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.

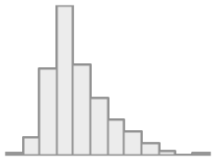
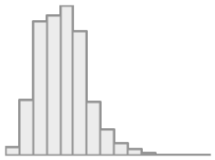
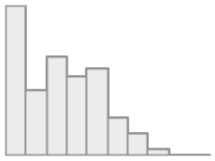
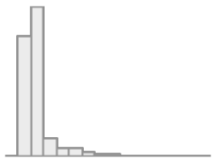
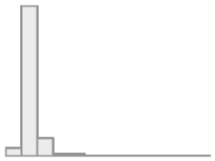
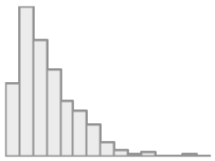
```
wine <- read.csv(
  url("https://raw.githubusercontent.com/OscarG-DataSci/ADS503/main/winequality-red.csv")
                      , header = TRUE)
```

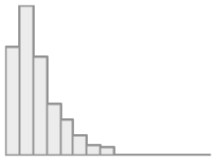
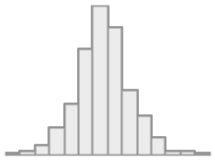
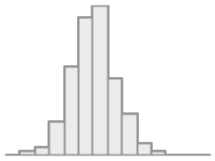
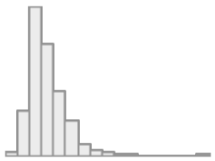
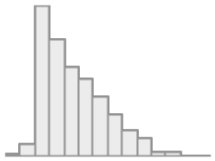
Data Summary

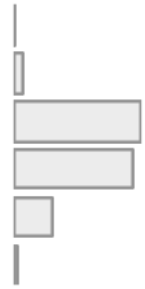
```
dfSummary(wine,
          plain.ascii  = FALSE,
          style        = "grid",
          graph.magnif = 0.75,
          valid.col    = FALSE,
          tmp.img.dir  = "/tmp")
```

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

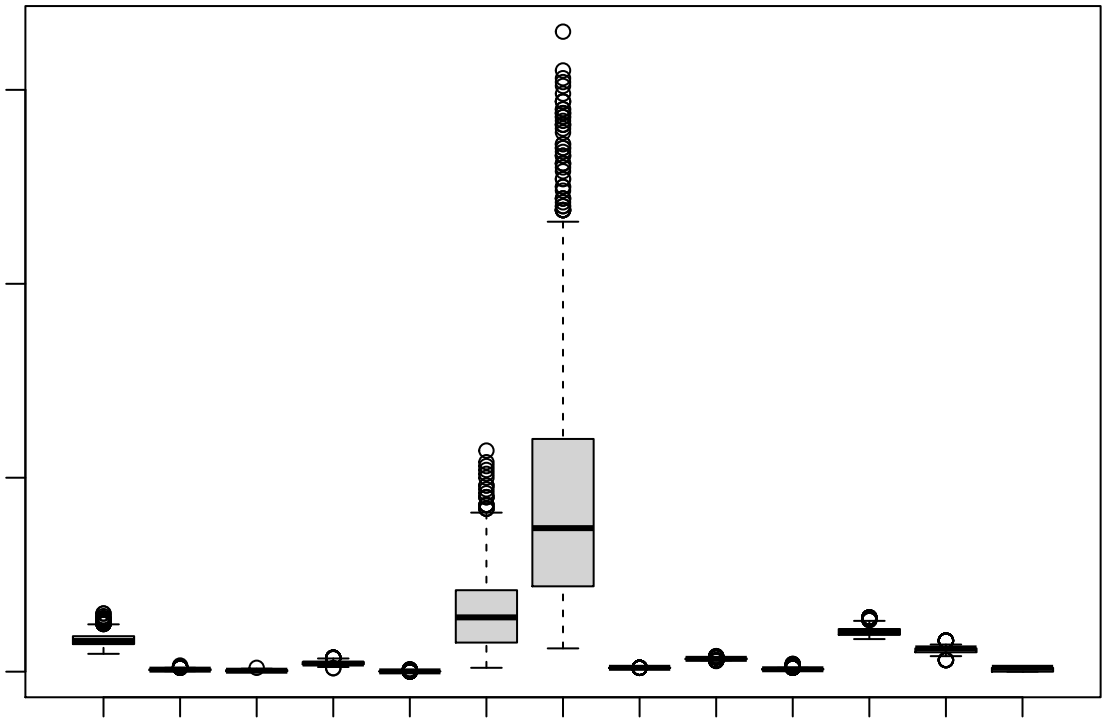
```
par(mar=c(1,1,1,1)) # to fix boxplot knit processing issues

# 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
```

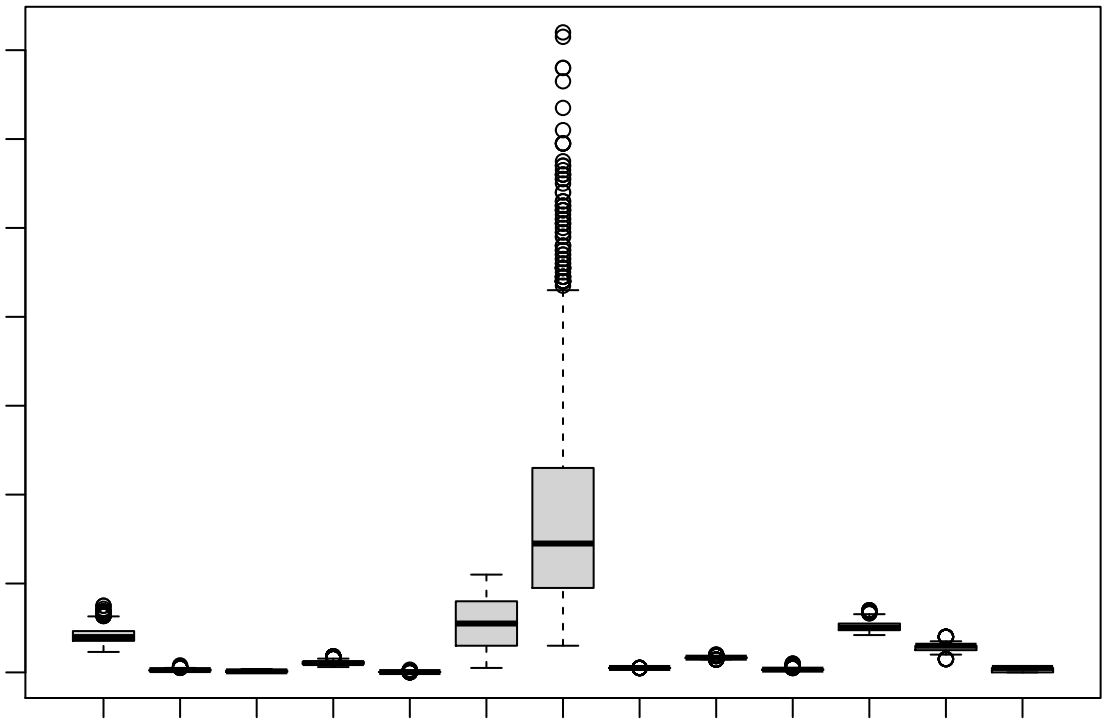
```
# Check for missing values in data set
wine %>% na.omit() %>% count() # there are no missing values
```

