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)
```

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

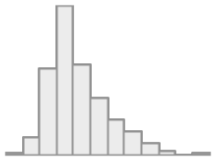
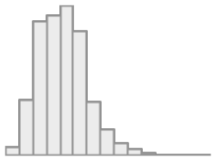
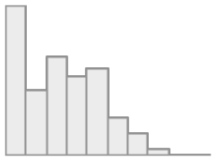
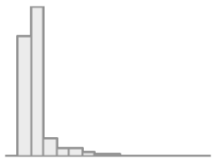
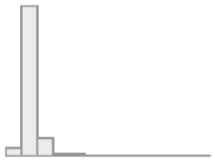
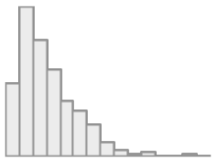
```
#par(mfcol=c(5,3), mar=c(0.5,0.5,0.5,0))

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

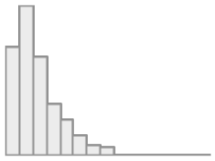
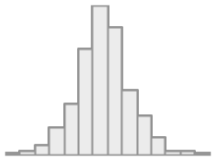
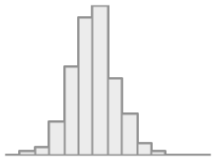
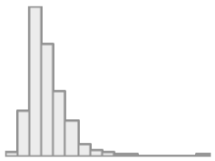
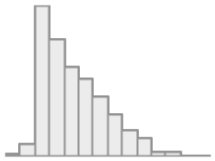
Data Summary

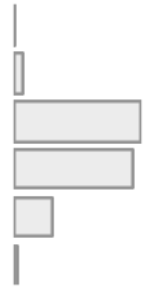
```
dfSummary(wine,
          plain.ascii  = FALSE,
          style        = "grid",
          graph.magnif = 0.75,
          valid.col    = FALSE,
          tmp.img.dir  = "/tmp")
```

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

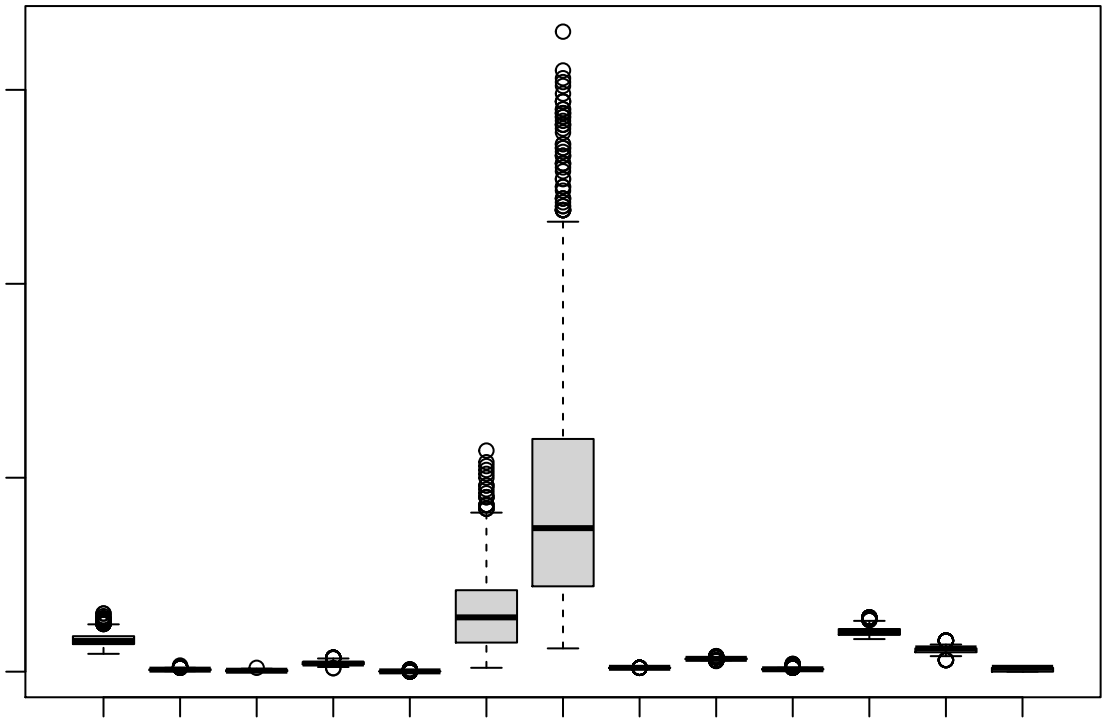
```
par(mar=c(1,1,1,1)) # to fix boxplot knit processing issues

# 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
```

