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.

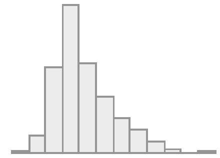
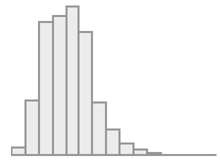
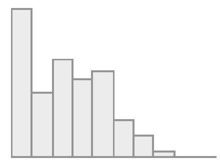
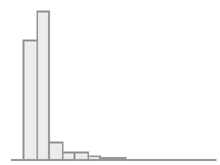
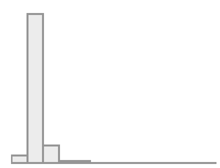
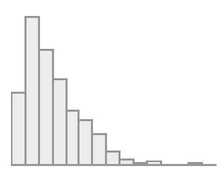
```r
wine <- read.csv(
  url("https://raw.githubusercontent.com/OscarG-DataSci/ADS503/main/winequality-red.csv")
                      , header = TRUE)
```

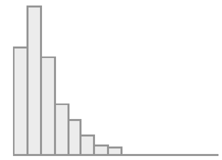
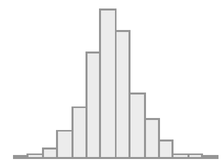
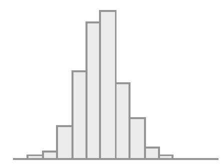
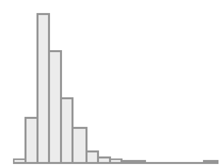
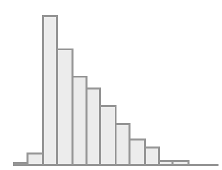
Data Summary

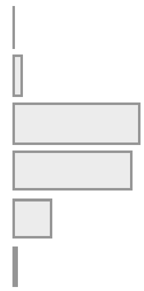
```r
dfSummary(wine,
          plain.ascii  = FALSE,
          style        = "grid",
          graph.magnif = 0.75,
          valid.col    = FALSE,
          tmp.img.dir  = "/tmp")
```

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

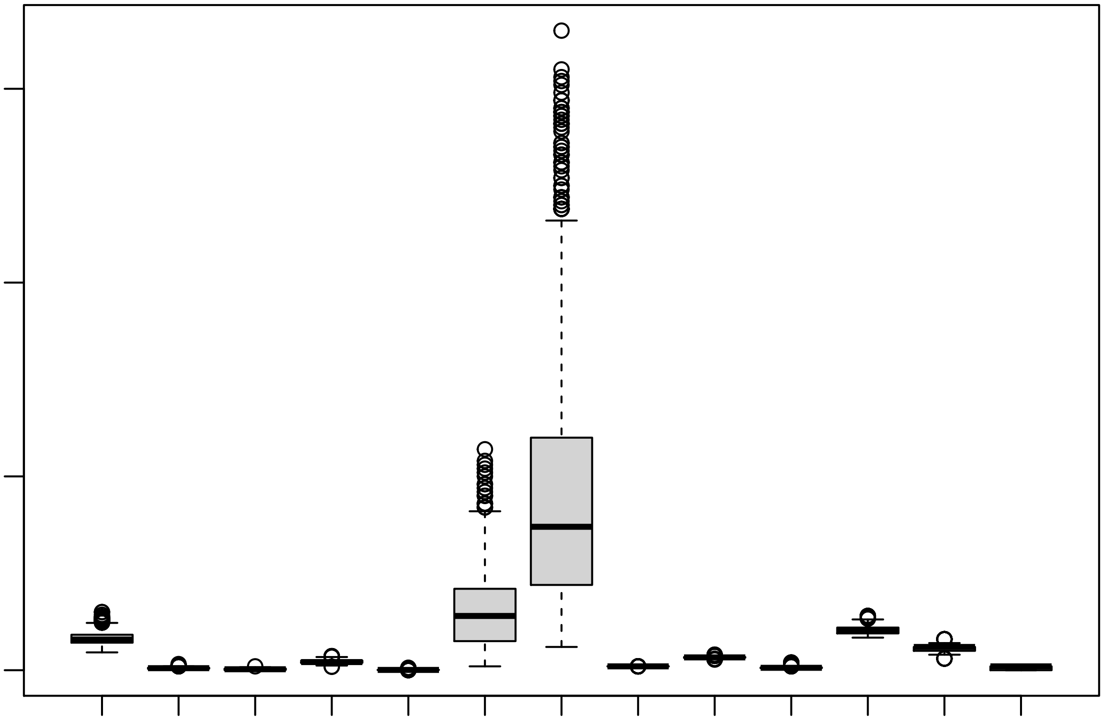
```
par(mar=c(1,1,1,1)) # to fix boxplot knit processing issues

# 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
```

