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)
library(MLeval)
```

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

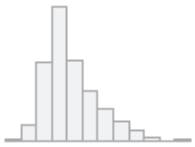
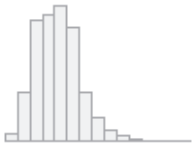
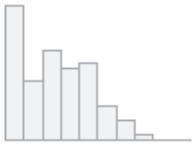
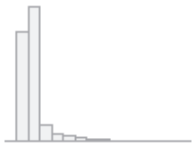
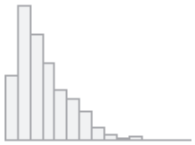
```
wine <- read.csv(
  url("https://raw.githubusercontent.com/OscarG-DataSci/ADS503/main/winequality-red.csv")
                       , header = TRUE)
```

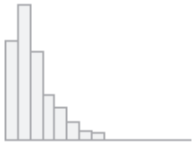
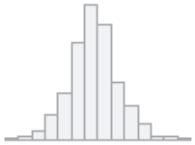
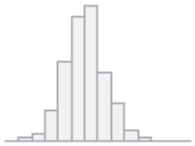
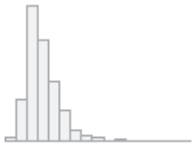
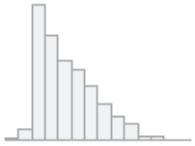
Data Summary

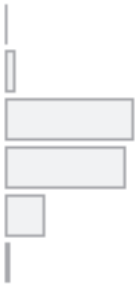
```
dfSummary(wine,
          plain.ascii  = FALSE,
          style        = "grid",
          graph.magnif = 0.75,
          valid.col    = FALSE,
          tmp.img.dir  = "/tmp")
```

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%)

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.dioxide + total.sulfur.dio
xide + density + pH + sulphates + alcohol"
```

```
set.seed(4)

# Model using "quality_target" as target variable
lmodel1 <- lm(quality_target~ volatile.acidity + sulphates + alcohol, data = wine_train)

summary(lmodel1)
```

```
##
## Call:
## lm(formula = quality_target ~ volatile.acidity + sulphates +
##     alcohol, data = wine_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4961 -0.3575 -0.0124  0.3852  1.0375
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.24491    0.14374  -8.661  < 2e-16 ***
## volatile.acidity -0.59412    0.07096  -8.372  < 2e-16 ***
## sulphates         0.39186    0.07526   5.207 2.24e-07 ***
## alcohol           0.17622    0.01154  15.276  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4301 on 1277 degrees of freedom
## Multiple R-squared:  0.2589, Adjusted R-squared:  0.2572
## F-statistic: 148.7 on 3 and 1277 DF,  p-value: < 2.2e-16
```

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

summary(lmodel2)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + sulphates + alcohol,
##     data = wine_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.73372 -0.38264 -0.05799  0.46563  2.16923
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.54533    0.22012  11.563  < 2e-16 ***
## volatile.acidity -1.21682    0.10867 -11.197  < 2e-16 ***
## sulphates         0.75134    0.11525   6.519 1.02e-10 ***
## alcohol           0.31087    0.01767  17.597  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6586 on 1277 degrees of freedom
## Multiple R-squared:  0.3398, Adjusted R-squared:  0.3383
## F-statistic: 219.1 on 3 and 1277 DF,  p-value: < 2.2e-16
```

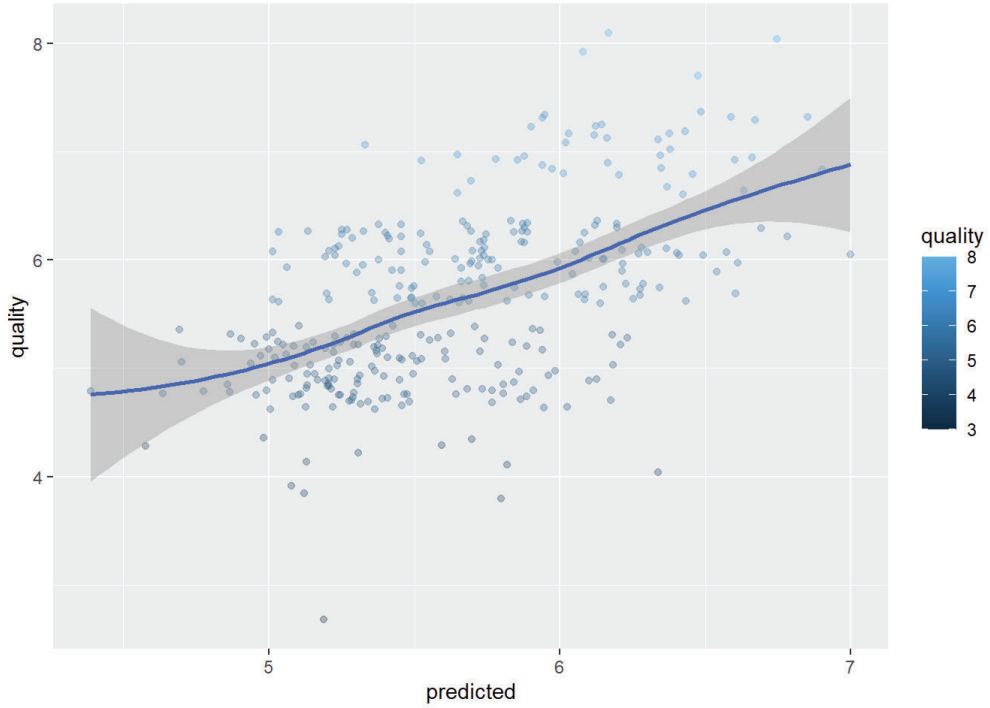
```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(lmodel2, newdata = wine_test)) -> df

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

```
##       fit             lwr             upr
##  Min.   :4.388   Min.   :3.087   Min.   :5.689
##  1st Qu.:5.268   1st Qu.:3.974   1st Qu.:6.562
##  Median :5.625   Median :4.329   Median :6.920
##  Mean   :5.650   Mean   :4.356   Mean   :6.945
##  3rd Qu.:5.990   3rd Qu.:4.697   3rd Qu.:7.283
##  Max.   :6.999   Max.   :5.700   Max.   :8.297
```

```
# Confusion Matrix
# Convert predicted values to whole numbers, so they match target values
df$predicted_int = as.integer(round(df$predicted, digits = 0))

union1 <- union(df$quality, df$predicted_int)
table1 <- table(factor(df$quality, union1), factor(df$predicted_int, union1))

confusionMatrix(table1)
```

```
## Confusion Matrix and Statistics
##
##
##      6  5  4  7  8  3
##   6 81 39  0  7  0  0
##   5 41 94  1  0  0  0
##   4  5  6  0  0  0  0
##   7 31  1  0  7  0  0
##   8  3  0  0  1  0  0
##   3  0  1  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5723
##                  95% CI : (0.5159, 0.6274)
##     No Information Rate : 0.5063
##     P-Value [Acc > NIR] : 0.01064
##
##                   Kappa : 0.2899
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 6 Class: 5 Class: 4 Class: 7 Class: 8 Class: 3
## Sensitivity            0.5031   0.6667 0.000000  0.46667       NA       NA
## Specificity            0.7070   0.7627 0.965300  0.89439  0.98742 0.996855
## Pos Pred Value         0.6378   0.6912 0.000000  0.17949       NA       NA
## Neg Pred Value         0.5812   0.7418 0.996743  0.97133       NA       NA
## Prevalence             0.5063   0.4434 0.003145  0.04717  0.00000 0.000000
## Detection Rate         0.2547   0.2956 0.000000  0.02201  0.00000 0.000000
## Detection Prevalence   0.3994   0.4277 0.034591  0.12264  0.01258 0.003145
## Balanced Accuracy      0.6051   0.7147 0.482650  0.68053       NA       NA
```

