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)
library(grid)
library(MLeval)
library(pROC)
```

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

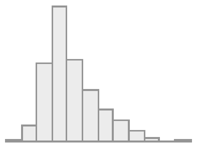
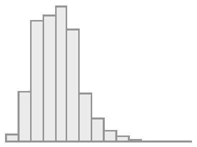
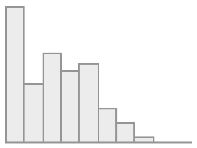
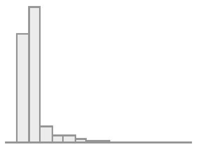
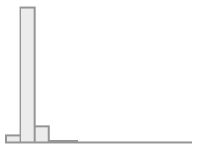
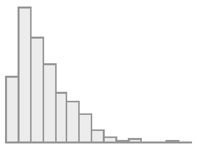
```r
wine <- read.csv(
  url("https://raw.githubusercontent.com/OscarG-DataSci/ADS503/main/winequality-red.csv")
                      , header = TRUE)
```

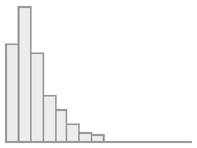
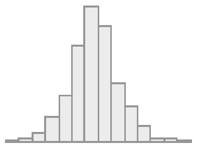
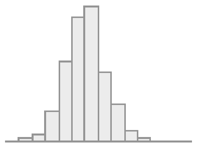
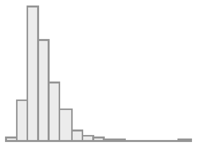
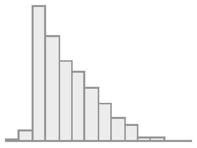
Data Summary

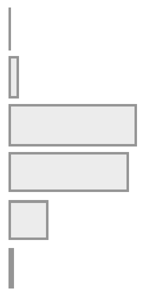
```r
# use the view function to view in R Studio
#view(
dfSummary(wine,
          plain.ascii  = FALSE,
          style        = "grid",
          graph.magnif = 0.75,
          valid.col    = FALSE,
          tmp.img.dir  = "NA")
```

Data Frame Summary

wine **Dimensions:** 1599 x 12
Duplicates: 240

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Missing
1	fixed.acidity [numeric]	Mean (sd) : 8.3 (1.7) min < med < max: 4.6 < 7.9 < 15.9 IQR (CV) : 2.1 (0.2)	96 distinct values		0 (0.0%)
2	volatile.acidity [numeric]	Mean (sd) : 0.5 (0.2) min < med < max: 0.1 < 0.5 < 1.6 IQR (CV) : 0.2 (0.3)	143 distinct values		0 (0.0%)
3	citric.acid [numeric]	Mean (sd) : 0.3 (0.2) min < med < max: 0 < 0.3 < 1 IQR (CV) : 0.3 (0.7)	80 distinct values		0 (0.0%)
4	residual.sugar [numeric]	Mean (sd) : 2.5 (1.4) min < med < max: 0.9 < 2.2 < 15.5 IQR (CV) : 0.7 (0.6)	91 distinct values		0 (0.0%)
5	chlorides [numeric]	Mean (sd) : 0.1 (0) min < med < max: 0 < 0.1 < 0.6 IQR (CV) : 0 (0.5)	153 distinct values		0 (0.0%)
6	free.sulfur.dioxide [numeric]	Mean (sd) : 15.9 (10.5) min < med < max: 1 < 14 < 72 IQR (CV) : 14 (0.7)	60 distinct values		0 (0.0%)

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Missing
7	total.sulfur.dioxide [numeric]	Mean (sd) : 46.5 (32.9) min < med < max: 6 < 38 < 289 IQR (CV) : 40 (0.7)	144 distinct values		0 (0.0%)
8	density [numeric]	Mean (sd) : 1 (0) min < med < max: 1 < 1 < 1 IQR (CV) : 0 (0)	436 distinct values		0 (0.0%)
9	pH [numeric]	Mean (sd) : 3.3 (0.2) min < med < max: 2.7 < 3.3 < 4 IQR (CV) : 0.2 (0)	89 distinct values		0 (0.0%)
10	sulphates [numeric]	Mean (sd) : 0.7 (0.2) min < med < max: 0.3 < 0.6 < 2 IQR (CV) : 0.2 (0.3)	96 distinct values		0 (0.0%)
11	alcohol [numeric]	Mean (sd) : 10.4 (1.1) min < med < max: 8.4 < 10.2 < 14.9 IQR (CV) : 1.6 (0.1)	65 distinct values		0 (0.0%)

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Missing
12	quality [integer]	Mean (sd) : 5.6 (0.8) min < med < max: 3 < 6 < 8 IQR (CV) : 1 (0.1)	3 : 10 (0.6%) 4 : 53 (3.3%) 5 : 681 (42.6%) 6 : 638 (39.9%) 7 : 199 (12.4%) 8 : 18 (1.1%)		0 (0.0%)

```
#    )
```

Pre-processing

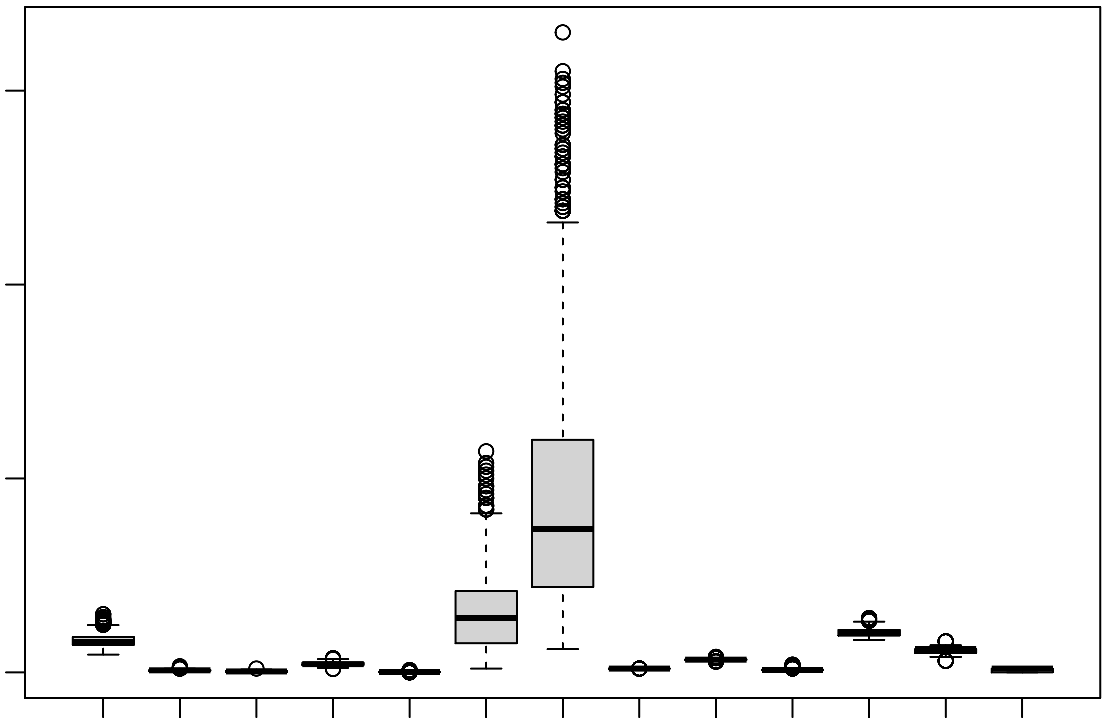
```
par(mar=c(1,1,1,1)) # to fix boxplot knit processing issues

# Create new variable, for quality values, split by half (0, 1)
wine$quality_target <- ifelse( wine$quality <= 5, 0, 1)

# Mean of new variable is at 0.5347 (close enough to 50% to maintain balance)
summary(wine$quality_target)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.5347  1.0000  1.0000
```

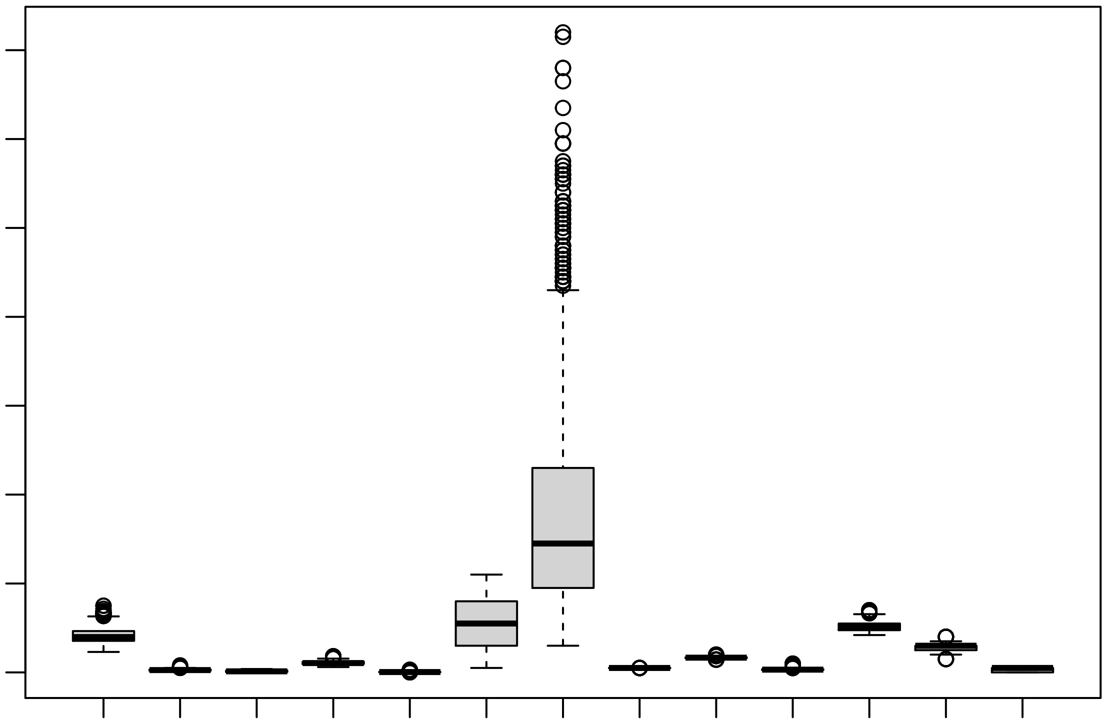
```
# Check for missing values in data set
wine %>% na.omit() %>% count() # there are no missing values
```

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

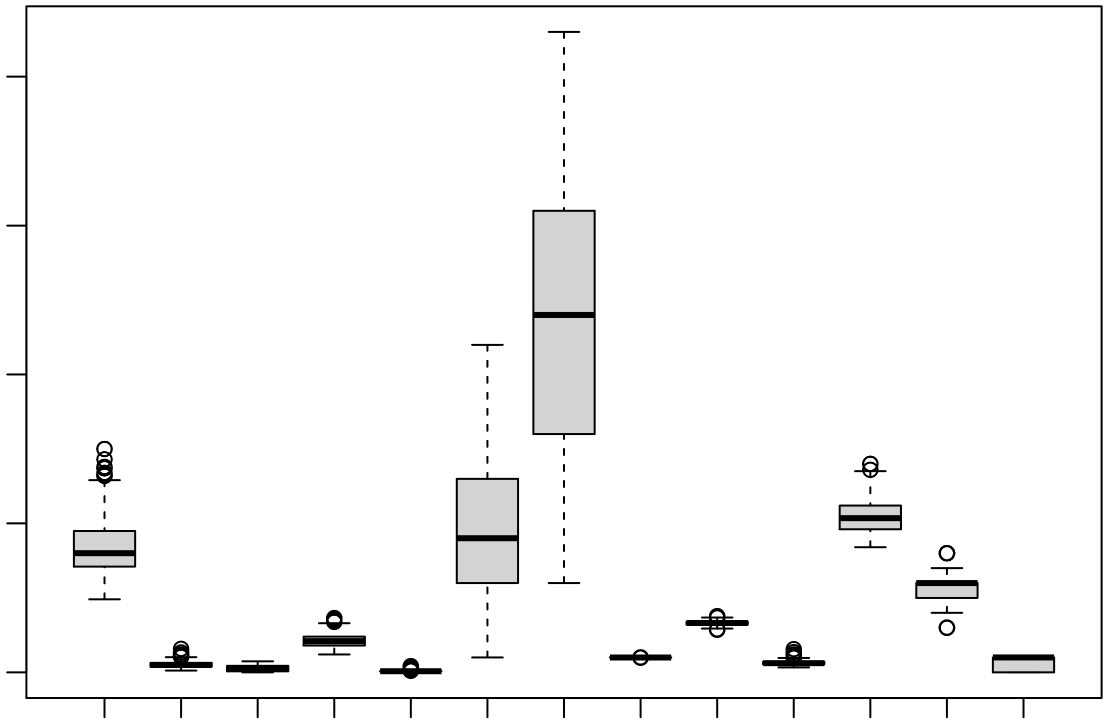
```
# Removing outliers for residual sugar:
Q <- quantile(wine$residual.sugar, probs=c(.25, .75), na.rm = FALSE)
iqr_rs <- IQR(wine$residual.sugar)
up_rs <-  Q[2]+1.5*iqr_rs # Upper Range
low_rs <- Q[1]-1.5*iqr_rs # Lower Range
eliminated_rs <- subset(wine, wine$residual.sugar > (Q[1] - 1.5*iqr_rs) &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



