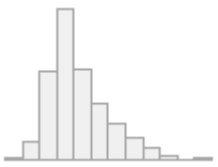
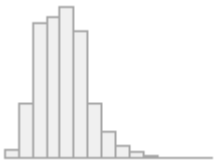
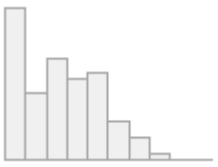
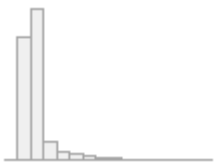
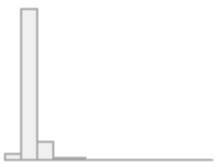
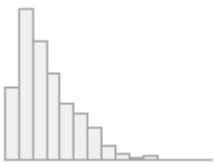
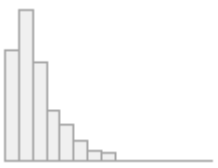
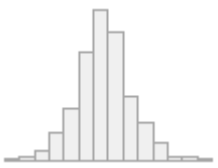
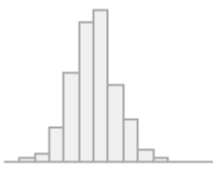
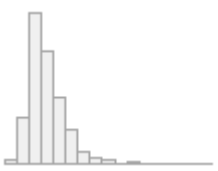
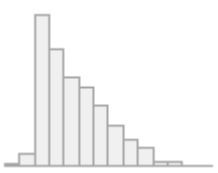
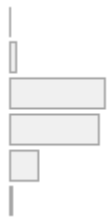
Data Frame Summary

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%)
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%)

Generated by summarytools 1.0.1 (R version 4.0.4)
2022-06-27

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)
library(pROC)
```

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

```
# 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")
    )
```

```
## temporary images written to 'G:\My Drive\Masters of Science in Applied Data Science (University of San Diego)\2022 - S5 - Summer Courses\ADS 503 Applied Predictive Modeling\Team Project\NA'
```

```
## Switching method to 'browser'
```

```
## Output file written: C:\Users\OscarGil\AppData\Local\Temp\RtmpApsrHA\file3dcc6c9238a8.html
```

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)
```

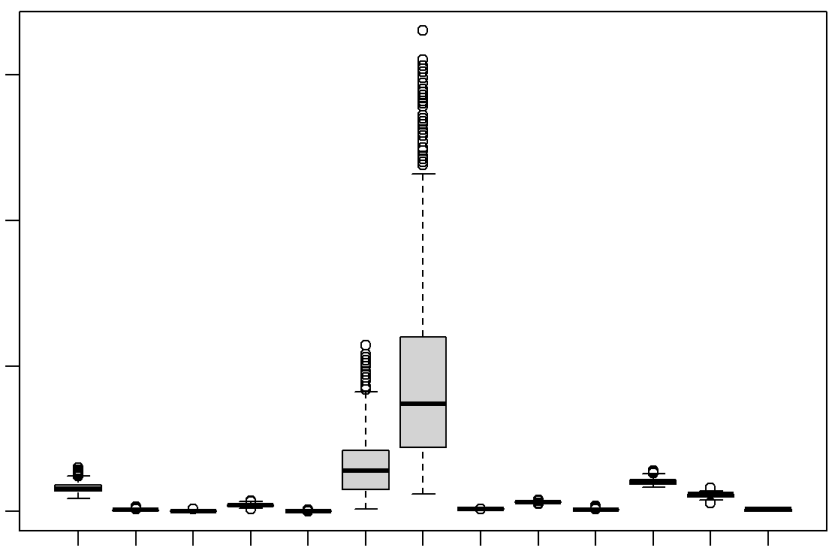
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.5347  1.0000  1.0000
```

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

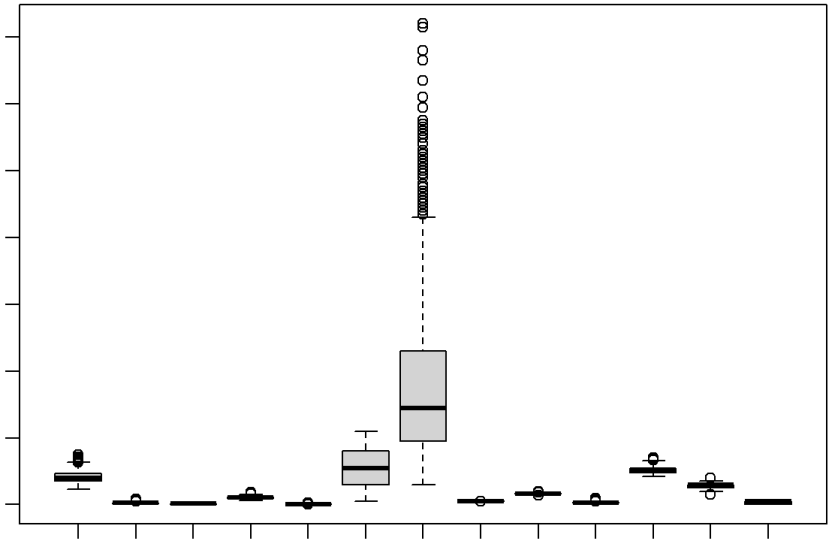
n
<int>
1599

1 row

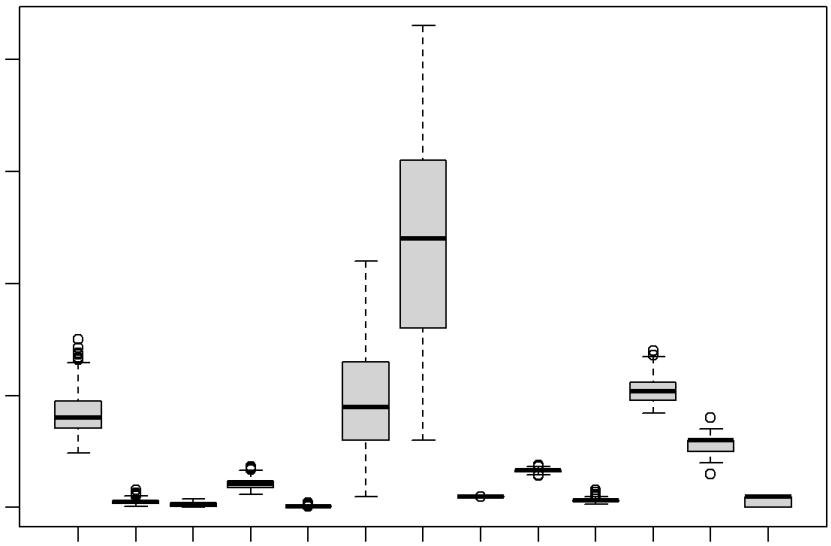
```
# Removing outliers for residual sugar:
Q <- quantile(wine$residual.sugar, probs=c(.25, .75), na.rm = FALSE)
iqr_rs <- IQR(wine$residual.sugar)
up_rs <-  Q[2]+1.5*iqr_rs # Upper Range
low_rs <- Q[1]-1.5*iqr_rs # Lower Range
eliminated_rs <- subset(wine, wine$residual.sugar > (Q[1] - 1.5*iqr_rs) &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



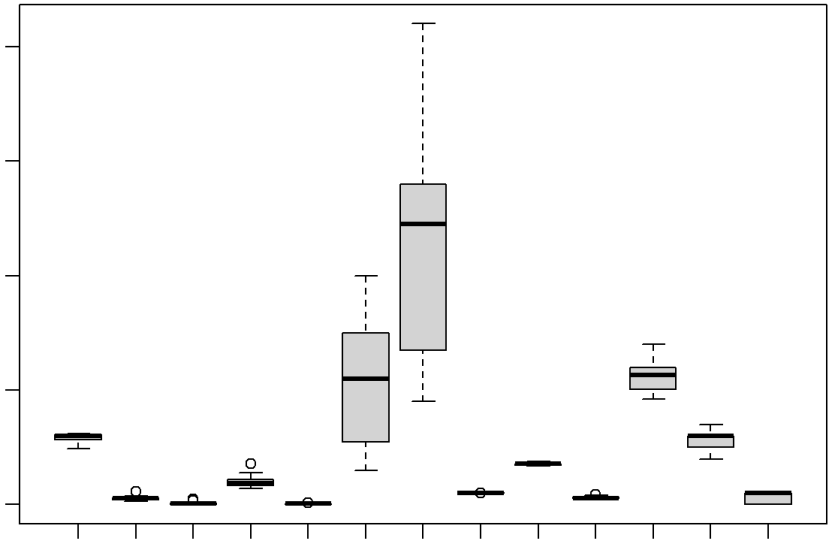
```
#Removing outliers for free.sulfur.dioxide:
Q2 <- quantile(wine$free.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
iqr_fs <- IQR(eliminated_rs$free.sulfur.dioxide)
up_fs <-  Q2[2]+1.5*iqr_fs # Upper Range
low_fs <- Q2[1]-1.5*iqr_fs # Lower Range
eliminated_fs <- subset(eliminated_rs, eliminated_rs$free.sulfur.dioxide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



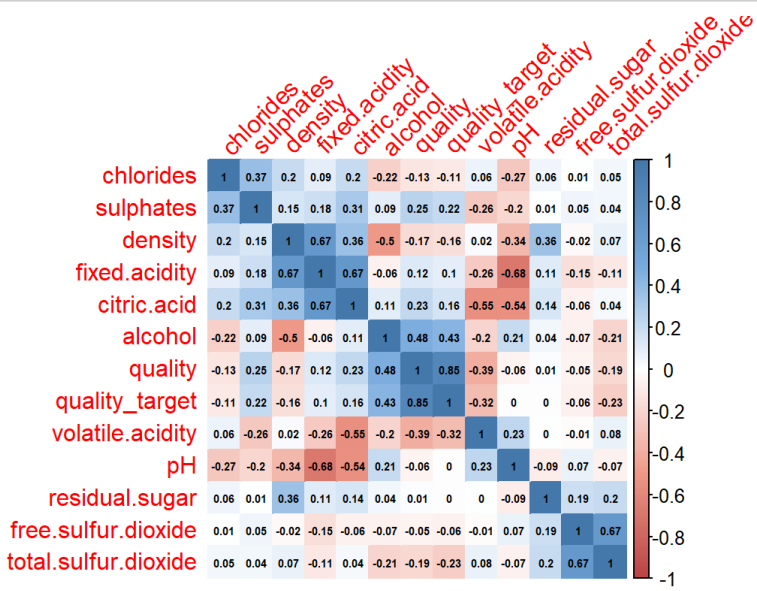
```
#Removing outliers for fixed.acidity:
Q4 <- quantile(wine$fixed.acidity, probs=c(.25, .75), na.rm = FALSE)
iqr_fa <- IQR(eliminated_ts$fixed.acidity)
up_fa <-  Q[2]+1.5*iqr_fa # Upper Range
low_fa <- Q[1]-1.5*iqr_fa # Lower Range
eliminated_fa <- subset(eliminated_ts, eliminated_ts$fixed.acidity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