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

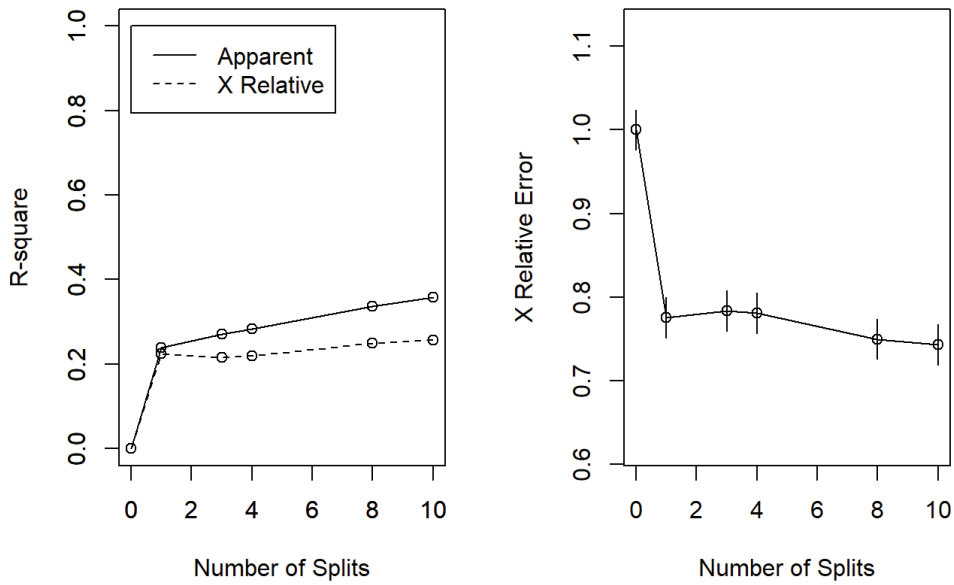



```
# The scatter plot supports the summary of the predicted interval, in the ranges of the fit,
# lower, and upper ranges. The R-squared value of 0.3283 of the model, indicates that this
# information can be predicted 33% of the time, with the data available, for the variance
# of the information.
```

CART

```
set.seed(4)
# Subset both train and test sets, to exclude "quality_target"
# Using non-transformed versions of train and test, to get actual values in the nodes
subset(wine_train, select = -c(quality_target)) -> rf_wine_train
subset(wine_test, select = -c(quality_target)) -> rf_wine_test

# Convert target variable to factor to ensure proper interpretation by model
rf_wine_train$quality <- as.factor(rf_wine_train$quality)

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

rpartTree2 <- as.party(rPartTree)

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

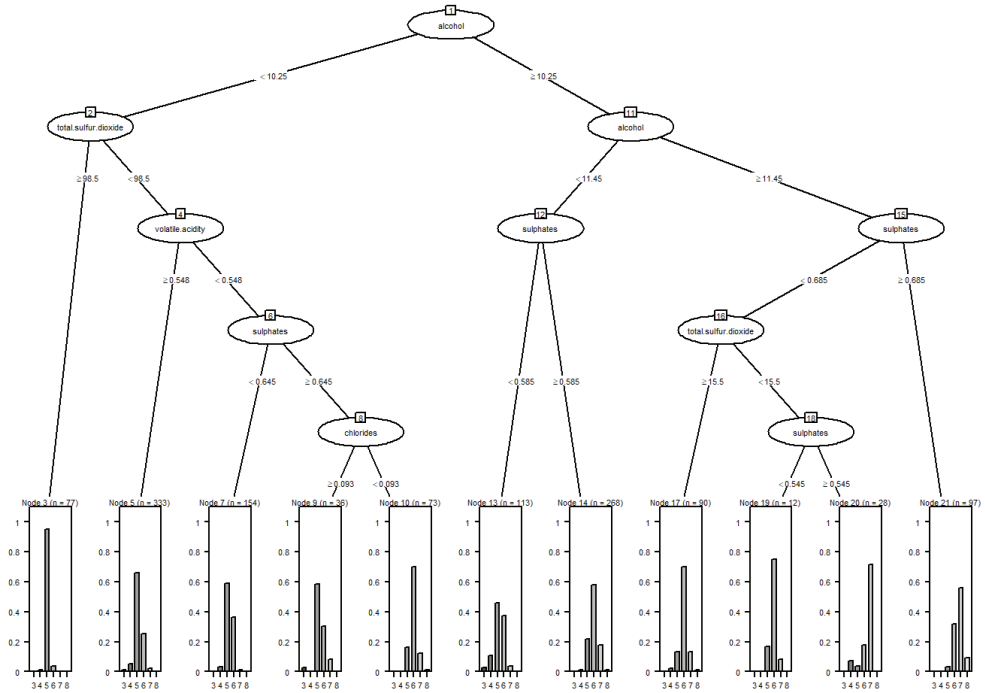
```
##
## Classification tree:
## rpart(formula = quality ~ ., data = rf_wine_train)
##
## Variables actually used in tree construction:
## [1] alcohol              chlorides            sulphates
## [4] total.sulfur.dioxide volatile.acidity
##
## Root node error: 736/1281 = 0.57455
##
## n= 1281
##
##         CP nsplit rel error  xerror     xstd
## 1 0.239130      0   1.00000 1.00000 0.024043
## 2 0.015625      1   0.76087 0.77582 0.024171
## 3 0.013587      3   0.72962 0.78397 0.024195
## 4 0.013134      4   0.71603 0.78125 0.024187
## 5 0.010190      8   0.66304 0.75000 0.024081
## 6 0.010000     10   0.64266 0.74321 0.024054
```