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

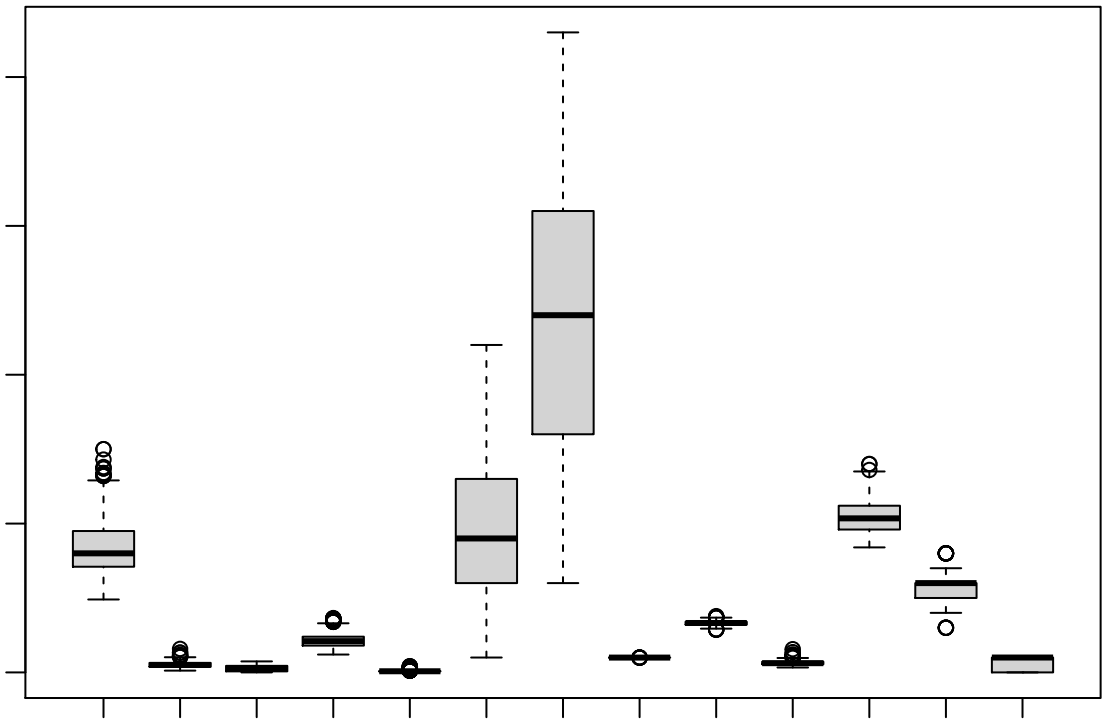
```
# Removing outliers for residual sugar:
Q <- quantile(wine$residual.sugar, probs=c(.25, .75), na.rm = FALSE)
iqr_rs <- IQR(wine$residual.sugar)
up_rs <-  Q[2]+1.5*iqr_rs # Upper Range
low_rs <- Q[1]-1.5*iqr_rs # Lower Range
eliminated_rs <- subset(wine, wine$residual.sugar > (Q[1] - 1.5*iqr_rs) &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



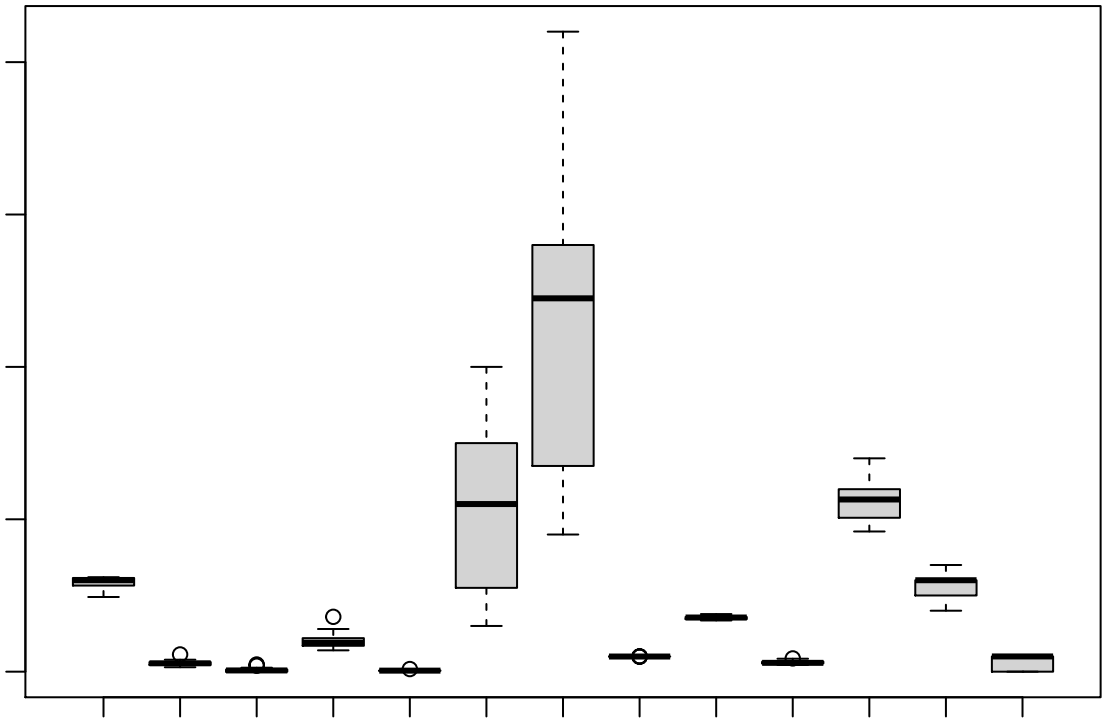
```r
#Removing outliers for free.sulfur.dioxide:
Q2 <- quantile(wine$free.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_fs <- IQR(eliminated_rs$free.sulfur.dioxide)
up_fs <-  Q2[2]+1.5*iqr_fs # Upper Range
low_fs <- Q2[1]-1.5*iqr_fs # Lower Range
eliminated_fs <- subset(eliminated_rs, eliminated_rs$free.sulfur.dioxide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



