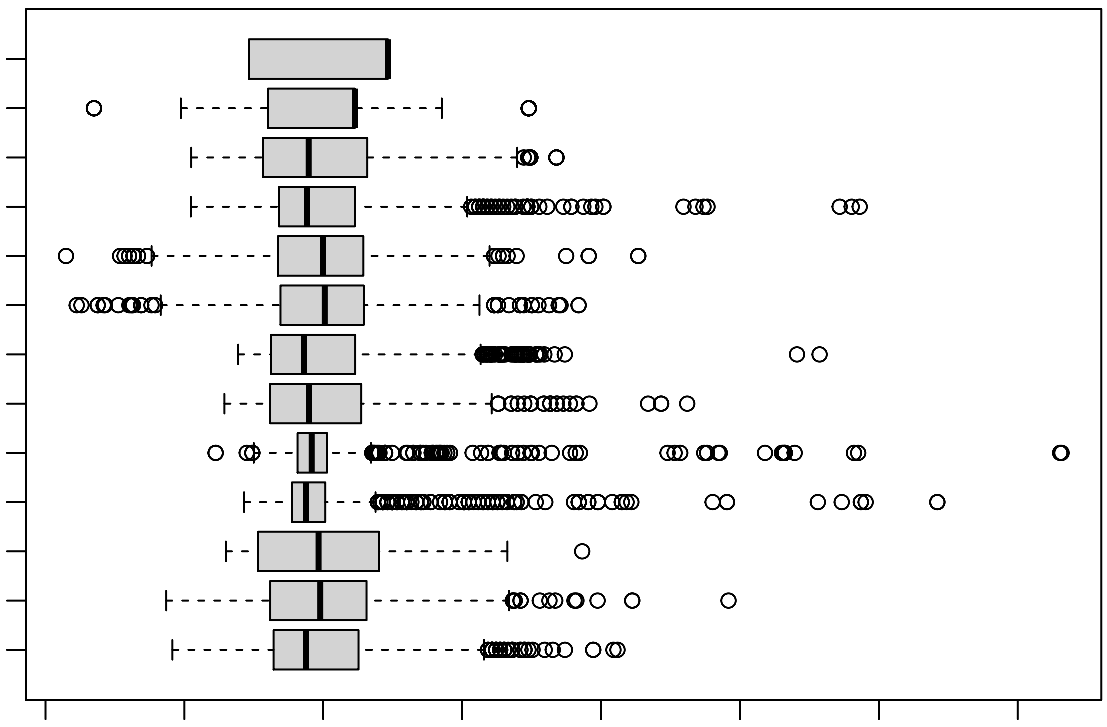
```
new_wine_data <- eliminated_fa

# 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
```

```
# 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