```
# 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 < 10.25
## |   |   [3] total.sulfur.dioxide >= 98.5: 5 (n = 77, err = 5.2%)
## |   |   [4] total.sulfur.dioxide < 98.5
## |   |   |   [5] volatile.acidity >= 0.5475: 5 (n = 333, err = 34.2%)
## |   |   |   [6] volatile.acidity < 0.5475
## |   |   |   |   [7] sulphates < 0.645: 5 (n = 154, err = 40.9%)
## |   |   |   |   [8] sulphates >= 0.645
## |   |   |   |   |   [9] chlorides >= 0.093: 5 (n = 36, err = 41.7%)
## |   |   |   |   |   [10] chlorides < 0.093: 6 (n = 73, err = 30.1%)
## |   [11] alcohol >= 10.25
## |   |   [12] alcohol < 11.45
## |   |   |   [13] sulphates < 0.585: 5 (n = 113, err = 54.0%)
## |   |   |   [14] sulphates >= 0.585: 6 (n = 268, err = 42.2%)
## |   |   [15] alcohol >= 11.45
## |   |   |   [16] sulphates < 0.685
## |   |   |   |   [17] total.sulfur.dioxide >= 15.5: 6 (n = 90, err = 30.0%)
## |   |   |   |   [18] total.sulfur.dioxide < 15.5
## |   |   |   |   |   [19] sulphates < 0.545: 6 (n = 12, err = 25.0%)
## |   |   |   |   |   [20] sulphates >= 0.545: 7 (n = 28, err = 28.6%)
## |   |   |   [21] sulphates >= 0.685: 7 (n = 97, err = 44.3%)
##
## Number of inner nodes:    10
## Number of terminal nodes: 11
```

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



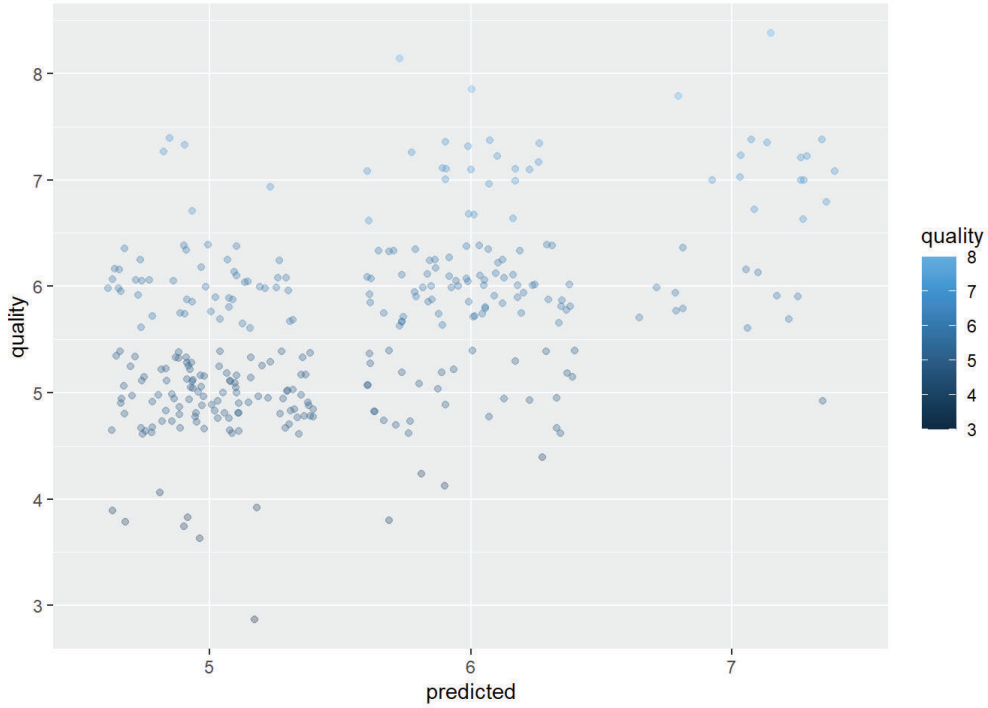

```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(rpartTree2, newdata = wine_test)) -> df2

# Summary of predicted values
predict(rpartTree2, newdata = wine_test, interval = "prediction") %>%
  summary()
```

```
##   3   4   5   6   7   8
##   0   0 166 123  29   0
```

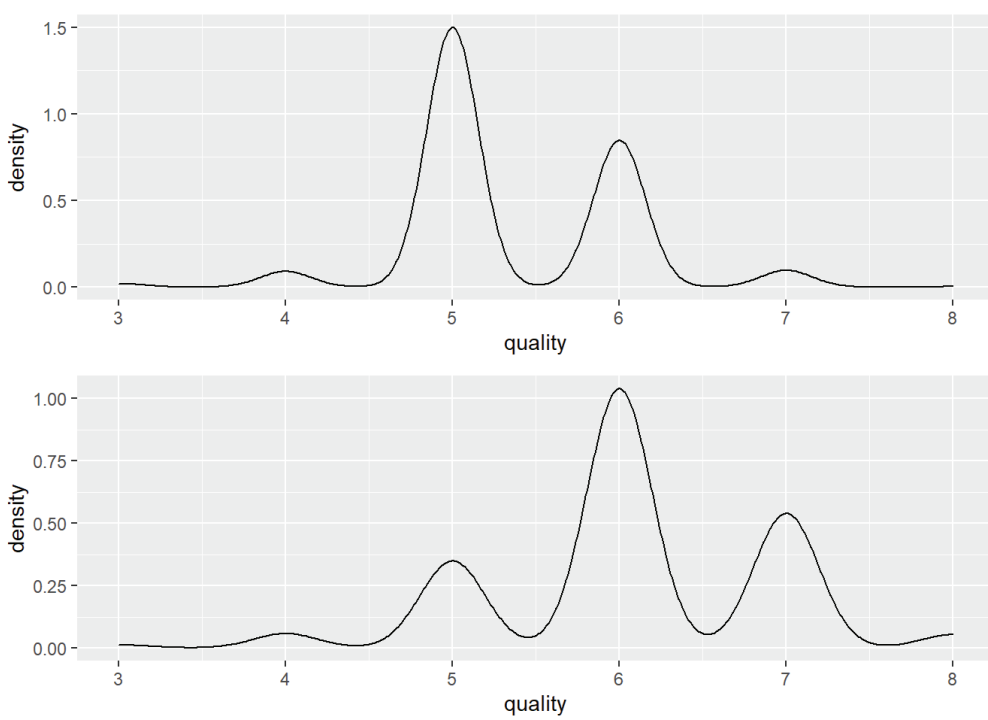
```
# Confusion Matrix
confusionMatrix(table(df2$quality, df2$predicted))
```

```
## Confusion Matrix and Statistics
##
##
##       3   4   5   6   7   8
##   3   0   0   1   0   0   0
##   4   0   0   7   4   0   0
##   5   0   0 106  29   1   0
##   6   0   0  47  68  12   0
##   7   0   0   5  20  14   0
##   8   0   0   0   2   2   0
##
## Overall Statistics
##
##                Accuracy : 0.5912
##                  95% CI : (0.5349, 0.6457)
##     No Information Rate : 0.522
##     P-Value [Acc > NIR] : 0.007744
##
##                   Kappa : 0.331
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity                NA       NA   0.6386   0.5528  0.48276       NA
## Specificity          0.996855  0.96541   0.8026   0.6974  0.91349  0.98742
## Pos Pred Value             NA       NA   0.7794   0.5354  0.35897       NA
## Neg Pred Value             NA       NA   0.6703   0.7120  0.94624       NA
## Prevalence           0.000000  0.00000   0.5220   0.3868  0.09119  0.00000
## Detection Rate       0.000000  0.00000   0.3333   0.2138  0.04403  0.00000
## Detection Prevalence 0.003145  0.03459   0.4277   0.3994  0.12264  0.01258
## Balanced Accuracy          NA       NA   0.7206   0.6251  0.69813       NA
```

```
# Scatter plot of predicted
ggplot(df2, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```




```
# 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)
```