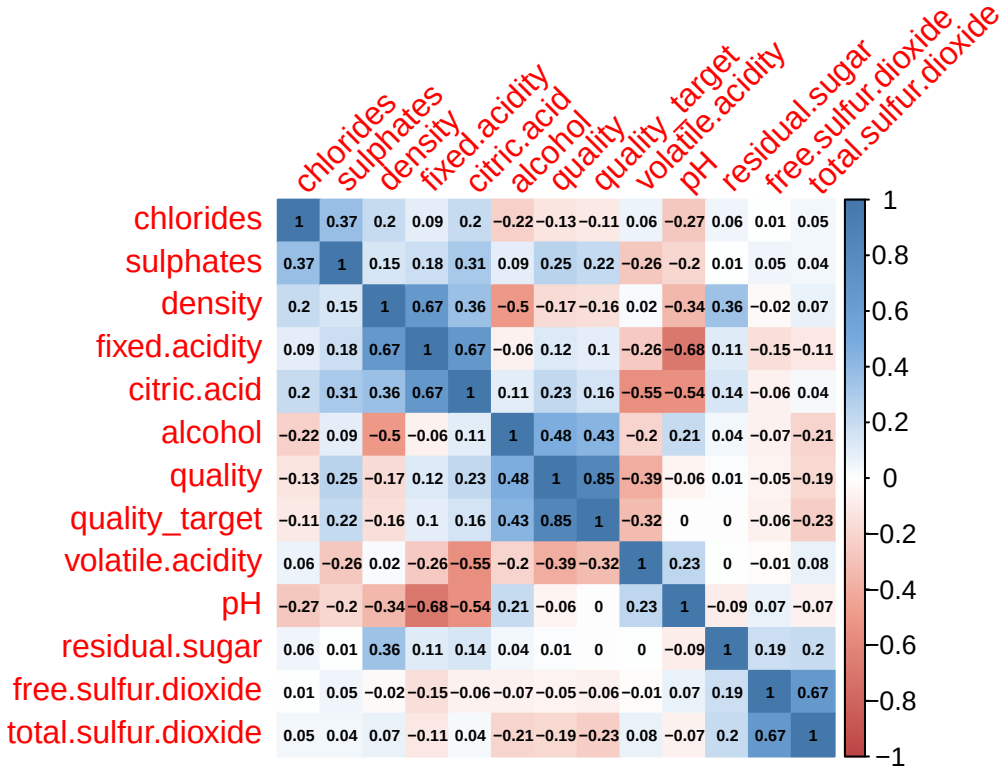
```r
#Removing outliers for fixed.acidity:
Q4 <- quantile(wine$fixed.acidity, probs=c(.25, .75), na.rm = FALSE)
iqr_fa <- IQR(eliminated_ts$fixed.acidity)
up_fa <-  Q[2]+1.5*iqr_fa # Upper Range
low_fa <- Q[1]-1.5*iqr_fa # Lower Range
eliminated_fa <- subset(eliminated_ts, eliminated_ts$fixed.acidity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



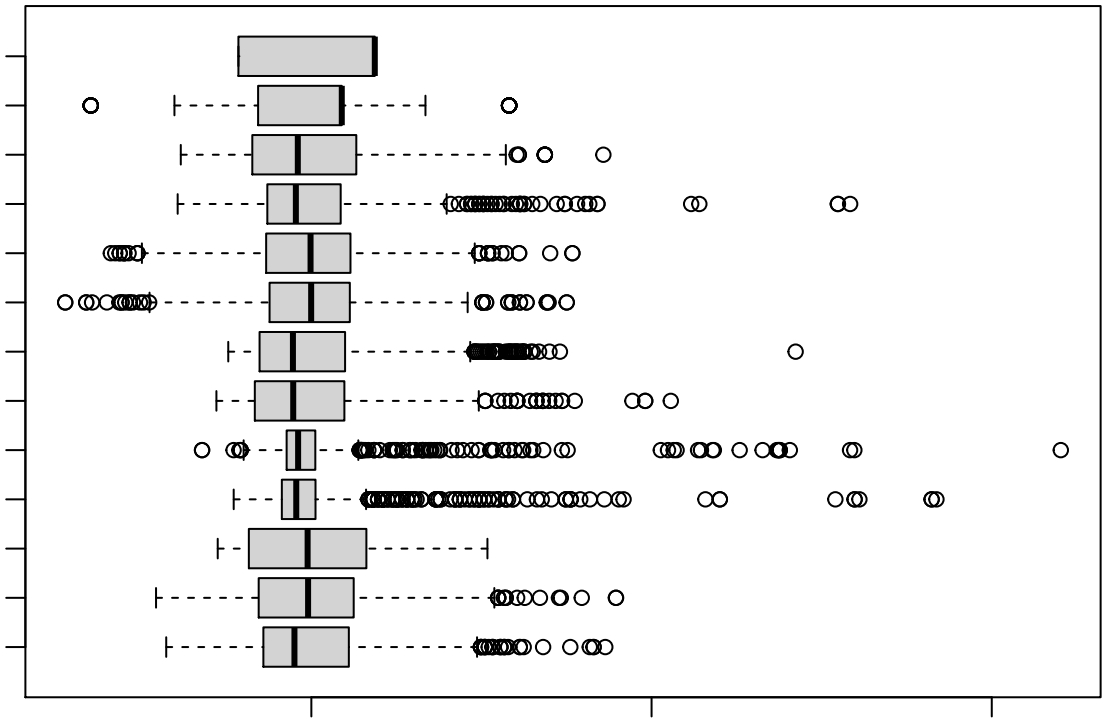
```r
new_wine_data <- eliminated_fa