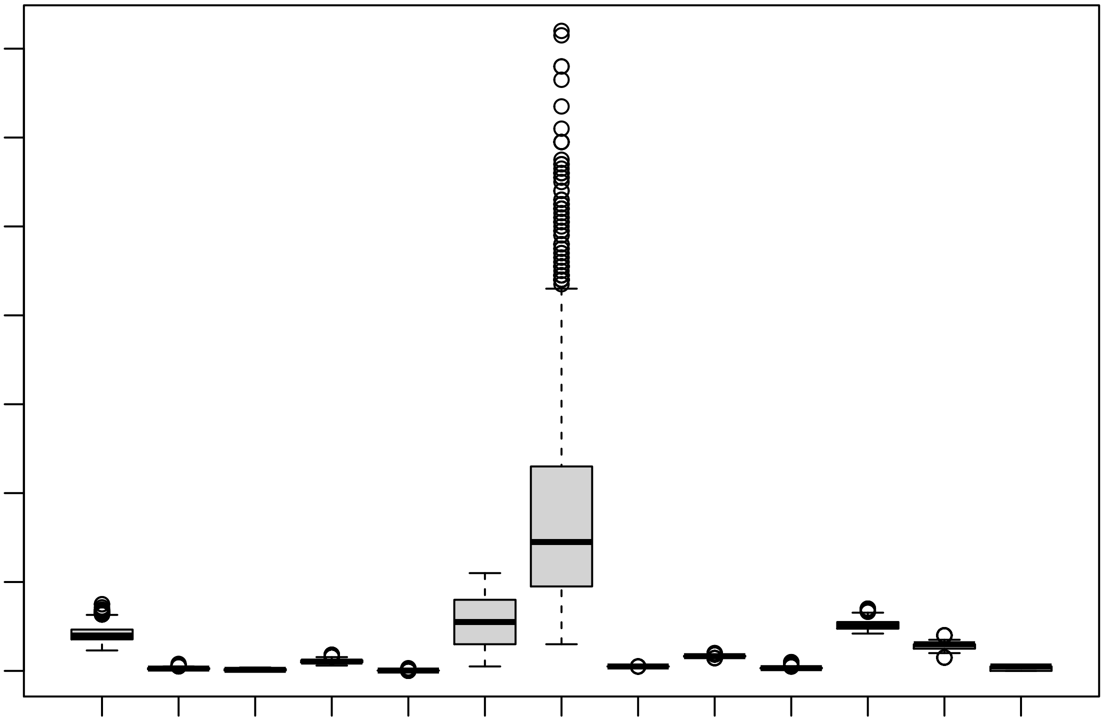
```
# Check for missing values in data set
wine %>% na.omit() %>% count() # there are no missing values
```

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

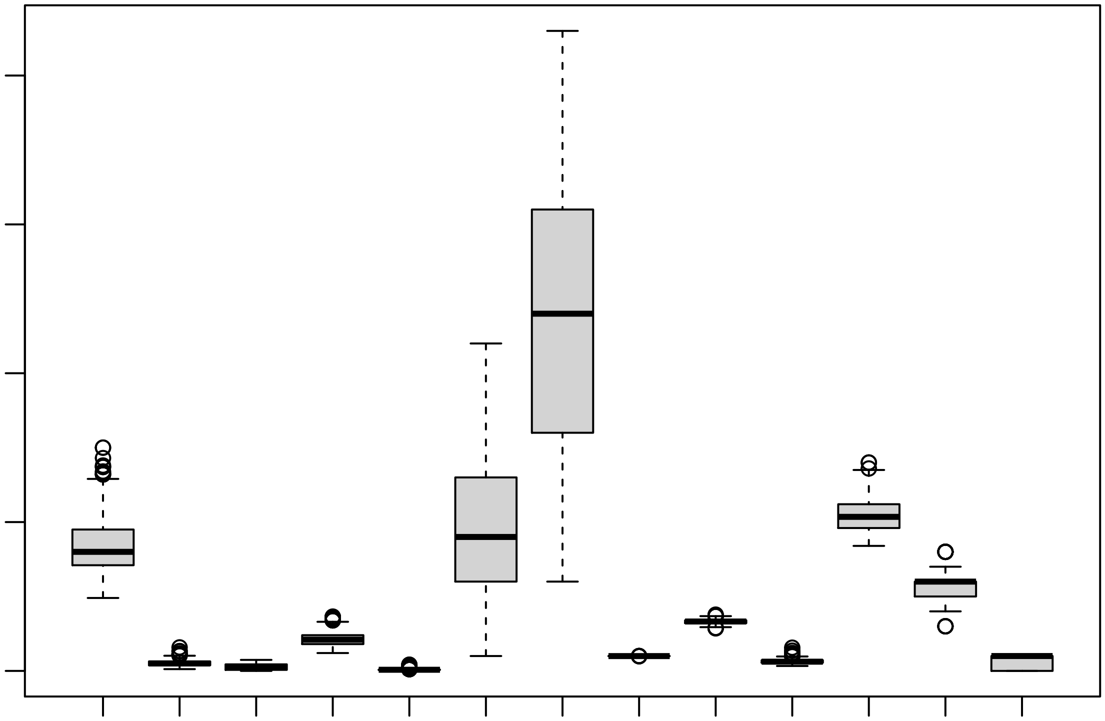
```
# Removing outliers for residual sugar:
Q <- quantile(wine$residual.sugar, probs=c(.25, .75), na.rm = FALSE)
iqr_rs <- IQR(wine$residual.sugar)
up_rs <-  Q[2]+1.5*iqr_rs # Upper Range
low_rs <- Q[1]-1.5*iqr_rs # Lower Range
eliminated_rs <- subset(wine, wine$residual.sugar > (Q[1] - 1.5*iqr_rs) &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



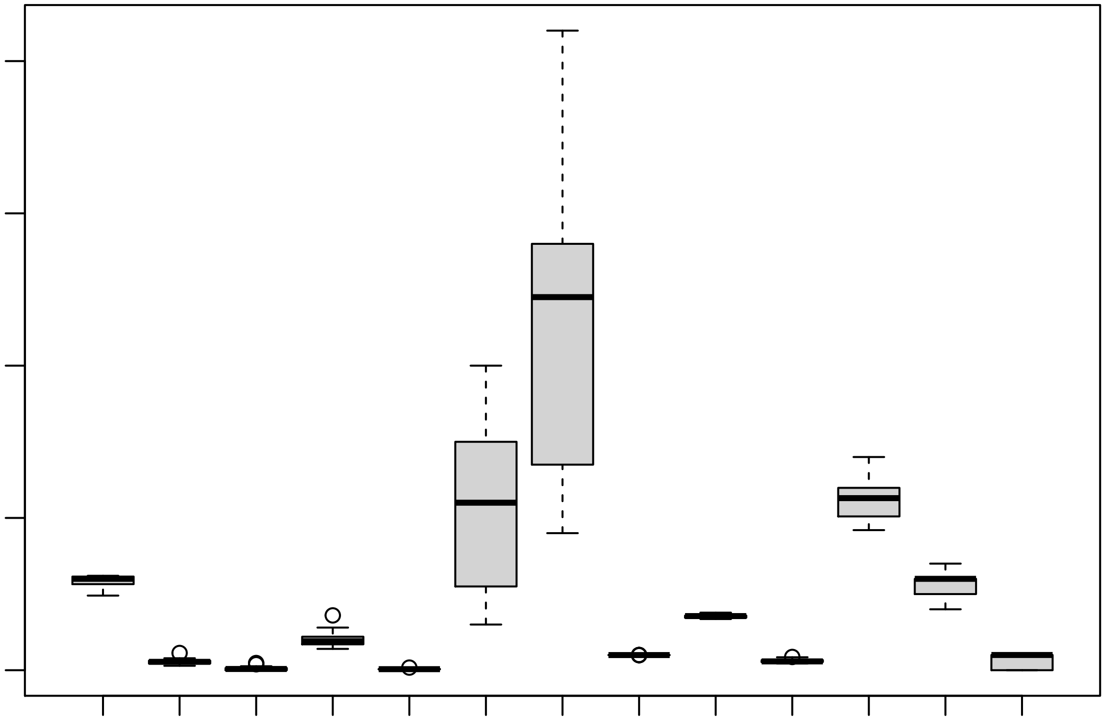
```
#Removing outliers for free.sulfur.dioxide:
Q2 <- quantile(wine$free.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_fs <- IQR(eliminated_rs$free.sulfur.dioxide)
up_fs <-  Q2[2]+1.5*iqr_fs # Upper Range
low_fs <- Q2[1]-1.5*iqr_fs # Lower Range
eliminated_fs <- subset(eliminated_rs, eliminated_rs$free.sulfur.dioxide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



```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) & elimin
boxplot(eliminated_ts)
```



