

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)
```

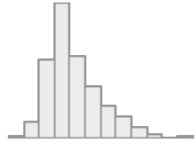
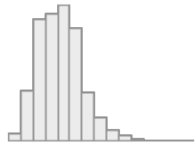
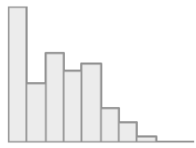
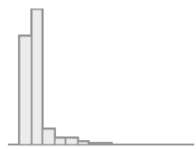
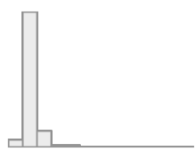
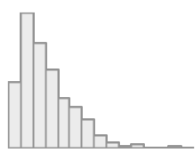
Data Summary

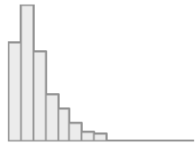
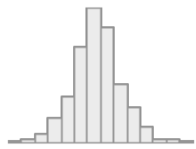
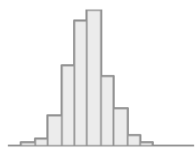
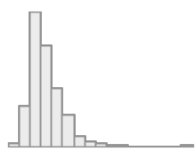
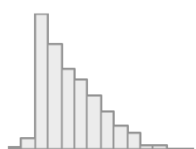
```
# use the view function to view in R Studio
#view(
dfSummary(wine,
  plain.ascii = FALSE,
  style       = "grid",
  graph.magnif = 0.75,
  valid.col   = FALSE,
  tmp.img.dir = "NA")
```

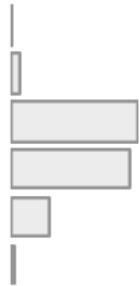
Data Frame Summary

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:
```

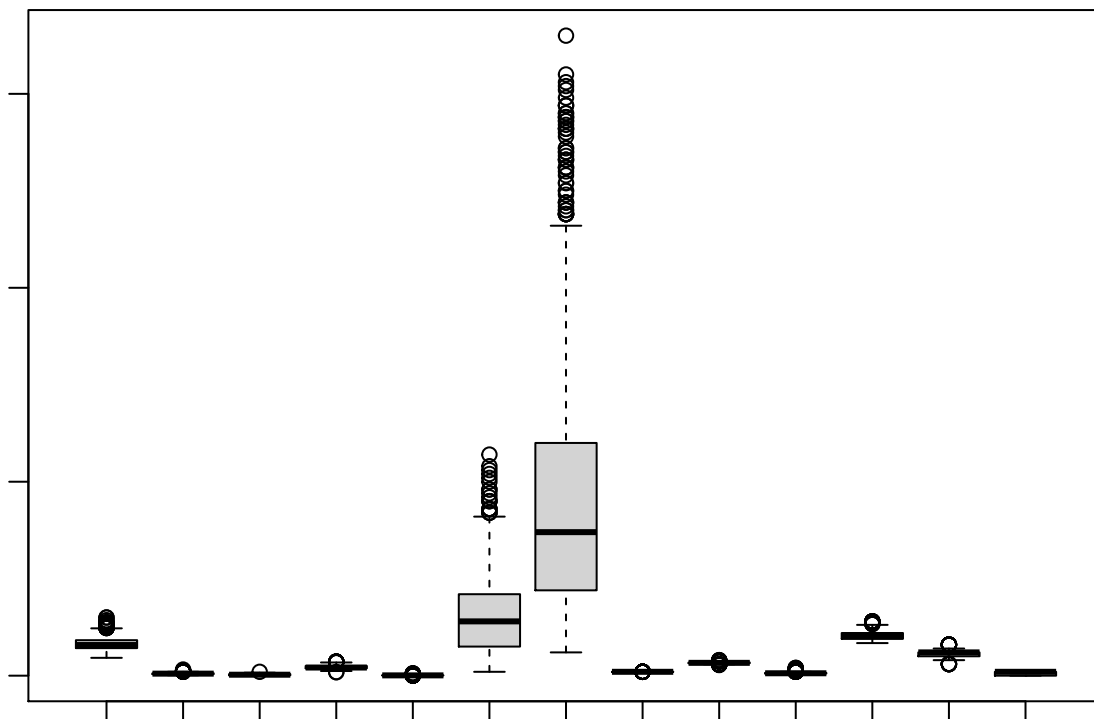
```
Q <- quantile(wine$residual.sugar, probs=c(.25, .75), na.rm = FALSE)
```

```
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]+1.5*iqr_rs))
boxplot(eliminated_rs)
```



#Removing outliers for free.sulfur.dioxide:

```
Q2 <- quantile(wine$free.sulfur.dioxide, probs=c(.25, .75), na.rm = FALSE)
```

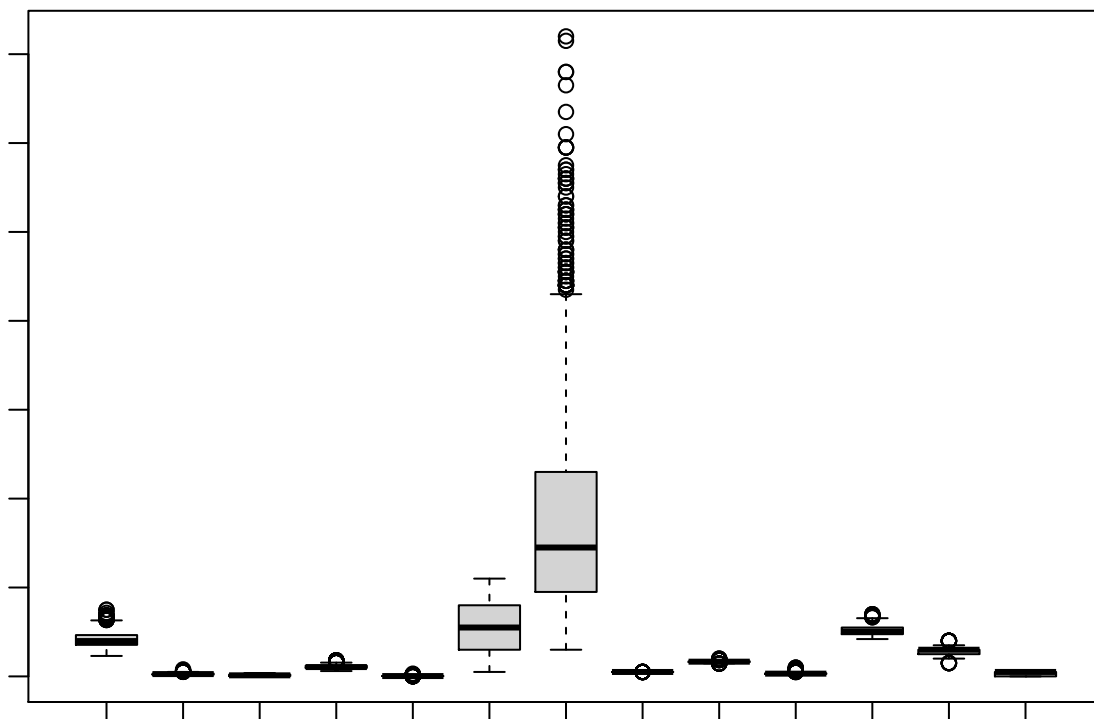
```
iqr_fs <- IQR(eliminated_rs$free.sulfur.dioxide)
```

```
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.dioxide < (Q[2] + 1.5*iqr_fs))
```

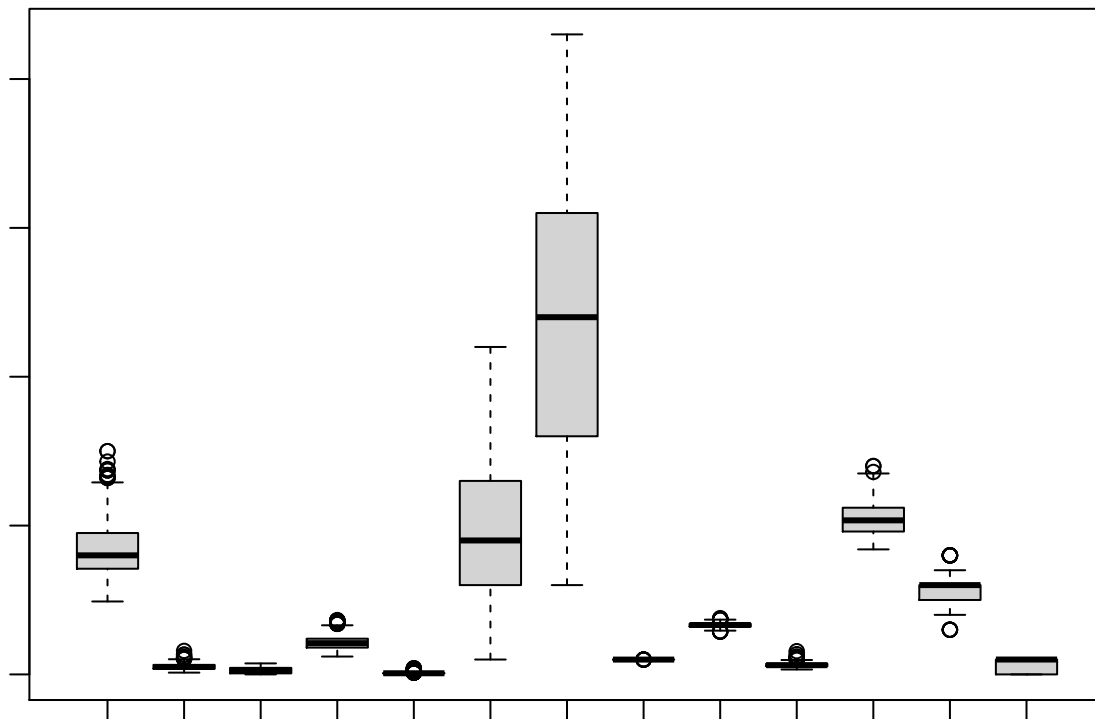
```
boxplot(eliminated_fs)
```



```

#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_ts < up_ts)
boxplot(eliminated_ts)

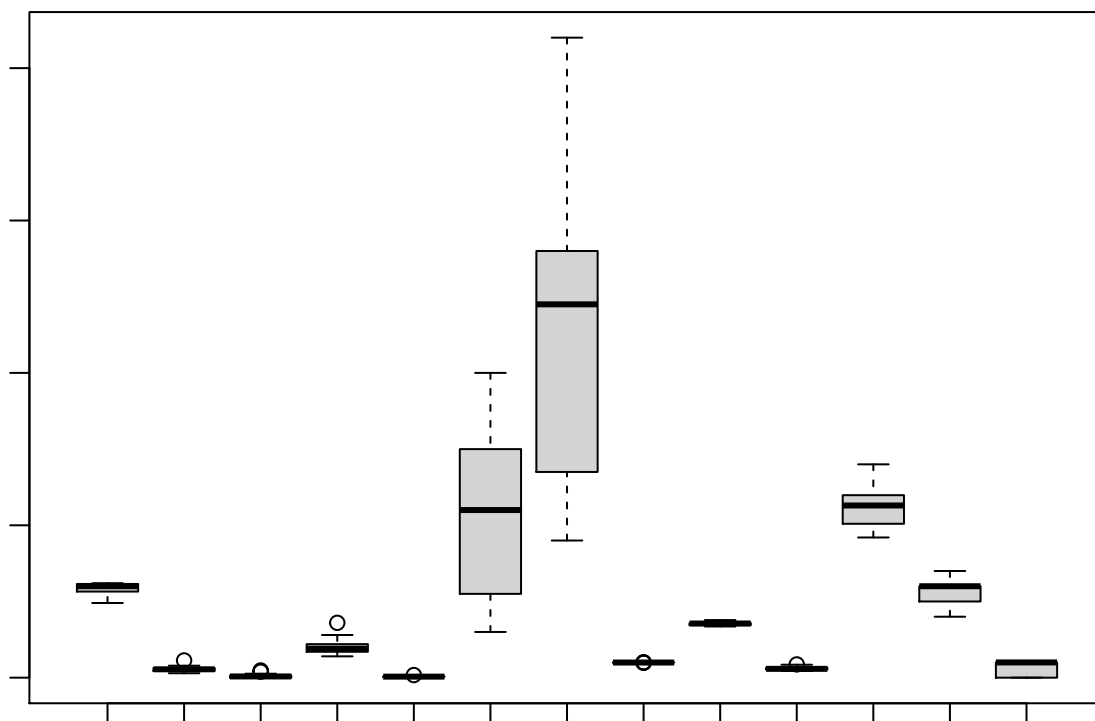
```



```

#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 < up_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
```

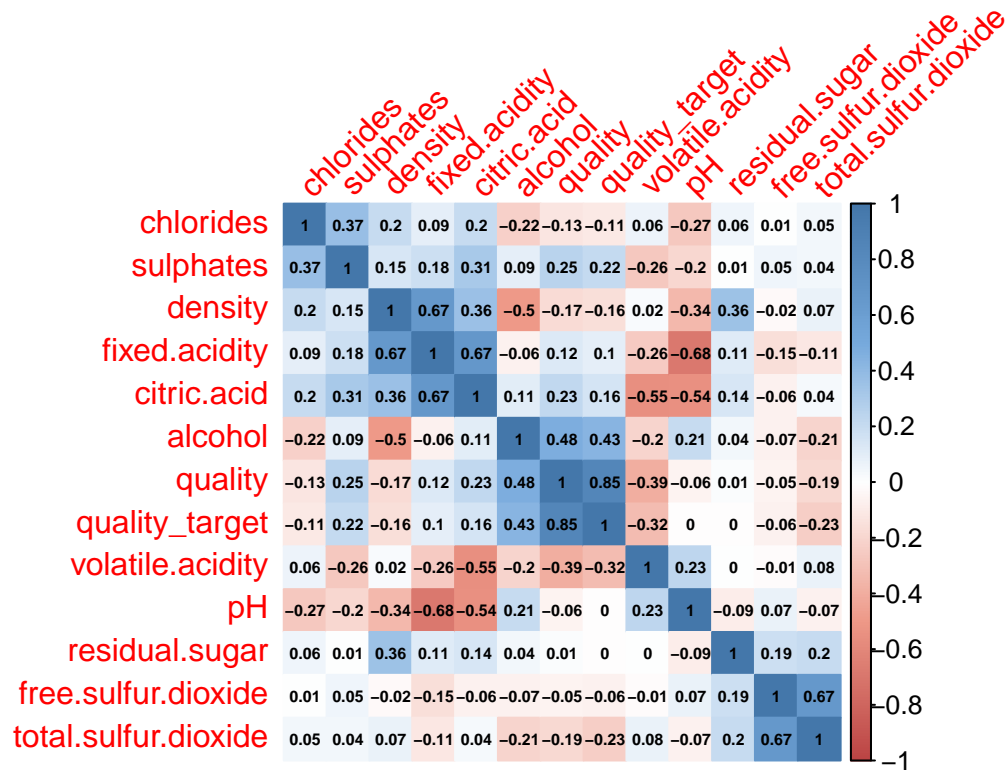
```
# Correlation Matrix
```

```
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))
```



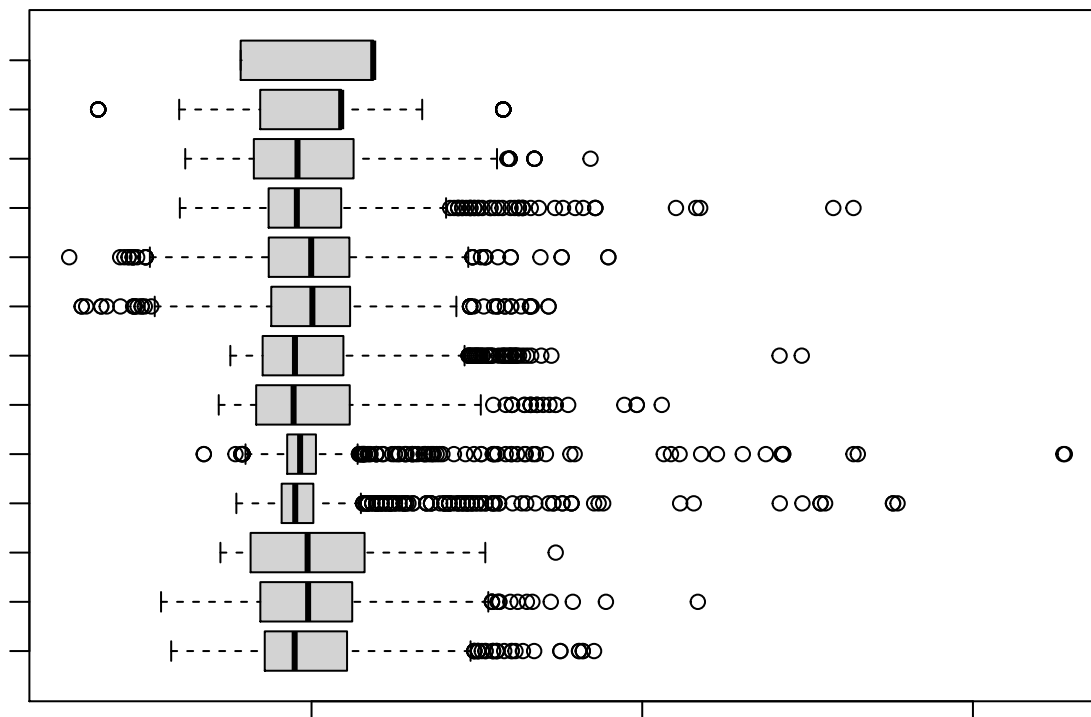
```
# Cutoff Correlation features
cutoffCorr <- findCorrelation(cor, cutoff = .8)
cutoffCorrFeatures <- wine[, -cutoffCorr]

# 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"
```

```
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
```

```
## -1.49852 -0.35196 -0.00665  0.38054  1.03178
```

```
##
```

```
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   -1.28481    0.14349  -8.954 < 2e-16 ***
```

```
## volatile.acidity -0.57138    0.07048  -8.107 1.20e-15 ***
```

```
## sulphates      0.44466    0.07604   5.847 6.33e-09 ***
```

```