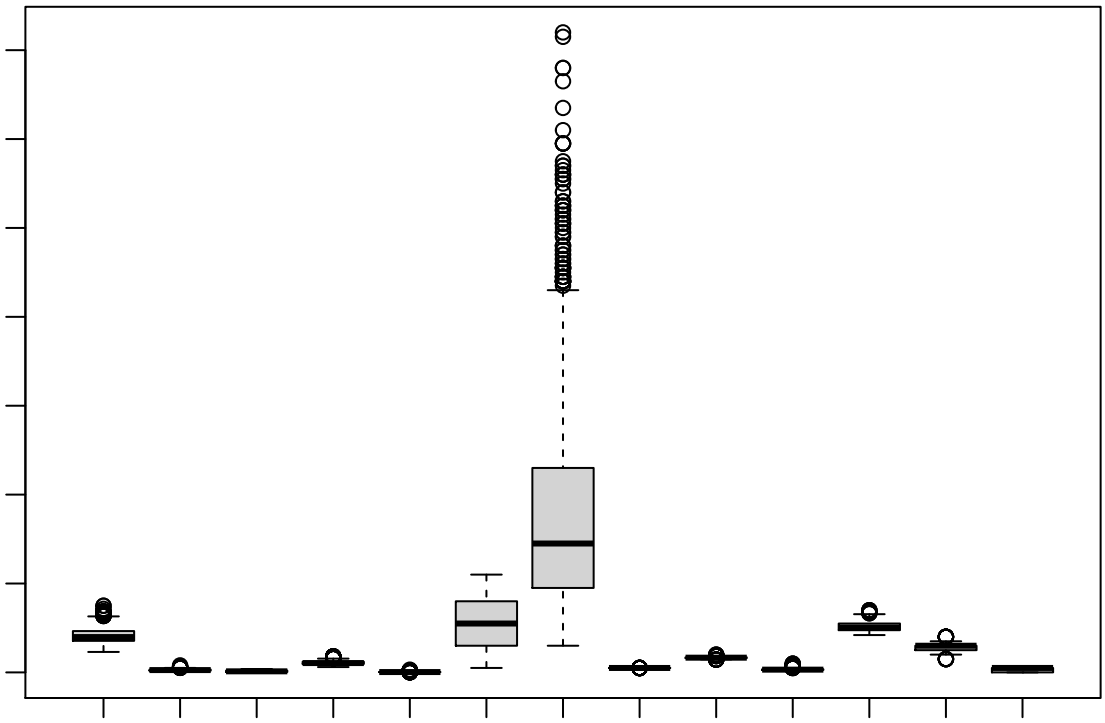
```
# Check for missing values in data set
wine %>% na.omit() %>% count() # there are no missing values
```

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

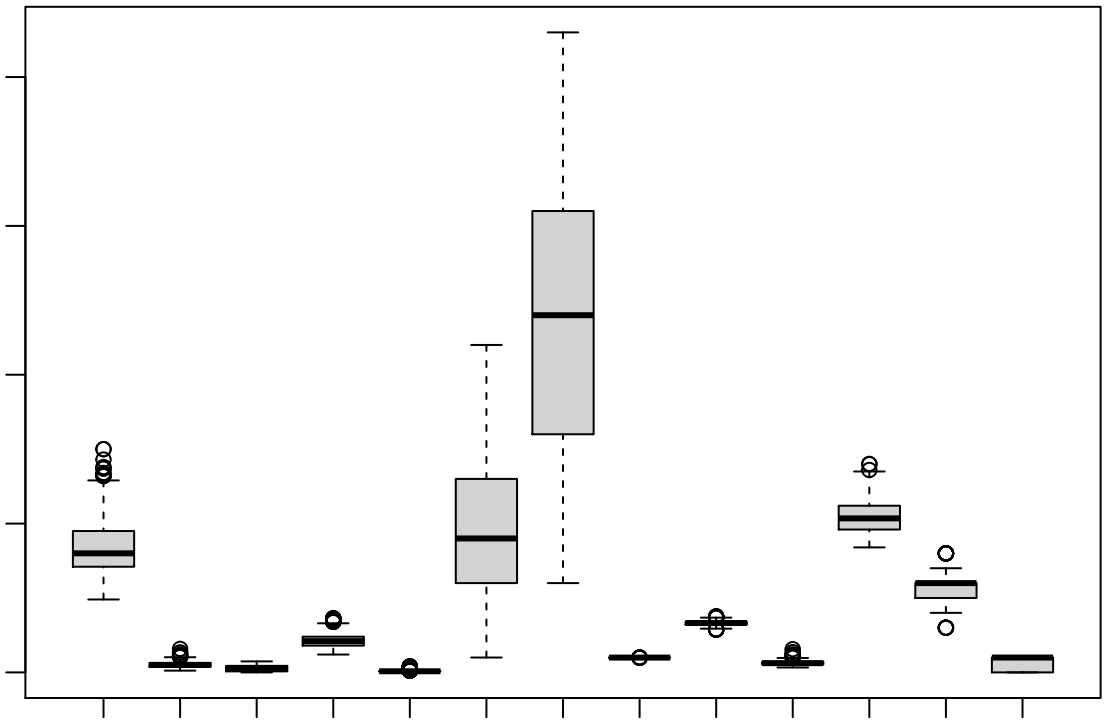
```
# Removing outliers for residual sugar:
Q <- quantile(wine$residual.sugar, probs=c(.25, .75), na.rm = FALSE)
iqr_rs <- IQR(wine$residual.sugar)
up_rs <-  Q[2]+1.5*iqr_rs # Upper Range
low_rs <- Q[1]-1.5*iqr_rs # Lower Range