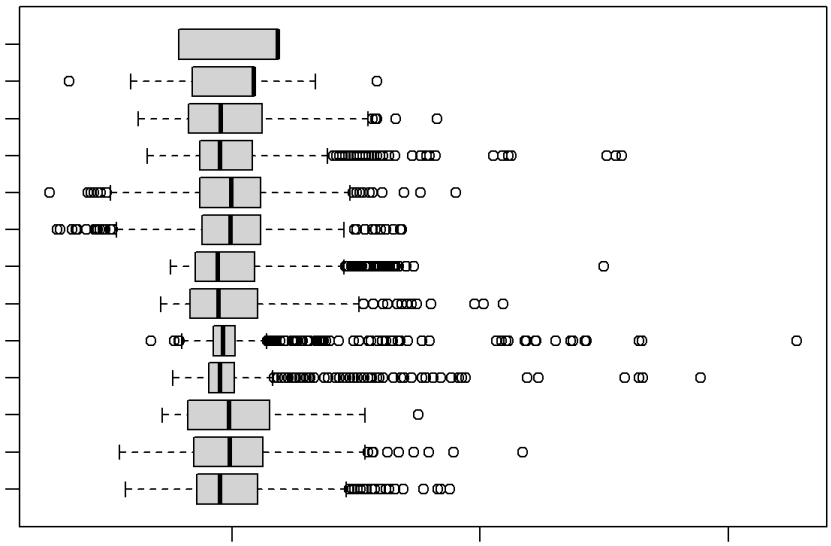
```
new_wine_data <- eliminated_fa

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

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

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

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

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



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

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

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

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

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



Logistic Regression Model

```
# 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
## -1.50873 -0.34784 -0.03531  0.38180  1.03439
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.30614    0.13967  -9.351  < 2e-16 ***
## volatile.acidity -0.56665    0.06979  -8.120 1.09e-15 ***
## sulphates         0.35047    0.07242   4.840 1.46e-06 ***
## alcohol           0.18285    0.01130  16.178  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4275 on 1277 degrees of freedom
## Multiple R-squared:  0.2676, Adjusted R-squared:  0.2659
## F-statistic: 155.5 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
## -2.69996 -0.37290 -0.06716  0.45889  2.20227
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.54400    0.21394   11.89  < 2e-16 ***
## volatile.acidity -1.19223    0.10689  -11.15  < 2e-16 ***
## sulphates         0.64005    0.11092    5.77 9.91e-09 ***
## alcohol           0.31617    0.01731   18.26  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6549 on 1277 degrees of freedom
## Multiple R-squared:  0.3433, Adjusted R-squared:  0.3418
## F-statistic: 222.5 on 3 and 1277 DF,  p-value: < 2.2e-16
```

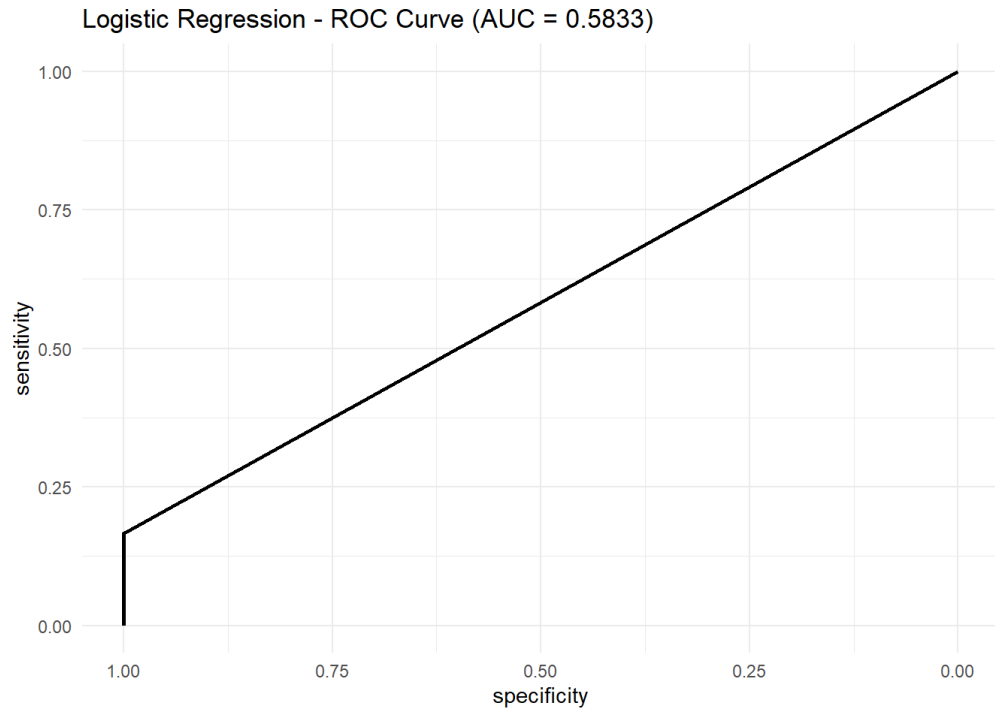
```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(lmodel2, newdata = wine_test)) -> df

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

```
##       fit             lwr             upr
##  Min.   :4.479   Min.   :3.189   Min.   :5.769
##  1st Qu.:5.241   1st Qu.:3.955   1st Qu.:6.528
##  Median :5.561   Median :4.275   Median :6.847
##  Mean   :5.606   Mean   :4.319   Mean   :6.892
##  3rd Qu.:5.898   3rd Qu.:4.612   3rd Qu.:7.183
##  Max.   :6.684   Max.   :5.396   Max.   :7.972
```

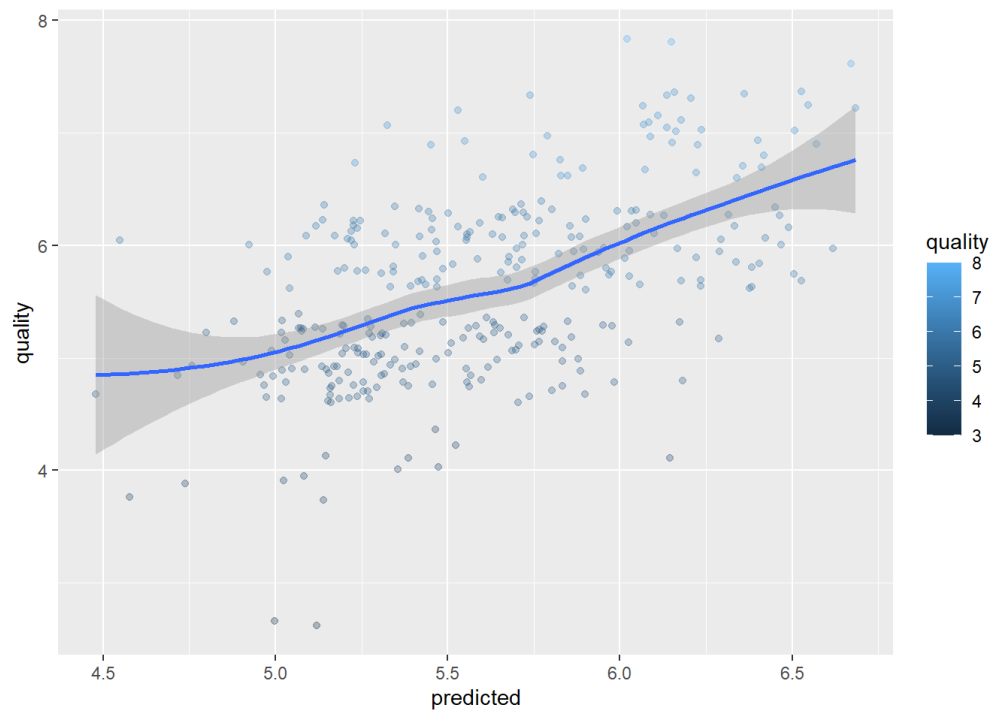
```r
# Confusion Matrix
# Convert predicted values to whole numbers, so they match target values
df$predicted_int = as.integer(round(df$predicted, digits = 0))

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

confusionMatrix(table1)
```

```
## Confusion Matrix and Statistics
##
##
##      5  7  6  4  8  3
##   5 84  0 49  1  0  0
##   7  3  5 32  0  0  0
##   6 44  3 80  0  0  0
##   4 10  0  2  0  0  0
##   8  0  1  2  0  0  0
##   3  2  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5314
##                  95% CI : (0.475, 0.5873)
##     No Information Rate : 0.5189
##     P-Value [Acc > NIR] : 0.3475
##
##                   Kappa : 0.2186
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 5 Class: 7 Class: 6 Class: 4 Class: 8 Class: 3
## Sensitivity            0.5874  0.55556   0.4848 0.000000       NA       NA
## Specificity            0.7143  0.88673   0.6928 0.962145 0.990566 0.993711
## Pos Pred Value         0.6269  0.12500   0.6299 0.000000       NA       NA
## Neg Pred Value         0.6793  0.98561   0.5550 0.996732       NA       NA
## Prevalence             0.4497  0.02830   0.5189 0.003145 0.000000 0.000000
## Detection Rate         0.2642  0.01572   0.2516 0.000000 0.000000 0.000000
## Detection Prevalence   0.4214  0.12579   0.3994 0.037736 0.009434 0.006289
## Balanced Accuracy      0.6508  0.72114   0.5888 0.481073       NA       NA
```