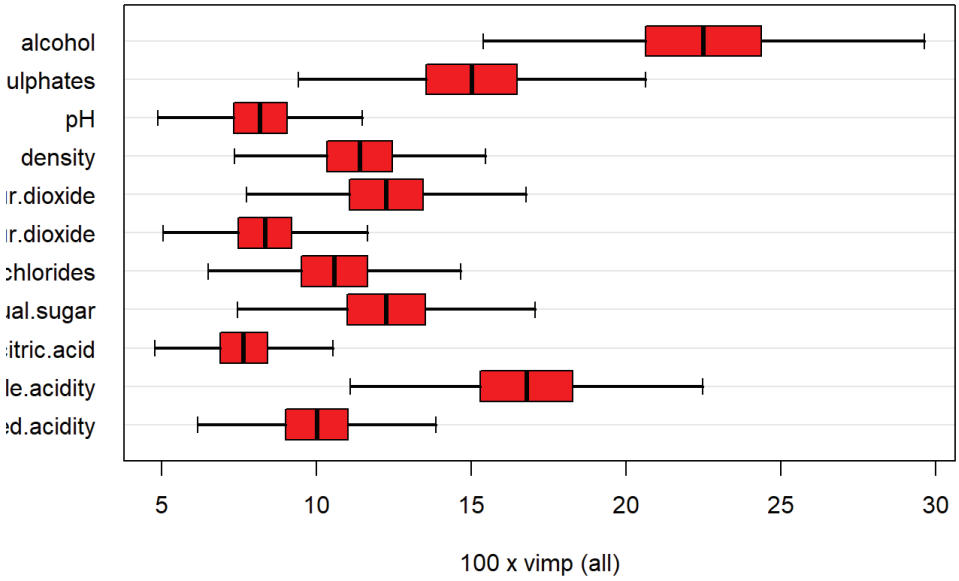
```
##                         Sample size: 1281
##            Frequency of class labels: 9, 42, 545, 511, 160, 14
##                      Number of trees: 500
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 253.004
## No. of variables tried at each split: 4
##               Total no. of variables: 11
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 810
##                             Analysis: RF-C
##                               Family: class
##                      Splitting rule: gini
##                    (OOB) Brier score: 0.0704743
##         (OOB) Normalized Brier score: 0.50741499
##                            (OOB) AUC: 0.83162878
##    (OOB) Requested performance error: 0.30523029, 1, 1, 0.20183486, 0.28767123, 0.44375, 0.85714286
##
## Confusion matrix:
##
##           predicted
##   observed 3 4   5   6  7 8 class.error
##          3 0 0   7   2  0 0      1.0000
##          4 0 0  30  11  1 0      1.0000
##          5 0 0 436 106  3 0      0.2000
##          6 0 2 121 362 26 0      0.2916
##          7 0 0  12  58 88 2      0.4500
##          8 0 0   0   6  6 2      0.8571
##
##       (OOB) Misclassification rate: 0.3067916
```

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

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

	lower <dbl>	mean <dbl>	upper <dbl>	pvalue <dbl>	signif <lgl>
fixed.acidity	7.097717	10.019533	12.941349	9.015586e-12	TRUE
volatile.acidity	12.450527	16.790234	21.129940	1.687493e-14	TRUE
citric.acid	5.470282	7.660722	9.851161	3.573883e-12	TRUE
residual.sugar	8.613770	12.264544	15.915317	2.284013e-11	TRUE
chlorides	7.493438	10.597028	13.700617	1.099326e-11	TRUE
free.sulfur.dioxide	5.840307	8.350743	10.861179	3.523638e-11	TRUE
total.sulfur.dioxide	8.825910	12.263694	15.701478	1.356620e-12	TRUE
density	8.316326	11.402972	14.489619	2.232223e-13	TRUE
pH	5.678052	8.188110	10.698169	8.099363e-11	TRUE
sulphates	10.756862	15.022081	19.287300	2.546078e-12	TRUE

1-10 of 11 rows Previous **1** 2 Next

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




```
# Confusion Matrix
# https://www.rdocumentation.org/packages/randomForestSRC/versions/3.1.0/topics/predict.rfsrc
randomForestSRC::predict.rfsrc(rf, rf_wine_test)
```

```
##   Sample size of test (predict) data: 318
##                 Number of grow trees: 500
##   Average no. of grow terminal nodes: 253.004
##          Total no. of grow variables: 11
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 810
##                             Analysis: RF-C
##                               Family: class
##                          Brier score: 0.06884518
##               Normalized Brier score: 0.49568529
##                                  AUC: 0.79949044
##          Requested performance error: 0.27987421, 1, 1, 0.19852941, 0.21259843, 0.48717949, 1
##
## Confusion matrix:
##
##           predicted
##   observed 3 4   5   6  7 8 class.error
##          3 0 0   1   0  0 0      1.0000
##          4 1 0   5   5  0 0      1.0000
##          5 0 0 109  26  1 0      0.1985
##          6 0 0  22 100  5 0      0.2126
##          7 0 0   0  19 20 0      0.4872
##          8 0 0   0   2  2 0      1.0000
##
##            Misclassification error: 0.2798742
```

Partial Least Squares

```
tctrl <- trainControl(method = "repeatedcv", repeats = 5, number =10)

set.seed(4)
pls_wine <- train(quality~ volatile.acidity + chlorides + total.sulfur.dioxide +
               sulphates + alcohol, data = wine_train,
                  method = "pls",
                  preProc = c("center", "scale", "BoxCox"),
                  tunelength =20,
                  trControl = tctrl)

pls_wine
```

```
## Partial Least Squares
##
## 1281 samples
##    5 predictor
##
## Pre-processing: centered (5), scaled (5), Box-Cox transformation (5)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 1153, 1153, 1153, 1154, 1153, 1152, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE       Rsquared   MAE
##   1      0.6498294  0.3586595  0.5040683
##   2      0.6488990  0.3607768  0.5041411
##   3      0.6487749  0.3610801  0.5032738
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 3.
```