eliminated_rs <- subset(wine, wine$residual.sugar > (Q[1] - 1.5*iqr_rs) &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



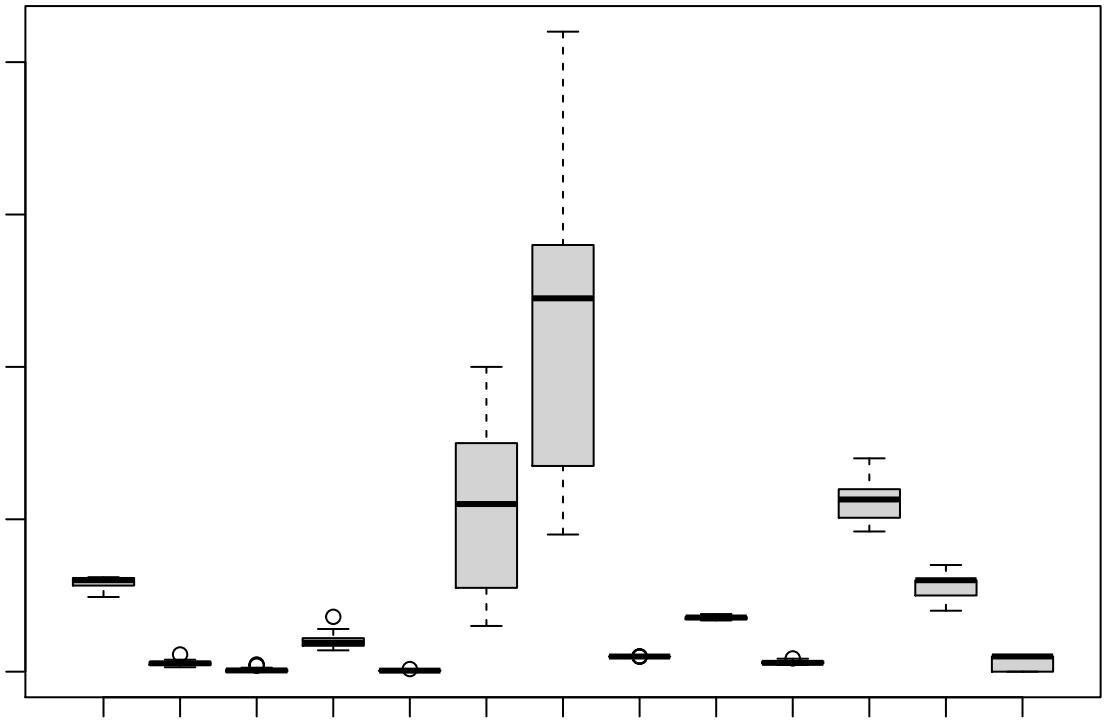
```r
#Removing outliers for free.sulfur.dioxide:
Q2 <- quantile(wine$free.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_fs <- IQR(eliminated_rs$free.sulfur.dioxide)
up_fs <-  Q2[2]+1.5*iqr_fs # Upper Range
low_fs <- Q2[1]-1.5*iqr_fs # Lower Range
eliminated_fs <- subset(eliminated_rs, eliminated_rs$free.sulfur.dioxide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