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

modelName1 <- 'Logistic Regression'
roc1 <- 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()
```

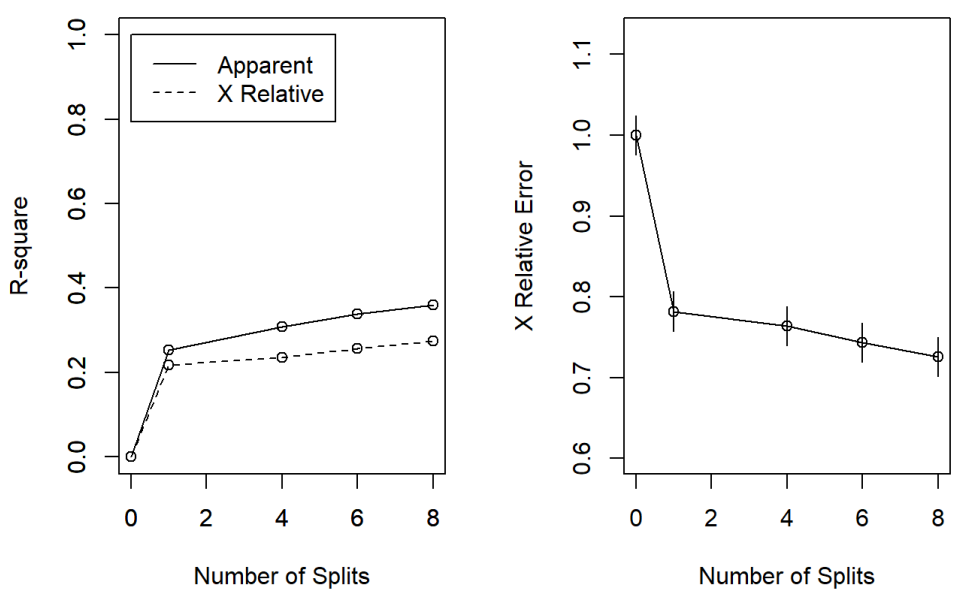



```
# The scatter plot supports the summary of the predicted interval, in the ranges of the fit,
# lower, and upper ranges. The R-squared value of 0.3283 of the model, indicates that this
# information can be predicted 33% of the time, with the data available, for the variance
# of the information.
```

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)
```

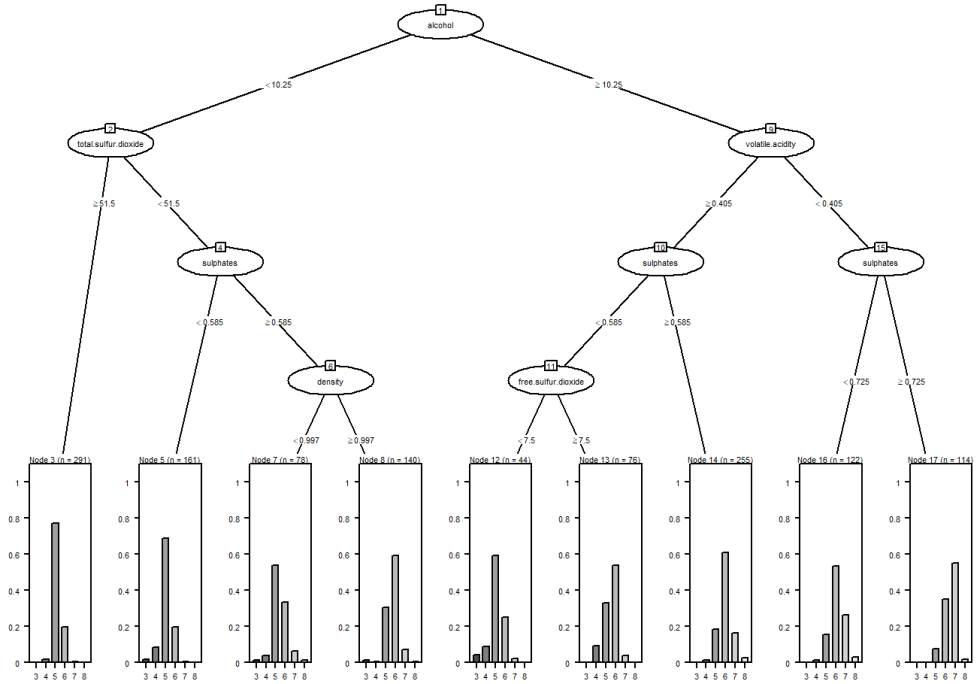
```
##
## Classification tree:
## rpart(formula = quality ~ ., data = rf_wine_train)
##
## Variables actually used in tree construction:
## [1] alcohol              density              free.sulfur.dioxide
## [4] sulphates            total.sulfur.dioxide volatile.acidity
##
## Root node error: 734/1281 = 0.57299
##
## n= 1281
##
##         CP nsplit rel error  xerror     xstd
## 1 0.253406      0   1.00000 1.00000 0.024120
## 2 0.016349      1   0.74659 0.78202 0.024249
## 3 0.015668      4   0.69210 0.76431 0.024192
## 4 0.010218      6   0.66076 0.74387 0.024114
## 5 0.010000      8   0.64033 0.72616 0.024035
```




```
# 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 >= 51.5: 5 (n = 291, err = 22.7%)
## |   |   [4] total.sulfur.dioxide < 51.5
## |   |   |   [5] sulphates < 0.585: 5 (n = 161, err = 31.1%)
## |   |   |   [6] sulphates >= 0.585
## |   |   |   |   [7] density < 0.99716: 5 (n = 78, err = 46.2%)
## |   |   |   |   [8] density >= 0.99716: 6 (n = 140, err = 40.7%)
## |   [9] alcohol >= 10.25
## |   |   [10] volatile.acidity >= 0.405
## |   |   |   [11] sulphates < 0.585
## |   |   |   |   [12] free.sulfur.dioxide < 7.5: 5 (n = 44, err = 40.9%)
## |   |   |   |   [13] free.sulfur.dioxide >= 7.5: 6 (n = 76, err = 46.1%)
## |   |   |   [14] sulphates >= 0.585: 6 (n = 255, err = 39.2%)
## |   |   [15] volatile.acidity < 0.405
## |   |   |   [16] sulphates < 0.725: 6 (n = 122, err = 46.7%)
## |   |   |   [17] sulphates >= 0.725: 7 (n = 114, err = 44.7%)
##
## Number of inner nodes:    8
## Number of terminal nodes: 9
```

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



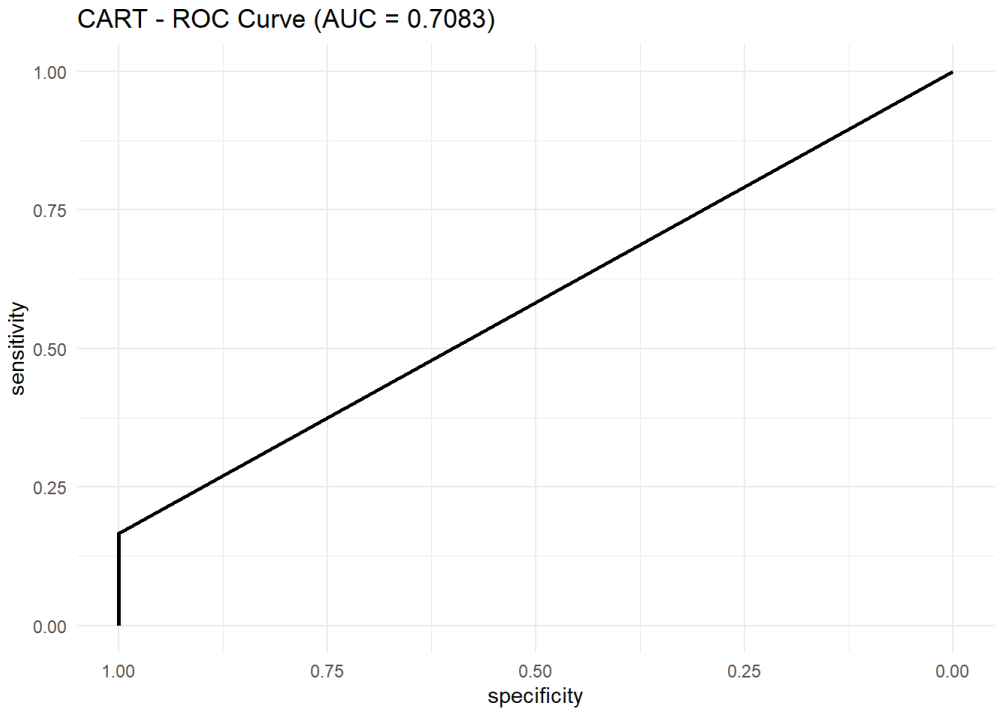

```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(rpartTree2, newdata = wine_test)) -> df2

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