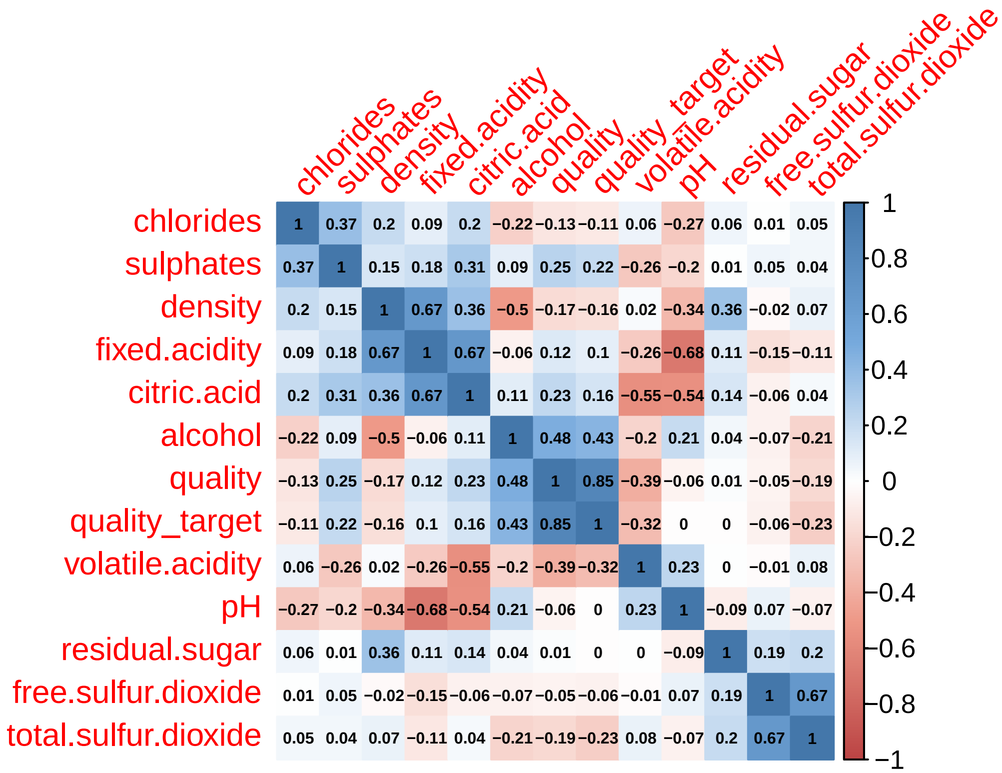
```r
#Removing outliers for fixed.acidity:
Q4 <- quantile(wine$fixed.acidity, probs=c(.25, .75), na.rm = FALSE)
iqr_fa <- IQR(eliminated_ts$fixed.acidity)
up_fa <-  Q[2]+1.5*iqr_fa # Upper Range
low_fa <- Q[1]-1.5*iqr_fa # Lower Range
eliminated_fa <- subset(eliminated_ts, eliminated_ts$fixed.acidity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



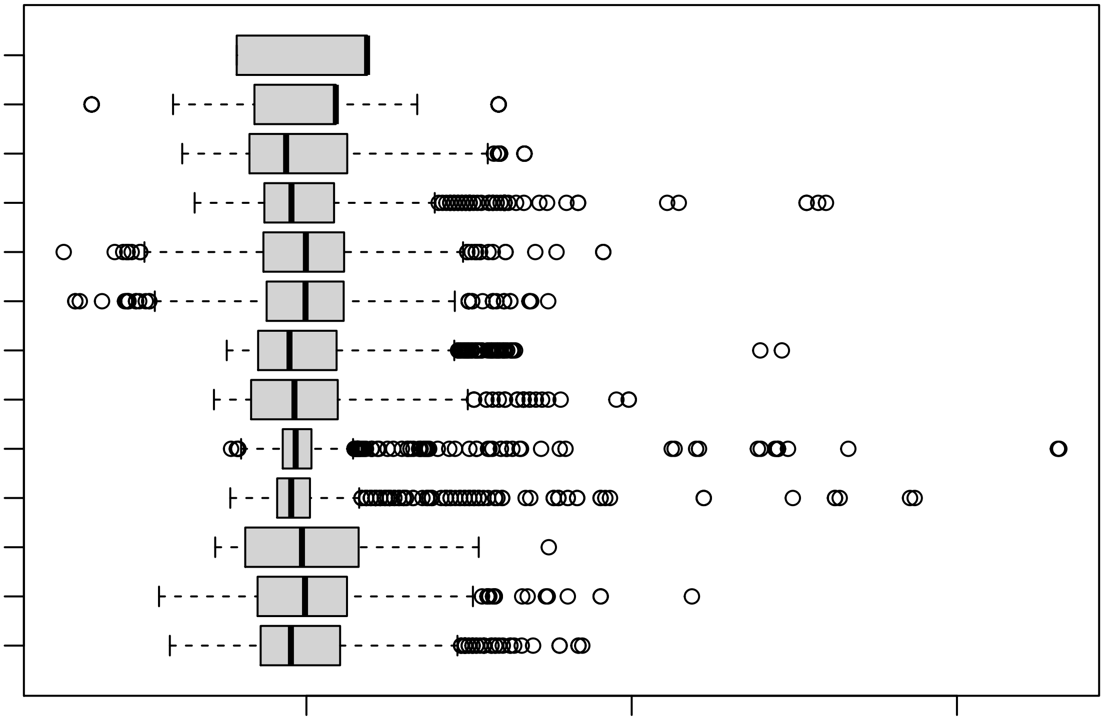
```
new_wine_data <- eliminated_fa