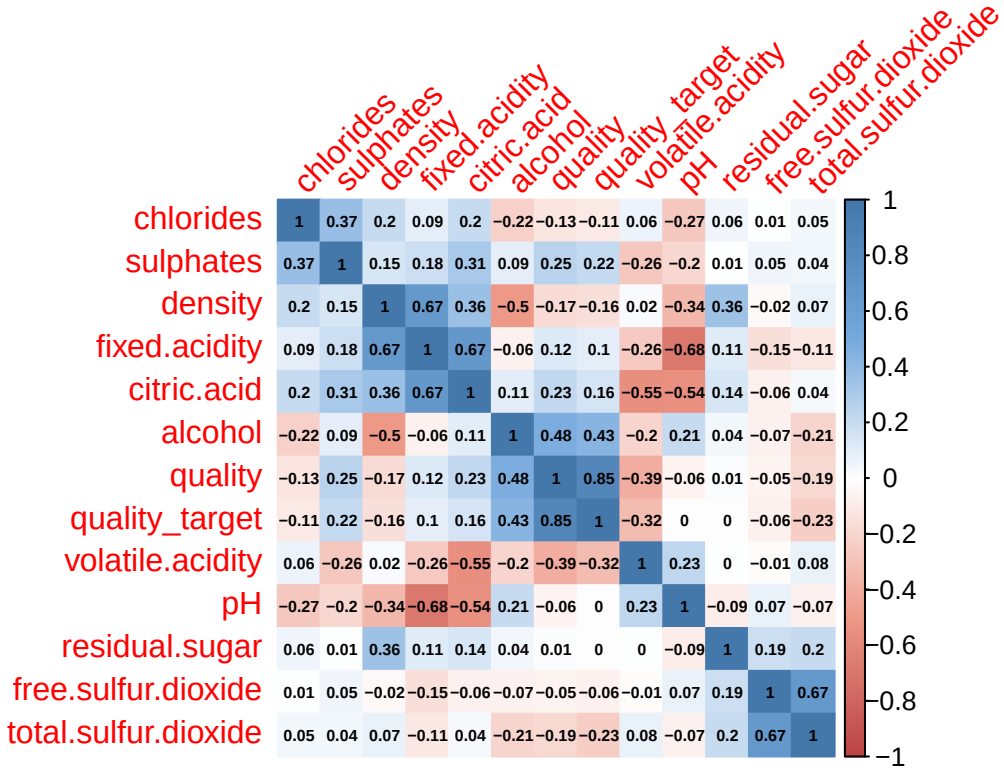
```r
#Removing outliers for fixed.acidity:
Q4 <- quantile(wine$fixed.acidity, probs=c(.25, .75), na.rm = FALSE)
iqr_fa <- IQR(eliminated_ts$fixed.acidity)
up_fa <-  Q[2]+1.5*iqr_fa # Upper Range
low_fa <- Q[1]-1.5*iqr_fa # Lower Range
eliminated_fa <- subset(eliminated_ts, eliminated_ts$fixed.acidity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



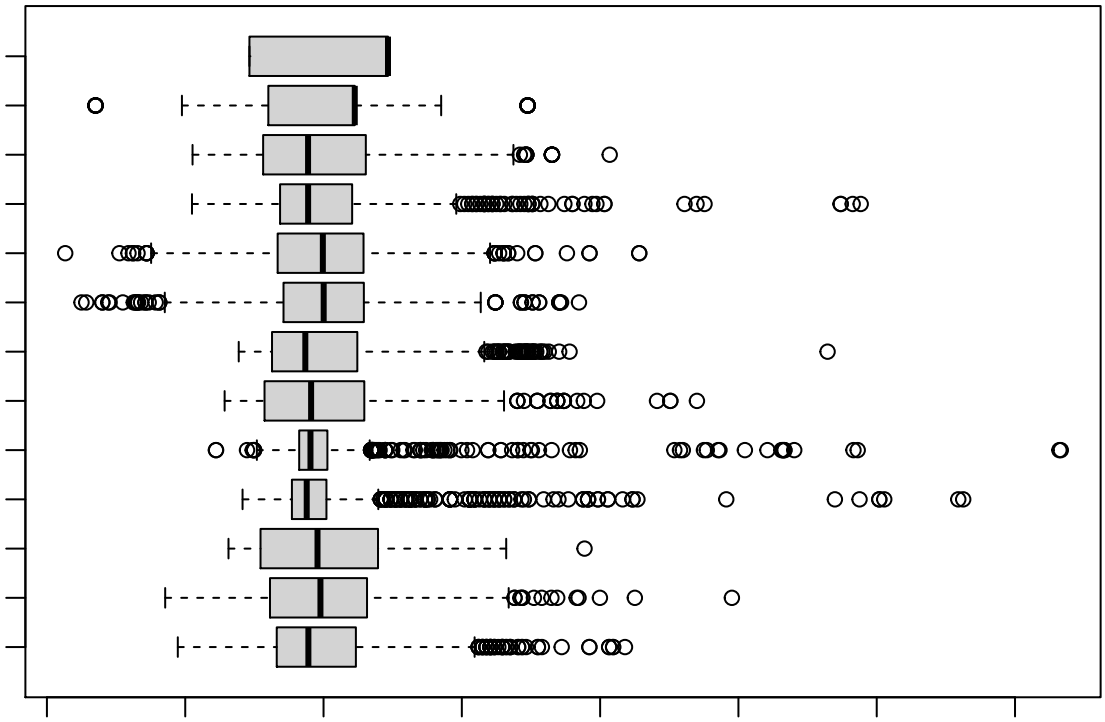
```r
new_wine_data <- eliminated_fa