```
##   3   4   5   6   7   8
##   0   0 150 148  20   0
```

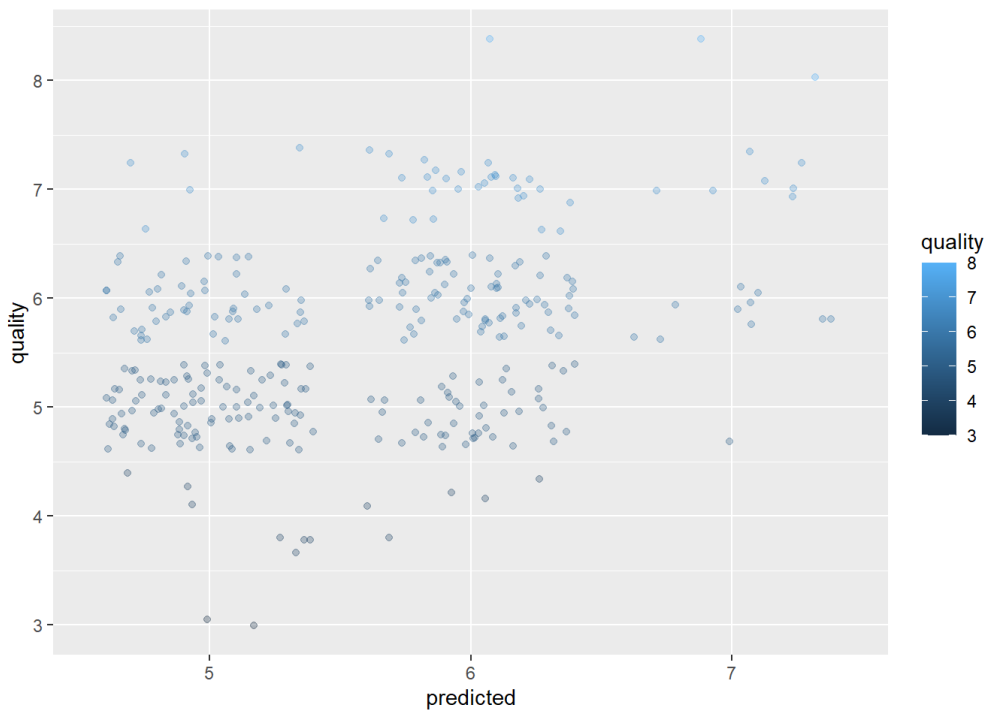
```
# Confusion Matrix
confusionMatrix(table(df2$quality, df2$predicted))
```

```
## Confusion Matrix and Statistics
##
##
##      3  4  5  6  7  8
##   3  0  0  2  0  0  0
##   4  0  0  7  5  0  0
##   5  0  0 89 44  1  0
##   6  0  0 47 70 10  0
##   7  0  0  5 28  7  0
##   8  0  0  0  1  2  0
##
## Overall Statistics
##
##                Accuracy : 0.522
##                  95% CI : (0.4656, 0.5781)
##     No Information Rate : 0.4717
##     P-Value [Acc > NIR] : 0.04093
##
##                   Kappa : 0.2131
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity               NA       NA   0.5933   0.4730  0.35000       NA
## Specificity          0.993711  0.96226   0.7321   0.6647  0.88926 0.990566
## Pos Pred Value            NA       NA   0.6642   0.5512  0.17500       NA
## Neg Pred Value            NA       NA   0.6685   0.5916  0.95324       NA
## Prevalence           0.000000  0.00000   0.4717   0.4654  0.06289 0.000000
## Detection Rate       0.000000  0.00000   0.2799   0.2201  0.02201 0.000000
## Detection Prevalence 0.006289  0.03774   0.4214   0.3994  0.12579 0.009434
## Balanced Accuracy         NA       NA   0.6627   0.5688  0.61963       NA
```

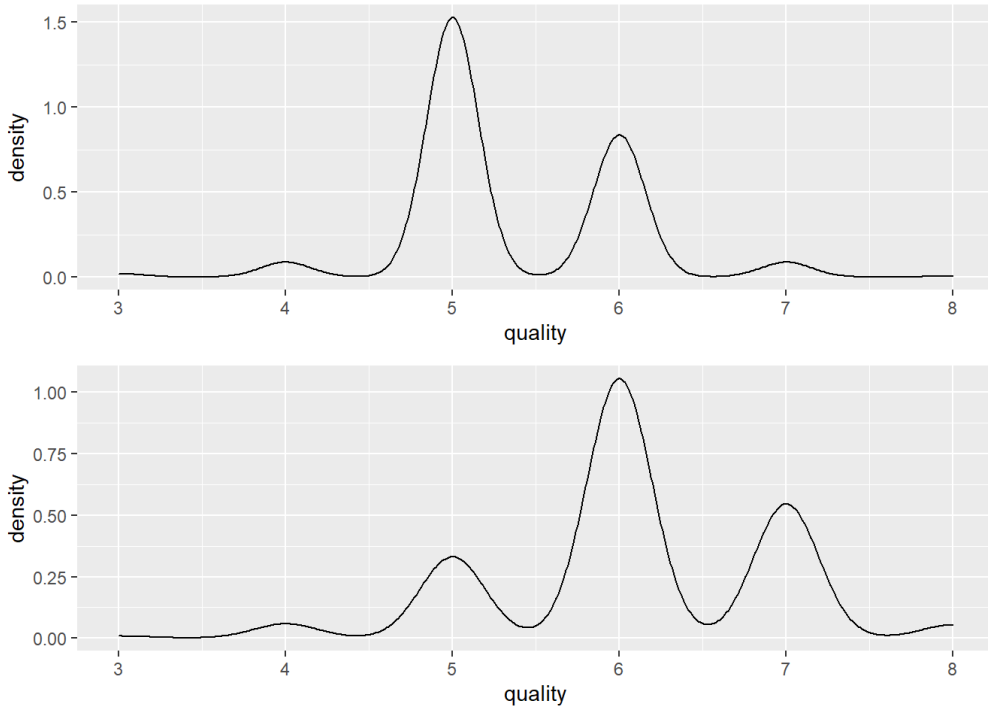
```
# ROC plot
df2$predicted_int = round(as.numeric(as.character(df2$predicted)), digits = 0)

modelName2 <- 'CART'
roc2 <- roc(df2$quality, df2$predicted_int)
auc2 <- round(auc(df2$quality, df2$predicted_int), 4)

ggroc(roc1, colours = 'red', size = 1) +
  ggtitle(paste0(modelName2, ' - ROC Curve ', '(AUC = ', auc2 , ')')) + theme_minimal()
```




```
# 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)
```

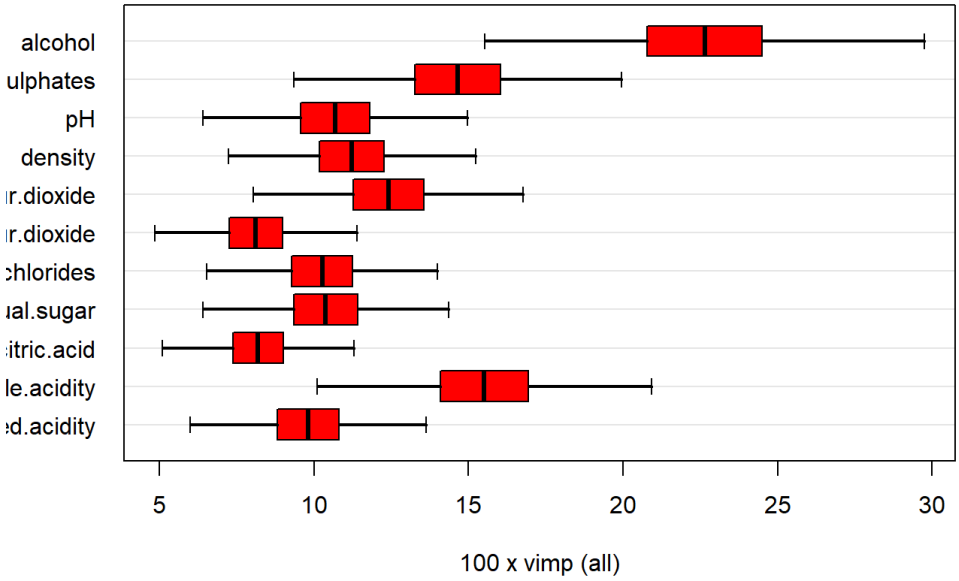
```
##                          Sample size: 1281
##            Frequency of class labels: 8, 41, 547, 511, 159, 15
##                      Number of trees: 500
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 250.732
## No. of variables tried at each split: 4
##               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: 0.07066667
##         (OOB) Normalized Brier score: 0.50880002
##                            (OOB) AUC: 0.79335591
##    (OOB) Requested performance error: 0.31850117, 1, 1, 0.19195612, 0.3072407, 0.51572327, 1
##
## Confusion matrix:
##
##           predicted
##   observed 3 4   5   6  7 8 class.error
##          3 0 1   6   1  0 0      1.0000
##          4 1 0  26  13  1 0      1.0000
##          5 0 0 443 100  4 0      0.1901
##          6 0 1 121 354 35 0      0.3072
##          7 0 0   9  71 77 2      0.5157
##          8 0 0   0   9  6 0      1.0000
##
##       (OOB) Misclassification rate: 0.3177205
```

```
# 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	6.907788	9.811890	12.71599	1.771907e-11	TRUE
volatile.acidity	11.389368	15.511253	19.63314	8.178505e-14	TRUE
citric.acid	5.831720	8.195329	10.55894	5.386653e-12	TRUE
residual.sugar	7.359686	10.379618	13.39955	8.114120e-12	TRUE
chlorides	7.424664	10.268790	13.11292	7.392005e-13	TRUE
free.sulfur.dioxide	5.633074	8.122718	10.61236	8.049236e-11	TRUE
total.sulfur.dioxide	9.089833	12.405520	15.72121	1.124165e-13	TRUE
density	8.189936	11.229299	14.26866	2.221587e-13	TRUE
pH	7.441979	10.690870	13.93976	5.611050e-11	TRUE
sulphates	10.614492	14.650955	18.68742	5.636940e-13	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: 250.732
##          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: 0.07071588
##               Normalized Brier score: 0.50915437
##                                  AUC: 0.85615164
##          Requested performance error: 0.29559748, 1, 1, 0.21641791, 0.23622047, 0.475, 0.66666667
##
## Confusion matrix:
##
##           predicted
##   observed 3 4   5  6  7 8 class.error
##          3 0 0   2  0  0 0      1.0000
##          4 0 0   6  6  0 0      1.0000
##          5 0 0 105 27  2 0      0.2164
##          6 0 0  28 97  2 0      0.2362
##          7 0 0   3 16 21 0      0.4750
##          8 0 0   0  0  2 1      0.6667
##
##            Misclassification error: 0.2955975
```

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      0.6437486  0.3675424  0.4966083
##   2      0.6422815  0.3699347  0.4963045
##   3      0.6422260  0.3702025  0.4954467
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 3.
```