```r
# 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
```

```r
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
## -2.07422 -0.69261 -0.05795  0.76130  2.13602
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -3.818e-15  2.385e-02   0.000        1
## volatile.acidity -2.337e-01  2.509e-02  -9.313  < 2e-16 ***
## sulphates         1.220e-01  2.470e-02   4.938 8.93e-07 ***
## alcohol           3.841e-01  2.431e-02  15.796  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8535 on 1277 degrees of freedom
## Multiple R-squared:  0.2733, Adjusted R-squared:  0.2716
## F-statistic: 160.1 on 3 and 1277 DF,  p-value: < 2.2e-16
```

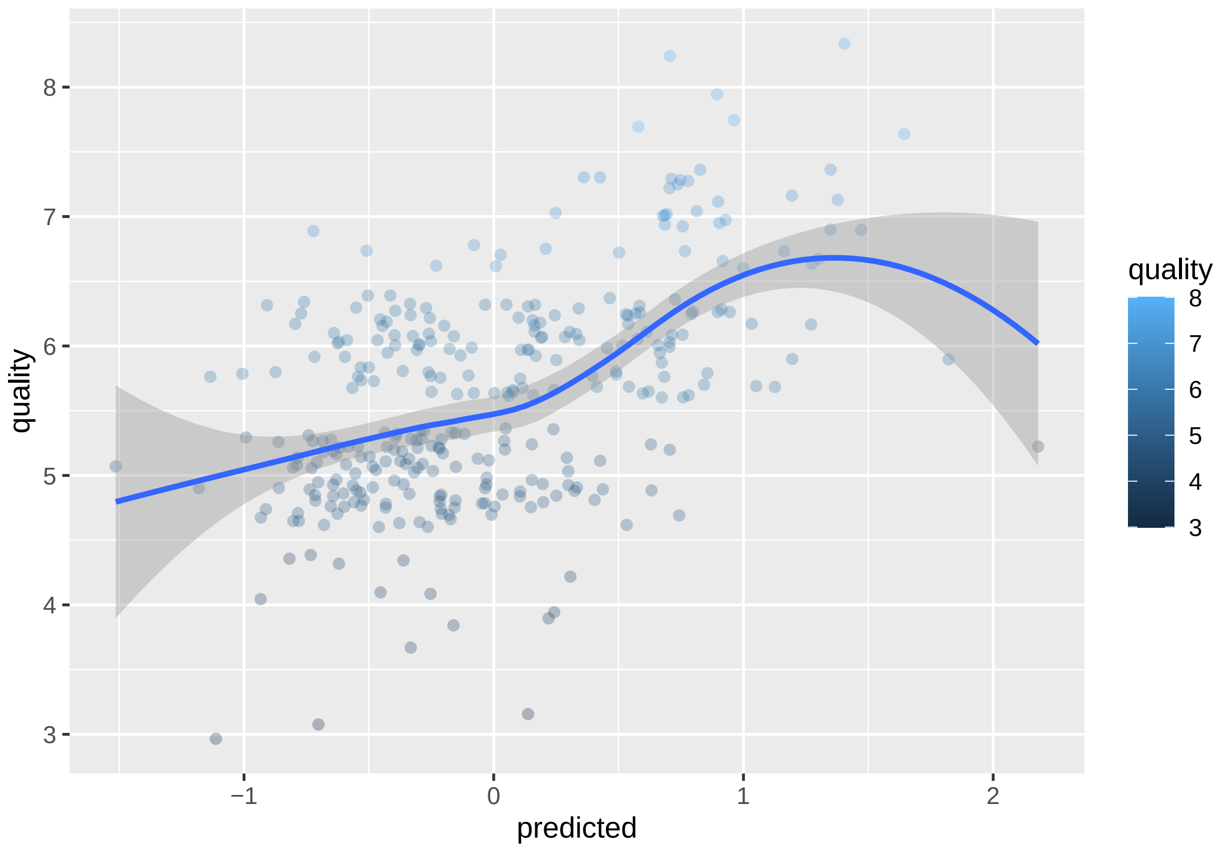
```
# Model using "quality" as target variable
lmodel2 <- lm(quality~ volatile.acidity + sulphates + alcohol, data = train_transformed)

summary(lmodel2)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + sulphates + alcohol,
##     data = train_transformed)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.01860 -0.47772 -0.06938  0.56533  2.72662
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.850e-15  2.261e-02   0.000        1
## volatile.acidity -2.885e-01  2.379e-02 -12.129  < 2e-16 ***
## sulphates         1.371e-01  2.342e-02   5.856 6.03e-09 ***
## alcohol           4.124e-01  2.305e-02  17.892  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8091 on 1277 degrees of freedom
## Multiple R-squared:  0.3469, Adjusted R-squared:  0.3453
## F-statistic: 226.1 on 3 and 1277 DF,  p-value: < 2.2e-16
```