# 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
## -3.01903 -0.69333 -0.04224  0.77411  2.08222
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -3.379e-15  2.391e-02   0.000        1
## volatile.acidity -2.070e-01  2.540e-02  -8.149 8.68e-16 ***
## sulphates         1.216e-01  2.485e-02   4.894 1.11e-06 ***
```

```
## alcohol           3.925e-01  2.454e-02  15.993  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8556 on 1277 degrees of freedom
## Multiple R-squared:  0.2697, Adjusted R-squared:  0.268
## F-statistic: 157.2 on 3 and 1277 DF,  p-value: < 2.2e-16
```

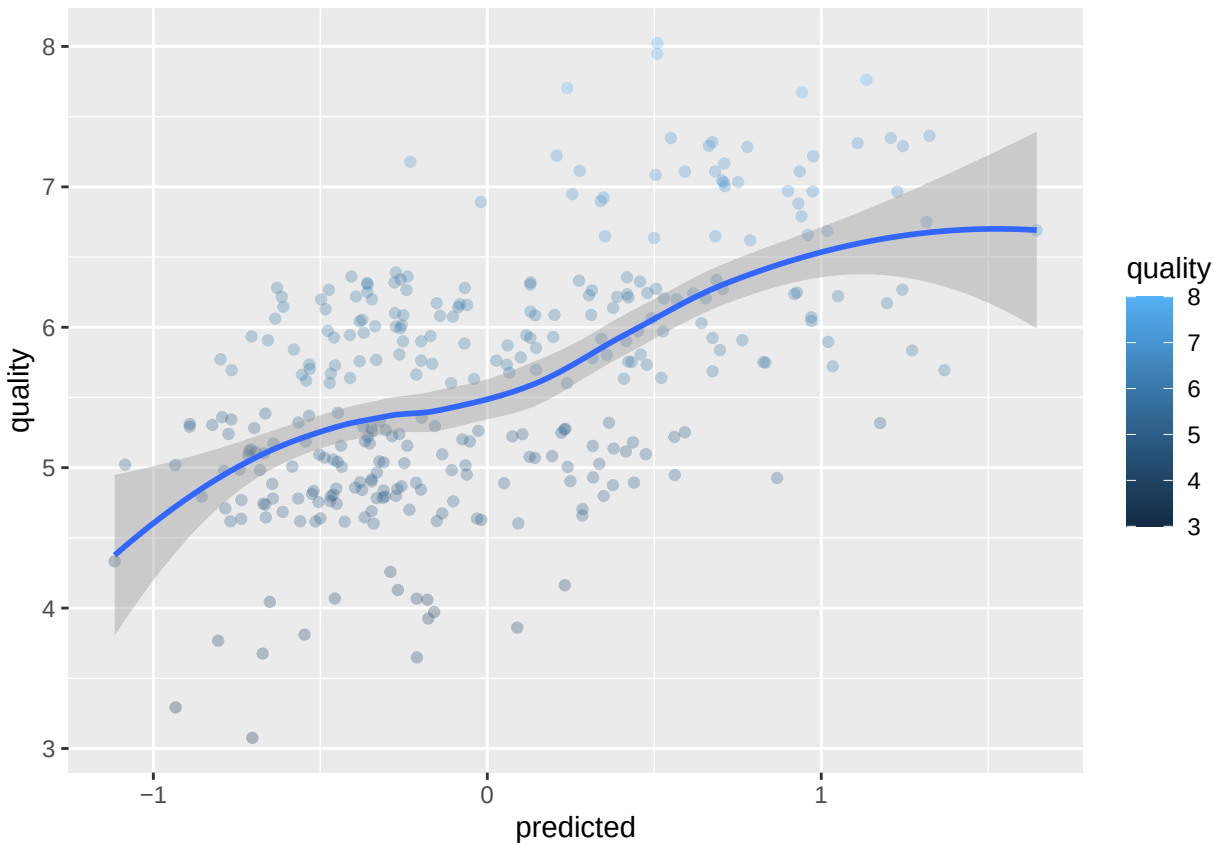
```r
# Model using "quality" as target variable
lmodel2 <- lm(quality~ volatile.acidity + sulphates + alcohol, data = train_transformed)

summary(lmodel2)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + sulphates + alcohol,
##     data = train_transformed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3902 -0.4700 -0.0888  0.5565  2.4556
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.051e-15  2.277e-02   0.000        1
## volatile.acidity -2.602e-01  2.420e-02 -10.754  < 2e-16 ***
## sulphates         1.466e-01  2.367e-02   6.192 8.01e-10 ***
## alcohol           4.115e-01  2.338e-02  17.599  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8151 on 1277 degrees of freedom
## Multiple R-squared:  0.3372, Adjusted R-squared:  0.3356
## F-statistic: 216.6 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.   :-1.1158   Min.   :-2.71926   Min.   :0.4877
##  1st Qu.:-0.4448   1st Qu.:-2.04507   1st Qu.:1.1555
##  Median :-0.1463   Median :-1.74696   Median :1.4541
##  Mean   : 0.0000   Mean   :-1.60158   Mean   :1.6016
##  3rd Qu.: 0.4177   3rd Qu.:-1.18318   3rd Qu.:2.0182
##  Max.   : 1.6452   Max.   : 0.03874   Max.   :3.2516
```

```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.
```

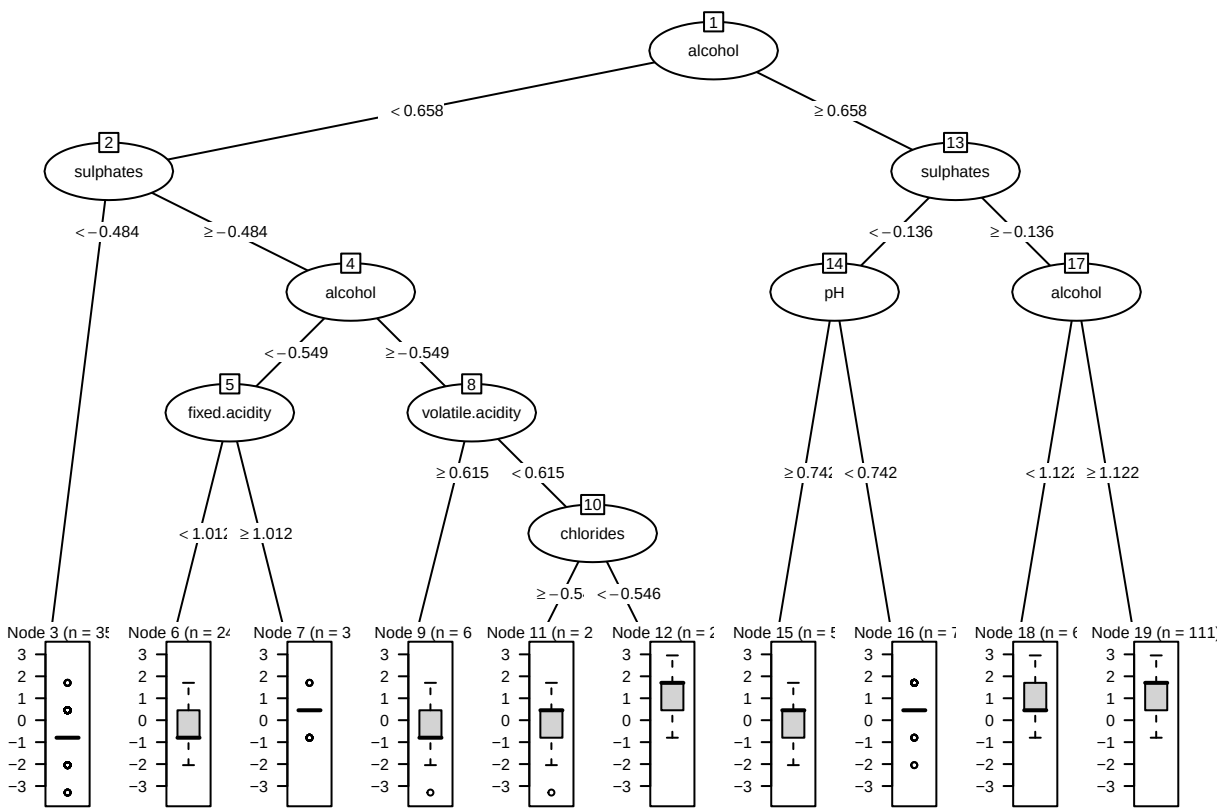
CART