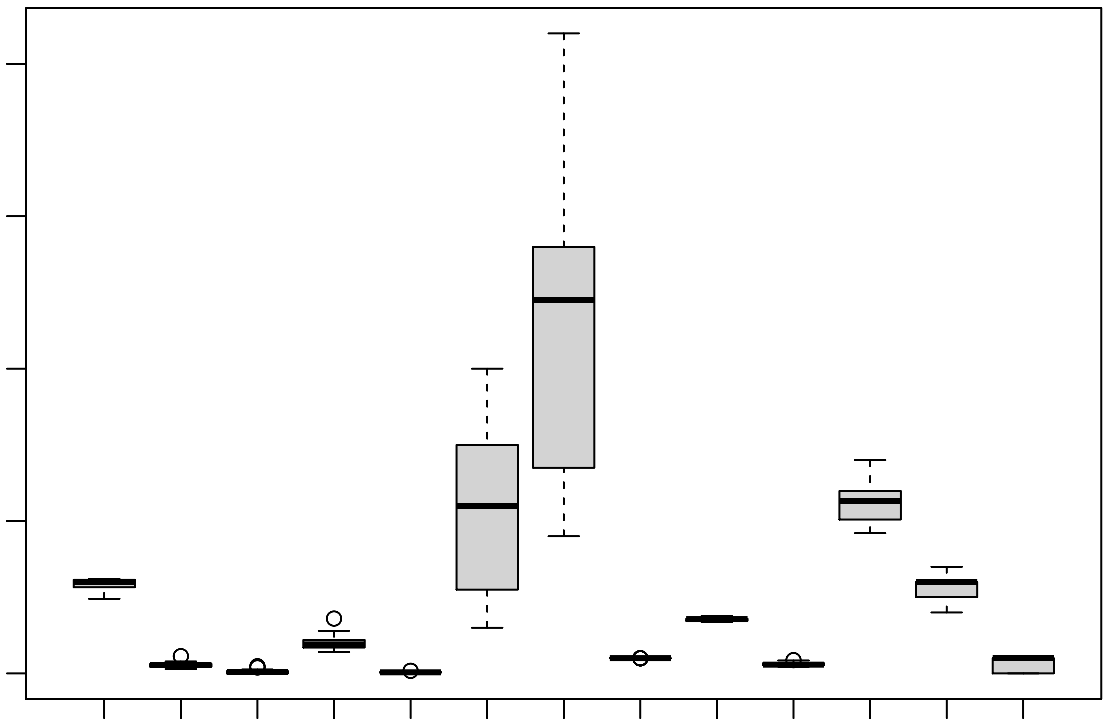
```
#Removing outliers for free.sulfur.dioxide:
Q2 <- quantile(wine$free.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_fs <- IQR(eliminated_rs$free.sulfur.dioxide)
up_fs <-  Q2[2]+1.5*iqr_fs # Upper Range
low_fs <- Q2[1]-1.5*iqr_fs # Lower Range
eliminated_fs <- subset(eliminated_rs, eliminated_rs$free.sulfur.dioxide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



```
#Removing outliers for total.sulfur.dioxide:
Q3 <- quantile(wine$total.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_ts <- IQR(eliminated_fs$total.sulfur.dioxide)
up_ts <-  Q3[2]+1.5*iqr_ts # Upper Range
low_ts <- Q3[1]-1.5*iqr_ts # Lower Range
eliminated_ts <- subset(eliminated_fs, eliminated_fs$total.sulfur.dioxide > (Q[1] - 1.5*iqr_ts) & elimi
boxplot(eliminated_ts)
```



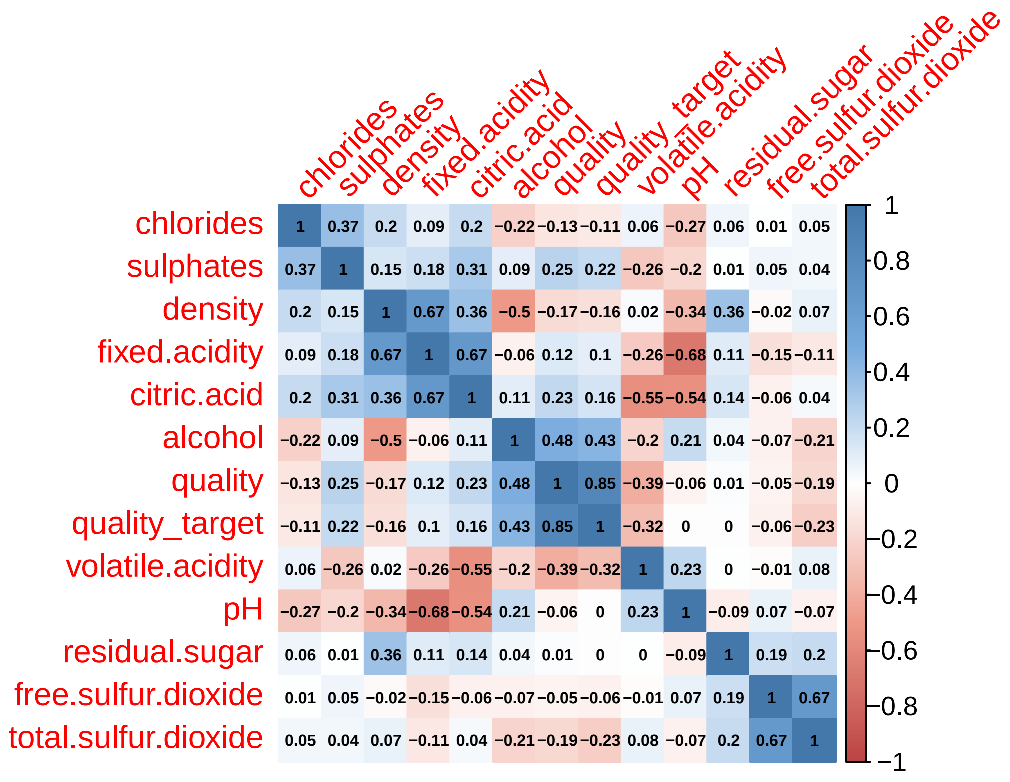
```
#Removing outliers for fixed.acidity:
Q4 <- quantile(wine$fixed.acidity, probs=c(.25, .75), na.rm = FALSE)
iqr_fa <- IQR(eliminated_ts$fixed.acidity)
up_fa <-  Q[2]+1.5*iqr_fa # Upper Range
low_fa <- Q[1]-1.5*iqr_fa # Lower Range
eliminated_fa <- subset(eliminated_ts, eliminated_ts$fixed.acidity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



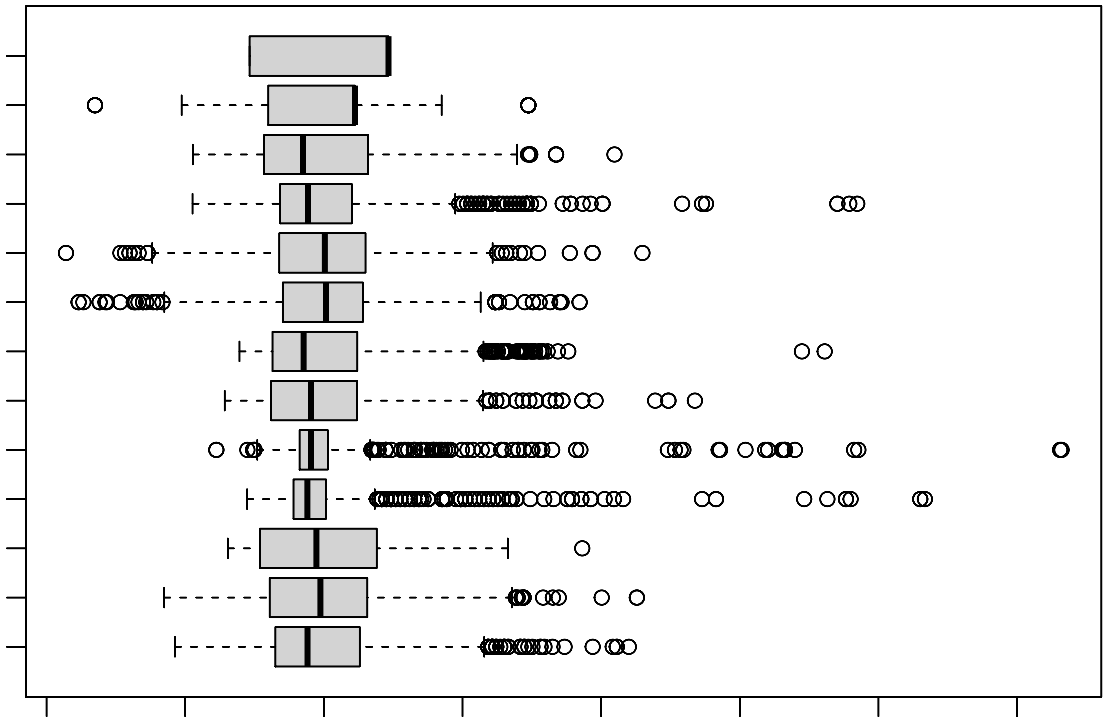
```
new_wine_data <- eliminated_fa

# Removing outliers reduced dimension of data set from 1599 observations to 48
# team opted not to use new_wine_data and keep outlier data
dim(new_wine_data)
```

```
## [1] 48 13
```

```
# Correlation Matrix
cor <- cor(wine)

# Colors for Correlation Matrix
colors <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))

corrplot(cor, order="hclust", method = "color", addCoef.col = "black"
         , tl.srt = 45, number.cex = 0.47, col=colors(200))
```



```r
# Cutoff Correlation features
cutoffCorr <- findCorrelation(cor, cutoff = .8)
cutoffCorrFeatures <- wine[, -cutoffCorr]

# Train and Test split
wine_split <- createDataPartition(wine$quality, p = .8, list = FALSE)
wine_train <- wine[ wine_split,]
wine_test  <- wine[-wine_split,]

# Transform Train Data
train_trans <- preProcess(wine_train, method = c("center", "scale"))
train_transformed <- predict(train_trans, wine_train)

# Transform Test Data
test_trans <- preProcess(wine_test, method = c("center", "scale"))
test_transformed <- predict(test_trans, wine_test)

# Boxplot of transformed train data
boxplot(train_transformed, horizontal = TRUE, las = 2, cex.axis = .65, cex.lab = 7)
```



Logistic Regression Model

```
# 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 = wine_train)

summary(lmodel1)
```

```
##
## Call:
## lm(formula = quality_target ~ volatile.acidity + sulphates +
##     alcohol, data = wine_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.49365 -0.35478 -0.02426  0.38227  1.03461
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.23371    0.14306  -8.624  < 2e-16 ***
## volatile.acidity -0.59164    0.07143  -8.283 3.00e-16 ***
## sulphates         0.34458    0.07124   4.837 1.48e-06 ***
## alcohol           0.17791    0.01154  15.424  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4292 on 1277 degrees of freedom
## Multiple R-squared:  0.262,  Adjusted R-squared:  0.2603
## F-statistic: 151.1 on 3 and 1277 DF,  p-value: < 2.2e-16
```

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

summary(lmodel2)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + sulphates + alcohol,
##     data = wine_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7039 -0.3831 -0.0719  0.4776  2.1995
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.62659    0.21803  12.047  < 2e-16 ***
## volatile.acidity -1.16761    0.10886 -10.726  < 2e-16 ***
## sulphates         0.60363    0.10858   5.559 3.29e-08 ***
## alcohol           0.31030    0.01758  17.651  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6541 on 1277 degrees of freedom
## Multiple R-squared:  0.3328, Adjusted R-squared:  0.3313
## F-statistic: 212.4 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()
```

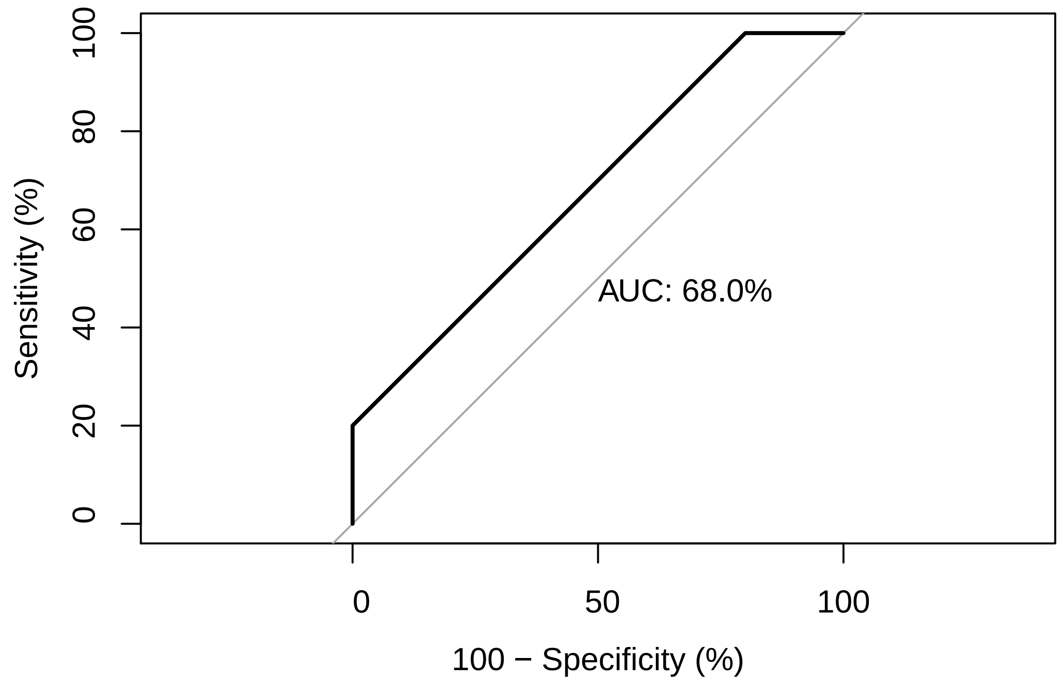
```
##       fit             lwr             upr
##  Min.   :4.405   Min.   :3.103   Min.   :5.708
##  1st Qu.:5.304   1st Qu.:4.019   1st Qu.:6.588
##  Median :5.628   Median :4.343   Median :6.914
##  Mean   :5.645   Mean   :4.360   Mean   :6.930
##  3rd Qu.:5.957   3rd Qu.:4.672   3rd Qu.:7.241
##  Max.   :7.051   Max.   :5.761   Max.   :8.340
```