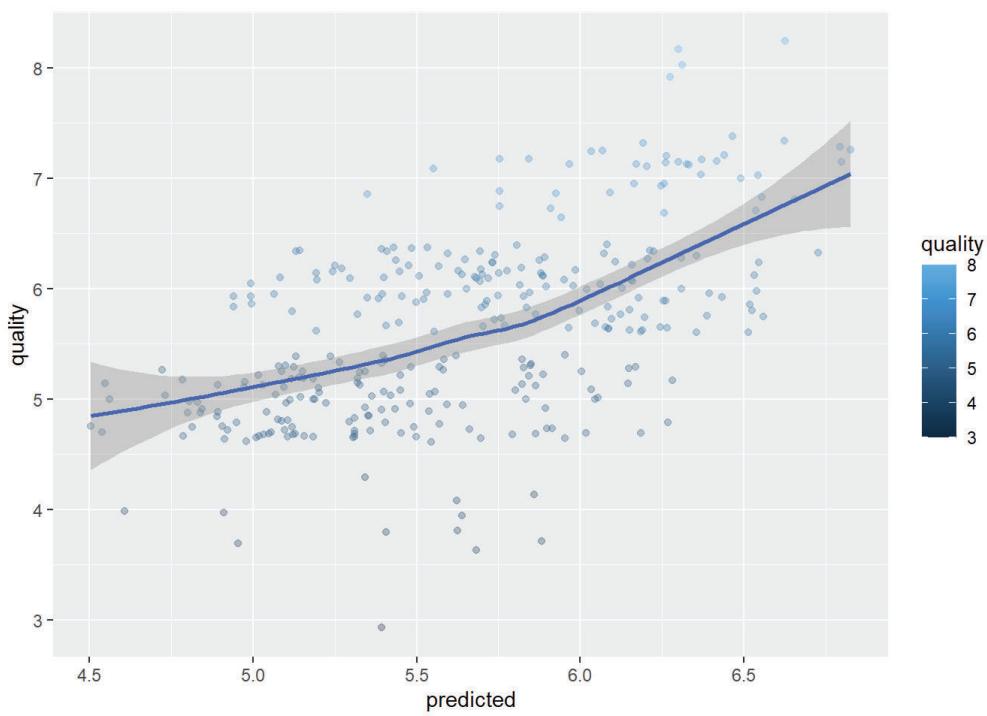
```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(pls_wine, newdata = wine_test)) -> df3

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.504   5.246   5.640   5.649   6.060   6.826
```

```
# Confusion Matrix
# Convert predicted values to whole numbers, so they match target values
df3$predicted_int = as.integer(round(df3$predicted, digits = 0))

union3 <- union(df3$quality, df3$predicted_int)
table3 <- table(factor(df3$quality, union3), factor(df3$predicted_int, union3))

confusionMatrix(table3)
```

```
## Confusion Matrix and Statistics
##
##
##      6  5  4  7  8  3
##   6 86 33  0  8  0  0
##   5 41 95  0  0  0  0
##   4  6  5  0  0  0  0
##   7 30  1  0  8  0  0
##   8  3  0  0  1  0  0
##   3  0  1  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5943
##                  95% CI : (0.5381, 0.6488)
##     No Information Rate : 0.522
##     P-Value [Acc > NIR] : 0.005638
##
##                   Kappa : 0.3277
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 6 Class: 5 Class: 4 Class: 7 Class: 8 Class: 3
## Sensitivity            0.5181   0.7037       NA  0.47059       NA       NA
## Specificity            0.7303   0.7760  0.96541  0.89701  0.98742 0.996855
## Pos Pred Value         0.6772   0.6985       NA  0.20513       NA       NA
## Neg Pred Value         0.5812   0.7802       NA  0.96774       NA       NA
## Prevalence             0.5220   0.4245  0.00000  0.05346  0.00000 0.000000
## Detection Rate         0.2704   0.2987  0.00000  0.02516  0.00000 0.000000
## Detection Prevalence   0.3994   0.4277  0.03459  0.12264  0.01258 0.003145
## Balanced Accuracy      0.6242   0.7398       NA  0.68380       NA       NA
```

```
# Scatter plot of predicted
ggplot(df3, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```



Mars Tuning

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

mars_wine
```

```
## Selected 14 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 13 (additive model)
## GCV 0.4105303    RSS 503.9543    GRSq 0.374189    RSq 0.3993544
```

```
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##             data=wine_train)
##
##                              coefficients
## (Intercept)                     19.879407
## h(0.84-volatile.acidity)         0.837011
## h(volatile.acidity-0.84)        -1.669799
## h(chlorides-0.041)              66.097055
## h(chlorides-0.065)             -15.210883
## h(0.153-chlorides)              54.140361
## h(chlorides-0.153)             -51.061898
## h(total.sulfur.dioxide-10)      -0.151609
## h(144-total.sulfur.dioxide)     -0.149050
## h(total.sulfur.dioxide-144)      0.159313
## h(0.76-sulphates)               -2.161289
## h(alcohol-11.1)                  0.268634
## h(12.4-alcohol)                 -0.228474
## h(alcohol-12.4)                 -0.419977
##
## Selected 14 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 13 (additive model)
## GCV 0.4105303    RSS 503.9543    GRSq 0.374189    RSq 0.3993544
```

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

set.seed(4)

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

marsModel_wine
```

```
## Multivariate Adaptive Regression Spline
##
## 1281 samples
##    5 predictor
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1281, 1281, 1281, 1281, 1281, 1281, ...
## Resampling results across tuning parameters:
##
##   degree  nprune  RMSE       Rsquared   MAE
##   1        2      0.7197874  0.2223572  0.5662870
##   1        3      0.6825800  0.3004366  0.5272729
##   1        4      0.6585235  0.3492197  0.5091231
##   1        5      0.6602654  0.3462091  0.5107050
##   1        6      0.6614945  0.3442458  0.5103259
##   1        7      0.6624907  0.3428342  0.5103855
##   1        8      0.6627237  0.3425995  0.5108516
##   1        9      0.6628261  0.3429884  0.5109822
##   1       10      0.6625510  0.3441167  0.5110786
##   1       11      0.6616949  0.3468105  0.5110675
##   1       12      0.6623159  0.3464811  0.5116324
##   1       13      0.6630560  0.3456522  0.5124295
##   1       14      0.6640396  0.3443099  0.5127140
##   1       15      0.6644512  0.3435491  0.5128637
##   1       16      0.6643828  0.3436181  0.5128149
##   1       17      0.6643828  0.3436181  0.5128149
##   1       18      0.6643828  0.3436181  0.5128149
##   1       19      0.6643828  0.3436181  0.5128149
##   1       20      0.6643828  0.3436181  0.5128149
##   1       21      0.6643828  0.3436181  0.5128149
##   1       22      0.6643828  0.3436181  0.5128149
##   1       23      0.6643828  0.3436181  0.5128149
##   1       24      0.6643828  0.3436181  0.5128149
##   1       25      0.6643828  0.3436181  0.5128149
##   1       26      0.6643828  0.3436181  0.5128149
##   1       27      0.6643828  0.3436181  0.5128149
##   1       28      0.6643828  0.3436181  0.5128149
##   1       29      0.6643828  0.3436181  0.5128149
##   1       30      0.6643828  0.3436181  0.5128149
##   1       31      0.6643828  0.3436181  0.5128149
##   1       32      0.6643828  0.3436181  0.5128149
##   1       33      0.6643828  0.3436181  0.5128149
##   1       34      0.6643828  0.3436181  0.5128149
##   1       35      0.6643828  0.3436181  0.5128149
##   1       36      0.6643828  0.3436181  0.5128149
##   1       37      0.6643828  0.3436181  0.5128149
##   1       38      0.6643828  0.3436181  0.5128149
##   2        2      0.7190828  0.2238948  0.5644229
##   2        3      0.6865904  0.2927491  0.5302202
##   2        4      0.6626047  0.3418392  0.5115419
##   2        5      0.6561552  0.3551095  0.5055840
##   2        6      0.6558502  0.3561448  0.5055829
##   2        7      0.6583839  0.3520635  0.5058738
##   2        8      0.6591888  0.3511027  0.5063643
##   2        9      0.6623530  0.3457923  0.5086520
##   2       10      0.6637473  0.3442736  0.5097710
##   2       11      0.6658840  0.3399237  0.5116165
##   2       12      0.6663448  0.3396822  0.5118317
##   2       13      0.6656417  0.3416827  0.5115254
##   2       14      0.6658405  0.3414090  0.5118307
##   2       15      0.6658675  0.3416468  0.5118444
##   2       16      0.6661386  0.3411474  0.5121211
##   2       17      0.6661386  0.3411474  0.5121211
##   2       18      0.6661386  0.3411474  0.5121211
##   2       19      0.6661386  0.3411474  0.5121211
##   2       20      0.6661386  0.3411474  0.5121211
##   2       21      0.6661386  0.3411474  0.5121211
##   2       22      0.6661386  0.3411474  0.5121211
```