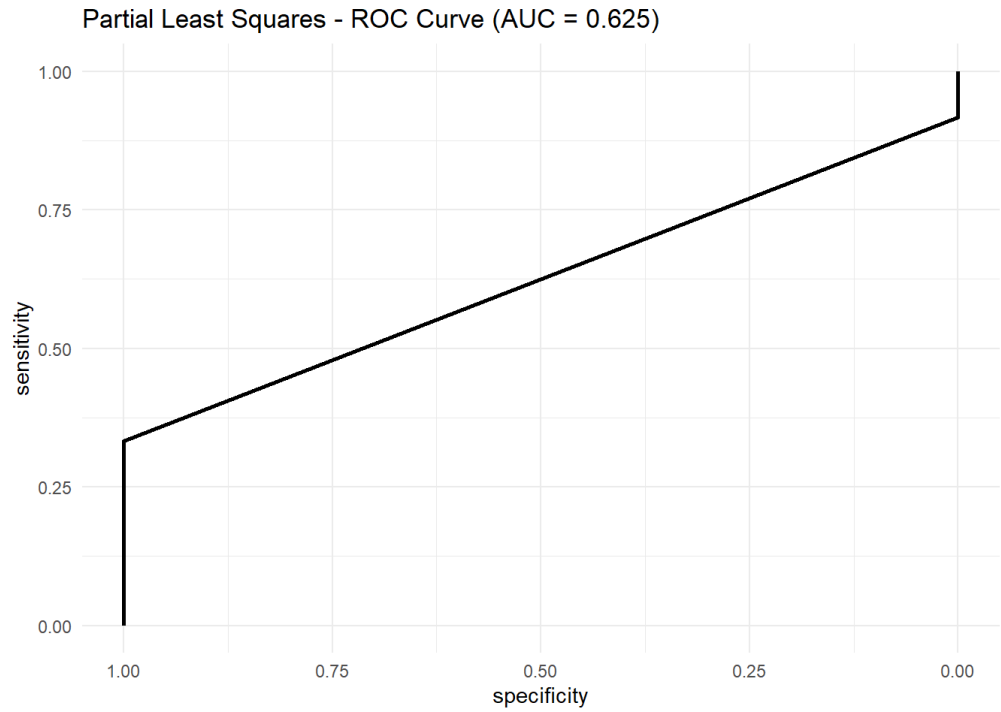
```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(pls_wine, newdata = wine_test)) -> df3

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.267   5.211   5.560   5.601   5.943   6.684
```

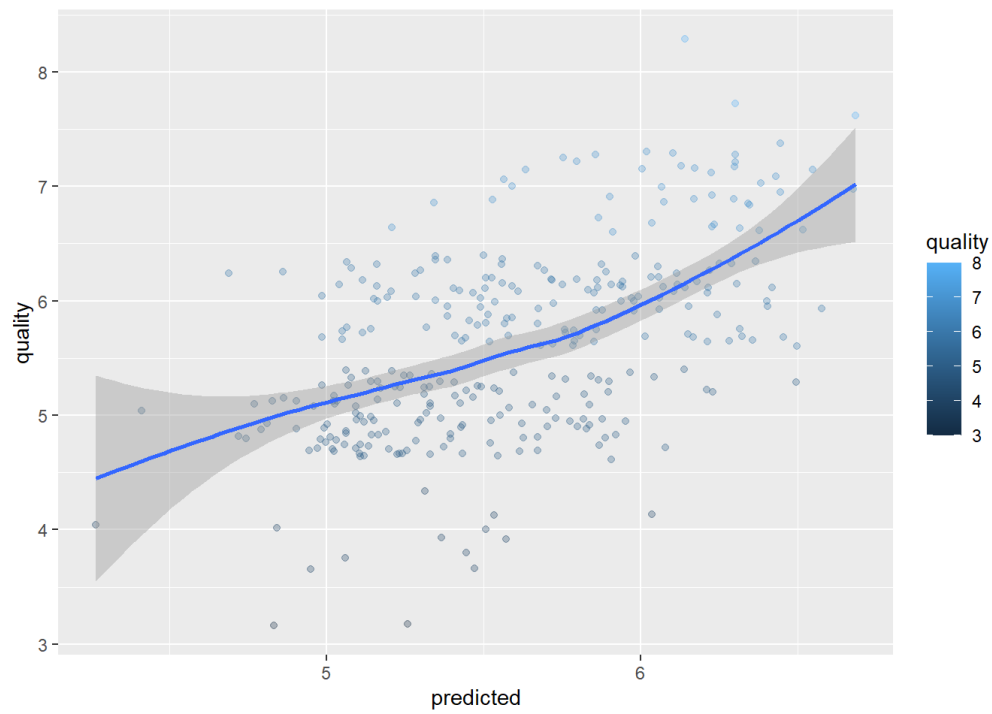
```
# Confusion Matrix
# Convert predicted values to whole numbers, so they match target values
df3$predicted_int = as.integer(round(df3$predicted, digits = 0))

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  7  6  4  8  3
##   5 89  0 44  1  0  0
##   7  2  3 35  0  0  0
##   6 39  1 87  0  0  0
##   4  7  0  4  1  0  0
##   8  0  1  2  0  0  0
##   3  2  0  0  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 : 0.2738
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 5 Class: 7 Class: 6 Class: 4 Class: 8 Class: 3
## Sensitivity            0.6403 0.600000   0.5058 0.500000       NA       NA
## Specificity            0.7486 0.881789   0.7260 0.965190 0.990566 0.993711
## Pos Pred Value         0.6642 0.075000   0.6850 0.083333       NA       NA
## Neg Pred Value         0.7283 0.992806   0.5550 0.996732       NA       NA
## Prevalence             0.4371 0.015723   0.5409 0.006289 0.000000 0.000000
## Detection Rate         0.2799 0.009434   0.2736 0.003145 0.000000 0.000000
## Detection Prevalence   0.4214 0.125786   0.3994 0.037736 0.009434 0.006289
## Balanced Accuracy      0.6944 0.740895   0.6159 0.732595       NA       NA
```

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

modelName3 <- 'Partial Least Squares'
roc3 <- 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

```
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 0.4041887    RSS 499.34    GRSq 0.3800956    RSq 0.4012217
```

```
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##             data=wine_train)
##
##                              coefficients
## (Intercept)                    29.6256957
## h(1-volatile.acidity)           0.8852370
## h(volatile.acidity-1)          -2.4127800
## h(chlorides-0.042)             19.9898403
## h(chlorides-0.093)            -12.2303109
## h(0.147-chlorides)             18.0260781
## h(chlorides-0.241)            -11.5122785
## h(total.sulfur.dioxide-9)      -0.2068679
## h(133-total.sulfur.dioxide)    -0.2039348
## h(total.sulfur.dioxide-133)     0.2142835
## h(0.82-sulphates)              -1.7611796
## h(12.4-alcohol)                -0.3093945
##
## 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 0.4041887    RSS 499.34    GRSq 0.3800956    RSq 0.4012217
```

```
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      0.6980937  0.2410713  0.5481180
##   1        3      0.6625528  0.3161322  0.5089983
##   1        4      0.6414653  0.3587293  0.4963981
##   1        5      0.6422392  0.3574602  0.4954879
##   1        6      0.6417237  0.3585564  0.4932711
##   1        7      0.6416446  0.3587798  0.4927852
##   1        8      0.6423400  0.3574198  0.4930751
##   1        9      0.6444475  0.3543354  0.4939707
##   1       10      0.6438232  0.3560011  0.4935398
##   1       11      0.6453399  0.3539126  0.4942555
##   1       12      0.6463176  0.3524901  0.4947600
##   1       13      0.6464563  0.3522562  0.4950027
##   1       14      0.6467683  0.3518672  0.4950933
##   1       15      0.6468107  0.3518250  0.4952098
##   1       16      0.6468107  0.3518250  0.4952098
##   1       17      0.6468107  0.3518250  0.4952098
##   1       18      0.6468107  0.3518250  0.4952098
##   1       19      0.6468107  0.3518250  0.4952098
##   1       20      0.6468107  0.3518250  0.4952098
##   1       21      0.6468107  0.3518250  0.4952098
##   1       22      0.6468107  0.3518250  0.4952098
##   1       23      0.6468107  0.3518250  0.4952098
##   1       24      0.6468107  0.3518250  0.4952098
##   1       25      0.6468107  0.3518250  0.4952098
##   1       26      0.6468107  0.3518250  0.4952098
##   1       27      0.6468107  0.3518250  0.4952098
##   1       28      0.6468107  0.3518250  0.4952098
##   1       29      0.6468107  0.3518250  0.4952098
##   1       30      0.6468107  0.3518250  0.4952098
##   1       31      0.6468107  0.3518250  0.4952098
##   1       32      0.6468107  0.3518250  0.4952098
##   1       33      0.6468107  0.3518250  0.4952098
##   1       34      0.6468107  0.3518250  0.4952098
##   1       35      0.6468107  0.3518250  0.4952098
##   1       36      0.6468107  0.3518250  0.4952098
##   1       37      0.6468107  0.3518250  0.4952098
##   1       38      0.6468107  0.3518250  0.4952098
##   2        2      0.6977630  0.2416075  0.5472608
##   2        3      0.6658207  0.3093708  0.5138282
##   2        4      0.6444153  0.3537541  0.4957689
##   2        5      0.6378347  0.3669557  0.4893667
##   2        6      0.6378938  0.3678589  0.4892333
##   2        7      0.6387058  0.3671135  0.4877267
##   2        8      0.6377459  0.3696433  0.4871677
##   2        9      0.6455242  0.3610463  0.4894594
##   2       10      0.6476644  0.3576680  0.4909094
##   2       11      0.6480229  0.3572479  0.4915958
##   2       12      0.6494303  0.3555074  0.4927250
##   2       13      0.6511946  0.3527971  0.4941743
##   2       14      0.6524934  0.3511155  0.4952042
##   2       15      0.6529988  0.3505030  0.4953743
##   2       16      0.6537915  0.3499405  0.4954884
##   2       17      0.6543992  0.3491393  0.4954318
##   2       18      0.6545145  0.3489970  0.4956122
##   2       19      0.6545332  0.3489349  0.4955728
##   2       20      0.6545332  0.3489349  0.4955728
##   2       21      0.6545332  0.3489349  0.4955728
##   2       22      0.6545332  0.3489349  0.4955728
```

```
##   2       23      0.6545332  0.3489349  0.4955728
##   2       24      0.6545332  0.3489349  0.4955728
##   2       25      0.6545332  0.3489349  0.4955728
##   2       26      0.6545332  0.3489349  0.4955728
##   2       27      0.6545332  0.3489349  0.4955728
##   2       28      0.6545332  0.3489349  0.4955728
##   2       29      0.6545332  0.3489349  0.4955728
##   2       30      0.6545332  0.3489349  0.4955728
##   2       31      0.6545332  0.3489349  0.4955728
##   2       32      0.6545332  0.3489349  0.4955728
##   2       33      0.6545332  0.3489349  0.4955728
##   2       34      0.6545332  0.3489349  0.4955728
##   2       35      0.6545332  0.3489349  0.4955728
##   2       36      0.6545332  0.3489349  0.4955728
##   2       37      0.6545332  0.3489349  0.4955728
##   2       38      0.6545332  0.3489349  0.4955728
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nprune = 8 and degree = 2.
```