# 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))
```




```
# 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
## -2.09174 -0.69098 -0.05791  0.76874  1.95454
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -4.401e-15  2.373e-02   0.000        1
## volatile.acidity -2.210e-01  2.504e-02  -8.824  < 2e-16 ***
## sulphates         1.349e-01  2.466e-02   5.469 5.44e-08 ***
```

```
## alcohol           3.890e-01  2.428e-02  16.018  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8492 on 1277 degrees of freedom
## Multiple R-squared:  0.2806, Adjusted R-squared:  0.2789
## F-statistic:   166 on 3 and 1277 DF,  p-value: < 2.2e-16
```

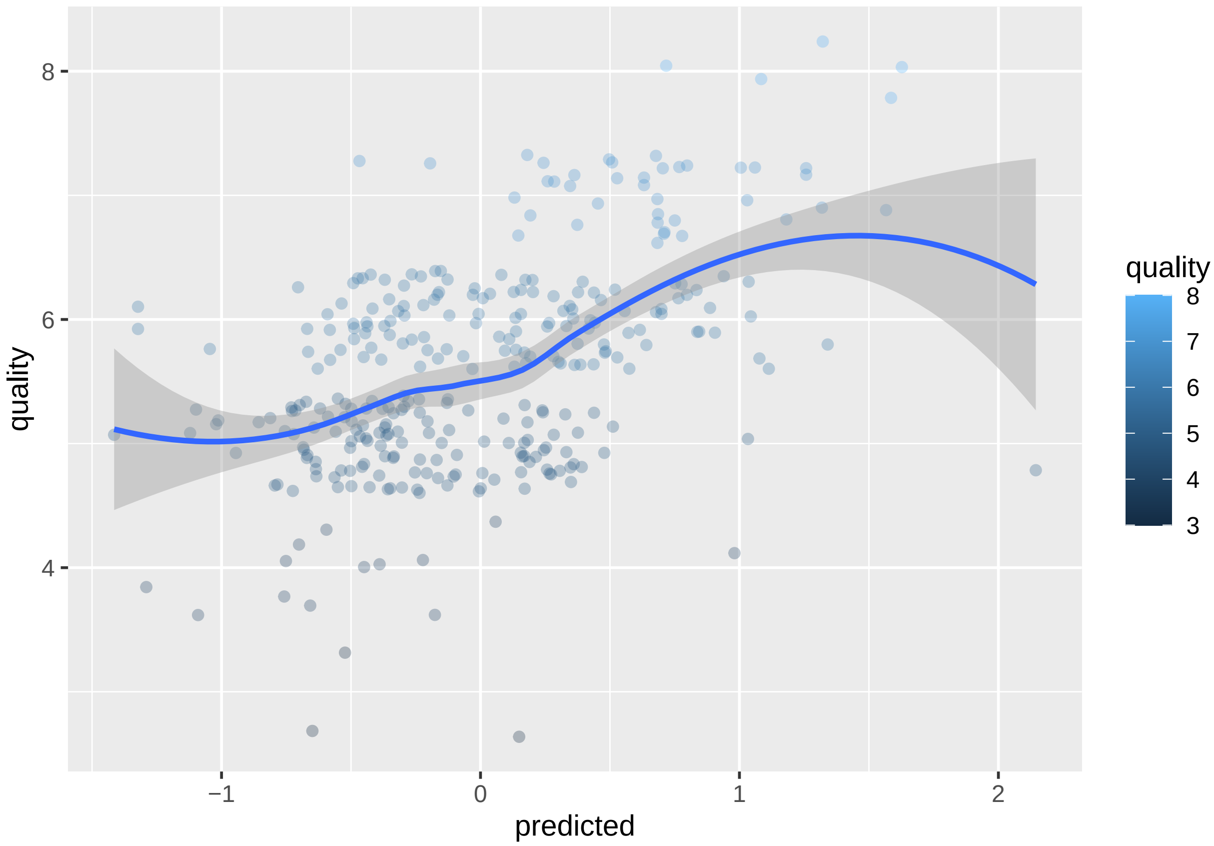
```r
# Model using "quality" as target variable
lmodel2 <- lm(quality~ volatile.acidity + sulphates + alcohol, data = train_transformed)

summary(lmodel2)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + sulphates + alcohol,
##     data = train_transformed)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.84361 -0.47510 -0.07854  0.55916  2.71337
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.081e-16  2.260e-02   0.000        1
## volatile.acidity -2.829e-01  2.385e-02 -11.861  < 2e-16 ***
## sulphates         1.477e-01  2.349e-02   6.286 4.45e-10 ***
## alcohol           4.043e-01  2.313e-02  17.478  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8089 on 1277 degrees of freedom
## Multiple R-squared:  0.3471, Adjusted R-squared:  0.3456
## F-statistic: 226.3 on 3 and 1277 DF,  p-value: < 2.2e-16
```