## alcohol          0.17554    0.01157  15.170 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4289 on 1277 degrees of freedom
## Multiple R-squared:  0.263, Adjusted R-squared:  0.2613
## F-statistic: 151.9 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
## -2.73657 -0.38252 -0.05755  0.45741  2.17094
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.52555    0.22310   11.320 < 2e-16 ***
## volatile.acidity -1.18982    0.10958  -10.858 < 2e-16 ***
## sulphates       0.78130    0.11824    6.608 5.71e-11 ***
## alcohol         0.30936    0.01799   17.194 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6668 on 1277 degrees of freedom
## Multiple R-squared:  0.3345, Adjusted R-squared:  0.333
## F-statistic: 214 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.      :4.642   Min.    :3.330   Min.    :5.955
## 1st Qu.:5.284   1st Qu.:3.975   1st Qu.:6.593
## Median :5.522   Median :4.210   Median :6.832
## Mean    :5.645   Mean    :4.335   Mean    :6.956
## 3rd Qu.:5.971   3rd Qu.:4.662   3rd Qu.:7.281
## Max.    :6.990   Max.    :5.676   Max.    :8.305

# 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)

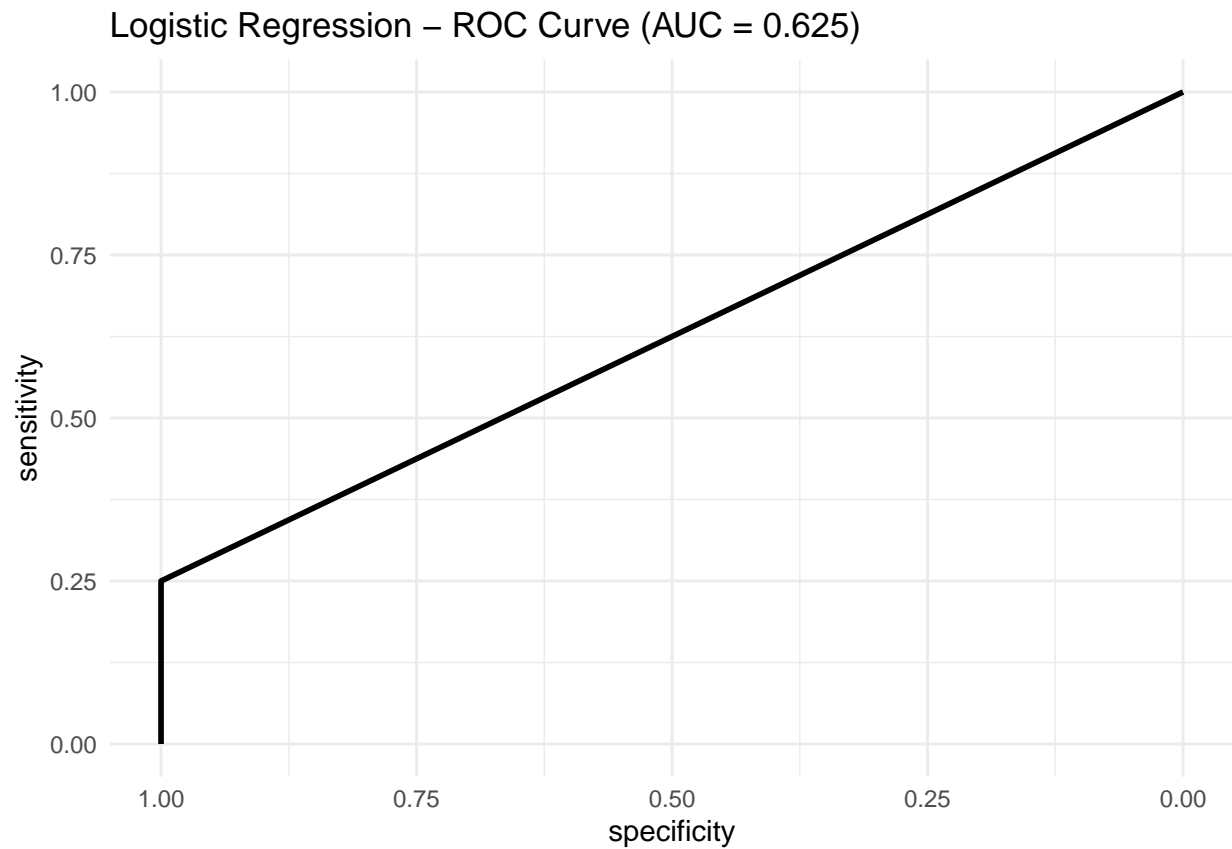
## Confusion Matrix and Statistics
##
##
##      5  7  6  4  8  3
## 5 97  2 40  0  0  0
## 7  2  8 31  0  0  0
## 6 47  6 74  0  0  0
## 4  6  0  2  0  0  0
## 8  0  1  1  0  0  0
## 3  1  0  0  0  0  0
##
## Overall Statistics
##
##              Accuracy : 0.5629
##              95% CI : (0.5064, 0.6182)
##      No Information Rate : 0.4811
##      P-Value [Acc > NIR] : 0.002103
##
##              Kappa : 0.2677
##
##  McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##              Class: 5 Class: 7 Class: 6 Class: 4 Class: 8 Class: 3
## Sensitivity      0.6340  0.47059  0.5000      NA      NA      NA
## Specificity      0.7455  0.89037  0.6882  0.97484  0.993711  0.996855
## Pos Pred Value   0.6978  0.19512  0.5827      NA      NA      NA
## Neg Pred Value   0.6872  0.96751  0.6126      NA      NA      NA
## Prevalence       0.4811  0.05346  0.4654  0.00000  0.000000  0.000000
## Detection Rate   0.3050  0.02516  0.2327  0.00000  0.000000  0.000000
## Detection Prevalence 0.4371  0.12893  0.3994  0.02516  0.006289  0.003145
## Balanced Accuracy 0.6897  0.68048  0.5941      NA      NA      NA

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

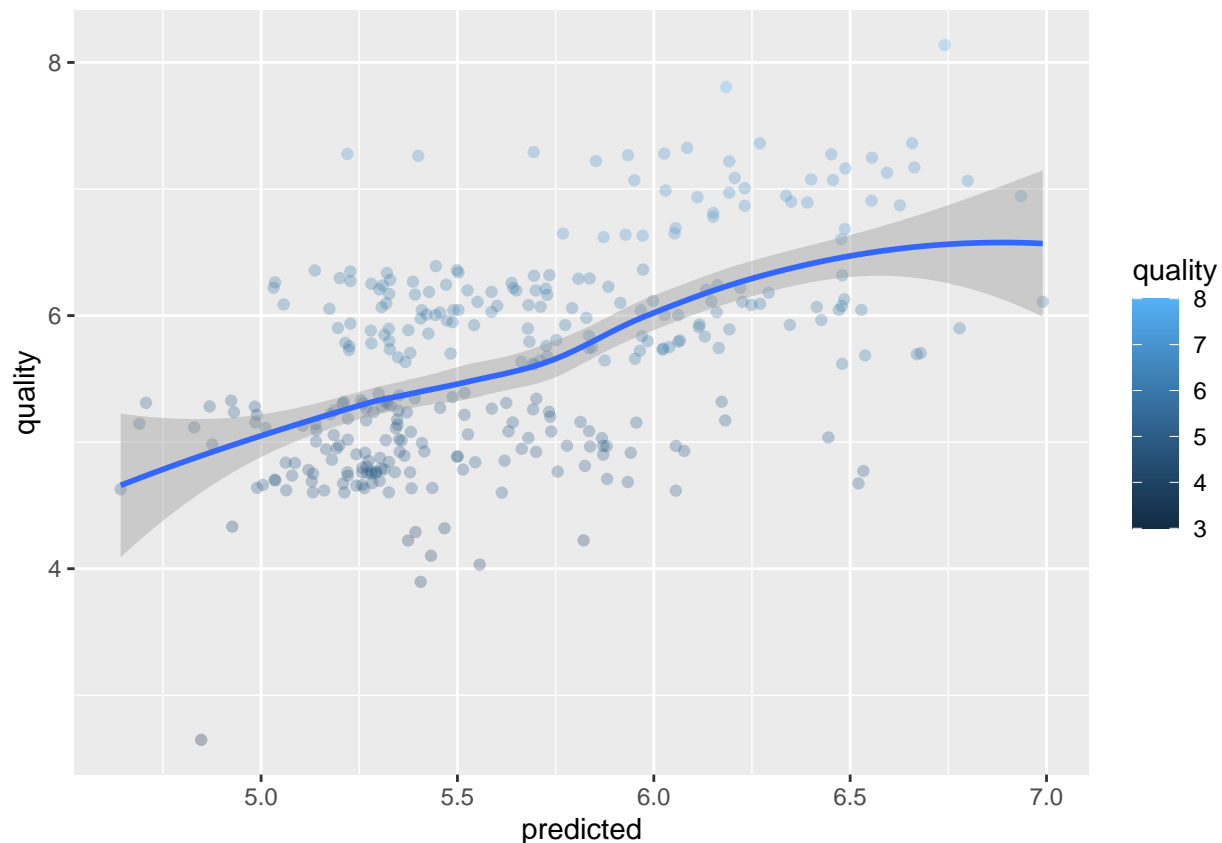
modelName1 <- 'Logistic Regression'
roc1 <- roc(df$quality, df$predicted_int)
auc1 <- round(auc(df$quality, df$predicted_int), 4)

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

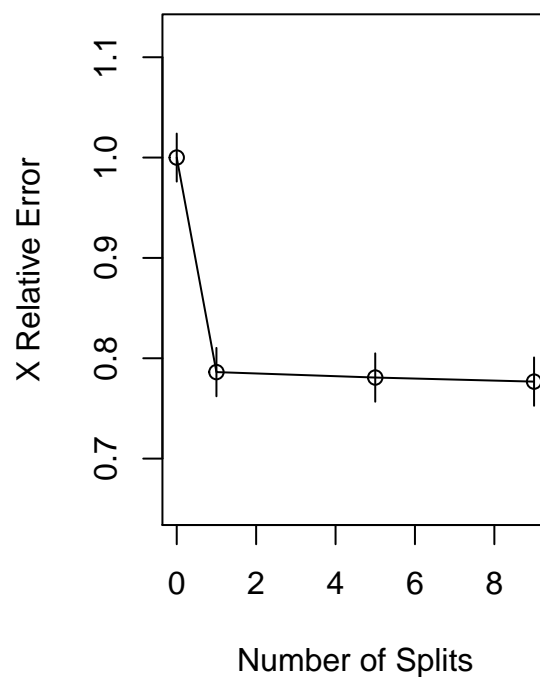
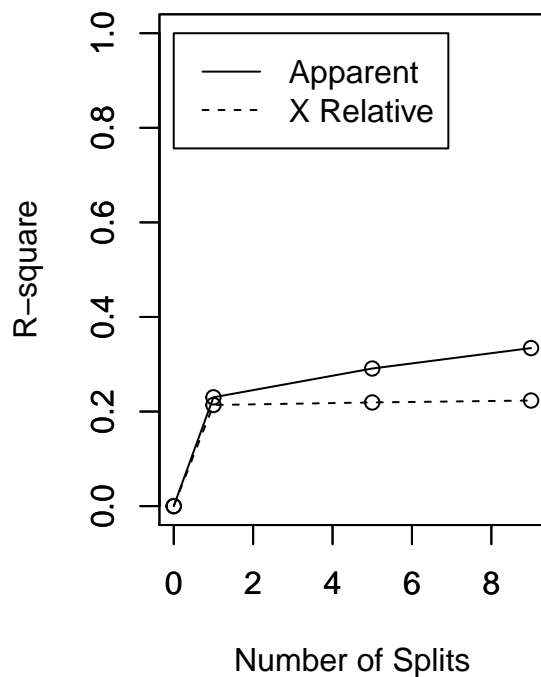
# Convert target variable to factor to ensure proper interpretation by model
rf_wine_train$quality <- as.factor(rf_wine_train$quality)