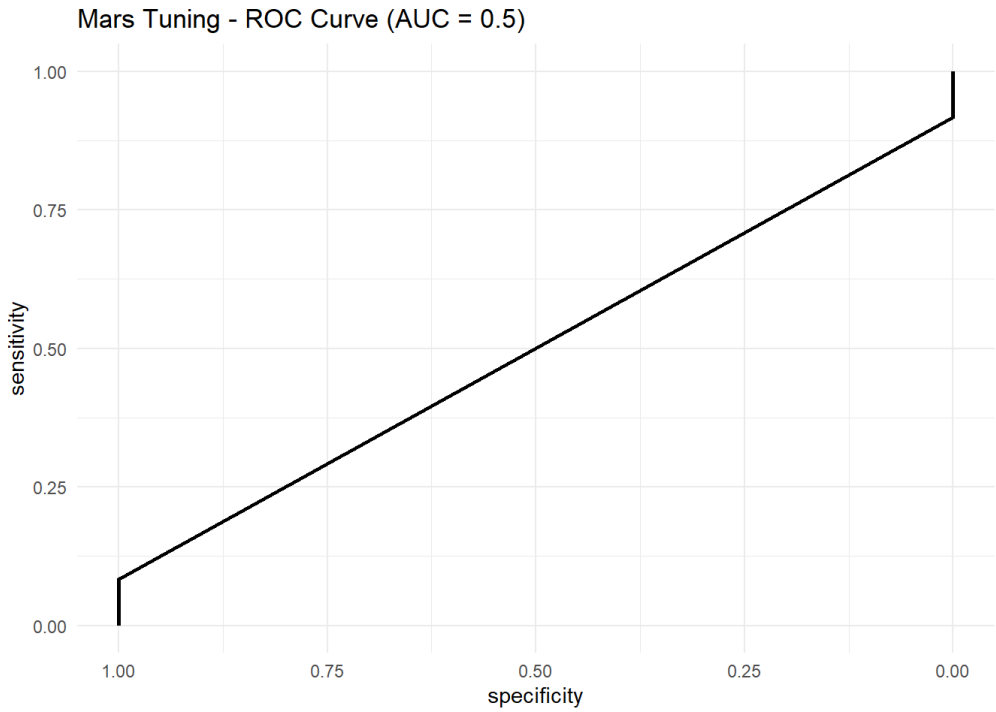
```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(marsModel_wine, newdata = wine_test)) -> df4

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

```
##        y
##  Min.   :4.324
##  1st Qu.:5.215
##  Median :5.512
##  Mean   :5.601
##  3rd Qu.:5.954
##  Max.   :7.165
```

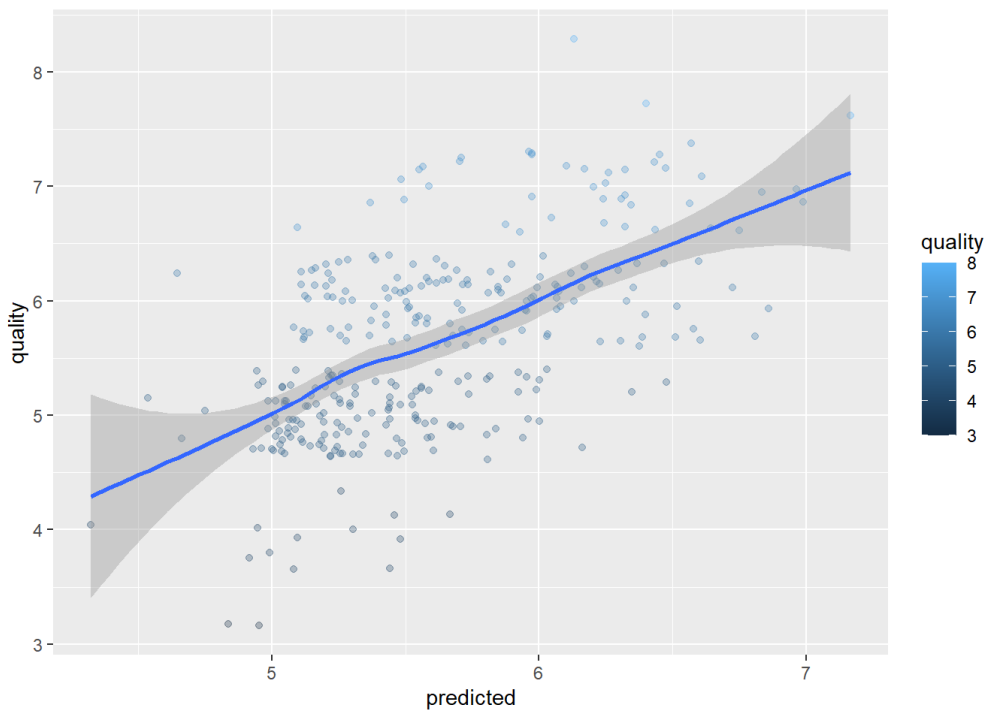
```
# Confusion Matrix
# Convert predicted values to whole numbers, so they match target values
df4$predicted_int = as.integer(round(df4$predicted, digits = 0))

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  7  6  4  8  3
##   5 96  0 38  0  0  0
##   7  4  8 28  0  0  0
##   6 43  8 76  0  0  0
##   4 10  0  1  1  0  0
##   8  0  1  2  0  0  0
##   3  2  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5692
##                  95% CI : (0.5128, 0.6243)
##     No Information Rate : 0.4874
##     P-Value [Acc > NIR] : 0.002099
##
##                   Kappa : 0.2887
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 5 Class: 7 Class: 6 Class: 4 Class: 8 Class: 3
## Sensitivity            0.6194  0.47059   0.5241 1.000000       NA       NA
## Specificity            0.7669  0.89369   0.7052 0.965300 0.990566 0.993711
## Pos Pred Value         0.7164  0.20000   0.5984 0.083333       NA       NA
## Neg Pred Value         0.6793  0.96763   0.6387 1.000000       NA       NA
## Prevalence             0.4874  0.05346   0.4560 0.003145 0.000000 0.000000
## Detection Rate         0.3019  0.02516   0.2390 0.003145 0.000000 0.000000
## Detection Prevalence   0.4214  0.12579   0.3994 0.037736 0.009434 0.006289
## Balanced Accuracy      0.6931  0.68214   0.6147 0.982650       NA       NA
```

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

modelName4 <- 'Mars Tuning'
roc4 <- 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  0.7521525  0.3203533  0.4316607
##    2  0.6985555  0.3264314  0.4715786
##    3  0.6653514  0.3569010  0.4746894
##    4  0.6651942  0.3490047  0.4870039
##    5  0.6574099  0.3566475  0.4861689
##    6  0.6492559  0.3667561  0.4831128
##    7  0.6399374  0.3809448  0.4785697
##    8  0.6385314  0.3826568  0.4800326
##    9  0.6402529  0.3782778  0.4839955
##   10  0.6405511  0.3769438  0.4862032
##   11  0.6388557  0.3791714  0.4883295
##   12  0.6384313  0.3799329  0.4888691
##   13  0.6392786  0.3774647  0.4906211
##   14  0.6393170  0.3770380  0.4908753
##   15  0.6388477  0.3775382  0.4900999
##   16  0.6390601  0.3768135  0.4904412
##   17  0.6395154  0.3753306  0.4909220
##   18  0.6393814  0.3755507  0.4917981
##   19  0.6405194  0.3730580  0.4933439
##   20  0.6405523  0.3728261  0.4935658
##   21  0.6398569  0.3738755  0.4931313
##   22  0.6384330  0.3766792  0.4922939
##   23  0.6385338  0.3764890  0.4932048
##   24  0.6374188  0.3788868  0.4930580
##   25  0.6375910  0.3786439  0.4938681
##   26  0.6385416  0.3770421  0.4948572
##   27  0.6371939  0.3796201  0.4940326
##   28  0.6355911  0.3828538  0.4937173
##   29  0.6366850  0.3804844  0.4946191
##   30  0.6361502  0.3816332  0.4938863
##   31  0.6360820  0.3819181  0.4942621
##   32  0.6358999  0.3822639  0.4942604
##   33  0.6364777  0.3813935  0.4949187
##   34  0.6360795  0.3821895  0.4947905
##   35  0.6358171  0.3828567  0.4938134
##   36  0.6360646  0.3824326  0.4939119
##   37  0.6368501  0.3809365  0.4945408
##   38  0.6367606  0.3812376  0.4943030
##   39  0.6358326  0.3831289  0.4937038
##   40  0.6364746  0.3818429  0.4937437
##   41  0.6358903  0.3830483  0.4936105
##   42  0.6358212  0.3831107  0.4931912
##   43  0.6356463  0.3834548  0.4934727
##   44  0.6359103  0.3830652  0.4937064
##   45  0.6349275  0.3851310  0.4928095
##   46  0.6349908  0.3850764  0.4927709
##   47  0.6348762  0.3853930  0.4928520
##   48  0.6347900  0.3855302  0.4929953
##   49  0.6347756  0.3855813  0.4931565
##   50  0.6349057  0.3854531  0.4933077
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 49.
```