```
# 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.   :-1.5138   Min.   :-3.1081   Min.   :0.08058
##  1st Qu.:-0.4639   1st Qu.:-2.0526   1st Qu.:1.12491
##  Median :-0.1398   Median :-1.7291   Median :1.44960
##  Mean   : 0.0000   Mean   :-1.5898   Mean   :1.58979
##  3rd Qu.: 0.4498   3rd Qu.:-1.1390   3rd Qu.:2.03853
##  Max.   : 2.1802   Max.   : 0.5808   Max.   :3.77952
```

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



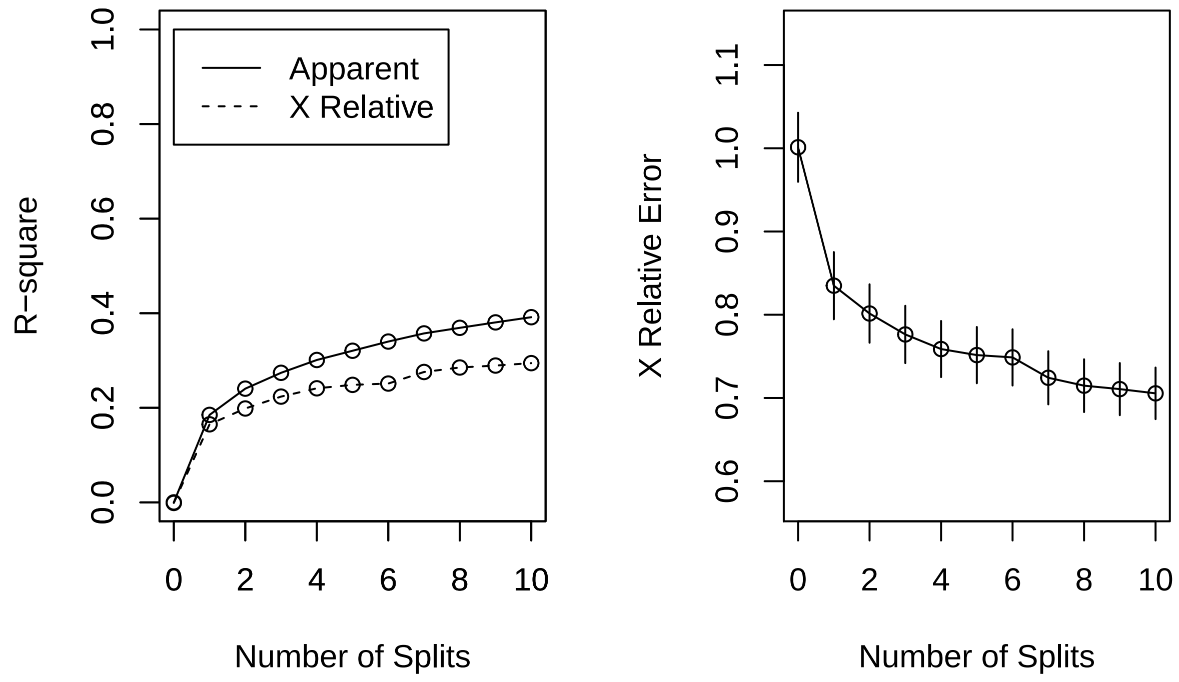
```
# The scatter plot supports the summary of the predicted interval, in the ranges of the fit,
# lower, and upper ranges. The R-squared value of 0.3283 of the model, indicates that this
# information can be predicted 33% of the time, with the data available, for the variance
# of the information.
```

CART

```
set.seed(4)
# Subset both train and test sets, to excluse "quality_target"
subset(train_transformed, select = -c(quality_target)) -> rf_wine_train
subset(test_transformed, select = -c(quality_target)) -> rf_wine_test

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

rpartTree2 <- as.party(rPartTree)

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

```
##
## Regression tree:
## rpart(formula = quality ~ ., data = rf_wine_train)
##
## Variables actually used in tree construction:
```