# 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          fixed.acidity      sulphates
## [4] total.sulfur.dioxide volatile.acidity
##
## Root node error: 739/1281 = 0.57689
##
## n= 1281
##
##      CP nsplit rel error  xerror   xstd
## 1 0.230041      0  1.00000 1.00000 0.023928
## 2 0.014434      1  0.76996 0.78620 0.024111
## 3 0.010374      5  0.70907 0.78078 0.024097
## 4 0.010000      9  0.66576 0.77673 0.024085
```

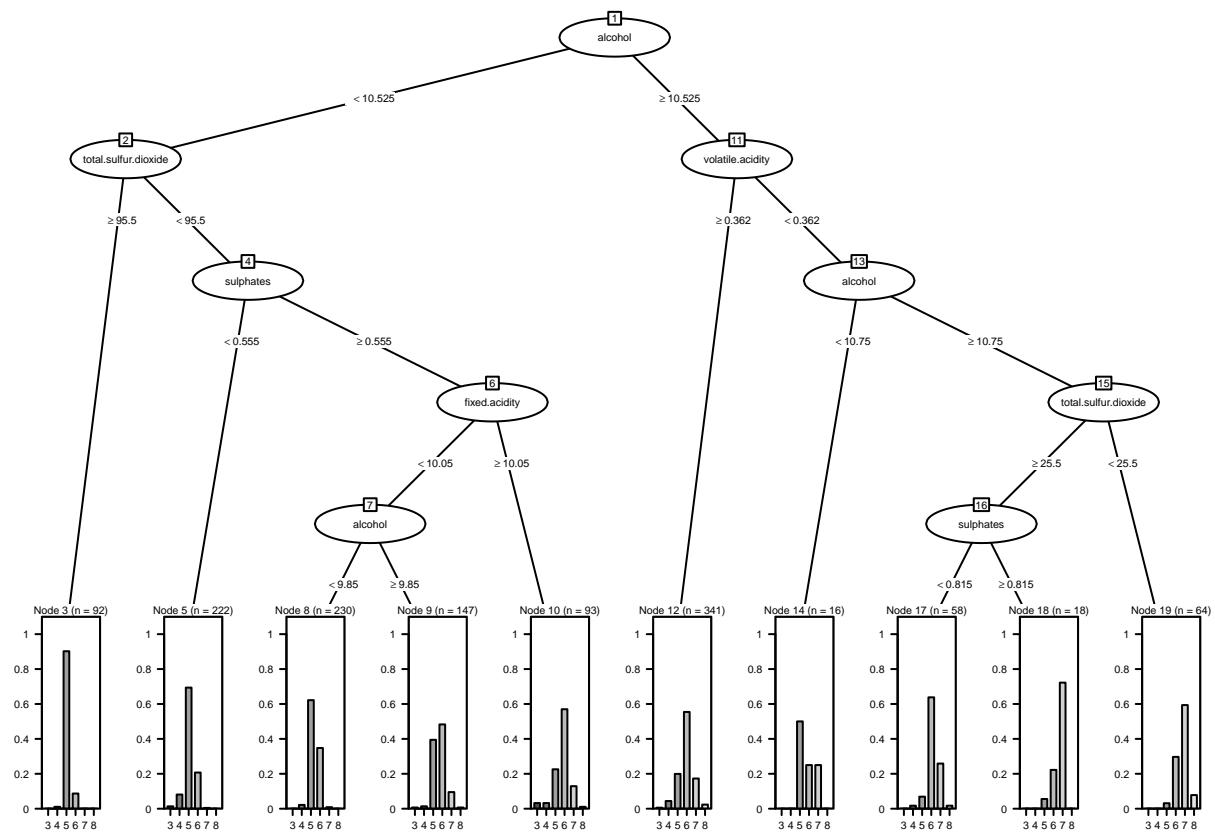


```
# Results
rpartTree2
```

```
##
## 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 < 10.525
## |   |   [3] total.sulfur.dioxide >= 95.5: 5 (n = 92, err = 9.8%)
## |   |   [4] total.sulfur.dioxide < 95.5
## |   |   |   [5] sulphates < 0.555: 5 (n = 222, err = 30.6%)
```

```
## |   |   |   [6] sulphates >= 0.555
## |   |   |   [7] fixed.acidity < 10.05
## |   |   |   [8] alcohol < 9.85: 5 (n = 230, err = 37.8%)
## |   |   |   [9] alcohol >= 9.85: 6 (n = 147, err = 51.7%)
## |   |   |   [10] fixed.acidity >= 10.05: 6 (n = 93, err = 43.0%)
## |   [11] alcohol >= 10.525
## |   |   [12] volatile.acidity >= 0.3625: 6 (n = 341, err = 44.6%)
## |   |   [13] volatile.acidity < 0.3625
## |   |   [14] alcohol < 10.75: 5 (n = 16, err = 50.0%)
## |   |   [15] alcohol >= 10.75
## |   |   [16] total.sulfur.dioxide >= 25.5
## |   |   [17] sulphates < 0.815: 6 (n = 58, err = 36.2%)
## |   |   [18] sulphates >= 0.815: 7 (n = 18, err = 27.8%)
## |   |   [19] total.sulfur.dioxide < 25.5: 7 (n = 64, err = 40.6%)
##
## Number of inner nodes:      9
## Number of terminal nodes: 10
```

```
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 0 143 154 21 0
```

```
# Confusion Matrix
```

```
confusionMatrix(table(df2$quality, df2$predicted))
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##
```

```
##      3  4  5  6  7  8
```

```
## 3  0  0  0  1  0  0
```

```
## 4  0  0  2  6  0  0
```

```
## 5  0  0 100 37  2  0
```

```
## 6  0  0 38 83  6  0
```

```
## 7  0  0  3 26 12  0
```

```
## 8  0  0  0  1  1  0
```

```
##
```

```
## Overall Statistics
```

```
##
```

```
##              Accuracy : 0.6132
```

```
##              95% CI : (0.5573, 0.667)
```

```
##      No Information Rate : 0.4843
```

```
##      P-Value [Acc > NIR] : 2.6e-06
```

```
##
```

```
##              Kappa : 0.357
```

```
##
```

```
## McNemar's Test P-Value : NA
```

```
##
```

```
## Statistics by Class:
```

```
##
```

```
##              Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
```

```
## Sensitivity          NA          NA  0.6993  0.5390  0.57143          NA
```

```
## Specificity          0.996855  0.97484  0.7771  0.7317  0.90236 0.993711
```

```