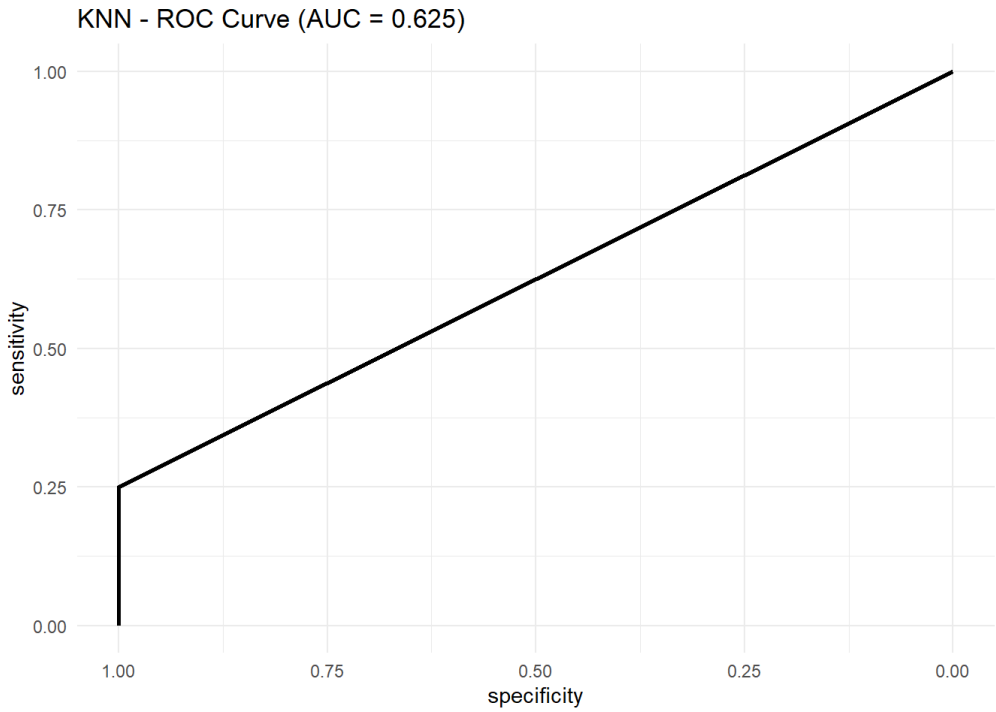
```r
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(knn_wine, newdata = wine_test)) -> df5

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.898   5.245   5.531   5.627   5.949   6.878
```

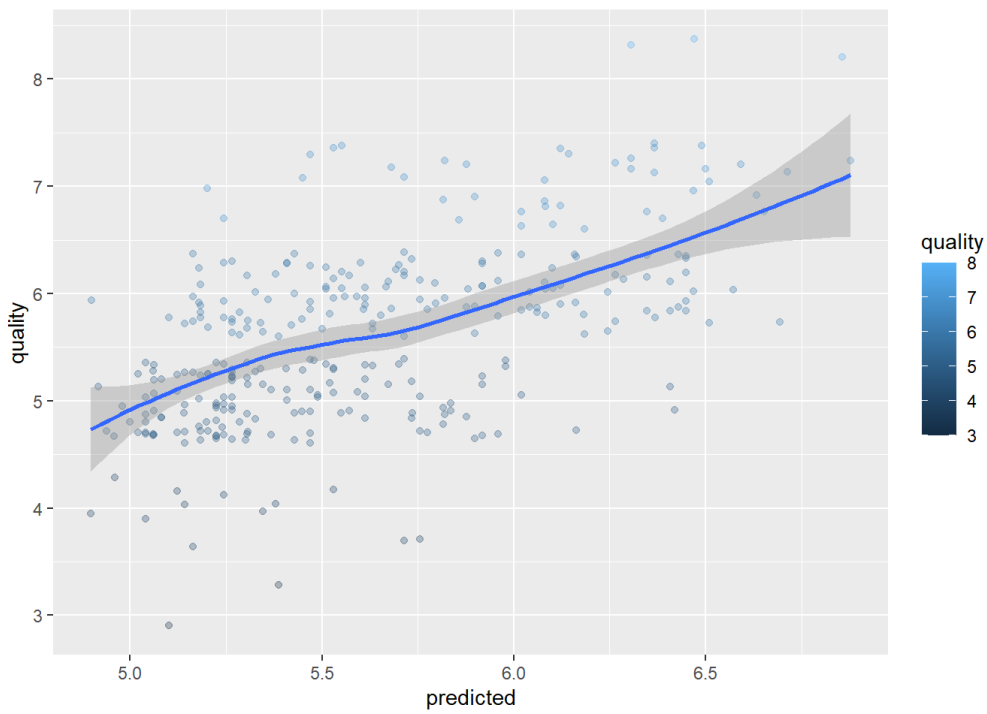
```r
# Confusion Matrix
# Convert predicted values to whole numbers, so they match target values
df5$predicted_int = as.integer(round(df5$predicted, digits = 0))

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  7  6  4  8  3
##   5 95  0 39  0  0  0
##   7  4  6 30  0  0  0
##   6 40  3 84  0  0  0
##   4  9  0  3  0  0  0
##   8  0  1  2  0  0  0
##   3  2  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5818
##                  95% CI : (0.5254, 0.6366)
##     No Information Rate : 0.4969
##     P-Value [Acc > NIR] : 0.001454
##
##                   Kappa : 0.3016
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 5 Class: 7 Class: 6 Class: 4 Class: 8 Class: 3
## Sensitivity            0.6333  0.60000   0.5316       NA       NA       NA
## Specificity            0.7679  0.88961   0.7312  0.96226 0.990566 0.993711
## Pos Pred Value         0.7090  0.15000   0.6614       NA       NA       NA
## Neg Pred Value         0.7011  0.98561   0.6126       NA       NA       NA
## Prevalence             0.4717  0.03145   0.4969  0.00000 0.000000 0.000000
## Detection Rate         0.2987  0.01887   0.2642  0.00000 0.000000 0.000000
## Detection Prevalence   0.4214  0.12579   0.3994  0.03774 0.009434 0.006289
## Balanced Accuracy      0.7006  0.74481   0.6314       NA       NA       NA
```

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

modelName5 <- 'KNN'
roc5 <- 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

```
set.seed(4)


svmTune <- train(quality ~ volatile.acidity + sulphates + alcohol, 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
##    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, 1153, 1154, 1151, 1153, 1152, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.6016946  0.3339034
##   0.50  0.5994118  0.3342121
##   1.00  0.5970925  0.3327318
##   2.00  0.5978799  0.3365259
##   4.00  0.6073292  0.3559663
##
## Tuning parameter 'sigma' was held constant at a value of 0.6439695
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.6439695 and C = 4.
```

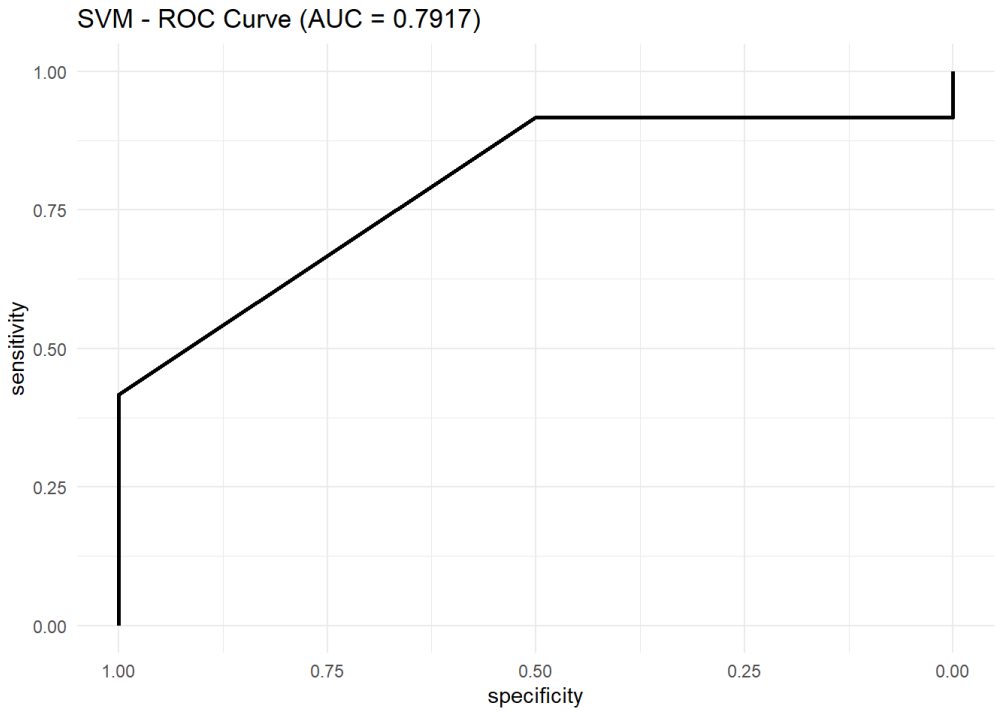
```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(svmTune, newdata = wine_test)) -> df6

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