```
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)

# 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 < 0.65806
## |   |   [3] sulphates < -0.48352: -0.607 (n = 354, err = 202.6)
## |   |   [4] sulphates >= -0.48352
## |   |   |   [5] alcohol < -0.54851
## |   |   |   |   [6] fixed.acidity < 1.01161: -0.419 (n = 247, err = 106.5)
## |   |   |   |   [7] fixed.acidity >= 1.01161: 0.254 (n = 38, err = 17.3)
## |   |   |   [8] alcohol >= -0.54851
## |   |   |   |   [9] volatile.acidity >= 0.61491: -0.362 (n = 66, err = 45.3)
## |   |   |   |   [10] volatile.acidity < 0.61491
## |   |   |   |   |   [11] chlorides >= -0.54577: 0.280 (n = 240, err = 169.4)
## |   |   |   |   |   [12] chlorides < -0.54577: 1.151 (n = 25, err = 15.9)
## |   [13] alcohol >= 0.65806
## |   |   [14] sulphates < -0.13609
## |   |   |   [15] pH >= 0.74182: -0.120 (n = 59, err = 54.1)
## |   |   |   [16] pH < 0.74182: 0.614 (n = 77, err = 47.9)
## |   |   [17] sulphates >= -0.13609
## |   |   |   [18] alcohol < 1.12212: 0.764 (n = 64, err = 46.8)
## |   |   |   [19] alcohol >= 1.12212: 1.330 (n = 111, err = 73.7)
##
## Number of inner nodes:     9
## Number of terminal nodes: 10
```

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



Random Forest

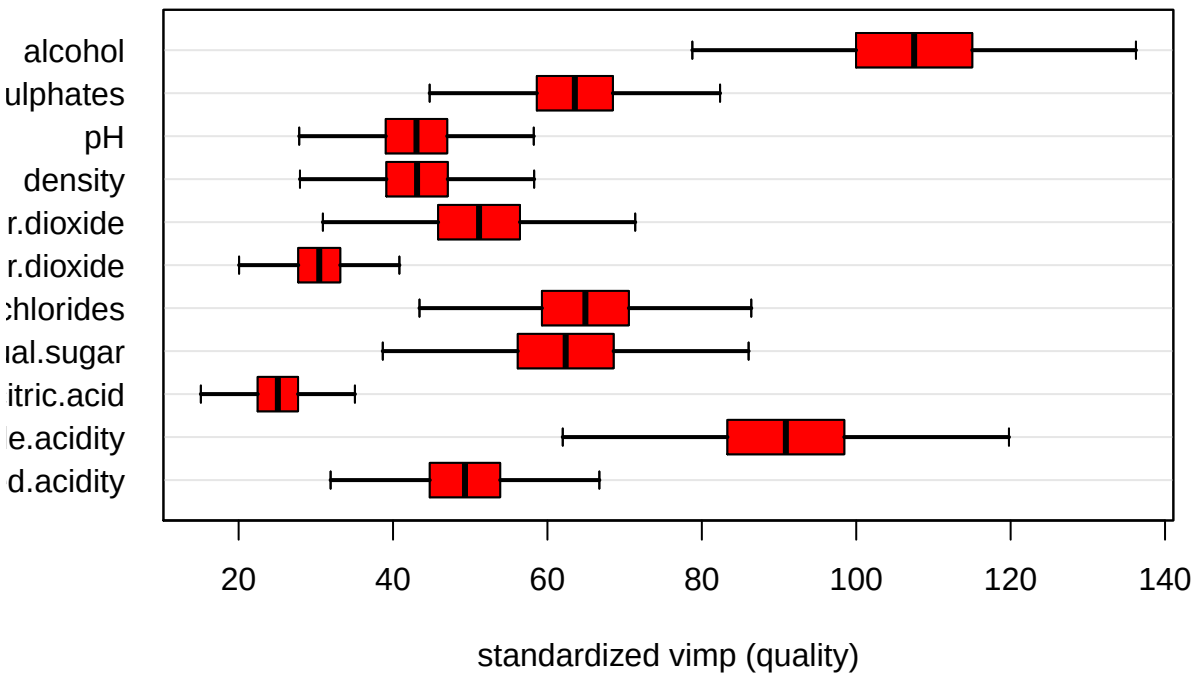
```r
set.seed(4)