## Pos Pred Value          NA          NA  0.7194  0.6535  0.29268          NA
```

```
## Neg Pred Value          NA          NA  0.7598  0.6283  0.96751          NA
```

```
## Prevalence            0.000000  0.00000  0.4497  0.4843  0.06604 0.000000
```

```
## Detection Rate            0.000000  0.00000  0.3145  0.2610  0.03774 0.000000
```

```
## Detection Prevalence 0.003145  0.02516  0.4371  0.3994  0.12893 0.006289
```

```
## Balanced Accuracy          NA          NA  0.7382  0.6353  0.73689          NA
```

```
# ROC plot
```

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

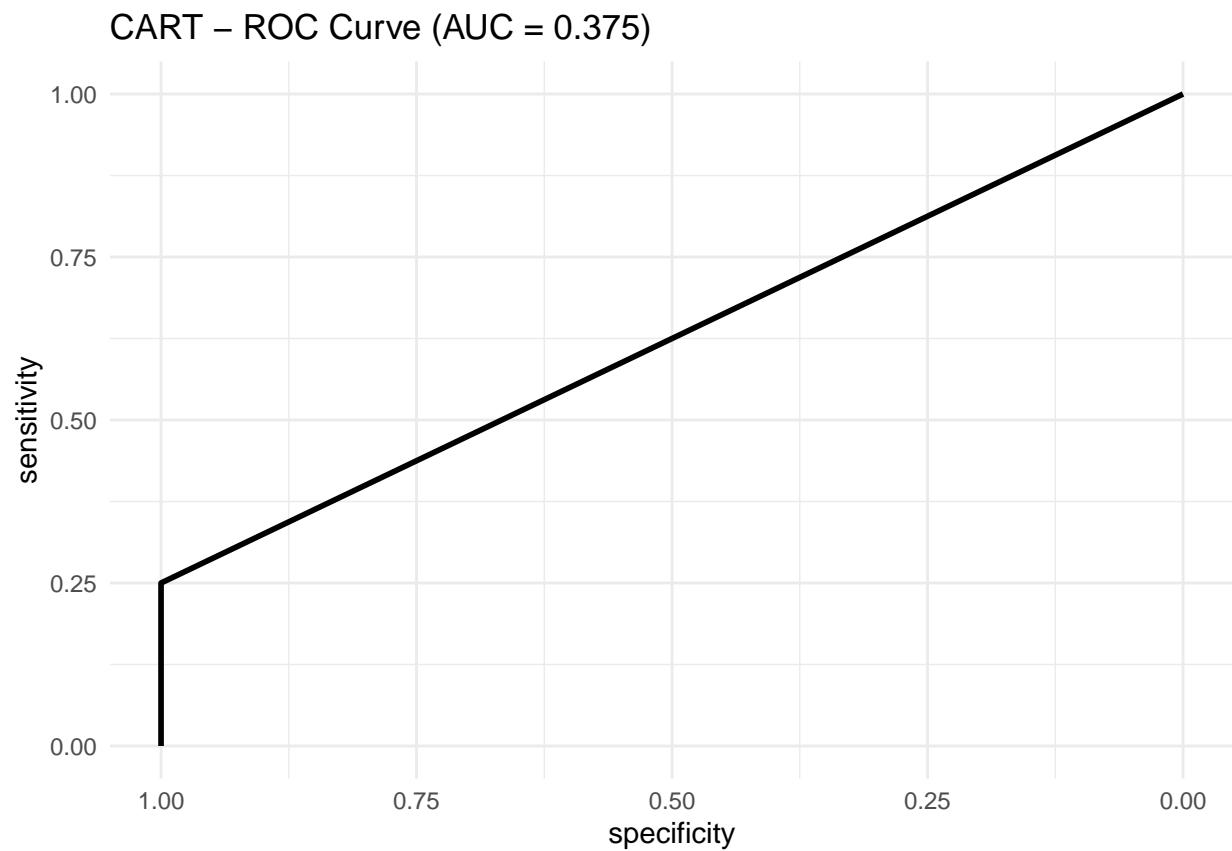
```
modelName2 <- 'CART'
```

```
roc2 <- roc(df2$quality, df2$predicted_int)
```

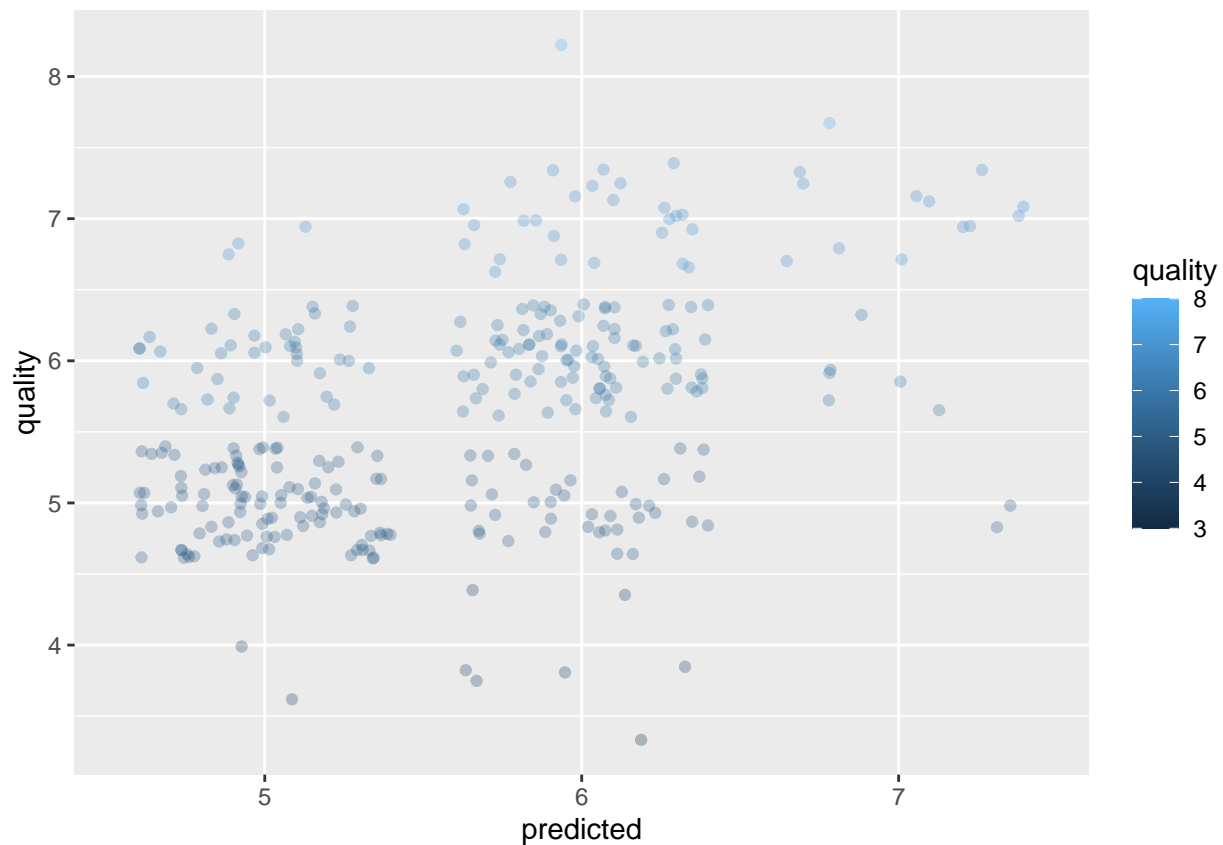
```
auc2 <- round(auc(df2$quality, df2$predicted_int), 4)
```

```
ggroc(roc1, colours = 'red', size = 1) +
```

```
  ggtitle(paste0(modelName2, ' - ROC Curve ', '(AUC = ', auc2, ', ')') + theme_minimal())
```

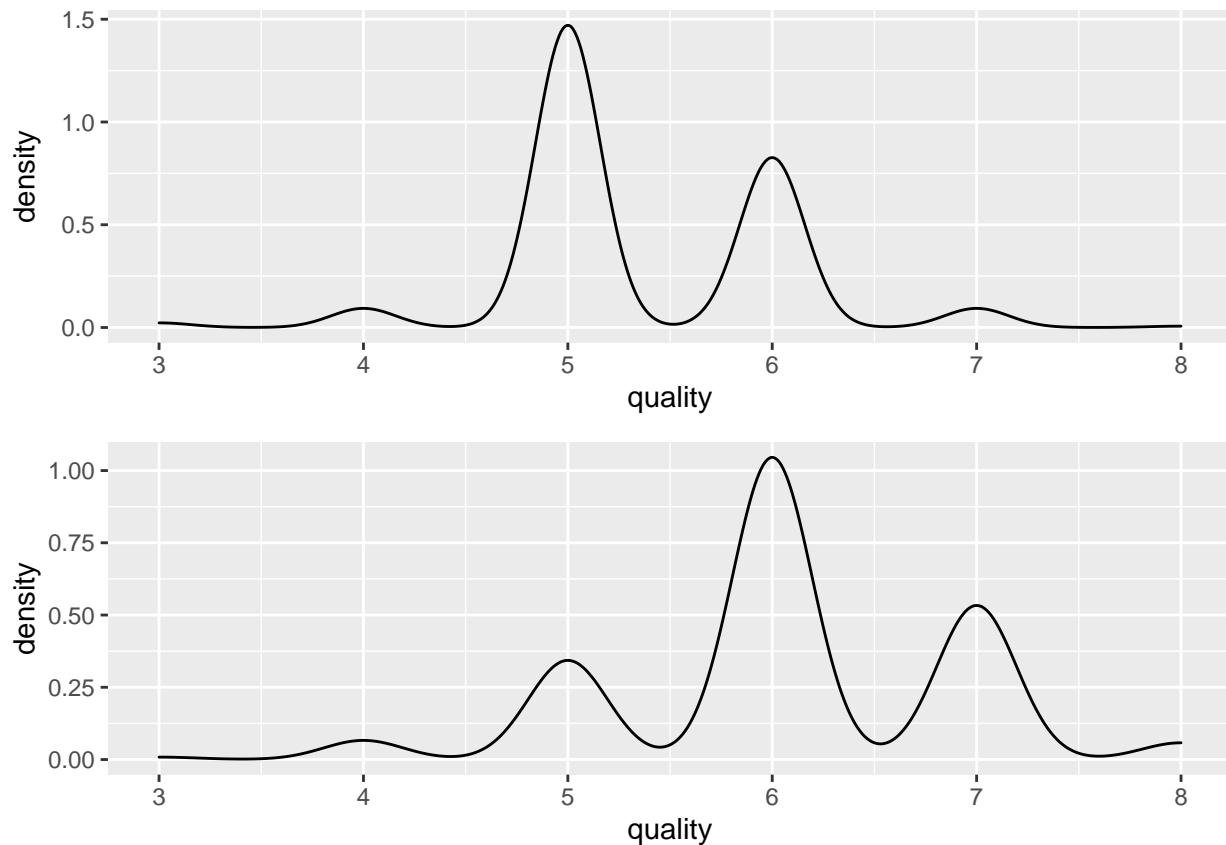
```
# Scatter plot of predicted  
ggplot(df2, aes(x = predicted, y = quality, colour = quality ))+  
geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```



```
# 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: 9, 45, 542, 511, 158, 16
##                   Number of trees: 500
##           Forest terminal node size: 1
##           Average no. of terminal nodes: 254.034
## 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: 0.0694003
##           (OOB) Normalized Brier score: 0.49968218
##                               (OOB) AUC: 0.81564077
##           (OOB) Requested performance error: 0.30288837, 1, 0.97777778, 0.18819188, 0.2778865, 0.48734177, 0.68734177, 0.88734177, 1.08734177, 1.28734177, 1.48734177, 1.68734177, 1.88734177, 2.08734177, 2.28734177, 2.48734177, 2.68734177, 2.88734177, 3.08734177, 3.28734177, 3.48734177, 3.68734177, 3.88734177, 4.08734177, 4.28734177, 4.48734177, 4.68734177, 4.88734177, 5.08734177, 5.28734177, 5.48734177, 5.68734177, 5.88734177, 6.08734177, 6.28734177, 6.48734177, 6.68734177, 6.88734177, 7.08734177, 7.28734177, 7.48734177, 7.68734177, 7.88734177, 8.08734177, 8.28734177, 8.48734177, 8.68734177, 8.88734177, 9.08734177, 9.28734177, 9.48734177, 9.68734177, 9.88734177, 10.08734177, 10.28734177, 10.48734177, 10.68734177, 10.88734177, 11.08734177, 11.28734177, 11.48734177, 11.68734177, 11.88734177, 12.08734177, 12.28734177, 12.48734177, 12.68734177, 12.88734177, 13.08734177, 13.28734177, 13.48734177, 13.68734177, 13.88734177, 14.08734177, 14.28734177, 14.48734177, 14.68734177, 14.88734177, 15.08734177, 15.28734177, 15.48734177, 15.68734177, 15.88734177, 16.08734177, 16.28734177, 