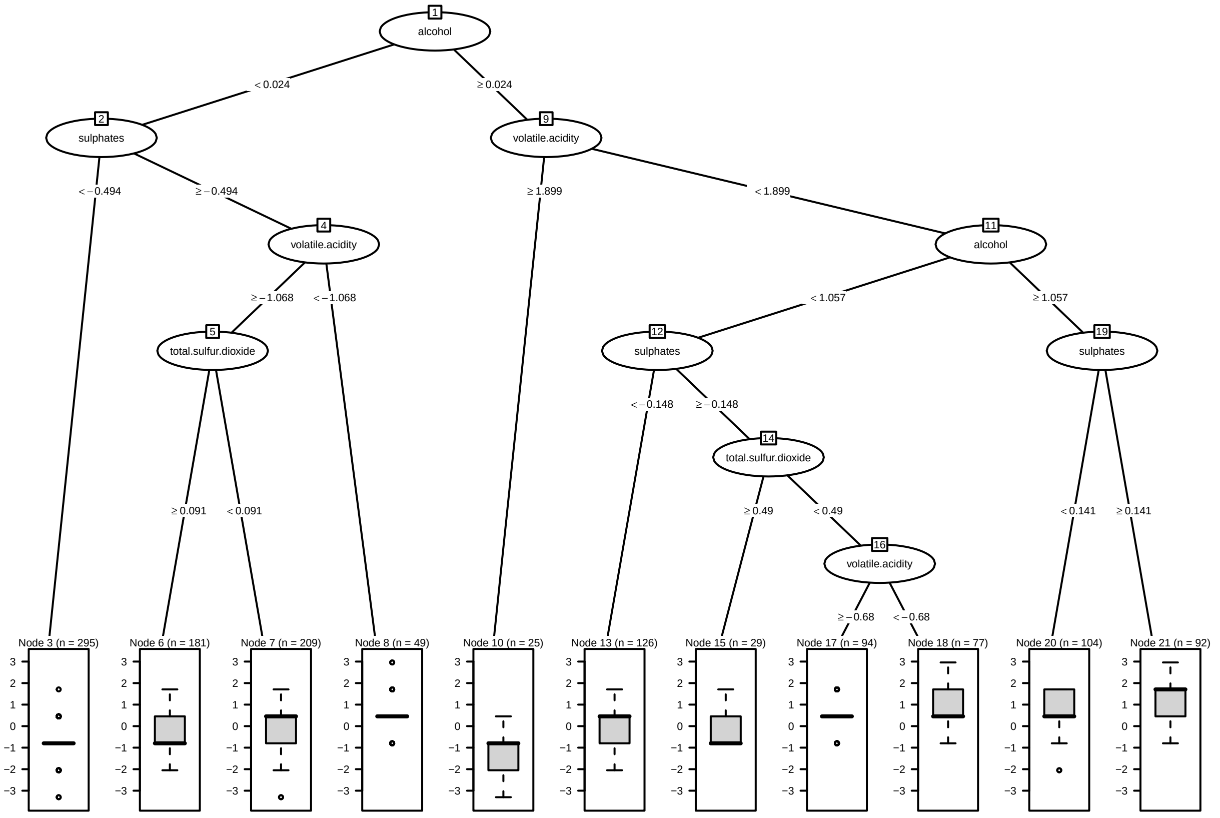
```
## [1] alcohol              sulphates            total.sulfur.dioxide
## [4] volatile.acidity
##
## Root node error: 1280/1281 = 0.99922
##
## n= 1281
##
##           CP nsplit rel error  xerror     xstd
## 1  0.185022      0   1.00000 1.00121 0.041490
## 2  0.055480      1   0.81498 0.83492 0.040396
## 3  0.033497      2   0.75950 0.80147 0.035118
## 4  0.027105      3   0.72600 0.77633 0.034420
## 5  0.019523      4   0.69890 0.75879 0.033713
## 6  0.019476      5   0.67937 0.75152 0.033808
## 7  0.017209      6   0.65990 0.74874 0.033799
## 8  0.011818      7   0.64269 0.72426 0.031918
## 9  0.011508      8   0.63087 0.71478 0.031714
## 10 0.010997      9   0.61936 0.71066 0.031343
## 11 0.010000     10   0.60837 0.70556 0.030991
```



```
# 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 < 0.02403
## |   |   [3] sulphates < -0.49395: -0.636 (n = 295, err = 133.5)
## |   |   [4] sulphates >= -0.49395
## |   |   |   [5] volatile.acidity >= -1.06828
## |   |   |   |   [6] total.sulfur.dioxide >= 0.09106: -0.486 (n = 181, err = 81.3)
## |   |   |   |   [7] total.sulfur.dioxide < 0.09106: -0.097 (n = 209, err = 140.4)
## |   |   |   [8] volatile.acidity < -1.06828: 0.480 (n = 49, err = 36.0)
## |   [9] alcohol >= 0.02403
## |   |   [10] volatile.acidity >= 1.89857: -1.148 (n = 25, err = 33.0)
## |   |   [11] volatile.acidity < 1.89857
## |   |   |   [12] alcohol < 1.05677
## |   |   |   |   [13] sulphates < -0.14788: 0.027 (n = 126, err = 97.7)
## |   |   |   |   [14] sulphates >= -0.14788
## |   |   |   |   |   [15] total.sulfur.dioxide >= 0.49048: -0.107 (n = 29, err = 20.7)
## |   |   |   |   |   [16] total.sulfur.dioxide < 0.49048
## |   |   |   |   |   |   [17] volatile.acidity >= -0.6801: 0.415 (n = 94, err = 48.5)
## |   |   |   |   |   |   [18] volatile.acidity < -0.6801: 0.991 (n = 77, err = 54.7)
## |   |   |   [19] alcohol >= 1.05677
## |   |   |   |   [20] sulphates < 0.1405: 0.611 (n = 104, err = 74.3)
## |   |   |   |   [21] sulphates >= 0.1405: 1.326 (n = 92, err = 58.8)
##
## Number of inner nodes:    10
## Number of terminal nodes: 11
```