```
# 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)
```

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

confusionMatrix(table1)
```

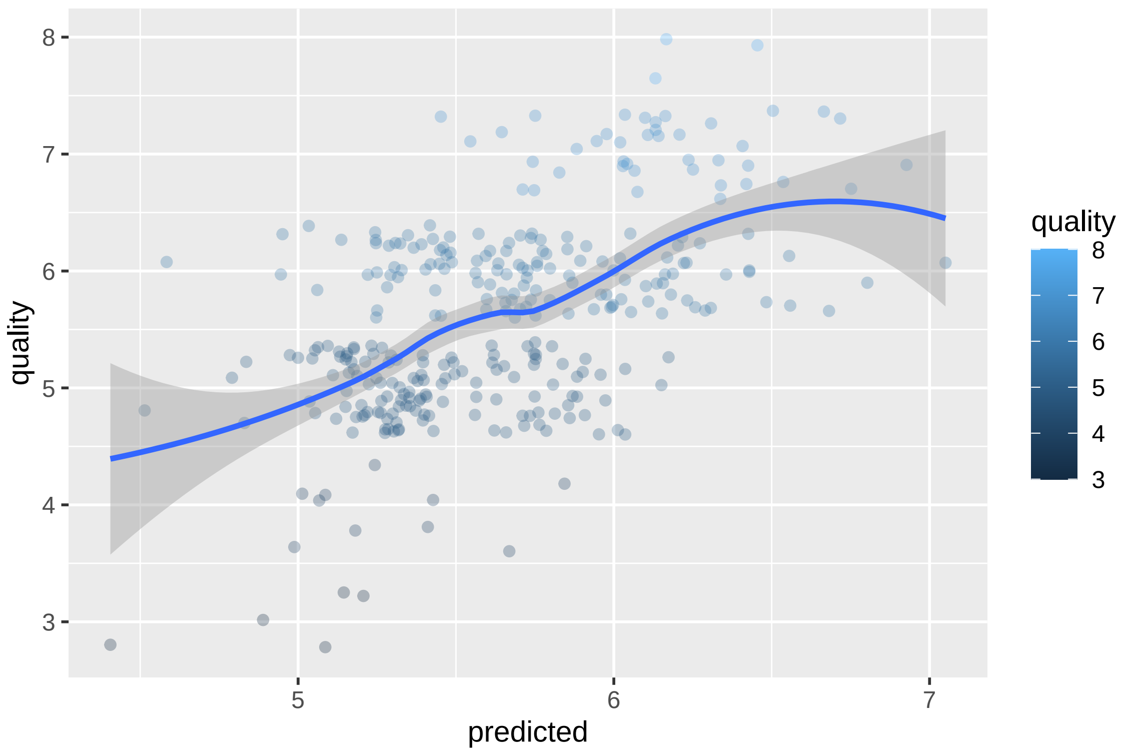
```
## Confusion Matrix and Statistics
##
##
##      5  6  4  7  8  3
##   5 87 46  0  0  0  0
##   6 39 83  0  5  0  0
##   4  8  2  0  0  0  0
##   7  1 33  0  6  0  0
##   8  0  3  0  0  0  0
##   3  4  0  1  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5535
##                  95% CI : (0.497, 0.6089)
##     No Information Rate : 0.5252
##     P-Value [Acc > NIR] : 0.1699
##
##                   Kappa : 0.2595
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 5 Class: 6 Class: 4 Class: 7 Class: 8 Class: 3
## Sensitivity            0.6259   0.4970 0.000000  0.54545       NA       NA
## Specificity            0.7430   0.7086 0.968454  0.88925 0.990566  0.98428
## Pos Pred Value         0.6541   0.6535 0.000000  0.15000       NA       NA
## Neg Pred Value         0.7189   0.5602 0.996753  0.98201       NA       NA
## Prevalence             0.4371   0.5252 0.003145  0.03459 0.000000  0.00000
## Detection Rate         0.2736   0.2610 0.000000  0.01887 0.000000  0.00000
## Detection Prevalence   0.4182   0.3994 0.031447  0.12579 0.009434  0.01572
## Balanced Accuracy      0.6845   0.6028 0.484227  0.71735       NA       NA
```

```r
# ROC plot
df$predicted_int = round(as.numeric(as.character(df$predicted)), digits = 0)

roc(df$quality, df$predicted_int, plot = TRUE, legacy.axes = TRUE, percent = TRUE, print.auc = TRUE)
```



```
##
## Call:
## roc.default(response = df$quality, predictor = df$predicted_int,     percent = TRUE, plot = TRUE, leg
##
## Data: df$predicted_int in 5 controls (df$quality 3) < 10 cases (df$quality 4).
## Area under the curve: 68%
```

```
#multiclass.roc(df$quality, df$predicted_int, plot = TRUE, legacy.axes = TRUE, percent = TRUE, print.au

# modelName1 <- 'Logistic Regression'
# roc1 <- multiclass.roc(df$quality, df$predicted_int)
# auc1 <- round(auc(df$quality, df$predicted_int), 4)
#
# ggroc(roc1, colours = 'red', size = 1) +
#   ggtitle(paste0(modelName1, ' - ROC Curve ', '(AUC = ', auc1 , ')')) + theme_minimal()


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



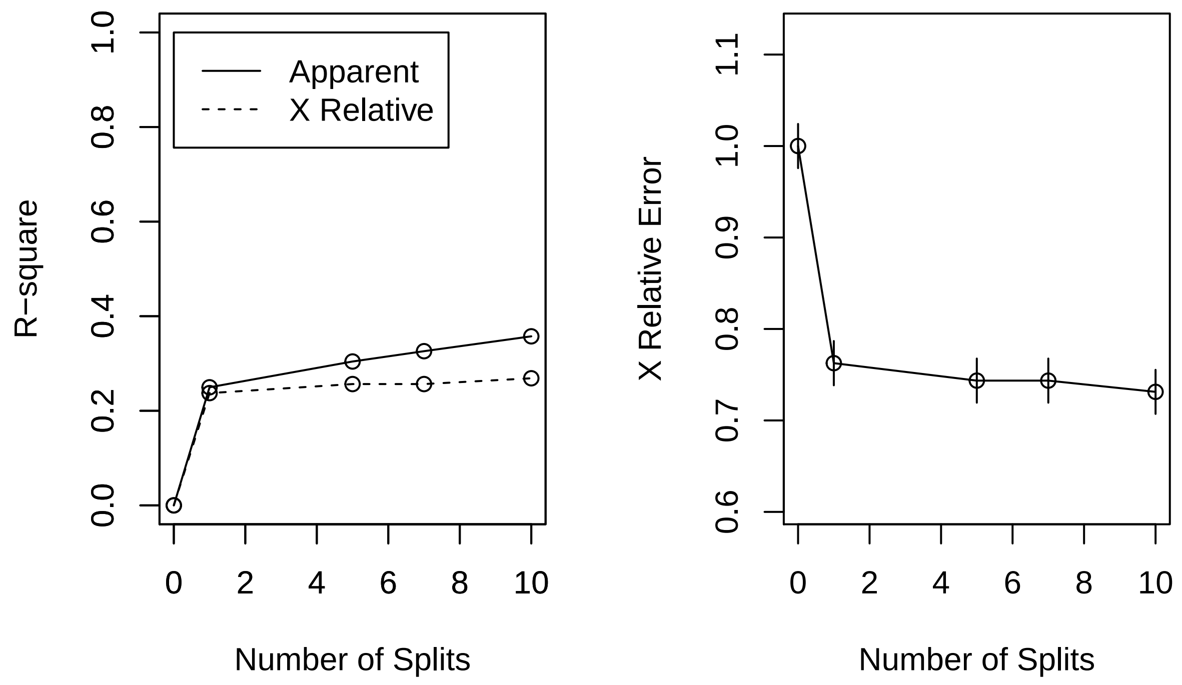
```
# The scatter plot supports the summary of the predicted interval, in the ranges of the fit,
# lower, and upper ranges. The R-squared value of 0.3283 of the model, indicates that this
# information can be predicted 33% of the time, with the data available, for the variance
# of the information.
```

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)
```

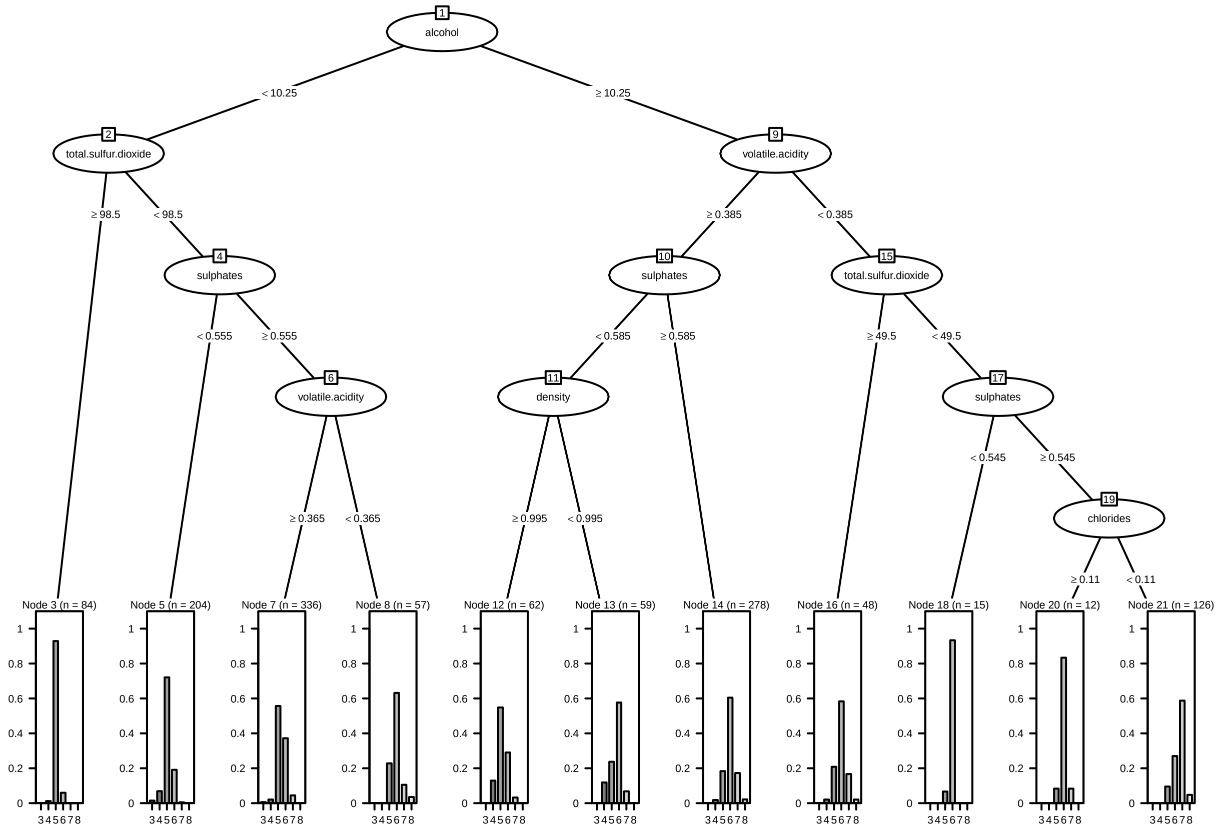
```
##
## Variables actually used in tree construction:
## [1] alcohol              chlorides            density
## [4] sulphates            total.sulfur.dioxide volatile.acidity
##
## Root node error: 733/1281 = 0.57221
##
## n= 1281
##
##         CP nsplit rel error  xerror     xstd
## 1 0.249659      0   1.00000 1.00000 0.024158
## 2 0.011596      1   0.75034 0.76262 0.024216
## 3 0.010914      5   0.69577 0.74352 0.024141
## 4 0.010459      7   0.67394 0.74352 0.024141
## 5 0.010000     10   0.64256 0.73124 0.024087
```



```
# 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 = 84, err = 7.1%)
## |   |   [4] total.sulfur.dioxide < 98.5
## |   |   |   [5] sulphates < 0.555: 5 (n = 204, err = 27.9%)
## |   |   |   [6] sulphates >= 0.555
## |   |   |   |   [7] volatile.acidity >= 0.365: 5 (n = 336, err = 44.3%)
## |   |   |   |   [8] volatile.acidity < 0.365: 6 (n = 57, err = 36.8%)
## |   [9] alcohol >= 10.25
## |   |   [10] volatile.acidity >= 0.385
## |   |   |   [11] sulphates < 0.585
## |   |   |   |   [12] density >= 0.99548: 5 (n = 62, err = 45.2%)
## |   |   |   |   [13] density < 0.99548: 6 (n = 59, err = 42.4%)
## |   |   |   [14] sulphates >= 0.585: 6 (n = 278, err = 39.6%)
## |   |   [15] volatile.acidity < 0.385
## |   |   |   [16] total.sulfur.dioxide >= 49.5: 6 (n = 48, err = 41.7%)
## |   |   |   [17] total.sulfur.dioxide < 49.5
## |   |   |   |   [18] sulphates < 0.545: 6 (n = 15, err = 6.7%)
## |   |   |   |   [19] sulphates >= 0.545
## |   |   |   |   |   [20] chlorides >= 0.1105: 6 (n = 12, err = 16.7%)
## |   |   |   |   |   [21] chlorides < 0.1105: 7 (n = 126, err = 41.3%)
##
## Number of inner nodes:    10
## Number of terminal nodes: 11
```