# 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
## -3.00967 -0.71610 -0.03202  0.75503  2.07175
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -3.737e-15  2.405e-02   0.000        1
## volatile.acidity -2.094e-01  2.538e-02  -8.249 3.95e-16 ***
## sulphates         1.381e-01  2.502e-02   5.521 4.07e-08 ***
```

```
## alcohol           3.706e-01  2.467e-02  15.020  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8608 on 1277 degrees of freedom
## Multiple R-squared:  0.2608, Adjusted R-squared:  0.2591
## F-statistic: 150.2 on 3 and 1277 DF,  p-value: < 2.2e-16
```

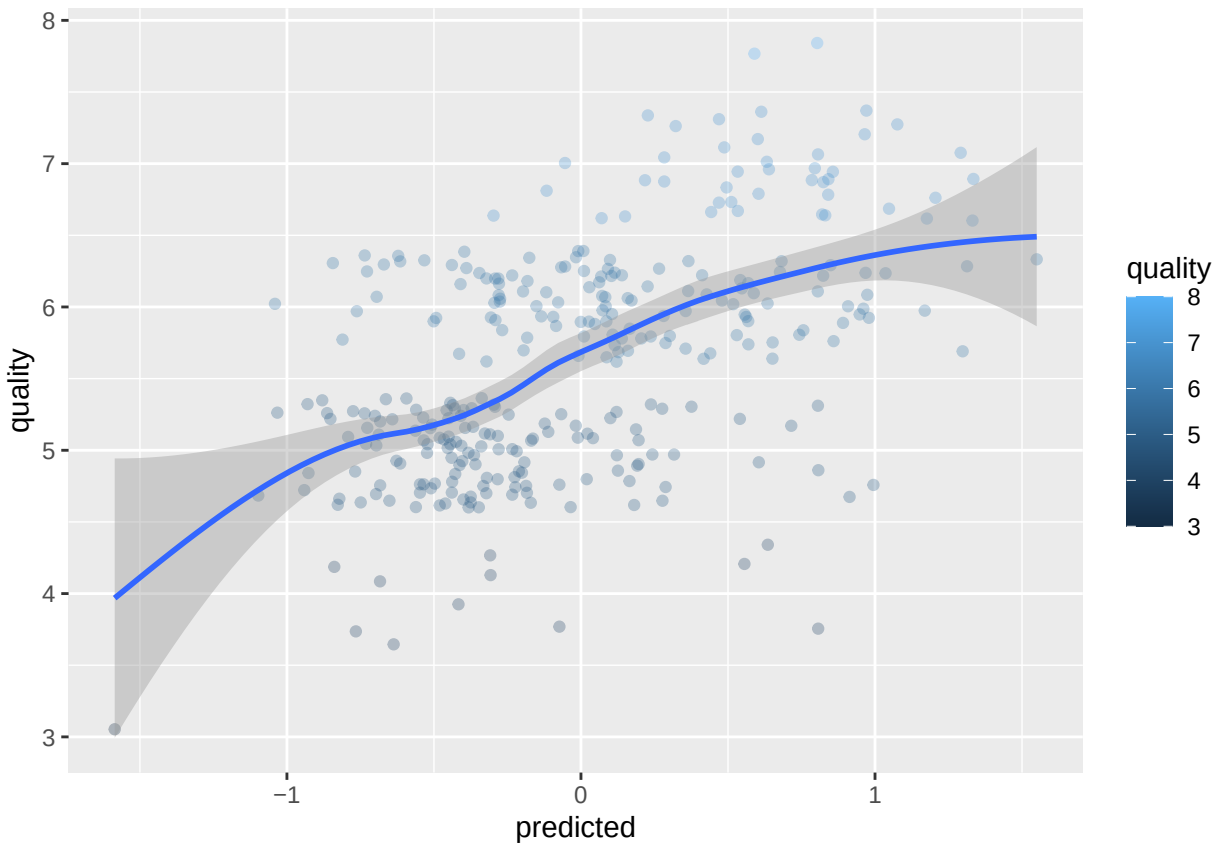
```r
# Model using "quality" as target variable
lmodel2 <- lm(quality~ volatile.acidity + sulphates + alcohol, data = train_transformed)