16.48734177, 16.68734177, 16.88734177, 17.08734177, 17.28734177, 17.48734177, 17.68734177, 17.88734177, 18.08734177, 18.28734177, 18.48734177, 18.68734177, 18.88734177, 19.08734177, 19.28734177, 19.48734177, 19.68734177, 19.88734177, 20.08734177, 20.28734177, 20.48734177, 20.68734177, 20.88734177, 21.08734177, 21.28734177, 21.48734177, 21.68734177, 21.88734177, 22.08734177, 22.28734177, 22.48734177, 22.68734177, 22.88734177, 23.08734177, 23.28734177, 23.48734177, 23.68734177, 23.88734177, 24.08734177, 24.28734177, 24.48734177, 24.68734177, 24.88734177, 25.08734177, 25.28734177, 25.48734177, 25.68734177, 25.88734177, 26.08734177, 26.28734177, 26.48734177, 26.68734177, 26.88734177, 27.08734177, 27.28734177, 27.48734177, 27.68734177, 27.88734177, 28.08734177, 28.28734177, 28.48734177, 28.68734177, 28.88734177, 29.08734177, 29.28734177, 29.48734177, 29.68734177, 29.88734177, 30.08734177, 30.28734177, 30.48734177, 30.68734177, 30.88734177, 31.08734177, 31.28734177, 31.48734177, 31.68734177, 31.88734177, 32.08734177, 32.28734177, 32.48734177, 32.68734177, 32.88734177, 33.08734177, 33.28734177, 33.48734177, 33.68734177, 33.88734177, 34.08734177, 34.28734177, 34.48734177, 34.68734177, 34.88734177, 35.08734177, 35.28734177, 35.48734177, 35.68734177, 35.88734177, 36.08734177, 36.28734177, 36.48734177, 36.68734177, 36.88734177, 37.08734177, 37.28734177, 37.48734177, 37.68734177, 37.88734177, 38.08734177, 38.28734177, 38.48734177, 38.68734177, 38.88734177, 39.08734177, 39.28734177, 39.48734177, 39.68734177, 39.88734177, 40.08734177, 40.28734177, 40.48734177, 40.68734177, 40.88734177, 41.08734177, 41.28734177, 41.48734177, 41.68734177, 41.88734177, 42.08734177, 42.28734177, 42.48734177, 42.68734177, 42.88734177, 43.08734177, 43.28734177, 43.48734177, 43.68734177, 43.88734177, 44.08734177, 44.28734177, 44.48734177, 44.68734177, 44.88734177, 45.08734177, 45.28734177, 45.48734177, 45.68734177, 45.88734177, 46.08734177, 46.28734177, 46.48734177, 46.68734177, 46.88734177, 47.08734177, 47.28734177, 47.48734177, 47.68734177, 47.88734177, 48.08734177, 48.28734177, 48.48734177, 48.68734177, 48.88734177, 49.08734177, 49.28734177, 49.48734177, 49.68734177, 49.88734177, 50.08734177, 50.28734177, 50.48734177, 50.68734177, 50.88734177, 51.08734177, 51.28734177, 51.48734177, 51.68734177, 51.88734177, 52.08734177, 52.28734177, 52.48734177, 52.68734177, 52.88734177, 53.08734177, 53.28734177, 53.48734177, 53.68734177, 53.88734177, 54.08734177, 54.28734177, 54.48734177, 54.68734177, 54.88734177, 55.08734177, 55.28734177, 55.48734177, 55.68734177, 55.88734177, 56.08734177, 56.28734177, 56.48734177, 56.68734177, 56.88734177, 57.08734177, 