```r
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(lmodel2, newdata = test_transformed)) -> df

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

```
##       fit                 lwr                upr
##  Min.   :-1.41473   Min.   :-3.0079   Min.   :0.1785
##  1st Qu.:-0.43915   1st Qu.:-2.0279   1st Qu.:1.1495
##  Median :-0.07877   Median :-1.6671   Median :1.5095
##  Mean   : 0.00000   Mean   :-1.5895   Mean   :1.5895
##  3rd Qu.: 0.36193   3rd Qu.:-1.2262   3rd Qu.:1.9501
##  Max.   : 2.14449   Max.   : 0.5457   Max.   :3.7433
```

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



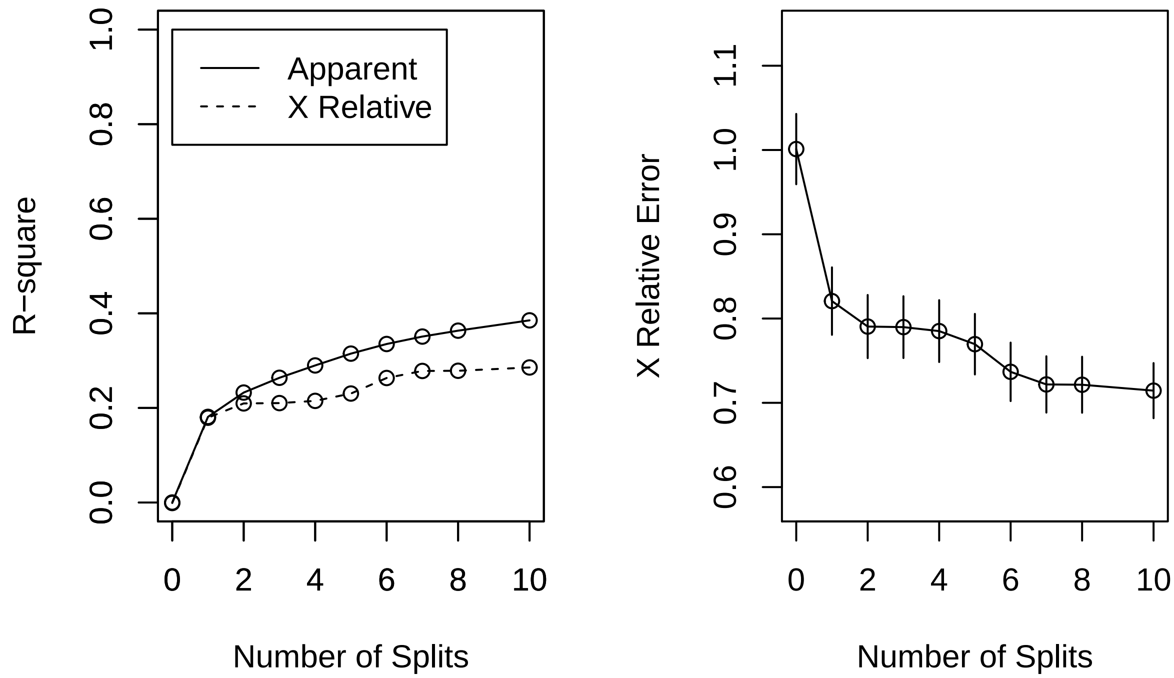

```
# The scatter plot supports the summary of the predicted interval, in the ranges of the fit,
# lower, and upper ranges. The R-squared value of 0.3283 of the model, indicates that this
# information can be predicted 33% of the time, with the data available, for the variance
# of the information.
```

CART

```
set.seed(4)
# Subset both train and test sets, to 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  0.181052      0   1.00000 1.00110 0.041762
## 2  0.051414      1   0.81895 0.82072 0.040061
## 3  0.031271      2   0.76753 0.79048 0.037492
## 4  0.026076      3   0.73626 0.78981 0.036679
## 5  0.024991      4   0.71019 0.78510 0.036738
## 6  0.020343      5   0.68520 0.76959 0.035964
## 7  0.015736      6   0.66485 0.73676 0.034727
## 8  0.012461      7   0.64912 0.72182 0.033440
## 9  0.010909      8   0.63666 0.72145 0.033255
## 10 0.010000     10   0.61484 0.71444 0.032724
```