summary(lmodel2)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + sulphates + alcohol,
##     data = train_transformed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3627 -0.4659 -0.0807  0.5825  2.6769
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -5.684e-16  2.273e-02   0.000        1
## volatile.acidity -2.563e-01  2.399e-02 -10.683  < 2e-16 ***
## sulphates         1.592e-01  2.364e-02   6.733  2.5e-11 ***
## alcohol           4.087e-01  2.332e-02  17.527  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8135 on 1277 degrees of freedom
## Multiple R-squared:  0.3397, Adjusted R-squared:  0.3382
## F-statistic:   219 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.   :-1.58572   Min.   :-3.20536   Min.   :0.03391
##  1st Qu.:-0.41602   1st Qu.:-2.01530   1st Qu.:1.18282
##  Median :-0.07018   Median :-1.66892   Median :1.52813
##  Mean   : 0.00000   Mean   :-1.59850   Mean   :1.59850
##  3rd Qu.: 0.41586   3rd Qu.:-1.18207   3rd Qu.:2.01379
##  Max.   : 1.55001   Max.   :-0.05255   Max.   :3.15257
```

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



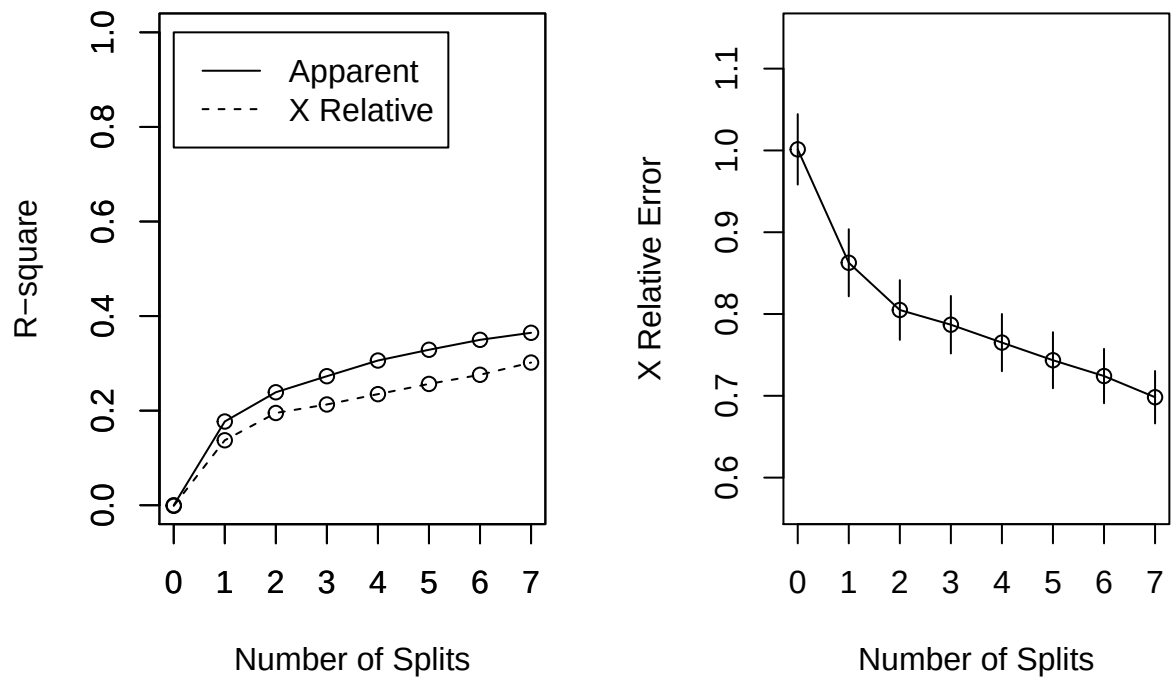

```
# The scatter plot supports the summary of the predicted interval, in the ranges of the fit,
# lower, and upper ranges. The R-squared value of 0.3283 of the model, indicates that this
# information can be predicted 33% of the time, with the data available, for the variance
# of the information.
```

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

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        volatile.acidity
##
## Root node error: 846.48/1281 = 0.6608
##
## n= 1281
##
##         CP nsplit rel error  xerror     xstd
## 1 0.177055      0   1.00000 1.00133 0.043043
## 2 0.061856      1   0.82294 0.86268 0.041009
## 3 0.033949      2   0.76109 0.80492 0.036499
## 4 0.033148      3   0.72714 0.78692 0.035181
## 5 0.022754      4   0.69399 0.76507 0.034959
## 6 0.021017      5   0.67124 0.74352 0.034234
## 7 0.014882      6   0.65022 0.72416 0.033199
## 8 0.010000      7   0.63534 0.69828 0.032057
```