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



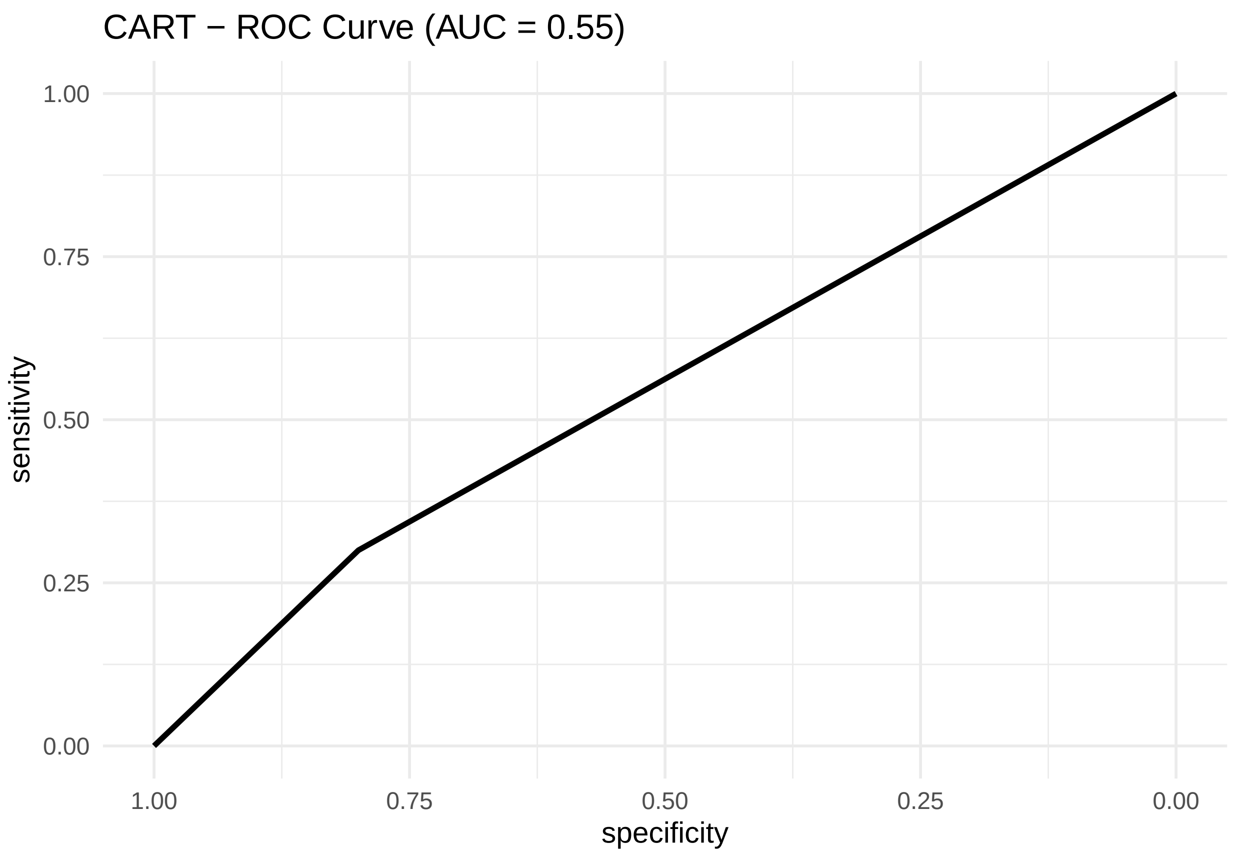
```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(rpartTree2, newdata = wine_test)) -> df2

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

```
##    3    4    5    6    7    8
##   0   0 167 115  36   0
```

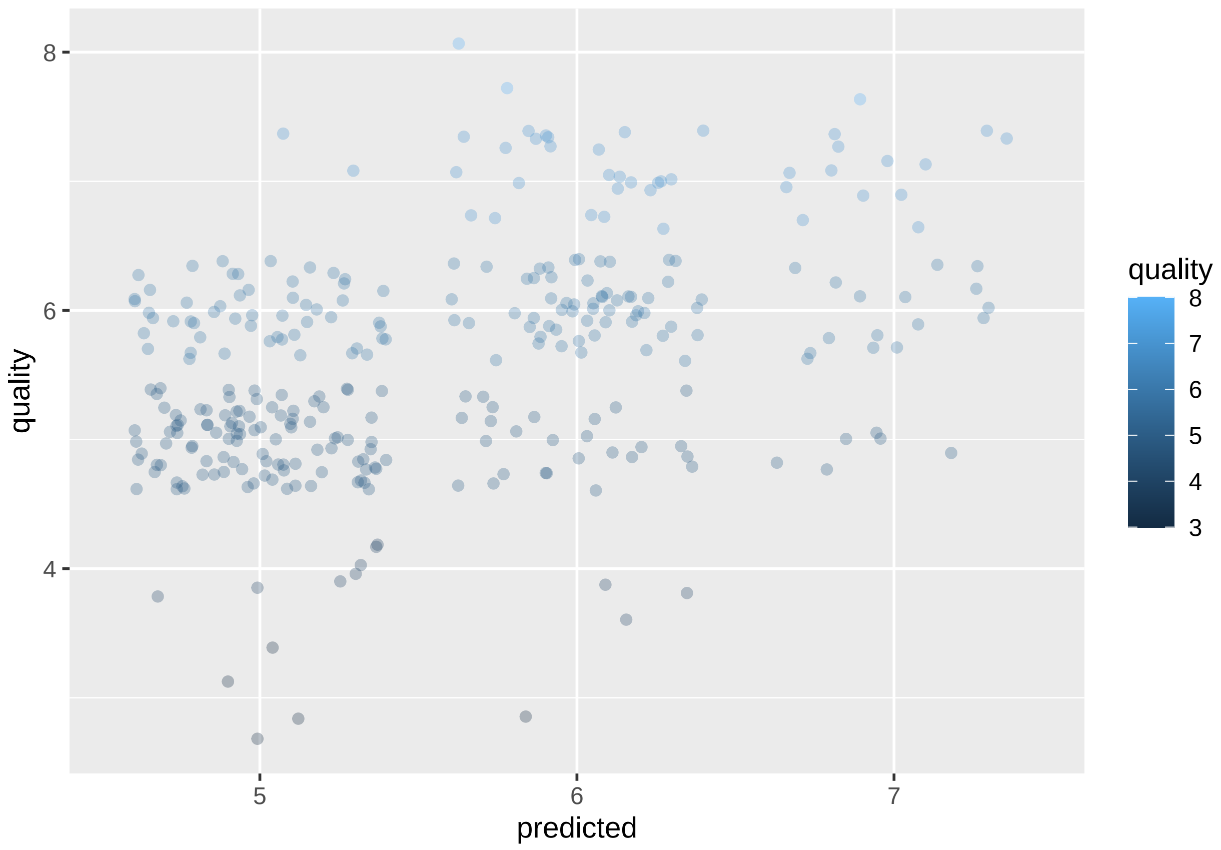
```
# Confusion Matrix
confusionMatrix(table(df2$quality, df2$predicted))
```

```
## Confusion Matrix and Statistics
##
##
##       3   4   5   6   7   8
##   3   0   0   4   1   0   0
##   4   0   0   7   3   0   0
##   5   0   0 101  26   6   0
##   6   0   0  53  58  16   0
##   7   0   0   2  25  13   0
##   8   0   0   0   2   1   0
##
## Overall Statistics
##
##                Accuracy : 0.5409
##                  95% CI : (0.4844, 0.5966)
##     No Information Rate : 0.5252
##     P-Value [Acc > NIR] : 0.3069
##
##                   Kappa : 0.2615
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity                NA       NA   0.6048   0.5043  0.36111       NA
## Specificity           0.98428  0.96855   0.7881   0.6601  0.90426 0.990566
## Pos Pred Value             NA       NA   0.7594   0.4567  0.32500       NA
## Neg Pred Value             NA       NA   0.6432   0.7016  0.91727       NA
## Prevalence            0.00000  0.00000   0.5252   0.3616  0.11321 0.000000
## Detection Rate        0.00000  0.00000   0.3176   0.1824  0.04088 0.000000
## Detection Prevalence  0.01572  0.03145   0.4182   0.3994  0.12579 0.009434
## Balanced Accuracy          NA       NA   0.6964   0.5822  0.63268       NA
```

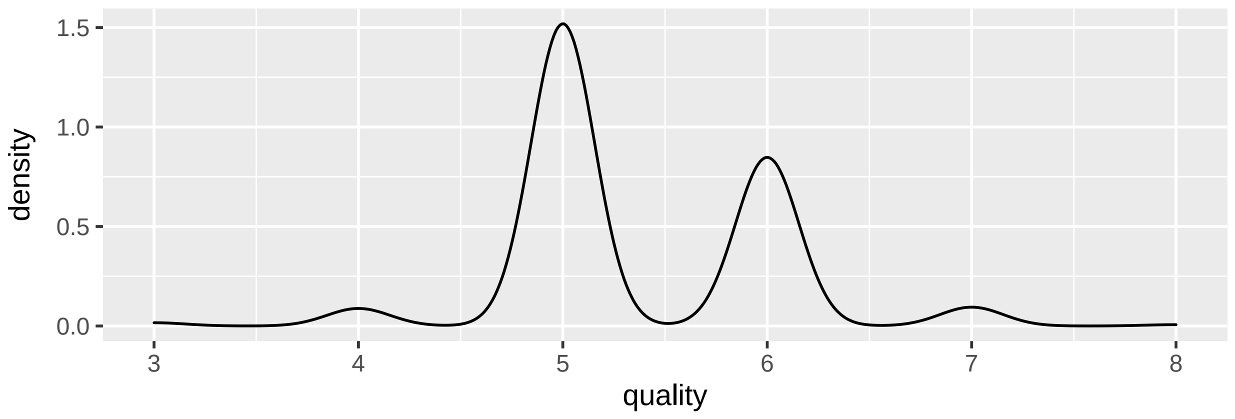
```
# ROC plot
df2$predicted_int = round(as.numeric(as.character(df2$predicted)), digits = 0)

#roc(df2$quality, df2$predicted_int, plot = TRUE, legacy.axes = TRUE, percent = TRUE, print.auc = TRUE)
#
modelName2 <- 'CART'
roc2 <- roc(df2$quality, df2$predicted_int)
auc2 <- round(auc(df2$quality, df2$predicted_int), 4)

ggroc(roc2, colours = 'red', size = 1) +
  ggtitle(paste0(modelName2, ' - ROC Curve ', '(AUC = ', auc2 , ')')) + theme_minimal()
```

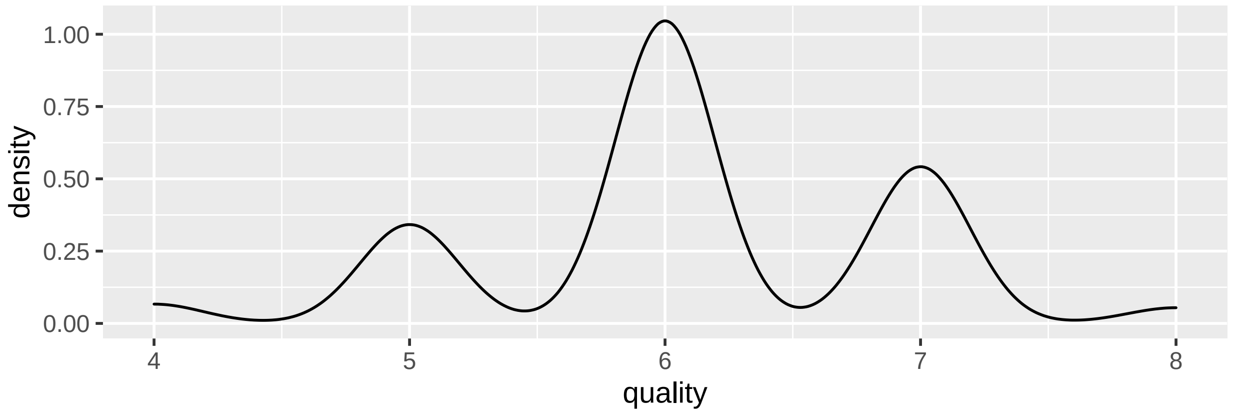