```
##   2       23      0.6661386  0.3411474  0.5121211
##   2       24      0.6661386  0.3411474  0.5121211
##   2       25      0.6661386  0.3411474  0.5121211
##   2       26      0.6661386  0.3411474  0.5121211
##   2       27      0.6661386  0.3411474  0.5121211
##   2       28      0.6661386  0.3411474  0.5121211
##   2       29      0.6661386  0.3411474  0.5121211
##   2       30      0.6661386  0.3411474  0.5121211
##   2       31      0.6661386  0.3411474  0.5121211
##   2       32      0.6661386  0.3411474  0.5121211
##   2       33      0.6661386  0.3411474  0.5121211
##   2       34      0.6661386  0.3411474  0.5121211
##   2       35      0.6661386  0.3411474  0.5121211
##   2       36      0.6661386  0.3411474  0.5121211
##   2       37      0.6661386  0.3411474  0.5121211
##   2       38      0.6661386  0.3411474  0.5121211
##
## 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.
```

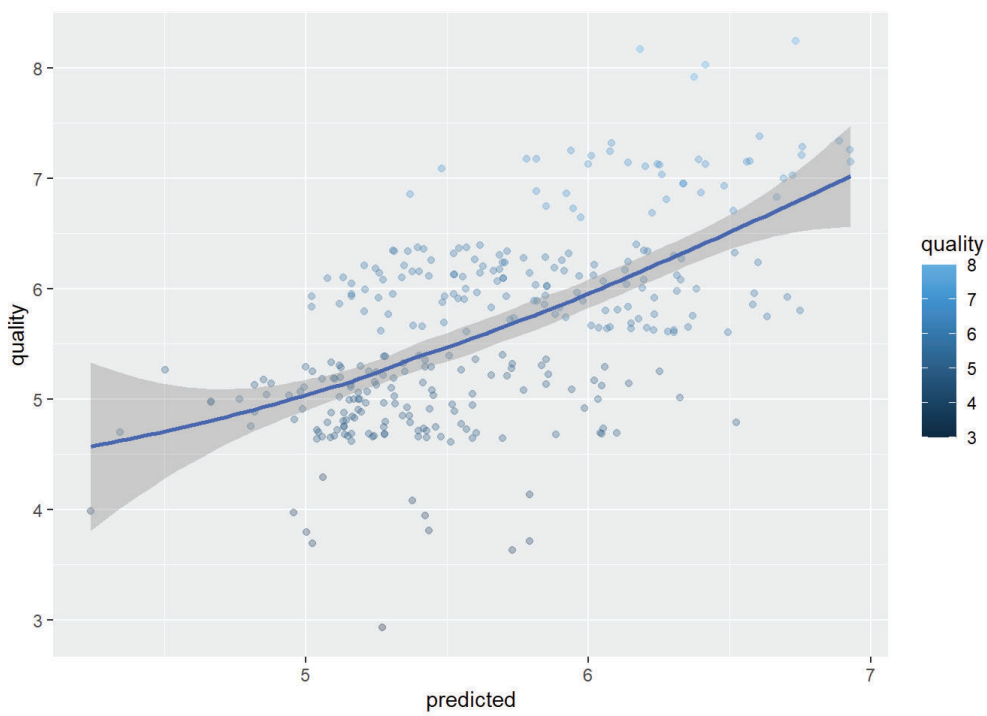
```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(marsModel_wine, newdata = wine_test)) -> df4

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

```
##        y
##  Min.   :4.239
##  1st Qu.:5.247
##  Median :5.569
##  Mean   :5.652
##  3rd Qu.:6.047
##  Max.   :6.929
```

```
# Confusion Matrix
# Convert predicted values to whole numbers, so they match target values
df4$predicted_int = as.integer(round(df4$predicted, digits = 0))

union4 <- union(df4$quality, df4$predicted_int)
table4 <- table(factor(df4$quality, union4), factor(df4$predicted_int, union4))

confusionMatrix(table4)
```

```
## Confusion Matrix and Statistics
##
##
##      6  5  4  7  8  3
##   6 86 34  0  7  0  0
##   5 37 97  1  1  0  0
##   4  3  7  1  0  0  0
##   7 25  2  0 12  0  0
##   8  3  0  0  1  0  0
##   3  0  1  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.6164
##                  95% CI : (0.5605, 0.6701)
##     No Information Rate : 0.4843
##     P-Value [Acc > NIR] : 1.506e-06
##
##                   Kappa : 0.3697
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 6 Class: 5 Class: 4 Class: 7 Class: 8 Class: 3
## Sensitivity            0.5584   0.6879 0.500000  0.57143       NA       NA
## Specificity            0.7500   0.7797 0.968354  0.90909  0.98742 0.996855
## Pos Pred Value         0.6772   0.7132 0.090909  0.30769       NA       NA
## Neg Pred Value         0.6440   0.7582 0.996743  0.96774       NA       NA
## Prevalence             0.4843   0.4434 0.006289  0.06604  0.00000 0.000000
## Detection Rate         0.2704   0.3050 0.003145  0.03774  0.00000 0.000000
## Detection Prevalence   0.3994   0.4277 0.034591  0.12264  0.01258 0.003145
## Balanced Accuracy      0.6542   0.7338 0.734177  0.74026       NA       NA
```

```
# Scatter plot of predicted
ggplot(df4, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```



KNN Neighbors

```
set.seed(4)

knn_wine <- train(quality~ volatile.acidity + chlorides + total.sulfur.dioxide +
                sulphates + alcohol, data =wine_train,
                method = "knn",
                preProc = c("center", "scale"),
                tuneGrid = data.frame(.k = 1:50),
                trControl = trainControl(method = "cv"))