57.28734177, 57.48734177, 57.68734177, 57.88734177, 58.08734177, 58.28734177, 58.48734177, 58.68734177, 58.88734177, 59.08734177, 59.28734177, 59.48734177, 59.68734177, 59.88734177, 60.08734177, 60.28734177, 60.48734177, 60.68734177, 60.88734177, 61.08734177, 61.28734177, 61.48734177, 61.68734177, 61.88734177, 62.08734177, 62.28734177, 62.48734177, 62.68734177, 62.88734177, 63.08734177, 63.28734177, 63.48734177, 63.68734177, 63.88734177, 64.08734177, 64.28734177, 64.48734177, 64.68734177, 64.88734177, 65.08734177, 65.28734177, 65.48734177, 65.68734177, 65.88734177, 66.08734177, 66.28734177, 66.48734177, 66.68734177, 66.88734177, 67.08734177, 67.28734177, 67.48734177, 67.68734177, 67.88734177, 68.08734177, 68.28734177, 68.48734177, 68.68734177, 68.88734177, 69.08734177, 69.28734177, 69.48734177, 69.68734177, 69.88734177, 70.08734177, 70.28734177, 70.48734177, 70.68734177, 70.88734177, 71.08734177, 71.28734177, 71.48734177, 71.68734177, 71.88734177, 72.08734177, 72.28734177, 72.48734177, 72.68734177, 72.88734177, 73.08734177, 73.28734177, 73.48734177, 73.68734177, 73.88734177, 74.08734177, 74.28734177, 74.48734177, 74.68734177, 74.88734177, 75.08734177, 75.28734177, 75.48734177, 75.68734177, 75.88734177, 76.08734177, 76.28734177, 76.48734177, 76.68734177, 76.88734177, 77.08734177, 77.28734177, 77.48734177, 77.68734177, 77.88734177, 78.08734177, 78.28734177, 78.48734177, 78.68734177, 78.88734177, 79.08734177, 79.28734177, 79.48734177, 79.68734177, 79.88734177, 80.08734177, 80.28734177, 80.48734177, 80.68734177, 80.88734177, 81.08734177, 81.28734177, 81.48734177, 81.68734177, 81.88734177, 82.08734177, 82.28734177, 82.48734177, 82.68734177, 82.88734177, 83.08734177, 83.28734177, 83.48734177, 83.68734177, 83.88734177, 84.08734177, 84.28734177, 84.48734177, 84.68734177, 84.88734177, 85.08734177, 85.28734177, 85.48734177, 85.68734177, 85.88734177, 86.08734177, 86.28734177, 86.48734177, 86.68734177, 86.88734177, 87.08734177, 87.28734177, 87.48734177, 87.68734177, 87.88734177, 88.08734177, 88.28734177, 88.48734177, 88.68734177, 88.88734177, 89.08734177, 89.28734177, 89.48734177, 89.68734177, 89.88734177, 90.08734177, 90.28734177, 90.48734177, 90.68734177, 90.88734177, 91.08734177, 91.28734177, 91.48734177, 91.68734177, 91.88734177, 92.08734177, 92.28734177, 92.48734177, 92.68734177, 92.88734177, 93.08734177, 93.28734177, 93.48734177, 93.68734177, 93.88734177, 94.08734177, 94.28734177, 94.48734177, 94.68734177, 94.88734177, 95.08734177, 95.28734177, 95.48734177, 95.68734177, 95.88734177, 96.08734177, 96.28734177, 96.48734177, 96.68734177, 96.88734177, 97.08734177, 97.28734177, 97.48734177, 97.68734177, 97.88734177, 98.08734177, 