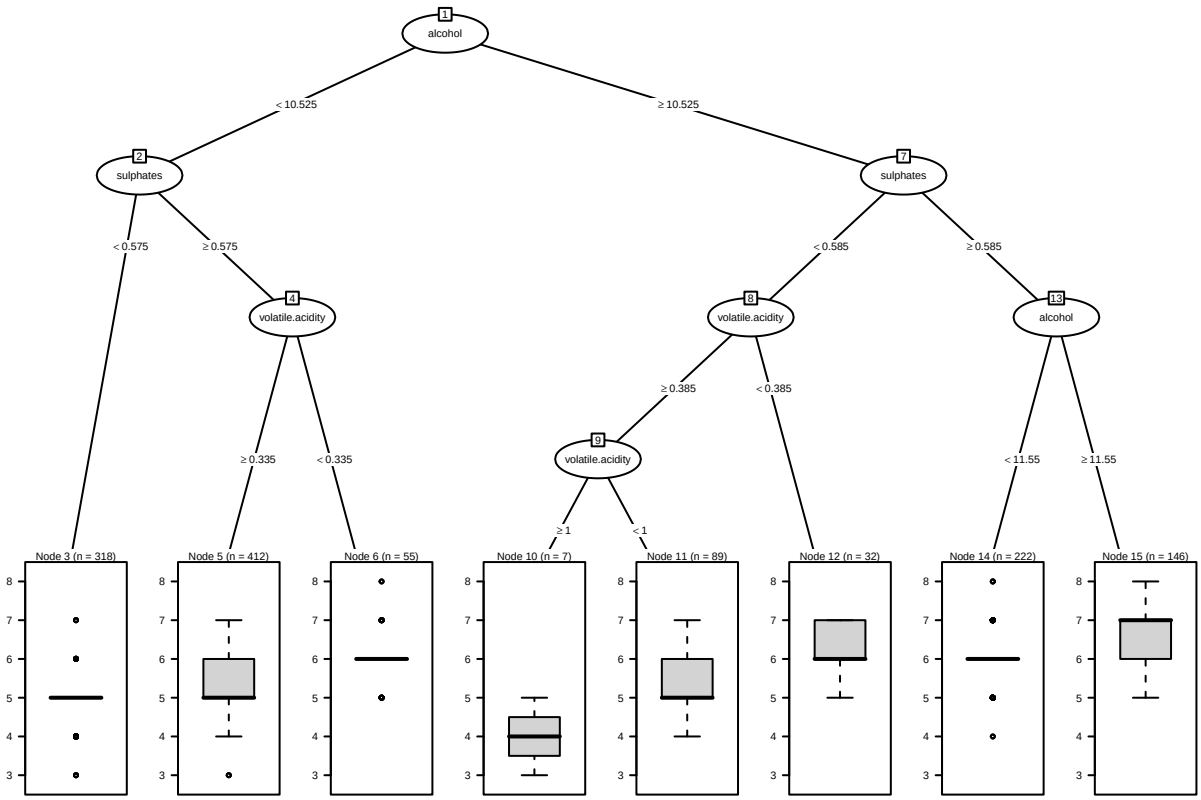



```r
# Results
rpartTree2
```

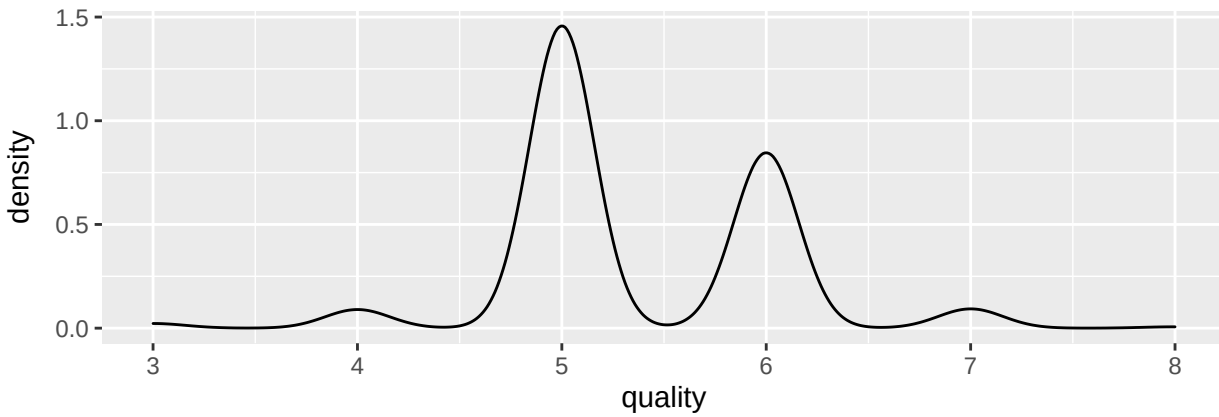
```
##
## Model formula:
## quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol
##
## Fitted party:
## [1] root
## |   [2] alcohol < 10.525
## |   |   [3] sulphates < 0.575: 5.135 (n = 318, err = 111.2)
## |   |   [4] sulphates >= 0.575
## |   |   |   [5] volatile.acidity >= 0.335: 5.449 (n = 412, err = 161.9)
## |   |   |   [6] volatile.acidity < 0.335: 6.055 (n = 55, err = 30.8)
```

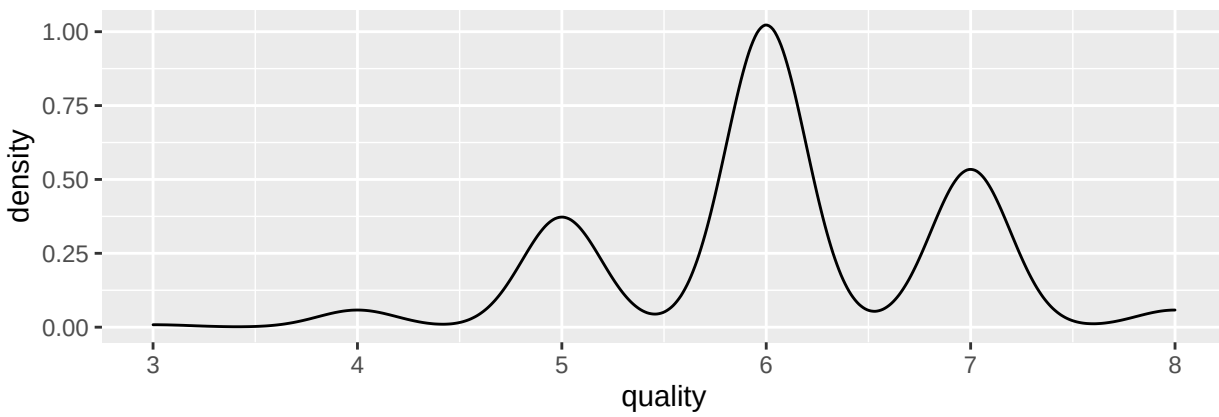
```
## |   [7] alcohol >= 10.525
## |   |   [8] sulphates < 0.585
## |   |   |   [9] volatile.acidity >= 0.385
## |   |   |   |   [10] volatile.acidity >= 1: 4.000 (n = 7, err = 4.0)
## |   |   |   |   [11] volatile.acidity < 1: 5.393 (n = 89, err = 47.2)
## |   |   |   [12] volatile.acidity < 0.385: 6.188 (n = 32, err = 10.9)
## |   |   [13] sulphates >= 0.585
## |   |   |   [14] alcohol < 11.55: 6.032 (n = 222, err = 106.8)
## |   |   |   [15] alcohol >= 11.55: 6.603 (n = 146, err = 65.0)
##
## Number of inner nodes:    7
## Number of terminal nodes: 8
```

```
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


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

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





Random Forest

```
set.seed(4)