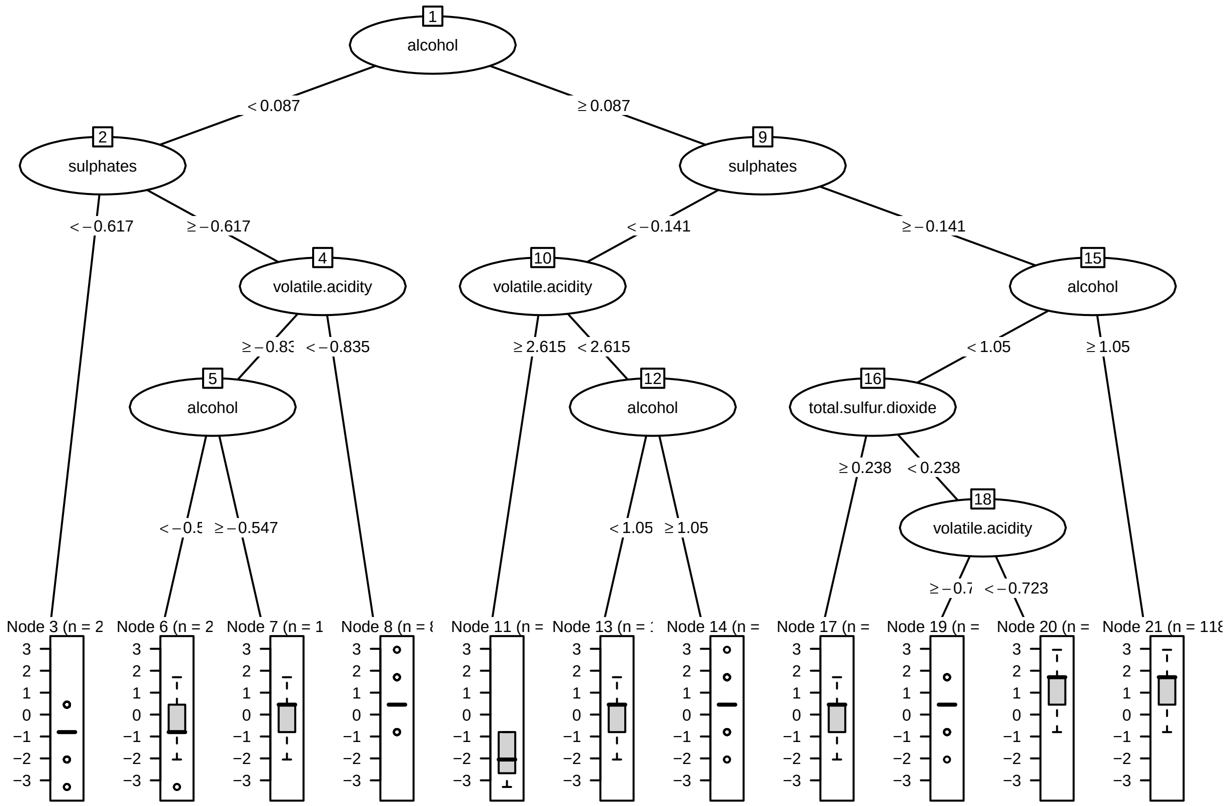



```
# Results
rpartTree2
```

```
##
## 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 < 0.08723
## |   |   [3] sulphates < -0.61681: -0.666 (n = 254, err = 106.7)
## |   |   [4] sulphates >= -0.61681
```

```
## |   |   |   [5] volatile.acidity >= -0.83496
## |   |   |   |   [6] alcohol < -0.54685: -0.438 (n = 274, err = 138.1)
## |   |   |   |   [7] alcohol >= -0.54685: -0.050 (n = 172, err = 124.2)
## |   |   |   [8] volatile.acidity < -0.83496: 0.391 (n = 82, err = 56.1)
## |   [9] alcohol >= 0.08723
## |   |   [10] sulphates < -0.14052
## |   |   |   [11] volatile.acidity >= 2.61549: -1.894 (n = 8, err = 7.6)
## |   |   |   [12] volatile.acidity < 2.61549
## |   |   |   |   [13] alcohol < 1.05011: -0.082 (n = 117, err = 101.2)
## |   |   |   |   [14] alcohol >= 1.05011: 0.553 (n = 87, err = 66.4)
## |   |   [15] sulphates >= -0.14052
## |   |   |   [16] alcohol < 1.05011
## |   |   |   |   [17] total.sulfur.dioxide >= 0.23772: 0.099 (n = 39, err = 31.1)
## |   |   |   |   [18] total.sulfur.dioxide < 0.23772
## |   |   |   |   |   [19] volatile.acidity >= -0.72275: 0.435 (n = 73, err = 42.2)
## |   |   |   |   |   [20] volatile.acidity < -0.72275: 1.133 (n = 57, err = 44.1)
## |   |   |   [21] alcohol >= 1.05011: 1.205 (n = 118, err = 69.3)
##
## Number of inner nodes:    10
## Number of terminal nodes: 11
```

```
plot(rpartTree2, gp = gpar(fontsize=6))
```



Random Forest

```
set.seed(4)
```

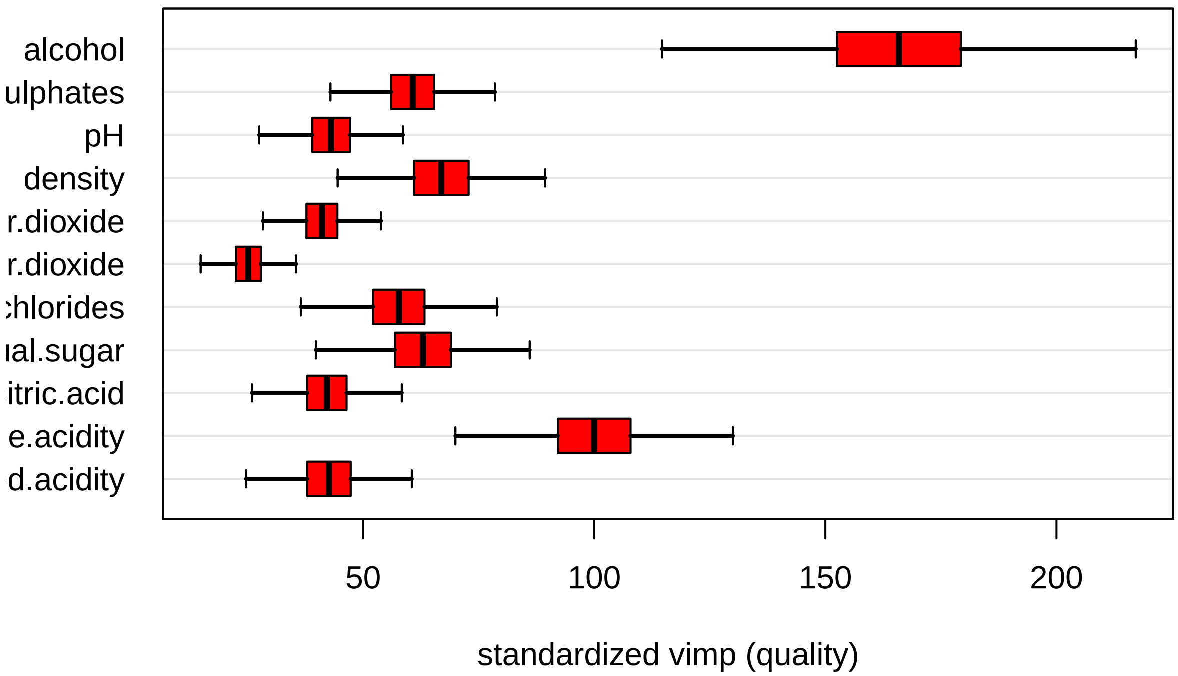
```r
rf <- rfsrc(quality ~ ., data = rf_wine_train)

print(rf)
```

```
##                          Sample size: 1281
##                      Number of trees: 500
##            Forest terminal node size: 5
##        Average no. of terminal nodes: 151.684
## 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: 0.46705408
##    (OOB) Requested performance error: 0.53294592
```