knn_wine
```

```
## k-Nearest Neighbors
##
## 1281 samples
##    5 predictor
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1153, 1153, 1153, 1154, 1153, 1152, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1  0.7935269  0.2769418  0.4590599
##    2  0.7213398  0.3064564  0.4954260
##    3  0.6944565  0.3154702  0.4948899
##    4  0.6856622  0.3172616  0.5046710
##    5  0.6789166  0.3220406  0.5052352
##    6  0.6722626  0.3288785  0.5031102
##    7  0.6619129  0.3435928  0.4999783
##    8  0.6562397  0.3527092  0.4964692
##    9  0.6523210  0.3583595  0.4955993
##   10  0.6494401  0.3619102  0.4952726
##   11  0.6498126  0.3604516  0.4983871
##   12  0.6510228  0.3576209  0.5008491
##   13  0.6494215  0.3592247  0.5016061
##   14  0.6495125  0.3586148  0.5026891
##   15  0.6469544  0.3629958  0.5024382
##   16  0.6456482  0.3655799  0.5030477
##   17  0.6471710  0.3626931  0.5040605
##   18  0.6479562  0.3613810  0.5048495
##   19  0.6473945  0.3626956  0.5043652
##   20  0.6480852  0.3611963  0.5050057
##   21  0.6461322  0.3646179  0.5038633
##   22  0.6440763  0.3688750  0.5036697
##   23  0.6448699  0.3671915  0.5034955
##   24  0.6447396  0.3676172  0.5027733
##   25  0.6460655  0.3651104  0.5036741
##   26  0.6456459  0.3658621  0.5043910
##   27  0.6454002  0.3661103  0.5042119
##   28  0.6460713  0.3647149  0.5053841
##   29  0.6465961  0.3634052  0.5061291
##   30  0.6467485  0.3633613  0.5068561
##   31  0.6465795  0.3637940  0.5075609
##   32  0.6470232  0.3628616  0.5076586
##   33  0.6463355  0.3643392  0.5074812
##   34  0.6464626  0.3642634  0.5076670
##   35  0.6471034  0.3631442  0.5080876
##   36  0.6464493  0.3644586  0.5071782
##   37  0.6470362  0.3630809  0.5072225
##   38  0.6465670  0.3641242  0.5067782
##   39  0.6464922  0.3644830  0.5066028
##   40  0.6462225  0.3649749  0.5064441
##   41  0.6461059  0.3652473  0.5059272
##   42  0.6462682  0.3651880  0.5061808
##   43  0.6456151  0.3666804  0.5054070
##   44  0.6451156  0.3679121  0.5053330
##   45  0.6456022  0.3671304  0.5056105
##   46  0.6457759  0.3669521  0.5059217
##   47  0.6455425  0.3673221  0.5057689
##   48  0.6451148  0.3685085  0.5053595
##   49  0.6458023  0.3672600  0.5061187
##   50  0.6455735  0.3678700  0.5059238
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 22.
```

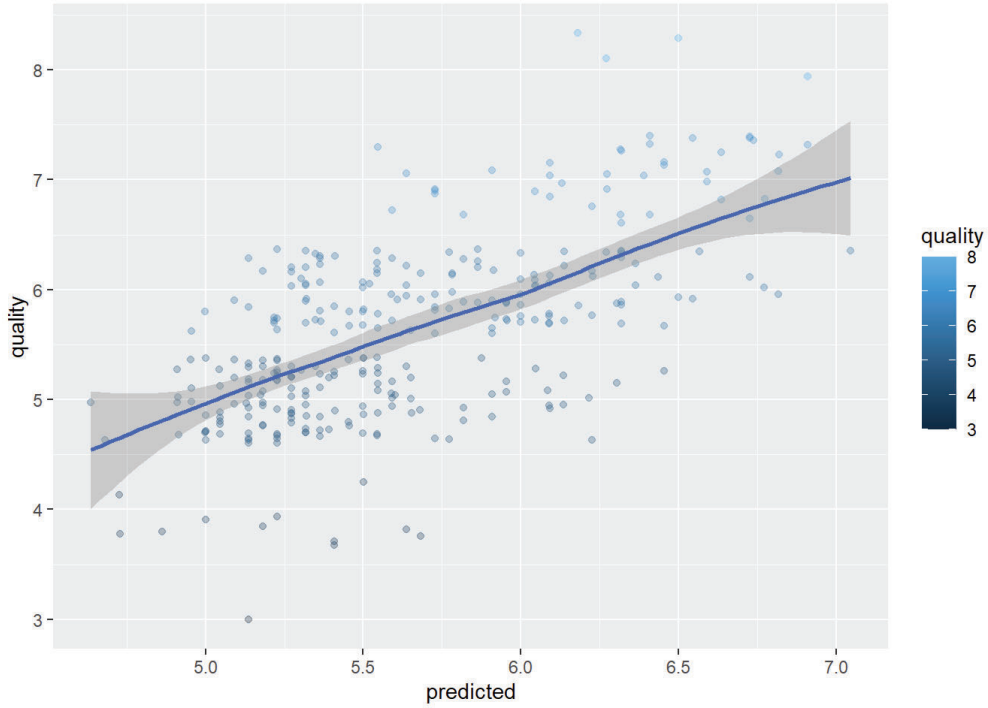
```r
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(knn_wine, newdata = wine_test)) -> df5

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.636   5.273   5.545   5.658   6.045   7.045
```

```r
# Confusion Matrix
# Convert predicted values to whole numbers, so they match target values
df5$predicted_int = as.integer(round(df5$predicted, digits = 0))

union5 <- union(df5$quality, df5$predicted_int)
table5 <- table(factor(df5$quality, union5), factor(df5$predicted_int, union5))

confusionMatrix(table5)
```

```
## Confusion Matrix and Statistics
##
##
##      6  5  4  7  8  3
##   6 85 36  0  6  0  0
##   5 44 92  0  0  0  0
##   4  3  8  0  0  0  0
##   7 26  0  0 13  0  0
##   8  3  0  0  1  0  0
##   3  0  1  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5975
##                  95% CI : (0.5413, 0.6518)
##     No Information Rate : 0.5063
##     P-Value [Acc > NIR] : 0.0006714
##
##                   Kappa : 0.3356
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 6 Class: 5 Class: 4 Class: 7 Class: 8 Class: 3
## Sensitivity            0.5280   0.6715       NA  0.65000       NA       NA
## Specificity            0.7325   0.7569  0.96541  0.91275  0.98742 0.996855
## Pos Pred Value         0.6693   0.6765       NA  0.33333       NA       NA
## Neg Pred Value         0.6021   0.7527       NA  0.97491       NA       NA
## Prevalence             0.5063   0.4308  0.00000  0.06289  0.00000 0.000000
## Detection Rate         0.2673   0.2893  0.00000  0.04088  0.00000 0.000000
## Detection Prevalence   0.3994   0.4277  0.03459  0.12264  0.01258 0.003145
## Balanced Accuracy      0.6302   0.7142       NA  0.78138       NA       NA
```

```r
# Scatter plot of predicted
ggplot(df5, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```



SVM

```
set.seed(4)


svmTune <- train(quality ~ ., data = rf_wine_train, # using the subset data as used in random forest
                 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
##    6 classes: '3', '4', '5', '6', '7', '8'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1154, 1153, 1154, 1151, 1153, 1153, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.5776131  0.2804971
##   0.50  0.5971880  0.3271770
##   1.00  0.6065035  0.3508358
##   2.00  0.6111674  0.3597917
##   4.00  0.6096231  0.3665122
##
## Tuning parameter 'sigma' was held constant at a value of 0.08887198
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.08887198 and C = 2.
```