```r
# Scatter plot of predicted
ggplot(df2, aes(x = predicted, y = quality, colour = quality ))+
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```



```r
# 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)
```

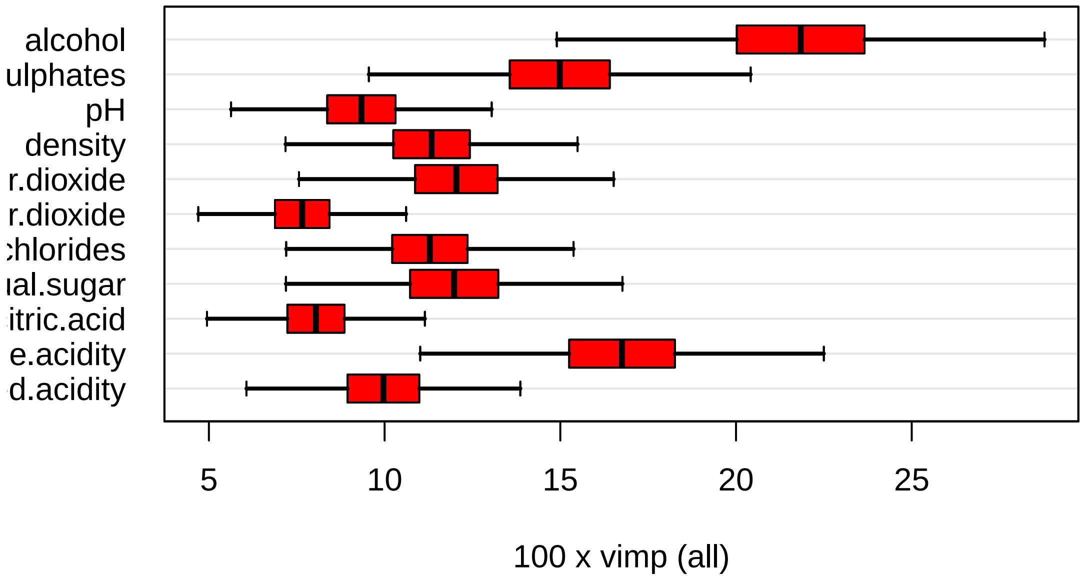
```
##                          Sample size: 1281
##            Frequency of class labels: 5, 43, 548, 511, 159, 15
##                      Number of trees: 500
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 252.772
## No. of variables tried at each split: 4
##              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: 0.07037366
##         (OOB) Normalized Brier score: 0.50669034
##                            (OOB) AUC: 0.78033697
##    (OOB) Requested performance error: 0.30523029, 1, 1, 0.20620438, 0.27201566, 0.49056604, 0.866666
##
```

```
## Confusion matrix:
##
##           predicted
##   observed 3 4   5   6  7 8 class.error
##          3 0 0   5   0  0 0      1.0000
##          4 0 0  29  13  1 0      1.0000
##          5 0 0 436 106  6 0      0.2044
##          6 0 3 113 370 24 1      0.2759
##          7 0 0   9  70 80 0      0.4969
##          8 0 0   0   7  6 2      0.8667
##
##      (OOB) Misclassification rate: 0.3067916
```

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

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

```
##                            lower      mean     upper       pvalue signif
## fixed.acidity           6.999708  9.965180 12.930653 2.255101e-11   TRUE
## volatile.acidity       12.386745 16.756263 21.125781 2.822027e-14   TRUE
## citric.acid             5.687040  8.045860 10.404680 1.151672e-11   TRUE
## residual.sugar          8.336291 11.979618 15.622945 5.797002e-11   TRUE
## chlorides               8.177184 11.288323 14.399461 5.741485e-13   TRUE
## free.sulfur.dioxide     5.405034  7.655202  9.905369 1.297337e-11   TRUE
## total.sulfur.dioxide    8.634538 12.041189 15.447840 2.138497e-12   TRUE
## density                 8.174147 11.335881 14.497615 1.054119e-12   TRUE
## pH                      6.517177  9.338854 12.160530 4.382196e-11   TRUE
## sulphates              10.850121 14.984289 19.118457 6.064748e-13   TRUE
## alcohol                16.558596 21.839066 27.119536 2.614475e-16   TRUE
```

```
# 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: 252.772
##          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: 0.07169764
##               Normalized Brier score: 0.516223
##                                  AUC: 0.84812834
##          Requested performance error: 0.32389937, 1, 1, 0.2406015, 0.26771654, 0.475, 1
##
## Confusion matrix:
##
##           predicted
##   observed 3 4   5  6  7 8 class.error
##          3 0 0   4  1  0 0      1.0000
##          4 0 0   8  2  0 0      1.0000
##          5 0 0 101 32  0 0      0.2406
##          6 0 1  30 93  3 0      0.2677
##          7 0 0   0 19 21 0      0.4750
##          8 0 0   0  1  2 0      1.0000
##
##            Misclassification error: 0.3238994
```

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 = 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      0.6406665  0.3615466  0.4955678
##   2      0.6401252  0.3626094  0.4954780
##   3      0.6401545  0.3625458  0.4949155
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 2.
```

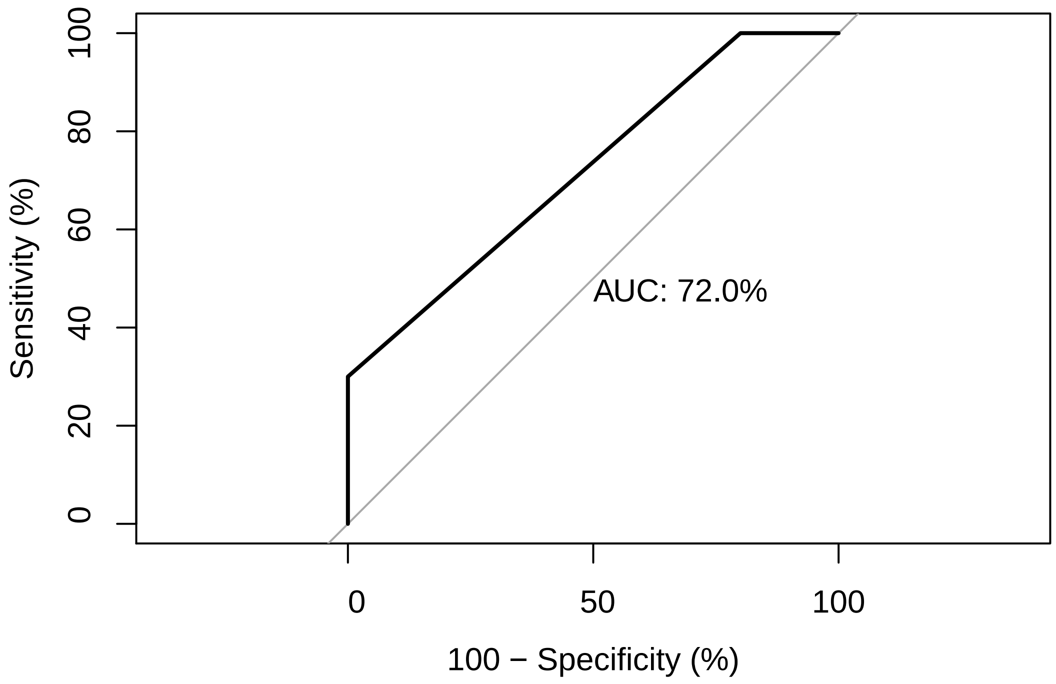
```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(pls_wine, newdata = wine_test)) -> df3

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.399   5.305   5.632   5.655   6.012   6.791
```

```
# 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)
```

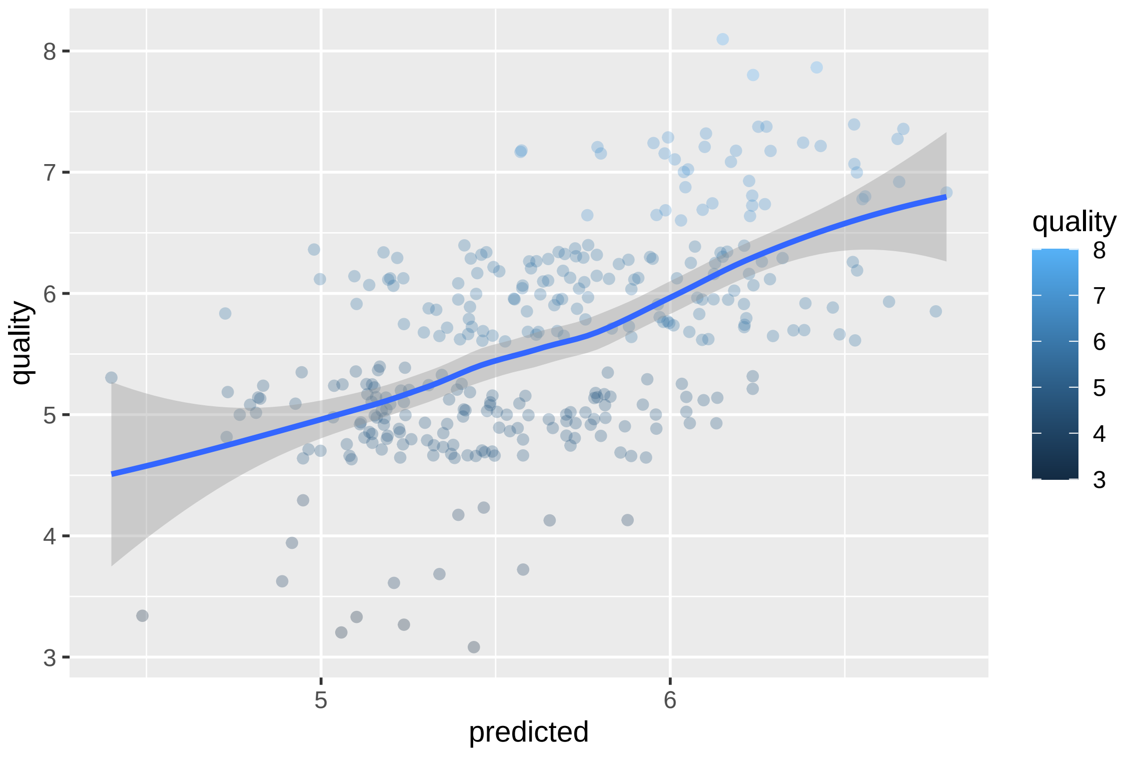
```
## Confusion Matrix and Statistics
##
##
##      5  6  4  7  8  3
##   5 84 48  1  0  0  0
##   6 35 87  0  5  0  0
##   4  7  3  0  0  0  0
##   7  0 31  0  9  0  0
##   8  0  3  0  0  0  0
##   3  4  0  1  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.566
##                  95% CI : (0.5096, 0.6212)
##     No Information Rate : 0.5409
##     P-Value [Acc > NIR] : 0.1995
##
##                   Kappa : 0.2854
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 5 Class: 6 Class: 4 Class: 7 Class: 8 Class: 3
## Sensitivity            0.6462   0.5058 0.000000  0.64286       NA       NA
## Specificity            0.7394   0.7260 0.968354  0.89803 0.990566  0.98428
## Pos Pred Value         0.6316   0.6850 0.000000  0.22500       NA       NA
## Neg Pred Value         0.7514   0.5550 0.993506  0.98201       NA       NA
## Prevalence             0.4088   0.5409 0.006289  0.04403 0.000000  0.00000
## Detection Rate         0.2642   0.2736 0.000000  0.02830 0.000000  0.00000
## Detection Prevalence   0.4182   0.3994 0.031447  0.12579 0.009434  0.01572
## Balanced Accuracy      0.6928   0.6159 0.484177  0.77044       NA       NA
```

```
# ROC plot
df3$predicted_int = round(as.numeric(as.character(df3$predicted)), digits = 0)

roc(df3$quality, df3$predicted_int, plot = TRUE, legacy.axes = TRUE, percent = TRUE, print.auc = TRUE)
```



```
##
## Call:
## roc.default(response = df3$quality, predictor = df3$predicted_int,     percent = TRUE, plot = TRUE, 
##
## Data: df3$predicted_int in 5 controls (df3$quality 3) < 10 cases (df3$quality 4).
## Area under the curve: 72%
```

```
#
# modelName3 <- 'Partial Least Squares'
# roc3 <- multiclass.roc(df3$quality, df3$predicted_int)
# auc3 <- round(auc(df3$quality, df3$predicted_int), 4)
#
# ggroc(roc3, colours = 'red', size = 1) +
#   ggtitle(paste0(modelName3, ' - ROC Curve ', '(AUC = ', auc3 , ')')) + theme_minimal()

# 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

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

mars_wine
```

```
## Selected 12 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 11 (additive model)
## GCV 0.4009379    RSS 495.3238    GRSq 0.3737305    RSq 0.3950735
```

```r
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##             data=wine_train)
##
##                                coefficients
## (Intercept)                       47.707645
## h(0.44-volatile.acidity)           1.285938
## h(volatile.acidity-0.44)          -0.806370
## h(chlorides-0.042)                20.942183
## h(chlorides-0.092)               -12.726397
## h(0.146-chlorides)                18.955464
## h(chlorides-0.226)               -10.995630
## h(total.sulfur.dioxide-9)         -0.321577
## h(144-total.sulfur.dioxide)       -0.318830
## h(total.sulfur.dioxide-144)        0.329662
## h(0.76-sulphates)              -2.012138
## h(12.5-alcohol)                -0.286173
##
## Selected 12 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 11 (additive model)
## GCV 0.4009379    RSS 495.3238    GRSq 0.3737305    RSq 0.3950735
```