```
##   3   4   5   6   7   8
##   2   1 144 154  17   0
```

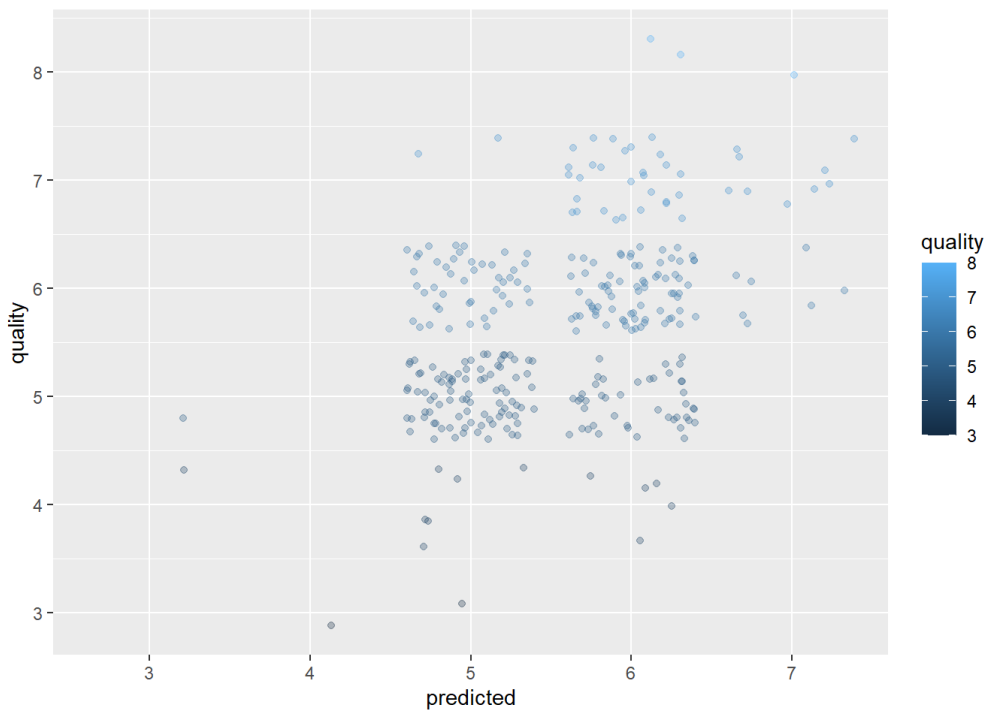
```
# Confusion Matrix
confusionMatrix(table(df6$quality, df6$predicted))
```

```
## Confusion Matrix and Statistics
##
##
##      3  4  5  6  7  8
##   3  0  1  1  0  0  0
##   4  1  0  6  5  0  0
##   5  1  0 89 44  0  0
##   6  0  0 46 74  7  0
##   7  0  0  2 29  9  0
##   8  0  0  0  2  1  0
##
## Overall Statistics
##
##                Accuracy : 0.5409
##                  95% CI : (0.4844, 0.5966)
##     No Information Rate : 0.4843
##     P-Value [Acc > NIR] : 0.0248
##
##                   Kappa : 0.246
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000 0.000000   0.6181   0.4805  0.52941       NA
## Specificity          0.993671 0.962145   0.7414   0.6768  0.89701 0.990566
## Pos Pred Value       0.000000 0.000000   0.6642   0.5827  0.22500       NA
## Neg Pred Value       0.993671 0.996732   0.7011   0.5812  0.97122       NA
## Prevalence           0.006289 0.003145   0.4528   0.4843  0.05346 0.000000
## Detection Rate       0.000000 0.000000   0.2799   0.2327  0.02830 0.000000
## Detection Prevalence 0.006289 0.037736   0.4214   0.3994  0.12579 0.009434
## Balanced Accuracy    0.496835 0.481073   0.6797   0.5787  0.71321       NA
```

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

modelName6 <- 'SVM'
roc6 <- 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: 1152, 1153, 1153, 1153, 1153, 1152, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda  Accuracy   Kappa
##   0.0    0.0100  0.5947686  0.3287572
##   0.0    0.0575  0.5877495  0.3046086
##   0.0    0.1050  0.5814809  0.2902285
##   0.0    0.1525  0.5806935  0.2870697
##   0.0    0.2000  0.5814748  0.2866086
##   0.1    0.0100  0.5932123  0.3290949
##   0.1    0.0575  0.5900932  0.3071147
##   0.1    0.1050  0.5838369  0.2923128
##   0.1    0.1525  0.5791554  0.2821370
##   0.1    0.2000  0.5776110  0.2792220
##   0.2    0.0100  0.5947808  0.3313302
##   0.2    0.0575  0.5869682  0.3014431
##   0.2    0.1050  0.5799550  0.2837216
##   0.2    0.1525  0.5752978  0.2752207
##   0.2    0.2000  0.5760850  0.2758423
##   0.4    0.0100  0.6010308  0.3399823
##   0.4    0.0575  0.5760364  0.2797152
##   0.4    0.1050  0.5721726  0.2695777
##   0.4    0.1525  0.5698470  0.2648390
##   0.4    0.2000  0.5643964  0.2550422
##   0.6    0.0100  0.5939995  0.3285497
##   0.6    0.0575  0.5705856  0.2684514
##   0.6    0.1050  0.5729538  0.2703826
##   0.6    0.1525  0.5628338  0.2521922
##   0.6    0.2000  0.5612836  0.2480027
##   0.8    0.0100  0.5924430  0.3238637
##   0.8    0.0575  0.5776292  0.2793124
##   0.8    0.1050  0.5675031  0.2607494
##   0.8    0.1525  0.5589399  0.2442030
##   0.8    0.2000  0.5495829  0.2267688
##   1.0    0.0100  0.5939871  0.3250939
##   1.0    0.0575  0.5737289  0.2721269
##   1.0    0.1050  0.5604901  0.2479148
##   1.0    0.1525  0.5527141  0.2329530
##   1.0    0.2000  0.4832423  0.1046889
##
## 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.
```

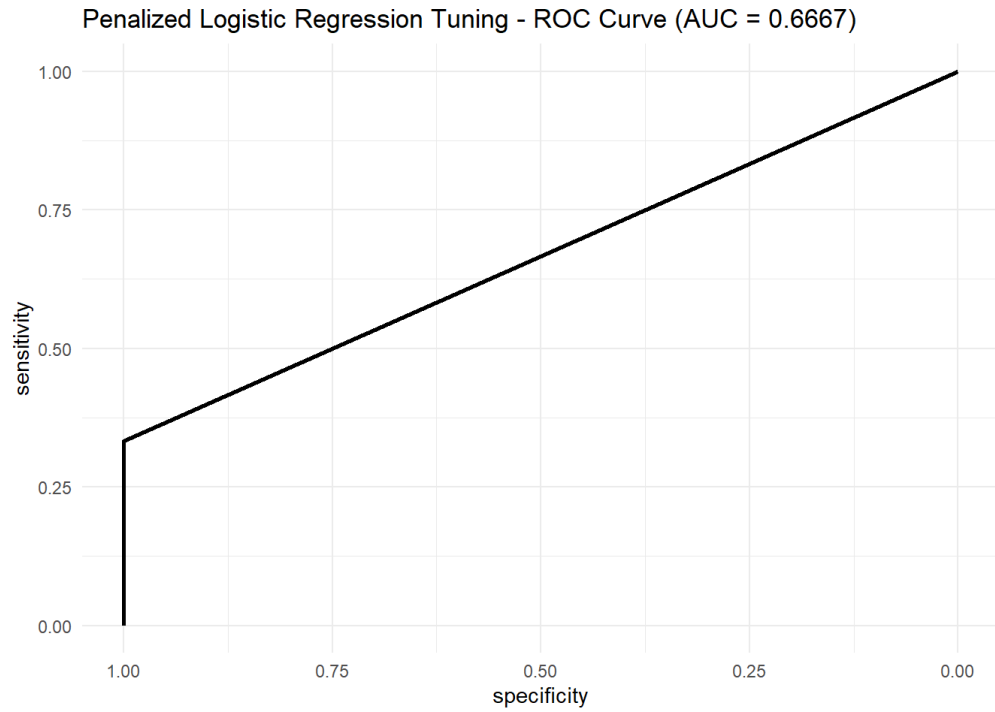
```
# Add predicted values to new data frame
wine_test %>%
  mutate(predicted = predict(glmnTune, newdata = wine_test)) -> df7

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

```
##   3   4   5   6   7   8
##   2   1 144 154  17   0
```

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

```
## Confusion Matrix and Statistics
##
##
##       3   4   5   6   7   8
##   3   0   0   2   0   0   0
##   4   0   0   8   4   0   0
##   5   0   0 101  33   0   0
##   6   0   0  52  73   2   0
##   7   0   0   5  28   7   0
##   8   0   0   0   2   1   0
##
## Overall Statistics
##
##                Accuracy : 0.5692
##                  95% CI : (0.5128, 0.6243)
##     No Information Rate : 0.5283
##     P-Value [Acc > NIR] : 0.07992
##
##                   Kappa : 0.2791
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity                NA       NA   0.6012   0.5214  0.70000       NA
## Specificity          0.993711  0.96226   0.7800   0.6966  0.89286 0.990566
## Pos Pred Value             NA       NA   0.7537   0.5748  0.17500       NA
## Neg Pred Value             NA       NA   0.6359   0.6492  0.98921       NA
## Prevalence           0.000000  0.00000   0.5283   0.4403  0.03145 0.000000
## Detection Rate       0.000000  0.00000   0.3176   0.2296  0.02201 0.000000
## Detection Prevalence 0.006289  0.03774   0.4214   0.3994  0.12579 0.009434
## Balanced Accuracy          NA       NA   0.6906   0.6090  0.79643       NA
```

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

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()
```