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

print(rf)
```

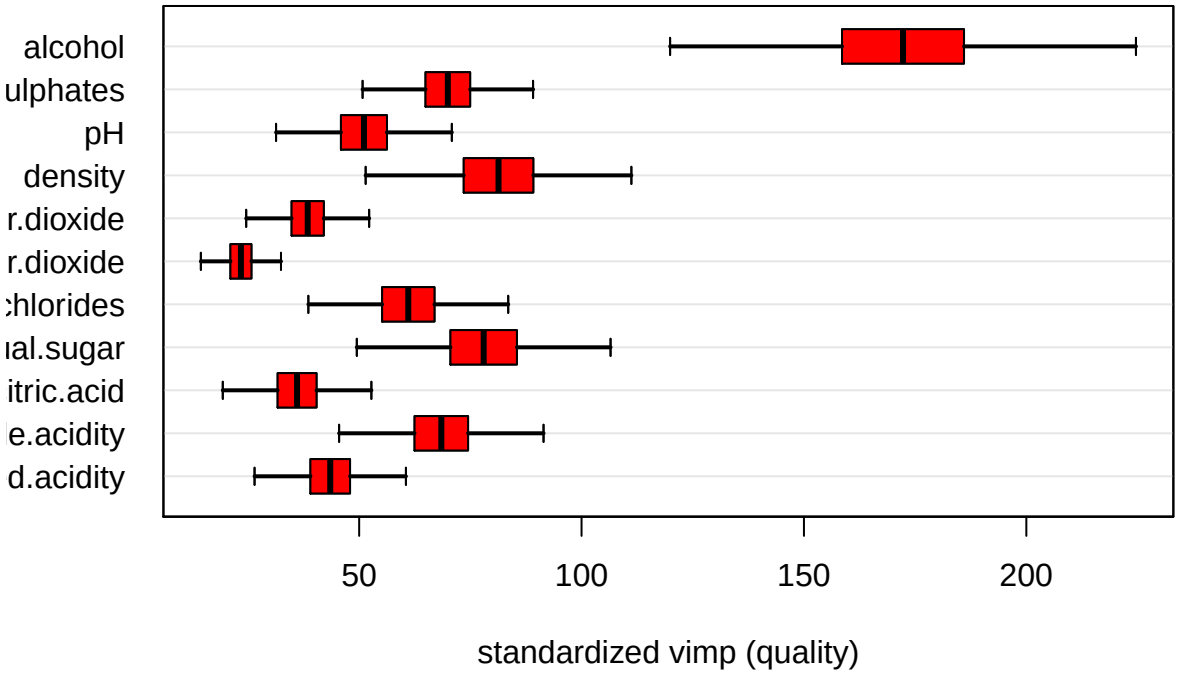
```
##                          Sample size: 1281
##                     Number of trees: 500
##            Forest terminal node size: 5
##        Average no. of terminal nodes: 154.44
## No. of variables tried at each split: 4
##               Total no. of variables: 11
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 810
##                             Analysis: RF-R
##                               Family: regr
##                      Splitting rule: mse
##                     (OOB) R squared: 0.47483782
##    (OOB) Requested performance error: 0.34729603
```

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

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