```
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      0.7014813  0.2419795  0.5549556
##   1        3      0.6684464  0.3108674  0.5216037
##   1        4      0.6485241  0.3512544  0.5051027
##   1        5      0.6512351  0.3469701  0.5048202
##   1        6      0.6504907  0.3485716  0.5022178
##   1        7      0.6507935  0.3481485  0.5022971
##   1        8      0.6488589  0.3515400  0.5017645
##   1        9      0.6503107  0.3492233  0.5024412
##   1       10      0.6533237  0.3449166  0.5036034
##   1       11      0.6554705  0.3422215  0.5043991
##   1       12      0.6552105  0.3429115  0.5041358
##   1       13      0.6555639  0.3424326  0.5046787
##   1       14      0.6558033  0.3422174  0.5046129
##   1       15      0.6563798  0.3412858  0.5050354
##   1       16      0.6563798  0.3412858  0.5050354
##   1       17      0.6563798  0.3412858  0.5050354
##   1       18      0.6563798  0.3412858  0.5050354
##   1       19      0.6563798  0.3412858  0.5050354
##   1       20      0.6563798  0.3412858  0.5050354
##   1       21      0.6563798  0.3412858  0.5050354
##   1       22      0.6563798  0.3412858  0.5050354
##   1       23      0.6563798  0.3412858  0.5050354
##   1        24      0.6563798  0.3412858  0.5050354
##   1        25      0.6563798  0.3412858  0.5050354
##   1        26      0.6563798  0.3412858  0.5050354
##   1        27      0.6563798  0.3412858  0.5050354
##   1        28      0.6563798  0.3412858  0.5050354
##   1        29      0.6563798  0.3412858  0.5050354
##   1        30      0.6563798  0.3412858  0.5050354
##   1        31      0.6563798  0.3412858  0.5050354
##   1        32      0.6563798  0.3412858  0.5050354
##   1        33      0.6563798  0.3412858  0.5050354
##   1        34      0.6563798  0.3412858  0.5050354
##   1        35      0.6563798  0.3412858  0.5050354
##   1        36      0.6563798  0.3412858  0.5050354
##   1        37      0.6563798  0.3412858  0.5050354
##   1        38      0.6563798  0.3412858  0.5050354
##   2         2      0.7014414  0.2419923  0.5548719
##   2         3      0.6702386  0.3075375  0.5230336
##   2         4      0.6524546  0.3440511  0.5064611
##   2         5      0.6462661  0.3563502  0.5004076
##   2         6      0.6452593  0.3592572  0.4978439
##   2         7      0.6465667  0.3567901  0.4989545
##   2         8      0.6494945  0.3521291  0.5005637
##   2         9      0.6505045  0.3507097  0.5006272
##   2        10      0.6530531  0.3462806  0.5010743
##   2        11      0.6560394  0.3415267  0.5022654
##   2        12      0.6560221  0.3419308  0.5026598
##   2        13      0.6570923  0.3406133  0.5033740
##   2        14      0.6586577  0.3389605  0.5040339
##   2        15      0.6595409  0.3376501  0.5044626
##   2        16      0.6614910  0.3343875  0.5057373
##   2        17      0.6618613  0.3339272  0.5059127
##   2        18      0.6618503  0.3340194  0.5060600
##   2        19      0.6620816  0.3337136  0.5061720
##   2        20      0.6621197  0.3336746  0.5061551
##   2        21      0.6621197  0.3336746  0.5061551
##   2        22      0.6621197  0.3336746  0.5061551
##   2        23      0.6621197  0.3336746  0.5061551
##   2        24      0.6621197  0.3336746  0.5061551
##   2        25      0.6621197  0.3336746  0.5061551
##   2        26      0.6621197  0.3336746  0.5061551
##   2        27      0.6621197  0.3336746  0.5061551
##   2        28      0.6621197  0.3336746  0.5061551
##   2        29      0.6621197  0.3336746  0.5061551
##   2        30      0.6621197  0.3336746  0.5061551
##   2        31      0.6621197  0.3336746  0.5061551
##   2        32      0.6621197  0.3336746  0.5061551
##   2        33      0.6621197  0.3336746  0.5061551
##   2        34      0.6621197  0.3336746  0.5061551
##   2        35      0.6621197  0.3336746  0.5061551
##   2        36      0.6621197  0.3336746  0.5061551
##   2        37      0.6621197  0.3336746  0.5061551
##   2        38      0.6621197  0.3336746  0.5061551
##
## 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.
```

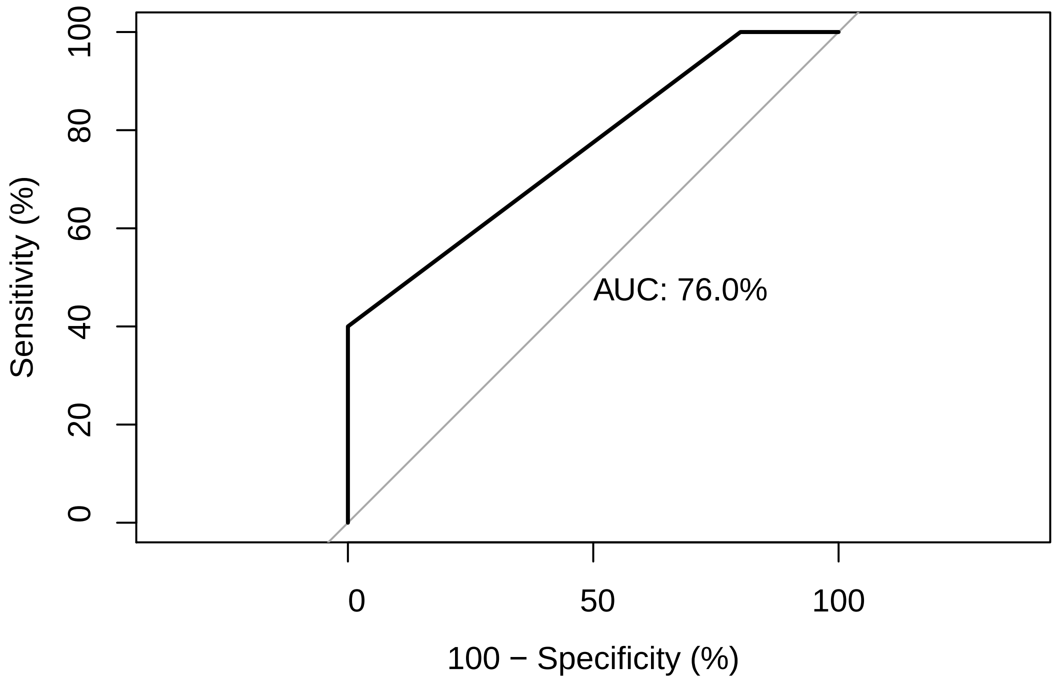
```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(marsModel_wine, newdata = wine_test)) -> df4

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

```
##        y
##  Min.   :3.974
##  1st Qu.:5.273
##  Median :5.596
##  Mean   :5.651
##  3rd Qu.:6.000
##  Max.   :6.804
```

```
# 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)
```

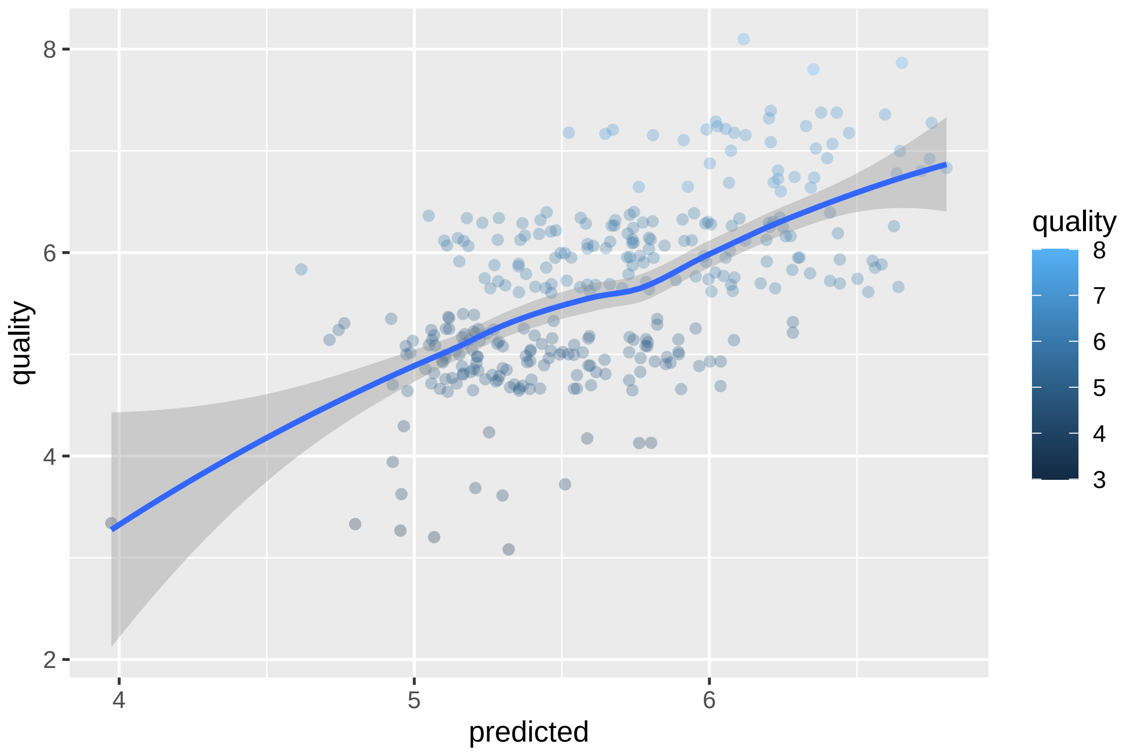
```
## Confusion Matrix and Statistics
##
##
##      5  6  4  7  8  3
##   5 88 45  0  0  0  0
##   6 36 84  0  7  0  0
##   4  6  4  0  0  0  0
##   7  0 33  0  7  0  0
##   8  0  2  0  1  0  0
##   3  4  0  1  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5629
##                  95% CI : (0.5064, 0.6182)
##     No Information Rate : 0.5283
##     P-Value [Acc > NIR] : 0.119
##
##                   Kappa : 0.2796
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 5 Class: 6 Class: 4 Class: 7 Class: 8 Class: 3
## Sensitivity            0.6567   0.5000 0.000000  0.46667       NA       NA
## Specificity            0.7554   0.7133 0.968454  0.89109 0.990566  0.98428
## Pos Pred Value         0.6617   0.6614 0.000000  0.17500       NA       NA
## Neg Pred Value         0.7514   0.5602 0.996753  0.97122       NA       NA
## Prevalence             0.4214   0.5283 0.003145  0.04717 0.000000  0.00000
## Detection Rate         0.2767   0.2642 0.000000  0.02201 0.000000  0.00000
## Detection Prevalence   0.4182   0.3994 0.031447  0.12579 0.009434  0.01572
## Balanced Accuracy      0.7061   0.6067 0.484227  0.67888       NA       NA
```

```
# ROC plot
df4$predicted_int = round(as.numeric(as.character(df4$predicted)), digits = 0)

roc(df4$quality, df4$predicted_int, plot = TRUE, legacy.axes = TRUE, percent = TRUE, print.auc = TRUE)
```



```
##
## Call:
## roc.default(response = df4$quality, predictor = df4$predicted_int,      percent = TRUE, plot = TRUE, 
##
## Data: df4$predicted_int in 5 controls (df4$quality 3) < 10 cases (df4$quality 4).
## Area under the curve: 76%
```

```
#
# modelName4 <- 'Mars Tuning'
# roc4 <- multiclass.roc(df4$quality, df4$predicted_int)
# auc4 <- round(auc(df4$quality, df4$predicted_int), 4)
#
# ggroc(roc4, colours = 'red', size = 1) +
#   ggtitle(paste0(modelName4, ' - ROC Curve ', '(AUC = ', auc4 , ')')) + theme_minimal()

# 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  0.7931263  0.2558058  0.4573932
##    2  0.7321340  0.2784365  0.5028836
##    3  0.7050894  0.2946256  0.5038454
##    4  0.6928700  0.3007317  0.5152042
##    5  0.6824992  0.3093344  0.5120538
##    6  0.6747969  0.3146795  0.5094303
##    7  0.6648928  0.3275608  0.5063056
##    8  0.6571997  0.3399876  0.5000693
##    9  0.6550702  0.3423538  0.5014153
##   10  0.6534921  0.3441642  0.5036200
##   11  0.6506971  0.3483598  0.5023332
##   12  0.6498040  0.3492250  0.5031206
##   13  0.6480800  0.3519414  0.5036606
##   14  0.6494254  0.3488478  0.5048960
##   15  0.6501741  0.3464884  0.5060485
##   16  0.6487798  0.3486218  0.5045870
##   17  0.6482825  0.3492570  0.5042702
##   18  0.6471091  0.3512579  0.5027609
##   19  0.6472611  0.3511440  0.5033137
##   20  0.6463476  0.3525807  0.5026752
##   21  0.6442849  0.3561775  0.5011751
##   22  0.6442341  0.3559528  0.5014678
##   23  0.6444608  0.3553065  0.5020588
##   24  0.6444540  0.3551886  0.5018186
##   25  0.6425414  0.3583346  0.5007683
##   26  0.6410799  0.3613750  0.4996561
##   27  0.6411065  0.3615645  0.5001382
##   28  0.6405786  0.3628237  0.4998135
##   29  0.6393510  0.3655226  0.4985531
##   30  0.6380821  0.3682214  0.4980229
##   31  0.6383581  0.3675952  0.4983101
##   32  0.6384969  0.3675095  0.4981278
##   33  0.6385071  0.3677068  0.4982906
##   34  0.6382805  0.3684733  0.4982302
##   35  0.6373637  0.3702965  0.4979635
##   36  0.6374273  0.3700186  0.4983200
##   37  0.6369241  0.3709937  0.4977353
##   38  0.6364424  0.3718126  0.4979826
##   39  0.6361429  0.3724760  0.4979734
##   40  0.6355148  0.3740180  0.4975481
##   41  0.6360932  0.3729431  0.4980664
##   42  0.6352480  0.3746541  0.4978058
##   43  0.6345464  0.3761934  0.4968382
##   44  0.6350254  0.3753575  0.4972071
##   45  0.6352044  0.3751202  0.4977586
##   46  0.6353060  0.3749478  0.4982237
##   47  0.6346586  0.3762757  0.4975144
##   48  0.6347369  0.3761214  0.4978120
##   49  0.6351326  0.3753620  0.4980585
##   50  0.6354252  0.3748959  0.4981426
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 43.
```