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



Random Forest

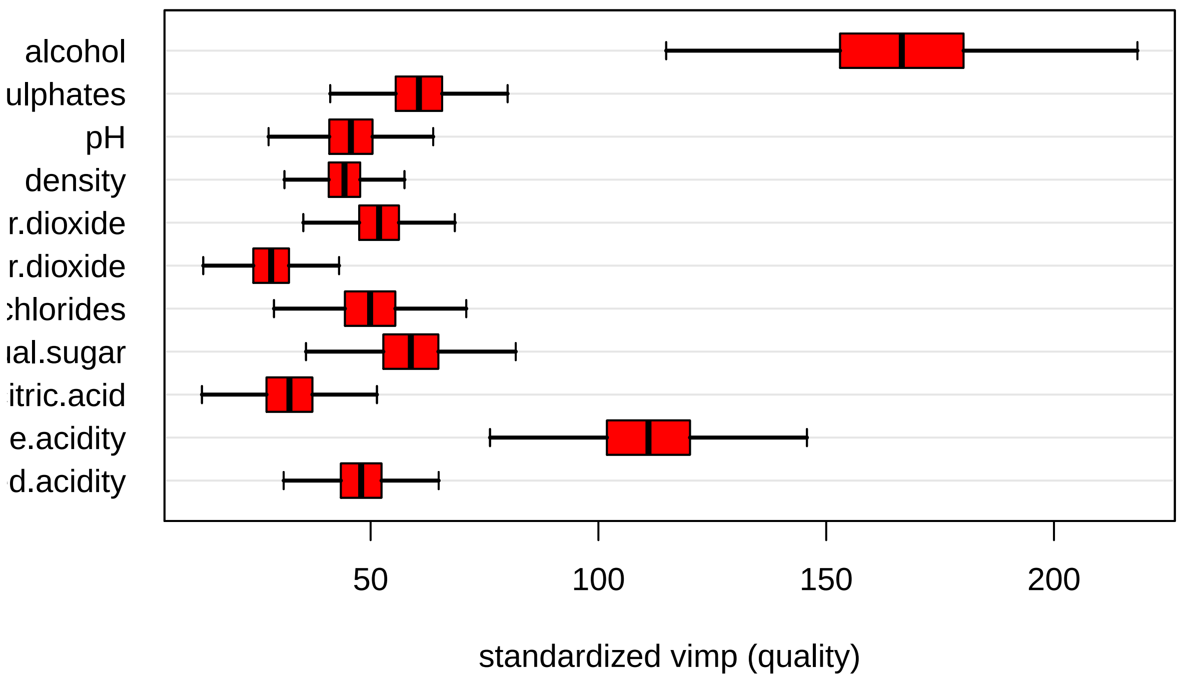
```
set.seed(4)

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

print(rf)
```

```
##                          Sample size: 1281
##                      Number of trees: 500
##            Forest terminal node size: 5
##        Average no. of terminal nodes: 152.262
## 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-R
##                               Family: regr
##                       Splitting rule: mse
##                      (OOB) R squared: 0.47760991
##    (OOB) Requested performance error: 0.52239009
```