98.28734177, 98.48734177, 98.68734177, 98.88734177, 99.08734177, 99.28734177, 99.48734177, 99.68734177, 99.88734177, 100.08734177, 100.28734177, 100.48734177, 100.68734177, 100.88734177, 101.08734177, 101.28734177, 101.48734177, 101.68734177, 101.88734177, 102.08734177, 102.28734177, 102.48734177, 102.68734177, 102.88734177, 103.08734177, 103.28734177, 103.48734177, 103.68734177, 103.88734177, 104.08734177, 104.28734177, 104.48734177, 104.68734177, 104.88734177, 105.08734177, 105.28734177, 105.48734177, 105.68734177, 105.88734177, 106.08734177, 106.28734177, 106.48734177, 106.68734177, 106.88734177, 107.08734177, 107.28734177, 107.48734177, 107.68734177, 107.88734177, 108.08734177, 108.28734177, 108.48734177, 108.68734177, 108.88734177, 109.08734177, 109.28734177, 109.48734177, 109.68734177, 109.88734177, 110.08734177, 110.28734177, 110.48734177, 110.68734177, 110.88734177, 111.08734177, 111.28734177, 111.48734177, 111.68734177, 111.88734177, 112.08734177, 112.28734177, 112.48734177, 112.68734177, 112.88734177, 113.08734177, 113.28734177, 113.48734177, 113.68734177, 113.88734177, 114.08734177, 114.28734177, 114.48734177, 114.68734177, 114.88734177, 115.08734177, 115.28734177, 115.48734177, 115.68734177, 115.88734177, 116.08734177, 116.28734177, 116.48734177, 116.68734177, 116.88734177, 117.08734177, 117.28734177, 117.48734177, 117.68734177, 117.88734177, 118.08734177, 118.28734177, 118.48734177, 118.68734177, 118.88734177, 119.08734177, 119.28734177, 119.48734177, 119.68734177, 119.88734177, 120.08734177, 120.28734177, 120.48734177, 120.68734177, 120.88734177, 121.08734177, 121.28734177, 121.48734177, 121.68734177, 121.88734177, 122.08734177, 122.28734177, 122.48734177, 122.68734177, 122.88734177, 123.08734177, 123.28734177, 123.48734177, 123.68734177, 123.88734177, 124.08734177, 124.28734177, 124.48734177, 124.68734177, 124.88734177, 125.08734177, 125.28734177, 125.48734177, 125.68734177, 125.88734177, 126.08734177, 126.28734177, 126.48734177, 126.68734177, 126.88734177, 127.08734177, 127.28734177, 127.48734177, 127.68734177, 127.88734177, 128.08734177, 128.28734177, 128.48734177, 128.68734177, 128.88734177, 129.08734177, 129.28734177, 129.48734177, 129.68734177, 129.88734177, 130.08734177, 130.28734177, 130.48734177, 130.68734177, 130.88734177, 131.08734177, 131.28734177, 131.48734177, 131.68734177, 131.88734177, 132.08734177, 132.28734177, 132.48734177, 132.68734177, 132.88734177, 133.08734177, 133.28734177, 133.48734177, 133.68734177, 133.88734177, 134.08734177, 134.28734177, 134.48734177, 134.68734177, 134.88734177, 135.08734177, 135.28734177, 135.48734177, 135.68734177, 135.88734177, 136.08734177, 