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

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

```
##                          lower      mean     upper       pvalue signif
## fixed.acidity         28.95938  42.59931  56.23924 4.642868e-10   TRUE
## volatile.acidity      77.13856  99.97660 122.81465 4.743454e-18   TRUE
## citric.acid           29.83193  42.15590  54.47988 1.011647e-11   TRUE
## residual.sugar        45.31073  62.90817  80.50560 1.221103e-12   TRUE
## chlorides             41.57913  57.71305  73.84698 1.182815e-12   TRUE
## free.sulfur.dioxide   17.31180  25.16301  33.01422 1.674946e-10   TRUE
## total.sulfur.dioxide  31.37751  41.08702  50.79654 5.482936e-17   TRUE
## density               49.85194  66.92942  84.00690 7.866840e-15   TRUE
## pH                    31.24334  43.06890  54.89446 4.727196e-13   TRUE
## sulphates             47.18117  60.72162  74.26206 7.519109e-19   TRUE
## alcohol              126.91884 165.91158 204.90433 3.729681e-17   TRUE
```

```r
# 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: 151.684
##          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: 0.45687818
##          Requested performance error: 0.54312182
```

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      0.8009483  0.3609283  0.6248827
##   2      0.8007016  0.3612925  0.6244046
##   3      0.8008902  0.3610147  0.6242525
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 2.
```

Mars Tuning

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

mars_wine
```

```
## Selected 10 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 9 (additive model)
## GCV 0.6192871    RSS 769.9483    GRSq 0.3811964    RSq 0.3984779
```

```r
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##             data=train_transformed)
##
##                                     coefficients
## (Intercept)                           31.2282353
## h(1.77391-volatile.acidity)            0.2069420
## h(volatile.acidity-1.77391)           -0.4839817
## h(chlorides- -0.385981)               -0.0786087
## h(total.sulfur.dioxide- -1.1342)      -8.4537879
## h(total.sulfur.dioxide-1.36842)       -0.3247463
## h(2.48405-total.sulfur.dioxide)       -8.3850064
## h(total.sulfur.dioxide-2.48405)        9.1601130
## h(0.960892-sulphates)                 -0.3652400
## h(1.94253-alcohol)                    -0.3674235
##
## Selected 10 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 9 (additive model)
## GCV 0.6192871    RSS 769.9483    GRSq 0.3811964    RSq 0.3984779
```

```r
preProc_Arguments = c("center", "scale")
marsGrid_wine = expand.grid(.degree=1:2, .nprune=2:38)

set.seed(4)

marsModel_wine = train(quality~ volatile.acidity + chlorides + total.sulfur.dioxide +
                       sulphates + alcohol, data =train_transformed,
                       method="earth",
                       preProc=preProc_Arguments,
                       tuneGrid=marsGrid_wine)

marsModel_wine
```

```
## Multivariate Adaptive Regression Spline
##
## 1281 samples
##    5 predictor
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1281, 1281, 1281, 1281, 1281, 1281, ...
## Resampling results across tuning parameters:
##
##   degree  nprune  RMSE       Rsquared   MAE
##   1        2      0.8931576  0.2276817  0.7087136
##   1        3      0.8411903  0.3132186  0.6523206
##   1        4      0.8085427  0.3659858  0.6312428
##   1        5      0.8066254  0.3690233  0.6291302
##   1        6      0.8088540  0.3658383  0.6301515
##   1        7      0.8092640  0.3654791  0.6298044
##   1        8      0.8113355  0.3621660  0.6305667
##   1        9      0.8121219  0.3612320  0.6307932
##   1       10      0.8129649  0.3597762  0.6312176
##   1       11      0.8133643  0.3593854  0.6313570
##   1       12      0.8140167  0.3585976  0.6317295
##   1       13      0.8139652  0.3588677  0.6317555
##   1       14      0.8156289  0.3565980  0.6330332
##   1       15      0.8153264  0.3570322  0.6328053
##   1       16      0.8153866  0.3569681  0.6327920
##   1       17      0.8153866  0.3569681  0.6327920
##   1       18      0.8153866  0.3569681  0.6327920
##   1       19      0.8153866  0.3569681  0.6327920
##   1       20      0.8153866  0.3569681  0.6327920
##   1       21      0.8153866  0.3569681  0.6327920
##   1       22      0.8153866  0.3569681  0.6327920
##   1       23      0.8153866  0.3569681  0.6327920
##   1       24      0.8153866  0.3569681  0.6327920
##   1       25      0.8153866  0.3569681  0.6327920
##   1       26      0.8153866  0.3569681  0.6327920
##   1       27      0.8153866  0.3569681  0.6327920
##   1       28      0.8153866  0.3569681  0.6327920
##   1       29      0.8153866  0.3569681  0.6327920
##   1       30      0.8153866  0.3569681  0.6327920
##   1       31      0.8153866  0.3569681  0.6327920
```

```