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

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

```
##                          lower      mean     upper       pvalue signif
## fixed.acidity         34.98371  47.92706  60.87042 1.972777e-13   TRUE
## volatile.acidity      84.51964 110.98665 137.45365 1.026847e-16   TRUE
## citric.acid           17.56287  32.17938  46.79589 7.979662e-06   TRUE
## residual.sugar        41.31289  58.83278  76.35266 2.325943e-11   TRUE
## chlorides             33.83073  49.88923  65.94774 5.678963e-10   TRUE
## free.sulfur.dioxide   16.82409  28.16609  39.50809 5.657645e-07   TRUE
## total.sulfur.dioxide  39.20016  51.85320  64.50624 4.791123e-16   TRUE
## density               34.23564  44.25832  54.28099 2.467949e-18   TRUE
## pH                    31.91963  45.66939  59.41915 3.758552e-11   TRUE
## sulphates             45.78972  60.60198  75.41424 5.335758e-16   TRUE
## alcohol              127.23911 166.59546 205.95181 5.359853e-17   TRUE
```

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



```
# 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: 152.262
##          Total no. of grow variables: 11
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 810
##                             Analysis: RF-R
##                               Family: regr
##                            R squared: 0.41116769
##          Requested performance error: 0.58883231
```

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      0.7998011  0.3632430  0.6276163
##   2      0.7993504  0.3641251  0.6271541
##   3      0.7994295  0.3640622  0.6267032
##
## 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 0.6128238    RSS 749.8859    GRSq 0.3876546    RSq 0.4141517
```

```
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##             data=train_transformed)
##
##                                      coefficients
## (Intercept)                            30.4984030
## h(1.73221-volatile.acidity)             0.1976638
## h(volatile.acidity-1.73221)            -0.4644967
## h(chlorides- -0.960266)                 3.2533522
## h(chlorides-0.0972729)                 -0.6250823
## h(1.39886-chlorides)                    3.0770129
## h(chlorides-1.39886)                   -2.4570771
## h(chlorides-5.14092)                   -0.4076001
## h(total.sulfur.dioxide- -1.1368)      -10.2398712
## h(2.53198-total.sulfur.dioxide)       -10.0963829
## h(total.sulfur.dioxide-2.53198)        10.5813790
## h(1.0345-sulphates)                    -0.4128928
## h(sulphates-1.0345)                    -0.1156520
## h(alcohol-0.634282)                     0.2129655
## h(1.76091-alcohol)                     -0.3017427
##
## 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 0.6128238    RSS 749.8859    GRSq 0.3876546    RSq 0.4141517
```