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

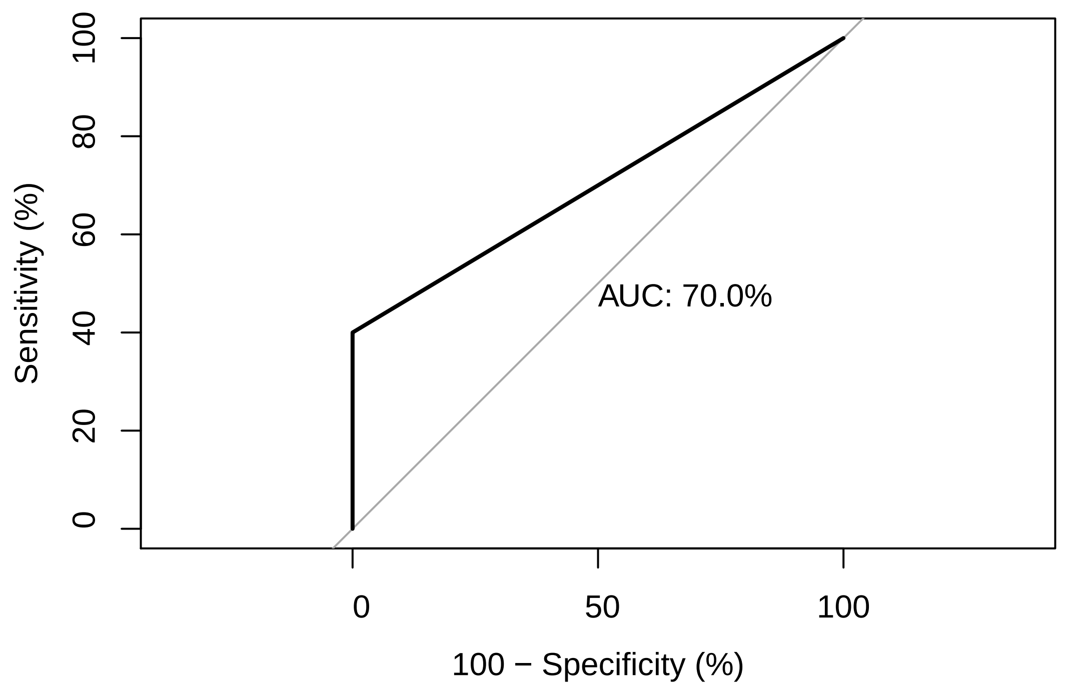
```r
# Summary of predicted interval
predict(knn_wine, newdata = wine_test, interval = "prediction") %>%
  summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.860   5.302   5.605   5.673   5.977   6.814
```

```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)
```

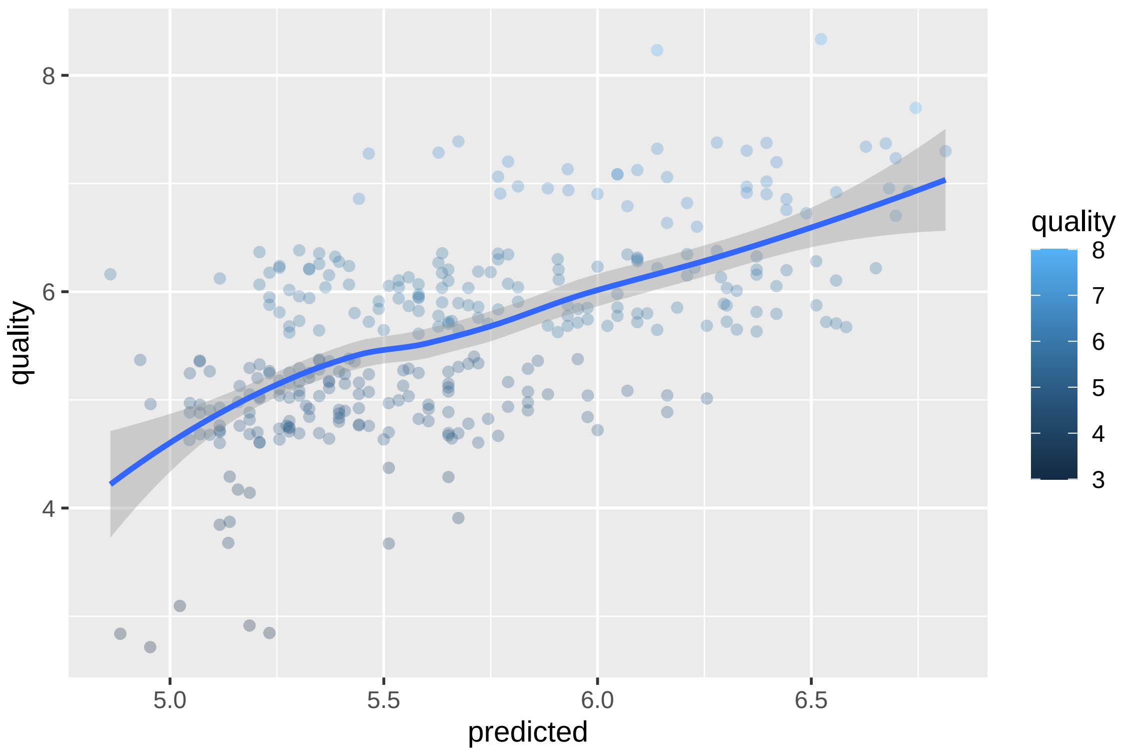
```
## Confusion Matrix and Statistics
##
##
##      5  6  4  7  8  3
##   5 87 46  0  0  0  0
##   6 32 88  0  7  0  0
##   4  6  4  0  0  0  0
##   7  2 30  0  8  0  0
##   8  0  1  0  2  0  0
##   3  5  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5755
##                  95% CI : (0.5191, 0.6304)
##     No Information Rate : 0.5314
##     P-Value [Acc > NIR] : 0.06436
##
##                   Kappa : 0.3011
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 5 Class: 6 Class: 4 Class: 7 Class: 8 Class: 3
## Sensitivity            0.6591   0.5207       NA  0.47059       NA       NA
## Specificity            0.7527   0.7383  0.96855  0.89369 0.990566  0.98428
## Pos Pred Value         0.6541   0.6929       NA  0.20000       NA       NA
## Neg Pred Value         0.7568   0.5759       NA  0.96763       NA       NA
## Prevalence             0.4151   0.5314  0.00000  0.05346 0.000000  0.00000
## Detection Rate         0.2736   0.2767  0.00000  0.02516 0.000000  0.00000
## Detection Prevalence   0.4182   0.3994  0.03145  0.12579 0.009434  0.01572
## Balanced Accuracy      0.7059   0.6295       NA  0.68214       NA       NA
```

```r
# ROC plot
df5$predicted_int = round(as.numeric(as.character(df5$predicted)), digits = 0)

roc(df5$quality, df5$predicted_int, plot = TRUE, legacy.axes = TRUE, percent = TRUE, print.auc = TRUE)
```



```
##
## Call:
## roc.default(response = df5$quality, predictor = df5$predicted_int,     percent = TRUE, plot = TRUE, 
##
## Data: df5$predicted_int in 5 controls (df5$quality 3) < 10 cases (df5$quality 4).
## Area under the curve: 70%
```

```
#
# modelName5 <- 'KNN'
# roc5 <- multiclass.roc(df5$quality, df5$predicted_int)
# auc5 <- round(auc(df5$quality, df5$predicted_int), 4)
#
# ggroc(roc5, colours = 'red', size = 1) +
#   ggtitle(paste0(modelName5, ' - ROC Curve ', '(AUC = ', auc5 , ')')) + theme_minimal()

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



SVM

```r
set.seed(4)


svmTune <- train(quality ~ volatile.acidity + sulphates + alcohol, data = rf_wine_train, # using the su
                 method = "svmRadial",
                 preProc = c("center", "scale"),
                 tuneLength= 5,
                 trControl = trainControl(method = "cv"))
svmTune
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1281 samples
##    3 predictor
##    6 classes: '3', '4', '5', '6', '7', '8'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1153, 1154, 1153, 1152, 1153, 1152, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.5910090  0.3127878
##   0.50  0.5925714  0.3185638
##   1.00  0.5933528  0.3215787
##   2.00  0.5870903  0.3128096
##   4.00  0.5901849  0.3205236
##
## Tuning parameter 'sigma' was held constant at a value of 0.6015525
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.6015525 and C = 1.
```

```
# 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()
```

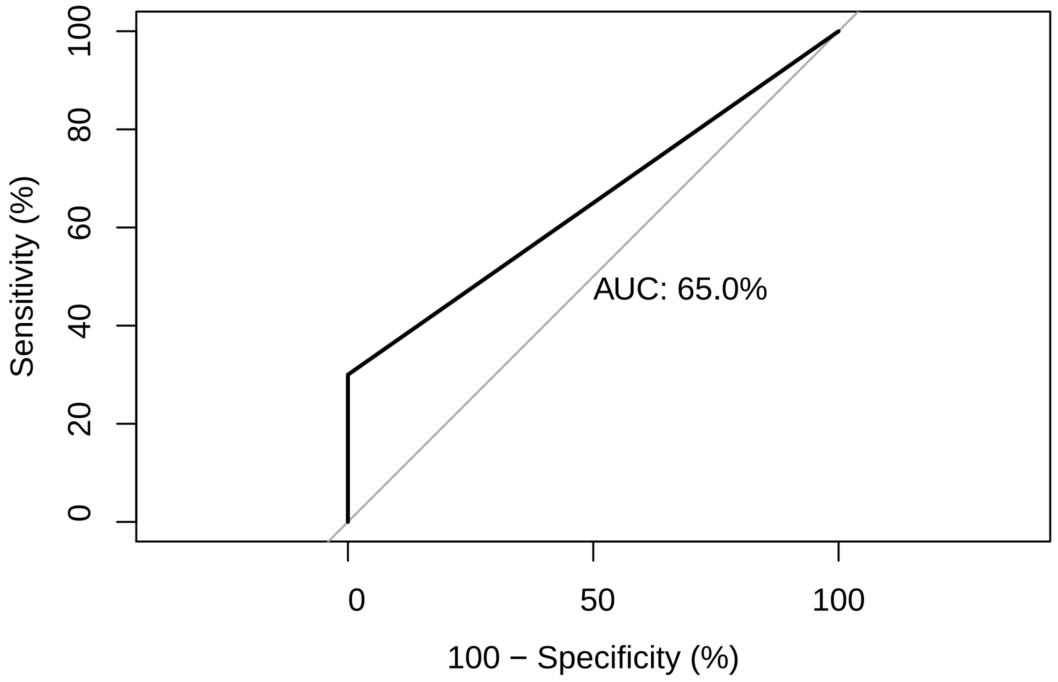
```
##   3   4   5   6   7   8
##   0   0 152 152  14   0
```

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