##   1       32      0.8153866  0.3569681  0.6327920
##   1       33      0.8153866  0.3569681  0.6327920
##   1       34      0.8153866  0.3569681  0.6327920
##   1       35      0.8153866  0.3569681  0.6327920
##   1       36      0.8153866  0.3569681  0.6327920
##   1       37      0.8153866  0.3569681  0.6327920
##   1       38      0.8153866  0.3569681  0.6327920
##   2        2      0.8915976  0.2298095  0.7072556
##   2        3      0.8348954  0.3239479  0.6462899
##   2        4      0.8065778  0.3694178  0.6277609
##   2        5      0.8052118  0.3718967  0.6252944
##   2        6      0.8030613  0.3747521  0.6231283
##   2        7      0.8083906  0.3673077  0.6256714
##   2        8      0.8143151  0.3592524  0.6277289
##   2        9      0.8191785  0.3523725  0.6307293
##   2       10      0.8216029  0.3493497  0.6322984
##   2       11      0.8230969  0.3476627  0.6341852
##   2       12      0.8229519  0.3481372  0.6339090
##   2       13      0.8249943  0.3455015  0.6353646
##   2       14      0.8263816  0.3437516  0.6360541
##   2       15      0.8271766  0.3428694  0.6365014
##   2       16      0.8279797  0.3419700  0.6365412
##   2       17      0.8281539  0.3417827  0.6365777
##   2       18      0.8281539  0.3417827  0.6365777
##   2       19      0.8281539  0.3417827  0.6365777
##   2       20      0.8281539  0.3417827  0.6365777
##   2       21      0.8281539  0.3417827  0.6365777
##   2       22      0.8281539  0.3417827  0.6365777
##   2       23      0.8281539  0.3417827  0.6365777
##   2       24      0.8281539  0.3417827  0.6365777
##   2       25      0.8281539  0.3417827  0.6365777
##   2       26      0.8281539  0.3417827  0.6365777
##   2       27      0.8281539  0.3417827  0.6365777
##   2       28      0.8281539  0.3417827  0.6365777
##   2       29      0.8281539  0.3417827  0.6365777
##   2       30      0.8281539  0.3417827  0.6365777
##   2       31      0.8281539  0.3417827  0.6365777
##   2       32      0.8281539  0.3417827  0.6365777
##   2       33      0.8281539  0.3417827  0.6365777
##   2       34      0.8281539  0.3417827  0.6365777
##   2       35      0.8281539  0.3417827  0.6365777
##   2       36      0.8281539  0.3417827  0.6365777
##   2       37      0.8281539  0.3417827  0.6365777
##   2       38      0.8281539  0.3417827  0.6365777
##
## 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)
```

```r
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  0.9644161  0.2847723  0.5544215
##    2  0.8907482  0.2982523  0.6109975
##    3  0.8522988  0.3217443  0.6080274
##    4  0.8403923  0.3282242  0.6084252
##    5  0.8295624  0.3359019  0.6115016
##    6  0.8244215  0.3388616  0.6159714
##    7  0.8142260  0.3511890  0.6119213
##    8  0.8035834  0.3645164  0.6072218
##    9  0.8007209  0.3666301  0.6069772
##   10  0.7961920  0.3717519  0.6060896
##   11  0.7966255  0.3701696  0.6055852
##   12  0.7966734  0.3696525  0.6085908
##   13  0.7934338  0.3744318  0.6085134
##   14  0.7934161  0.3742458  0.6097105
##   15  0.7953655  0.3712114  0.6117051
##   16  0.7942764  0.3721523  0.6118128
##   17  0.7975641  0.3672746  0.6147680
##   18  0.7976932  0.3671614  0.6162870
##   19  0.7961296  0.3692235  0.6153625
##   20  0.7959554  0.3696504  0.6162101
##   21  0.7939922  0.3721527  0.6161404
##   22  0.7944903  0.3715377  0.6166030
##   23  0.7921240  0.3747960  0.6143588
##   24  0.7927152  0.3737572  0.6153292
##   25  0.7949951  0.3701811  0.6177137
##   26  0.7926743  0.3739579  0.6173598
##   27  0.7916786  0.3758599  0.6164459
##   28  0.7909634  0.3770866  0.6159224
##   29  0.7909416  0.3771263  0.6163367
##   30  0.7907118  0.3777643  0.6165737
##   31  0.7901812  0.3786153  0.6157317
##   32  0.7913889  0.3767080  0.6171446
##   33  0.7910876  0.3772137  0.6175664
##   34  0.7899098  0.3790728  0.6162648
```

```
##   35  0.7894065  0.3801164  0.6156748
##   36  0.7890555  0.3809280  0.6152512
##   37  0.7882048  0.3824843  0.6143572
##   38  0.7881174  0.3829143  0.6139271
##   39  0.7889098  0.3818575  0.6153589
##   40  0.7884420  0.3826805  0.6150324
##   41  0.7896330  0.3807016  0.6160041
##   42  0.7901894  0.3797496  0.6164813
##   43  0.7906953  0.3789899  0.6165895
##   44  0.7904117  0.3795530  0.6155542
##   45  0.7894754  0.3813497  0.6151459
##   46  0.7889741  0.3826017  0.6160122
##   47  0.7887943  0.3830243  0.6162245
##   48  0.7887811  0.3830731  0.6165738
##   49  0.7882914  0.3841124  0.6168664
##   50  0.7877509  0.3851382  0.6165076
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 50.
```

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  0.7838408  0.3910168  0.5760806
##   0.50  0.7754609  0.4027211  0.5643058
##   1.00  0.7694682  0.4115229  0.5554067
##   2.00  0.7718078  0.4109309  0.5536613
##   4.00  0.7840375  0.3987760  0.5599795
##
## Tuning parameter 'sigma' was held constant at a value of 0.09900808
```

```
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were sigma = 0.09900808 and C = 1.
```