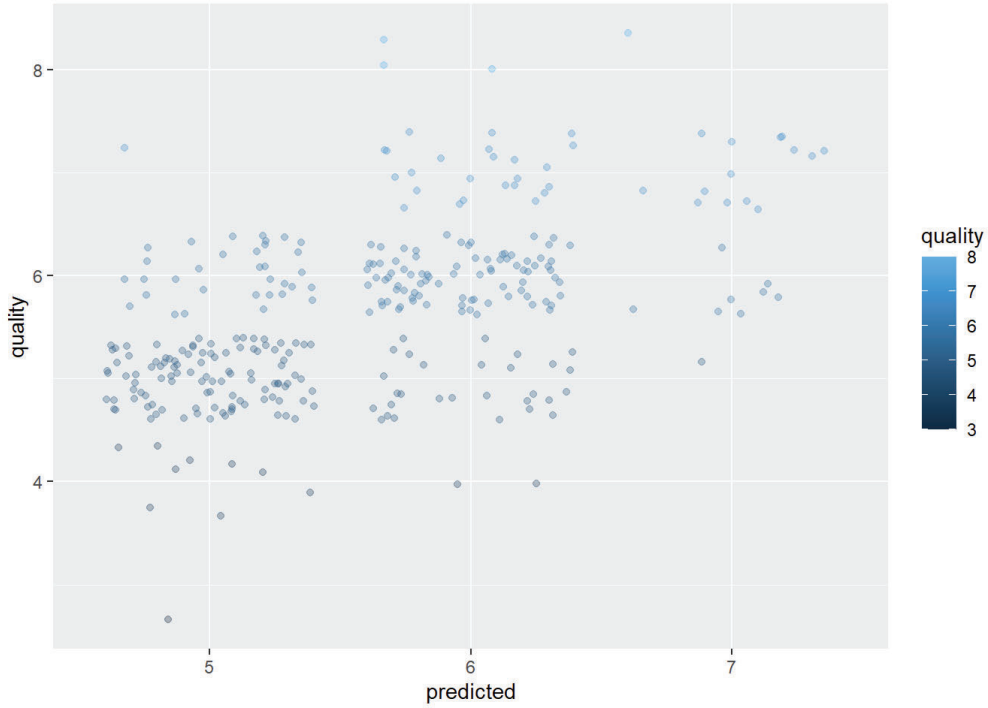
```r
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(svmTune, newdata = wine_test)) -> df6

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

```
##   3   4   5   6   7   8
##   0   0 150 144  24   0
```

```r
# Confusion Matrix
confusionMatrix(table(df6$quality, df6$predicted))
```

```
## Confusion Matrix and Statistics
##
##
##       3   4   5   6   7   8
##   3   0   0   1   0   0   0
##   4   0   0   9   2   0   0
##   5   0   0 106  29   1   0
##   6   0   0  33  86   8   0
##   7   0   0   1  24  14   0
##   8   0   0   0   3   1   0
##
## Overall Statistics
##
##                Accuracy : 0.6478
##                  95% CI : (0.5925, 0.7003)
##     No Information Rate : 0.4717
##     P-Value [Acc > NIR] : 1.96e-10
##
##                   Kappa : 0.4209
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity                NA       NA   0.7067   0.5972  0.58333       NA
## Specificity          0.996855  0.96541   0.8214   0.7644  0.91497  0.98742
## Pos Pred Value             NA       NA   0.7794   0.6772  0.35897       NA
## Neg Pred Value             NA       NA   0.7582   0.6963  0.96416       NA
## Prevalence           0.000000  0.00000   0.4717   0.4528  0.07547  0.00000
## Detection Rate       0.000000  0.00000   0.3333   0.2704  0.04403  0.00000
## Detection Prevalence 0.003145  0.03459   0.4277   0.3994  0.12264  0.01258
## Balanced Accuracy          NA       NA   0.7640   0.6808  0.74915       NA
```

```r
# Scatter plot of predicted
ggplot(df6, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```



Penalized Logistic Regression Tuning

```
#tuning parameters, alpha is associated with the ridge(0) versus lasso regression(1)
glmnGrid <- expand.grid(alpha = c(0,  .1,  .2, .4, .6, .8, 1),
                        lambda = seq(.01, .2, length = 5))
glmnTune <- train(quality ~ ., data = rf_wine_train, # using the subset data as used in random forest,
                  method = "glmnet",
                  tuneGrid = glmnGrid,
                  preProc = c("center", "scale"),
                  trControl = trainControl(method = "cv"))
glmnTune
```

```
## glmnet
##
## 1281 samples
##   11 predictor
##    6 classes: '3', '4', '5', '6', '7', '8'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1152, 1154, 1152, 1153, 1152, 1152, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda  Accuracy   Kappa
##   0.0    0.0100  0.5814959  0.3031037
##   0.0    0.0575  0.5729322  0.2821836
##   0.0    0.1050  0.5674388  0.2666047
##   0.0    0.1525  0.5690014  0.2669168
##   0.0    0.2000  0.5690075  0.2658045
##   0.1    0.0100  0.5807451  0.3066803
##   0.1    0.0575  0.5721204  0.2780257
##   0.1    0.1050  0.5682141  0.2664099
##   0.1    0.1525  0.5666881  0.2616358
##   0.1    0.2000  0.5596687  0.2491799
##   0.2    0.0100  0.5854145  0.3141001
##   0.2    0.0575  0.5728957  0.2774506
##   0.2    0.1050  0.5690075  0.2662369
##   0.2    0.1525  0.5651437  0.2585185
##   0.2    0.2000  0.5588934  0.2471309
##   0.4    0.0100  0.5838704  0.3066920
##   0.4    0.0575  0.5674266  0.2660310
##   0.4    0.1050  0.5581182  0.2464599
##   0.4    0.1525  0.5588994  0.2469575
##   0.4    0.2000  0.5620002  0.2515601
##   0.6    0.0100  0.5807391  0.3020759
##   0.6    0.0575  0.5604254  0.2524909
##   0.6    0.1050  0.5596808  0.2485583
##   0.6    0.1525  0.5581120  0.2450883
##   0.6    0.2000  0.5510253  0.2315798
##   0.8    0.0100  0.5815387  0.3032196
##   0.8    0.0575  0.5589117  0.2482871
##   0.8    0.1050  0.5557620  0.2413896
##   0.8    0.1525  0.5518005  0.2329377
##   0.8    0.2000  0.5362051  0.2050039
##   1.0    0.0100  0.5823016  0.3039459
##   1.0    0.0575  0.5604560  0.2504024
##   1.0    0.1050  0.5526124  0.2349537
##   1.0    0.1525  0.5447871  0.2204280
##   1.0    0.2000  0.4340631  0.0166399
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.2 and lambda = 0.01.
```

```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(glmnTune, newdata = wine_test)) -> df7

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

```
##   3   4   5   6   7   8
##   0   0 150 144  24   0
```

```
# Confusion Matrix
confusionMatrix(table(df7$quality, df7$predicted))
```

```
## Confusion Matrix and Statistics
##
##
##       3   4   5   6   7   8
##   3   0   0   1   0   0   0
##   4   0   0   7   4   0   0
##   5   0   0 104  32   0   0
##   6   0   0  36  83   8   0
##   7   0   0   1  27  11   0
##   8   0   0   0   2   2   0
##
## Overall Statistics
##
##                Accuracy : 0.6226
##                  95% CI : (0.5668, 0.6761)
##     No Information Rate : 0.4686
##     P-Value [Acc > NIR] : 2.411e-08
##
##                   Kappa : 0.3769
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity                NA       NA   0.6980   0.5608  0.52381       NA
## Specificity          0.996855  0.96541   0.8107   0.7412  0.90572  0.98742
## Pos Pred Value             NA       NA   0.7647   0.6535  0.28205       NA
## Neg Pred Value             NA       NA   0.7527   0.6597  0.96416       NA
## Prevalence           0.000000  0.00000   0.4686   0.4654  0.06604  0.00000
## Detection Rate       0.000000  0.00000   0.3270   0.2610  0.03459  0.00000
## Detection Prevalence 0.003145  0.03459   0.4277   0.3994  0.12264  0.01258
## Balanced Accuracy          NA       NA   0.7543   0.6510  0.71477       NA
```

```
# Scatter plot of predicted
ggplot(df7, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```