```
## Confusion Matrix and Statistics
##
##
##      3  4  5  6  7  8
##   3  0  0  5  0  0  0
##   4  0  0  7  3  0  0
##   5  0  0 96 37  0  0
##   6  0  0 43 80  4  0
##   7  0  0  1 30  9  0
##   8  0  0  0  2  1  0
##
## Overall Statistics
##
##                Accuracy : 0.5818
##                  95% CI : (0.5254, 0.6366)
##     No Information Rate : 0.478
##     P-Value [Acc > NIR] : 0.000131
##
##                   Kappa : 0.3072
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity                NA       NA   0.6316   0.5263  0.64286       NA
## Specificity           0.98428  0.96855   0.7771   0.7169  0.89803 0.990566
## Pos Pred Value             NA       NA   0.7218   0.6299  0.22500       NA
## Neg Pred Value             NA       NA   0.6973   0.6230  0.98201       NA
## Prevalence            0.00000  0.00000   0.4780   0.4780  0.04403 0.000000
## Detection Rate        0.00000  0.00000   0.3019   0.2516  0.02830 0.000000
## Detection Prevalence  0.01572  0.03145   0.4182   0.3994  0.12579 0.009434
## Balanced Accuracy          NA       NA   0.7043   0.6216  0.77044       NA
```

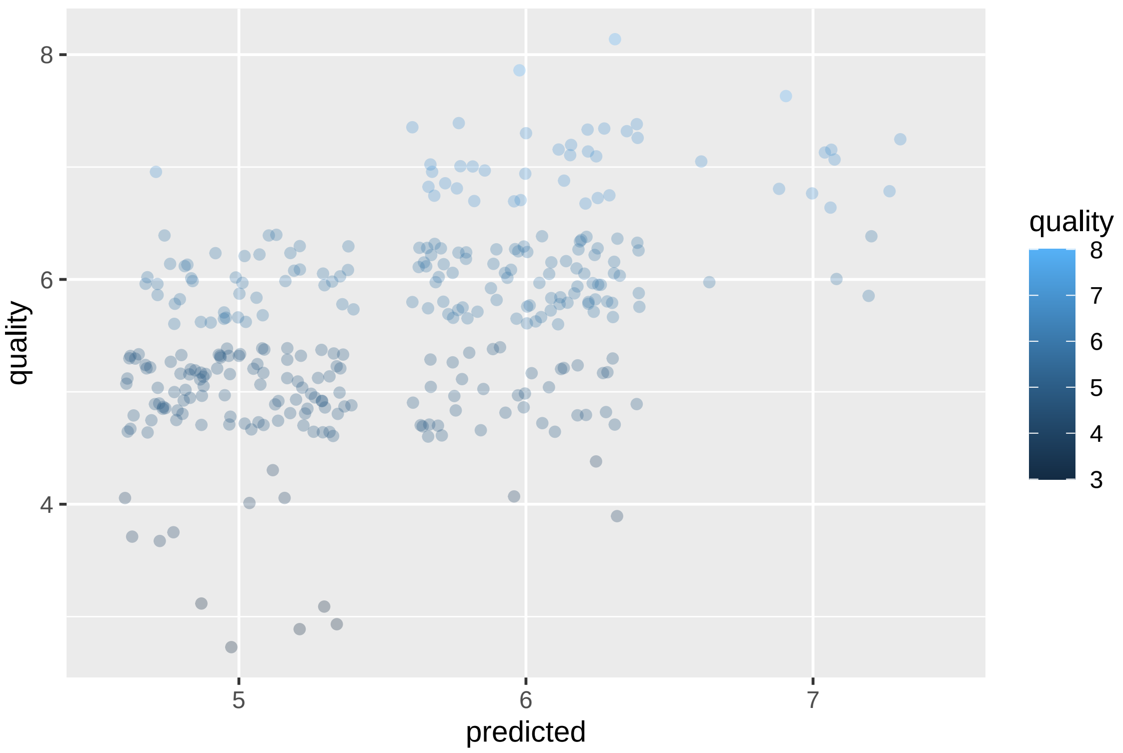
```
# ROC plot
df6$predicted_int = round(as.numeric(as.character(df6$predicted)), digits = 0)
```

```
roc(df6$quality, df6$predicted_int, plot = TRUE, legacy.axes = TRUE, percent = TRUE, print.auc = TRUE)
```



```
##
## Call:
## roc.default(response = df6$quality, predictor = df6$predicted_int,     percent = TRUE, plot = TRUE, 
##
## Data: df6$predicted_int in 5 controls (df6$quality 3) < 10 cases (df6$quality 4).
## Area under the curve: 65%
```

```
#
# modelName6 <- 'SVM'
# roc6 <- multiclass.roc(df6$quality, df6$predicted_int)
# auc6 <- round(auc(df6$quality, df6$predicted_int), 4)
#
# ggroc(roc6, colours = 'red', size = 1) +
#   ggtitle(paste0(modelName6, ' - ROC Curve ', '(AUC = ', auc6 , ')')) + theme_minimal()

# 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

```r
#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: 1154, 1154, 1152, 1152, 1153, 1152, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda  Accuracy   Kappa
##   0.0    0.0100  0.6010676  0.33503660
##   0.0    0.0575  0.5878469  0.30509871
##   0.0    0.1050  0.5864007  0.29798813
##   0.0    0.1525  0.5856007  0.29432135
##   0.0    0.2000  0.5832932  0.28882488
##   0.1    0.0100  0.6034296  0.34232307
##   0.1    0.0575  0.5918024  0.30996661
##   0.1    0.1050  0.5832260  0.28973398
##   0.1    0.1525  0.5809246  0.28441948
##   0.1    0.2000  0.5777625  0.27857370
##   0.2    0.0100  0.6018732  0.33995345
##   0.2    0.0575  0.5839765  0.29403575
##   0.2    0.1050  0.5824382  0.28718508
##   0.2    0.1525  0.5739230  0.27193416
##   0.2    0.2000  0.5739167  0.27137366
##   0.4    0.0100  0.6049557  0.34275408
##   0.4    0.0575  0.5778233  0.28107401
##   0.4    0.1050  0.5731295  0.27041928
##   0.4    0.1525  0.5676840  0.26012440
##   0.4    0.2000  0.5645157  0.25395588
##   0.6    0.0100  0.6002494  0.33458111
##   0.6    0.0575  0.5723418  0.27043675
##   0.6    0.1050  0.5661034  0.25782801
##   0.6    0.1525  0.5613780  0.24867794
##   0.6    0.2000  0.5566783  0.23925609
##   0.8    0.0100  0.6010799  0.33417339
##   0.8    0.0575  0.5723667  0.26954382
##   0.8    0.1050  0.5637524  0.25326580
##   0.8    0.1525  0.5558909  0.23814660
##   0.8    0.2000  0.5472657  0.22180362
##   1.0    0.0100  0.5987421  0.32925132
##   1.0    0.0575  0.5739351  0.27169914
##   1.0    0.1050  0.5605787  0.24682498
##   1.0    0.1525  0.5559031  0.23773006
##   1.0    0.2000  0.4646061  0.06934246
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.4 and lambda = 0.01.
```

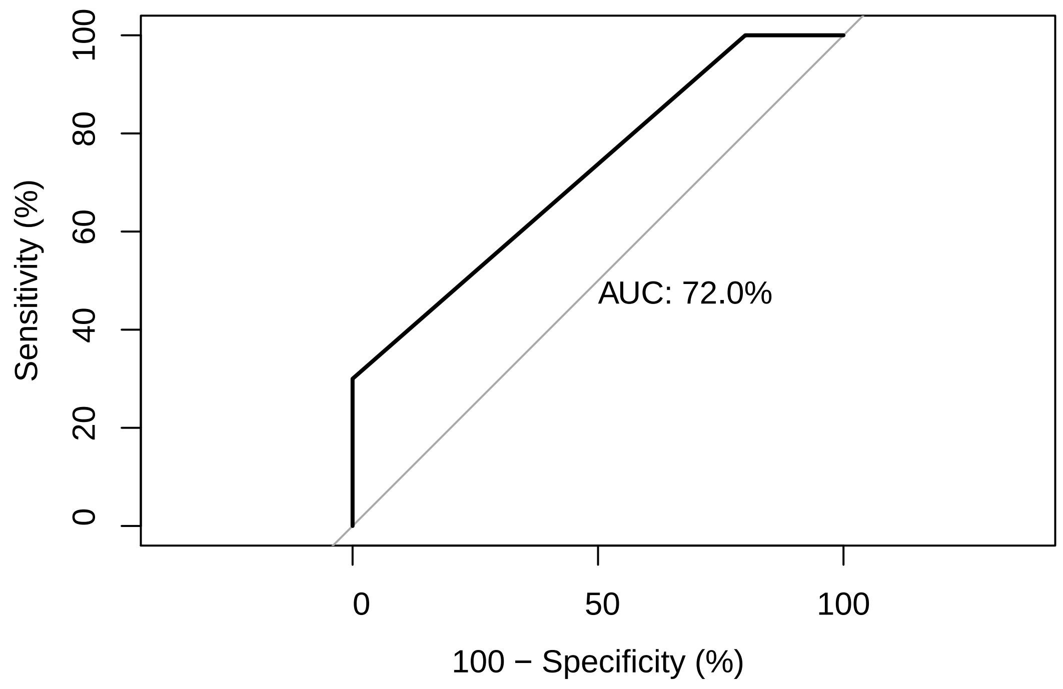
```r
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(glmnTune, newdata = wine_test)) -> df7

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

```
##   3   4   5   6   7   8
##   0   1 146 156  15   0
```

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

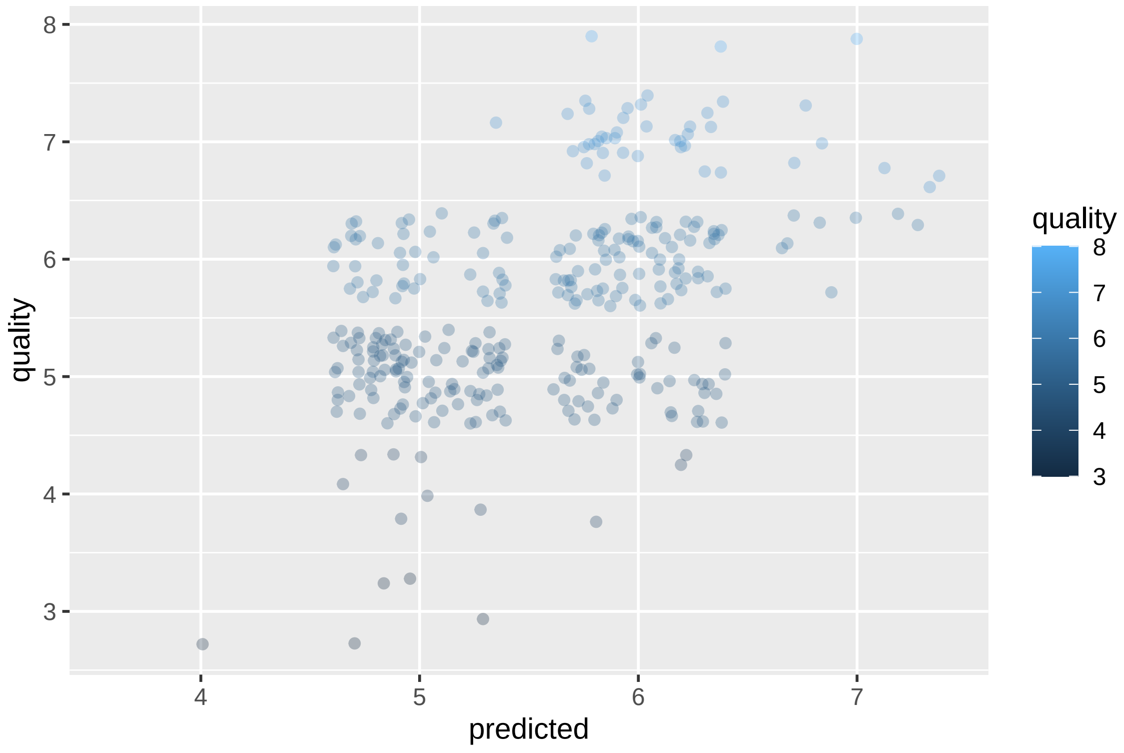
```
## Confusion Matrix and Statistics
##
##
##      3  4  5  6  7  8
##   3  0  1  4  0  0  0
##   4  0  0  7  3  0  0
##   5  0  0 91 42  0  0
##   6  0  0 43 76  8  0
##   7  0  0  1 33  6  0
##   8  0  0  0  2  1  0
##
## Overall Statistics
##
##                Accuracy : 0.544
##                  95% CI : (0.4875, 0.5997)
##     No Information Rate : 0.4906
##     P-Value [Acc > NIR] : 0.03207
##
##                   Kappa : 0.2476
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity                NA 0.000000   0.6233   0.4872  0.40000       NA
## Specificity           0.98428 0.968454   0.7558   0.6852  0.88779 0.990566
## Pos Pred Value             NA 0.000000   0.6842   0.5984  0.15000       NA
## Neg Pred Value             NA 0.996753   0.7027   0.5812  0.96763       NA
## Prevalence            0.00000 0.003145   0.4591   0.4906  0.04717 0.000000
## Detection Rate        0.00000 0.000000   0.2862   0.2390  0.01887 0.000000
## Detection Prevalence  0.01572 0.031447   0.4182   0.3994  0.12579 0.009434
## Balanced Accuracy          NA 0.484227   0.6896   0.5862  0.64389       NA
```

```r
# ROC plot
df7$predicted_int = round(as.numeric(as.character(df7$predicted)), digits = 0)

roc(df7$quality, df7$predicted_int, plot = TRUE, legacy.axes = TRUE, percent = TRUE, print.auc = TRUE)
```



```
##
## Call:
## roc.default(response = df7$quality, predictor = df7$predicted_int,      percent = TRUE, plot = TRUE, 
##
## Data: df7$predicted_int in 5 controls (df7$quality 3) < 10 cases (df7$quality 4).
## Area under the curve: 72%
```

```
#
# modelName7 <- 'Penalized Logistic Regression Tuning'
# roc7 <- roc(df7$quality, df7$predicted_int)
# auc7 <- round(auc(df7$quality, df7$predicted_int), 4)
#
# ggroc(roc7, colours = 'red', size = 1) +
#   ggtitle(paste0(modelName7, ' - ROC Curve ', '(AUC = ', auc7 , ')')) + theme_minimal()

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


quality
predicted
quality