```
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      0.8767725  0.2286331  0.6983669
##   1        3      0.8281246  0.3129505  0.6469912
##   1        4      0.7986723  0.3598732  0.6239193
##   1        5      0.7973767  0.3627609  0.6213442
##   1        6      0.7954260  0.3658170  0.6189612
##   1        7      0.7933841  0.3690469  0.6184932
##   1        8      0.7955313  0.3658275  0.6195302
##   1        9      0.7933719  0.3687607  0.6178130
##   1       10      0.7943316  0.3680007  0.6182851
##   1       11      0.7949170  0.3675983  0.6183133
##   1       12      0.7955150  0.3670106  0.6189397
##   1       13      0.7956669  0.3668727  0.6185138
##   1       14      0.7961519  0.3667770  0.6185384
##   1       15      0.7962221  0.3667518  0.6180695
##   1       16      0.7962221  0.3667518  0.6180695
##   1       17      0.7962221  0.3667518  0.6180695
##   1       18      0.7962221  0.3667518  0.6180695
##   1       19      0.7962221  0.3667518  0.6180695
##   1       20      0.7962221  0.3667518  0.6180695
##   1       21      0.7962221  0.3667518  0.6180695
##   1       22      0.7962221  0.3667518  0.6180695
##   1       23      0.7962221  0.3667518  0.6180695
##   1       24      0.7962221  0.3667518  0.6180695
##   1       25      0.7962221  0.3667518  0.6180695
##   1       26      0.7962221  0.3667518  0.6180695
##   1       27      0.7962221  0.3667518  0.6180695
##   1       28      0.7962221  0.3667518  0.6180695
##   1        29      0.7962221  0.3667518  0.6180695
##   1        30      0.7962221  0.3667518  0.6180695
##   1        31      0.7962221  0.3667518  0.6180695
##   1        32      0.7962221  0.3667518  0.6180695
##   1        33      0.7962221  0.3667518  0.6180695
##   1        34      0.7962221  0.3667518  0.6180695
##   1        35      0.7962221  0.3667518  0.6180695
##   1        36      0.7962221  0.3667518  0.6180695
##   1        37      0.7962221  0.3667518  0.6180695
##   1        38      0.7962221  0.3667518  0.6180695
##   2         2      0.8778478  0.2264638  0.6948882
##   2         3      0.8217598  0.3220552  0.6392462
##   2         4      0.7891923  0.3744973  0.6172698
##   2         5      0.7846781  0.3822662  0.6125307
##   2         6      0.7837052  0.3840279  0.6103097
##   2         7      0.7852206  0.3823850  0.6102977
##   2         8      0.7874277  0.3797090  0.6117221
##   2         9      0.7910195  0.3749172  0.6135831
##   2        10      0.7948285  0.3699249  0.6152409
##   2        11      0.7987378  0.3648440  0.6158673
##   2        12      0.8045943  0.3572502  0.6164939
##   2        13      0.8044160  0.3578839  0.6160370
##   2        14      0.8041707  0.3588251  0.6158924
##   2        15      0.8048573  0.3580218  0.6163916
##   2        16      0.8052540  0.3575887  0.6162449
##   2        17      0.8057625  0.3572440  0.6164627
##   2        18      0.8058331  0.3571200  0.6164736
##   2        19      0.8058331  0.3571200  0.6164736
##   2        20      0.8058331  0.3571200  0.6164736
##   2        21      0.8058331  0.3571200  0.6164736
##   2        22      0.8058331  0.3571200  0.6164736
##   2        23      0.8058331  0.3571200  0.6164736
##   2        24      0.8058331  0.3571200  0.6164736
##   2        25      0.8058331  0.3571200  0.6164736
##   2        26      