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: 151.804
## 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: 0.48360166
##    (OOB) Requested performance error: 0.51639834
```

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

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

```
##                        lower      mean     upper       pvalue signif
## fixed.acidity        36.07507  49.31879  62.56250 1.451857e-13   TRUE
## volatile.acidity     68.89114  90.88204 112.87293 2.748727e-16   TRUE
## citric.acid          17.48953  25.08148  32.67343 4.736827e-11   TRUE
## residual.sugar       44.33437  62.36363  80.39288 6.027012e-12   TRUE
## chlorides            48.56247  64.91367  81.26487 3.598066e-15   TRUE
## free.sulfur.dioxide  22.54100  30.44024  38.33948 2.129132e-14   TRUE
## total.sulfur.dioxide 35.74904  51.14179  66.53454 3.710241e-11   TRUE
## density              31.57222  43.11301  54.65379 1.222839e-13   TRUE
## pH                   31.47835  43.03740  54.59645 1.466208e-13   TRUE
## sulphates            49.23997  63.55182  77.86368 1.612918e-18   TRUE
## alcohol              85.63911 107.50070 129.36228 2.768202e-22   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: 151.804
##           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: 0.38271735
##          Requested performance error: 0.61728265
```

Partial Least Squares

```r
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      0.8047329  0.3549636  0.6253385
##   2      0.8039932  0.3561886  0.6238560
##   3      0.8043480  0.3556864  0.6233929
##
## 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 13 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 12 (additive model)
## GCV 0.6253416    RSS 770.1007    GRSq 0.3751465    RSq 0.3983588
```

```
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##             data=train_transformed)
##
##                                   coefficients
## (Intercept)                          40.299926
## h(2.64973-volatile.acidity)           0.184415
## h(volatile.acidity-2.64973)          -0.640620
## h(chlorides- -0.94349)                2.162832
## h(1.1777-chlorides)                   2.268855
## h(chlorides-1.1777)                  -2.244383
## h(total.sulfur.dioxide- -1.13597)   -12.444535
## h(2.47483-total.sulfur.dioxide)     -12.315898
## h(total.sulfur.dioxide-2.47483)      12.689417
## h(0.935176-sulphates)                -0.407212
## h(alcohol-0.518838)                   0.323892
## h(1.91103-alcohol)                   -0.265515
## h(alcohol-1.91103)                   -0.584056
##
## Selected 13 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 12 (additive model)
```

```
## GCV 0.6253416    RSS 770.1007    GRSq 0.3751465    RSq 0.3983588
```

```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      0.8756930  0.2354502  0.6936302
##   1        3      0.8314504  0.3093290  0.6425392
##   1        4      0.8073480  0.3488156  0.6257890
##   1        5      0.8041485  0.3542999  0.6226520
##   1        6      0.8017427  0.3585575  0.6188208
##   1        7      0.8020412  0.3583713  0.6174441
##   1        8      0.8434366  0.3420549  0.6225569
##   1        9      0.8454422  0.3399539  0.6240305
##   1       10      0.8466531  0.3382266  0.6239170
##   1       11      0.8486616  0.3374556  0.6253726
##   1       12      0.8504126  0.3352609  0.6263172
##   1       13      0.8565933  0.3329037  0.6272741
##   1       14      0.8581101  0.3321964  0.6271843
##   1       15      0.8574906  0.3320197  0.6273578
##   1       16      0.8554511  0.3322485  0.6273276
##   1       17      0.8554511  0.3322485  0.6273276
##   1       18      0.8554511  0.3322485  0.6273276
##   1       19      0.8554511  0.3322485  0.6273276
##   1       20      0.8554511  0.3322485  0.6273276
##   1       21      0.8554511  0.3322485  0.6273276
##   1       22      0.8554511  0.3322485  0.6273276
##   1       23      0.8554511  0.3322485  0.6273276
##   1       24      0.8554511  0.3322485  0.6273276
##   1       25      0.8554511  0.3322485  0.6273276
##   1       26      0.8554511  0.3322485  0.6273276
##   1       27      0.8554511  0.3322485  0.6273276
##   1       28      0.8554511  0.3322485  0.6273276
##   1       29      0.8554511  0.3322485  0.6273276
##   1       30      0.8554511  0.3322485  0.6273276
```