```
##                          lower      mean     upper       pvalue signif
## fixed.acidity         30.55005  43.49516  56.44027 2.268062e-11   TRUE
## volatile.acidity      50.98391  68.47067  85.95743 8.311031e-15   TRUE
## citric.acid           23.32641  36.03789  48.74937 1.375194e-08   TRUE
## residual.sugar        56.28516  77.99340  99.70164 9.491215e-13   TRUE
## chlorides             43.94843  61.04596  78.14348 1.298541e-12   TRUE
## free.sulfur.dioxide   16.55683  23.41198  30.26714 1.087685e-11   TRUE
## total.sulfur.dioxide  27.90961  38.42750  48.94539 4.010368e-13   TRUE
## density               58.60194  81.33127 104.06059 1.164247e-12   TRUE
## pH                    36.03962  51.07339  66.10716 1.383412e-11   TRUE
## sulphates             55.34834  69.93480  84.52126 2.805752e-21   TRUE
## alcohol              132.42797 172.27663 212.12529 1.191053e-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: 154.44
##           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: 0.4738014
##          Requested performance error: 0.32482477
```

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      0.8087530  0.3502403  0.6295864
##   2      0.8076009  0.3523703  0.6280769
##   3      0.8078026  0.3520609  0.6279273
##
## 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 14 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 13 (additive model)
## GCV 0.6237366    RSS 765.6798    GRSq 0.3767503    RSq 0.4018126
```

```
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##             data=train_transformed)
##
##                                    coefficients
## (Intercept)                           45.995356
## h(-1.22088-volatile.acidity)          -0.646512
```

```
## h(volatile.acidity- -1.22088)         -0.870343
## h(volatile.acidity- -0.77398)          0.695562
## h(1.34361-chlorides)                   0.148853
## h(total.sulfur.dioxide- -1.09835)    -34.386053
## h(total.sulfur.dioxide- -1.06769)     24.754346
## h(total.sulfur.dioxide- -0.914415)    -2.095996
## h(2.70295-total.sulfur.dioxide)      -11.632045
## h(total.sulfur.dioxide-2.70295)       11.949287
## h(0.491101-sulphates)                 -0.455812
## h(alcohol-0.565375)                    0.264400
## h(2.8584-alcohol)                     -0.258158
## h(alcohol-2.8584)                     -1.472078
##
## Selected 14 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 13 (additive model)
## GCV 0.6237366    RSS 765.6798    GRSq 0.3767503    RSq 0.4018126
```

```
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      0.8850014  0.2312022  0.7053687
##   1        3      0.8377666  0.3099770  0.6500428
##   1        4      0.8122972  0.3507056  0.6324137
##   1        5      0.8114462  0.3525609  0.6303420
##   1        6      0.8136694  0.3495223  0.6310561
##   1        7      0.8156143  0.3463378  0.6319505
##   1        8      0.8167655  0.3447020  0.6332684
##   1        9      0.8161338  0.3460706  0.6317048
##   1       10      0.8168680  0.3457867  0.6312923
##   1       11      0.8174787  0.3455033  0.6318670
##   1       12      0.8194375  0.3431718  0.6319792
##   1       13      0.8189981  0.3435539  0.6315615
```

```
##   1        14       0.8199610  0.3423760  0.6319039
##   1        15       0.8199553  0.3423410  0.6320242
##   1        16       0.8199553  0.3423410  0.6320242
##   1        17       0.8199553  0.3423410  0.6320242
##   1        18       0.8199553  0.3423410  0.6320242
##   1        19       0.8199553  0.3423410  0.6320242
##   1        20       0.8199553  0.3423410  0.6320242
##   1        21       0.8199553  0.3423410  0.6320242
##   1        22       0.8199553  0.3423410  0.6320242
##   1        23       0.8199553  0.3423410  0.6320242
##   1        24       0.8199553  0.3423410  0.6320242
##   1        25       0.8199553  0.3423410  0.6320242
##   1        26       0.8199553  0.3423410  0.6320242
##   1        27       0.8199553  0.3423410  0.6320242
##   1        28       0.8199553  0.3423410  0.6320242
##   1        29       0.8199553  0.3423410  0.6320242
##   1        30       0.8199553  0.3423410  0.6320242
##   1        31       0.8199553  0.3423410  0.6320242
##   1        32       0.8199553  0.3423410  0.6320242
##   1        33       0.8199553  0.3423410  0.6320242
##   1        34       0.8199553  0.3423410  0.6320242
##   1        35       0.8199553  0.3423410  0.6320242
##   1        36       0.8199553  0.3423410  0.6320242
##   1        37       0.8199553  0.3423410  0.6320242
##   1        38       0.8199553  0.3423410  0.6320242
##   2         2       0.8825968  0.2353626  0.7012423
##   2         3       0.8363814  0.3126894  0.6481621
##   2         4       0.8127832  0.3501297  0.6325345
##   2         5       0.8109811  0.3532361  0.6290777
##   2         6       0.8104837  0.3553752  0.6276453
##   2         7       0.8132258  0.3526718  0.6287959
##   2         8       0.8152948  0.3509569  0.6286829
##   2         9       0.8173011  0.3481685  0.6294273
##   2        10       0.8225046  0.3415505  0.6328101
##   2        11       0.8238200  0.3399769  0.6341545
##   2        12       0.8242273  0.3399923  0.6347160
##   2        13       0.8268848  0.3366763  0.6353942
##   2        14       0.8274111  0.3359898  0.6356580
##   2        15       0.8282707  0.3351960  0.6362737
##   2        16       0.8287824  0.3346814  0.6368447
##   2        17       0.8290523  0.3343626  0.6370059
##   2        18       0.8293898  0.3339941  0.6370909
##   2        19       0.8297997  0.3334811  0.6372150
##   2        20       0.8297997  0.3334811  0.6372150
##   2        21       0.8297997  0.3334811  0.6372150
##   2        22       0.8297997  0.3334811  0.6372150
##   2        23       0.8297997  0.3334811  0.6372150
##   2        24       0.8297997  0.3334811  0.6372150
##   2        25       0.8297997  0.3334811  0.6372150
##   2        26       0.8297997  0.3334811  0.6372150
##   2        27       0.8297997  0.3334811  0.6372150
##   2        28       0.8297997  0.3334811  0.6372150
##   2        29       0.8297997  0.3334811  0.6372150
##   2        30       0.8297997  0.3334811  0.6372150
```

```
##   2       31      0.8297997  0.3334811  0.6372150
##   2       32      0.8297997  0.3334811  0.6372150
##   2       33      0.8297997  0.3334811  0.6372150
##   2       34      0.8297997  0.3334811  0.6372150
##   2       35      0.8297997  0.3334811  0.6372150
##   2       36      0.8297997  0.3334811  0.6372150
##   2       37      0.8297997  0.3334811  0.6372150
##   2       38      0.8297997  0.3334811  0.6372150
##
## 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  0.9427691  0.3097511  0.5501662
##    2  0.8629831  0.3328498  0.5969983
##    3  0.8368764  0.3405317  0.6030573
##    4  0.8279773  0.3431394  0.6124601
##    5  0.8221784  0.3450349  0.6199145
##    6  0.8160131  0.3486007  0.6194493
##    7  0.8095582  0.3567541  0.6199686
##    8  0.8036826  0.3628989  0.6199448
##    9  0.8021998  0.3639614  0.6214743
##   10  0.8031749  0.3613022  0.6234298
##   11  0.7984903  0.3675699  0.6202376
##   12  0.7981756  0.3682820  0.6217025
##   13  0.7960425  0.3708261  0.6220512
##   14  0.7999649  0.3652795  0.6254455
##   15  0.7988554  0.3663144  0.6247548
##   16  0.7988248  0.3663441  0.6247859
```

```
##   17  0.8004743  0.3639946  0.6249979
##   18  0.7991022  0.3659568  0.6231131
##   19  0.8003750  0.3639124  0.6252625
##   20  0.8012480  0.3627424  0.6268121
##   21  0.7990107  0.3662109  0.6258791
##   22  0.7995840  0.3654909  0.6262162
##   23  0.8009324  0.3641209  0.6285904
##   24  0.8008281  0.3643124  0.6300182
##   25  0.7985497  0.3681201  0.6281662
##   26  0.7983149  0.3682929  0.6288287
##   27  0.7972389  0.3702646  0.6280531
##   28  0.7958480  0.3720008  0.6272963
##   29  0.7954800  0.3721615  0.6265835
##   30  0.7954088  0.3724957  0.6265261
##   31  0.7959198  0.3716672  0.6274353
##   32  0.7954233  0.3726451  0.6276219
##   33  0.7937028  0.3754105  0.6269257
##   34  0.7948998  0.3732350  0.6273414
##   35  0.7953070  0.3725203  0.6274869
##   36  0.7952432  0.3728494  0.6280948
##   37  0.7956865  0.3720231  0.6277415
##   38  0.7951890  0.3730701  0.6274628
##   39  0.7945010  0.3741482  0.6271546
##   40  0.7948020  0.3739603  0.6268306
##   41  0.7945344  0.3746530  0.6268870
##   42  0.7943813  0.3750086  0.6265858
##   43  0.7944566  0.3751135  0.6267804
##   44  0.7945235  0.3749604  0.6272007
##   45  0.7938029  0.3763423  0.6263954
##   46  0.7937111  0.3765468  0.6259715
##   47  0.7945843  0.3750564  0.6266835
##   48  0.7944131  0.3753720  0.6266935
##   49  0.7934151  0.3770704  0.6262354
##   50  0.7937482  0.3764827  0.6262795
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 49.
```

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  0.7924102  0.3776305  0.5919455
##   0.50  0.7820502  0.3919129  0.5818012
##   1.00  0.7784482  0.3977921  0.5757303
##   2.00  0.7764093  0.4021745  0.5719077
##   4.00  0.7774002  0.4057215  0.5687248
##
## Tuning parameter 'sigma' was held constant at a value of 0.0959568
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were sigma = 0.0959568 and C = 2.
```