0.8058331  0.3571200  0.6164736
##   2        27      0.8058331  0.3571200  0.6164736
##   2        28      0.8058331  0.3571200  0.6164736
##   2        29      0.8058331  0.3571200  0.6164736
##   2        30      0.8058331  0.3571200  0.6164736
##   2        31      0.8058331  0.3571200  0.6164736
##   2        32      0.8058331  0.3571200  0.6164736
##   2        33      0.8058331  0.3571200  0.6164736
##   2        34      0.8058331  0.3571200  0.6164736
##   2        35      0.8058331  0.3571200  0.6164736
##   2        36      0.8058331  0.3571200  0.6164736
##   2        37      0.8058331  0.3571200  0.6164736
##   2        38      0.8058331  0.3571200  0.6164736
##
## 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 =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  0.9486037  0.3101279  0.5521179
##    2  0.8946471  0.3055655  0.6207649
##    3  0.8596248  0.3173637  0.6192937
##    4  0.8384012  0.3314930  0.6215991
##    5  0.8202779  0.3497374  0.6164375
##    6  0.8096196  0.3604723  0.6099141
##    7  0.7961261  0.3766117  0.6026718
##    8  0.7917952  0.3808891  0.6029282
##    9  0.7930582  0.3770572  0.6082138
##   10  0.7915187  0.3787846  0.6105603
##   11  0.7868061  0.3848931  0.6104114
##   12  0.7883179  0.3813246  0.6116129
##   13  0.7877965  0.3815857  0.6121286
##   14  0.7869724  0.3826114  0.6130522
##   15  0.7867557  0.3823977  0.6126466
##   16  0.7862693  0.3831196  0.6100775
##   17  0.7877585  0.3804738  0.6100929
##   18  0.7882592  0.3799676  0.6112571
##   19  0.7858700  0.3836689  0.6103664
##   20  0.7853813  0.3844932  0.6113662
##   21  0.7863263  0.3826412  0.6120682
##   22  0.7866219  0.3821987  0.6121664
##   23  0.7864929  0.3826019  0.6124847
##   24  0.7866213  0.3827113  0.6129548
##   25  0.7839553  0.3868342  0.6113461
##   26  0.7830884  0.3882729  0.6103765
##   27  0.7840037  0.3868623  0.6121240
##   28  0.7832355  0.3877893  0.6114887
##   29  0.7828527  0.3886391  0.6111209
##   30  0.7815061  0.3908631  0.6108329
##   31  0.7816921  0.3905510  0.6112563
##   32  0.7813698  0.3911467  0.6116271
##   33  0.7824046  0.3896750  0.6119544
##   34  0.7812635  0.3915225  0.6107396
##   35  0.7821713  0.3899855  0.6115484
##   36  0.7829359  0.3888554  0.6126616
##   37  0.7826544  0.3894240  0.6128846
##   38  0.7825614  0.3898820  0.6133241
##   39  0.7824786  0.3901287  0.6128711
##   40  0.7819865  0.3911644  0.6130563
##   41  0.7831124  0.3894069  0.6142938
##   42  0.7831814  0.3892890  0.6141271
##   43  0.7825908  0.3904415  0.6145867
##   44  0.7828766  0.3900157  0.6150421
##   45  0.7832661  0.3894957  0.6155824
##   46  0.7831180  0.3896859  0.6153044
##   47  0.7826061  0.3905436  0.6153951
##   48  0.7814354  0.3926554  0.6139669
##   49  0.7817824  0.3922137  0.6144282
##   50  0.7819978  0.3919336  0.6143947
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 34.
```

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  0.7807204  0.3936554  0.5841701
##   0.50  0.7724359  0.4053327  0.5722882
##   1.00  0.7748364  0.4034222  0.5722157
##   2.00  0.7804447  0.3991023  0.5747985
##   4.00  0.7910344  0.3920209  0.5818857
##
## Tuning parameter 'sigma' was held constant at a value of 0.1020787
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were sigma = 0.1020787 and C = 0.5.
```