```
##   1        31       0.8554511  0.3322485  0.6273276
##   1        32       0.8554511  0.3322485  0.6273276
##   1        33       0.8554511  0.3322485  0.6273276
##   1        34       0.8554511  0.3322485  0.6273276
##   1        35       0.8554511  0.3322485  0.6273276
##   1        36       0.8554511  0.3322485  0.6273276
##   1        37       0.8554511  0.3322485  0.6273276
##   1        38       0.8554511  0.3322485  0.6273276
##   2         2       0.8752931  0.2362860  0.6918094
##   2         3       0.8299070  0.3123018  0.6393580
##   2         4       0.8018899  0.3581254  0.6196514
##   2         5       0.7979669  0.3645300  0.6180754
##   2         6       0.7983150  0.3647554  0.6168513
##   2         7       0.7987696  0.3650468  0.6169129
##   2         8       0.7980430  0.3664906  0.6153796
##   2         9       0.8019168  0.3624040  0.6167713
##   2        10       0.8052411  0.3581898  0.6173896
##   2        11       0.8097894  0.3533775  0.6189852
##   2        12       0.8099257  0.3539874  0.6191938
##   2        13       0.8133970  0.3500473  0.6204395
##   2        14       0.8130922  0.3508676  0.6207732
##   2        15       0.8160599  0.3479666  0.6221147
##   2        16       0.8190965  0.3445479  0.6233901
##   2        17       0.8192989  0.3441989  0.6235858
##   2        18       0.8190578  0.3445551  0.6233230
##   2        19       0.8190578  0.3445551  0.6233230
##   2        20       0.8190578  0.3445551  0.6233230
##   2        21       0.8190578  0.3445551  0.6233230
##   2        22       0.8190578  0.3445551  0.6233230
##   2        23       0.8190578  0.3445551  0.6233230
##   2        24       0.8190578  0.3445551  0.6233230
##   2        25       0.8190578  0.3445551  0.6233230
##   2        26       0.8190578  0.3445551  0.6233230
##   2        27       0.8190578  0.3445551  0.6233230
##   2        28       0.8190578  0.3445551  0.6233230
##   2        29       0.8190578  0.3445551  0.6233230
##   2        30       0.8190578  0.3445551  0.6233230
##   2        31       0.8190578  0.3445551  0.6233230
##   2        32       0.8190578  0.3445551  0.6233230
##   2        33       0.8190578  0.3445551  0.6233230
##   2        34       0.8190578  0.3445551  0.6233230
##   2        35       0.8190578  0.3445551  0.6233230
##   2        36       0.8190578  0.3445551  0.6233230
##   2        37       0.8190578  0.3445551  0.6233230
##   2        38       0.8190578  0.3445551  0.6233230
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nprune = 5 and degree = 2.
```

KNN Neighbors

```
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  0.9413877  0.3058371  0.5279069
##    2  0.8612857  0.3382435  0.5799643
##    3  0.8351696  0.3473631  0.6000973
##    4  0.8157720  0.3588492  0.6063523
##    5  0.8108712  0.3591337  0.6070467
##    6  0.8020065  0.3689034  0.6048653
##    7  0.7954283  0.3756290  0.6058771
##    8  0.7901812  0.3830179  0.6012335
##    9  0.7888980  0.3845511  0.6000482
##   10  0.7849860  0.3887880  0.6006464
##   11  0.7821155  0.3923894  0.6001655
##   12  0.7814038  0.3922440  0.6036986
##   13  0.7843517  0.3872926  0.6058842
##   14  0.7840747  0.3875626  0.6058779
##   15  0.7835280  0.3885711  0.6053609
##   16  0.7844098  0.3873707  0.6068986
##   17  0.7855601  0.3858868  0.6081172
##   18  0.7877121  0.3826103  0.6103087
##   19  0.7873093  0.3831278  0.6106442
##   20  0.7837525  0.3881698  0.6072053
##   21  0.7851736  0.3857656  0.6079892
##   22  0.7861932  0.3842359  0.6086314
##   23  0.7849386  0.3865515  0.6088558
##   24  0.7844942  0.3870588  0.6088049
##   25  0.7829696  0.3896934  0.6077639
##   26  0.7817699  0.3918113  0.6073675
##   27  0.7807344  0.3936582  0.6061010
##   28  0.7807894  0.3937854  0.6057185
##   29  0.7794957  0.3956485  0.6054457
##   30  0.7791325  0.3965426  0.6050119
```

```
##   31  0.7797593  0.3957354  0.6055871
##   32  0.7803991  0.3947046  0.6064823
##   33  0.7798827  0.3958997  0.6061321
##   34  0.7802847  0.3953651  0.6061262
##   35  0.7805149  0.3950339  0.6058392
##   36  0.7801082  0.3957980  0.6065722
##   37  0.7804597  0.3952672  0.6066267
##   38  0.7804860  0.3952059  0.6061595
##   39  0.7795539  0.3969491  0.6052725
##   40  0.7796581  0.3968186  0.6060280
##   41  0.7803455  0.3958819  0.6073018
##   42  0.7811408  0.3946118  0.6082041
##   43  0.7802834  0.3960826  0.6074135
##   44  0.7809098  0.3951477  0.6075255
##   45  0.7817781  0.3937682  0.6084861
##   46  0.7815741  0.3943479  0.6082870
##   47  0.7811918  0.3948631  0.6079341
##   48  0.7816923  0.3940424  0.6087511
##   49  0.7825869  0.3926080  0.6102180
##   50  0.7825933  0.3927708  0.6107923
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 30.
```

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  0.7805393  0.3958471  0.5804411
##   0.50  0.7718950  0.4080347  0.5699780
##   1.00  0.7653200  0.4188130  0.5630520
```

```
##   2.00  0.7642745  0.4225274  0.5604507
##   4.00  0.7728294  0.4160370  0.5642298
##
## Tuning parameter 'sigma' was held constant at a value of 0.1021364
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were sigma = 0.1021364 and C = 2.
```