136.28734177, 136.48734177, 136.68734177, 136.88734177, 137.08734177, 137.28734177, 137.48734177, 137.68734177, 137.88734177, 138.08734177, 138.28734177, 138.48734177, 138.68734177, 138.88734177, 139.08734177, 139.28734177, 139.48734177, 139.68734177, 139.88734177, 140.08734177, 140.28734177, 140.48734177, 140.68734177, 140.88734177, 141.08734177, 141.28734177, 141.48734177, 141.68734177, 141.88734177, 142.08734177, 142.28734177, 142.48734177, 142.68734177, 142.88734177, 143.08734177, 143.28734177, 143.48734177, 143.68734177, 143.88734177, 144.08734177, 144.28734177, 144.48734177, 144.68734177, 144.88734177, 145.08734177, 145.28734177, 145.48734177, 145.68734177, 145.88734177, 146.08734177, 146.28734177, 146.48734177, 146.68734177, 146.88734177, 147.08734177, 147.28734177, 147.48734177, 147.68734177, 147.88734177, 148.08734177, 148.28734177, 148.48734177, 148.68734177, 148.88734177, 149.08734177, 149.28734177, 149.48734177, 149.68734177, 149.88734177, 150.08734177, 150.28734177, 150.48734177, 150.68734177, 150.88734177, 151.08734177, 151.28734177, 151.48734177, 151.68734177, 151.88734177, 152.08734177, 152.28734177, 152.48734177, 152.68734177, 152.88734177, 153.08734177, 153.28734177, 153.48734177, 153.68734177, 153.88734177, 154.08734177, 154.28734177, 154.48734177, 154.68734177, 154.88734177, 155.08734177, 155.28734177, 155.48734177, 155.68734177, 155.88734177, 156.08734177, 156.28734177, 156.48734177, 156.68734177, 156.88734177, 157.08734177, 157.28734177, 157.48734177, 157.68734177, 157.88734177, 158.08734177, 158.28734177, 158.48734177, 158.68734177, 158.88734177, 159.08734177, 159.28734177, 159.48734177, 159.68734177, 159.88734177, 160.08734177, 160.28734177, 160.48734177, 160.68734177, 160.88734177, 161.08734177, 161.28734177, 161.48734177, 161.68734177, 161.88734177, 162.08734177, 162.28734177, 162.48734177, 162.68734177, 162.88734177, 163.08734177, 163.28734177, 163.48734177, 163.68734177, 163.88734177, 164.08734177, 164.28734177, 164.48734177, 164.68734177, 164.88734177, 165.08734177, 165.28734177, 165.48734177, 165.68734177, 165.88734177, 166.08734177, 166.28734177, 166.48734177, 166.68734177, 166.88734177, 167.08734177, 167.28734177, 167.48734177, 167.68734177, 167.88734177, 168.08734177, 168.28734177, 168.48734177, 168.68734177, 168.88734177, 169.08734177, 169.28734177, 169.48734177, 169.68734177, 169.88734177, 170.08734177, 170.28734177, 170.48734177, 170.68734177, 170.88734177, 171.08734177, 171.28734177, 171.48734177, 171.68734177, 171.88734177, 172.08734177, 172.28734177, 172.48734177, 172.68734177, 172.88734177, 173.08734177, 173.28734177, 173.48734177, 173.68734177, 173.88734177, 17
```

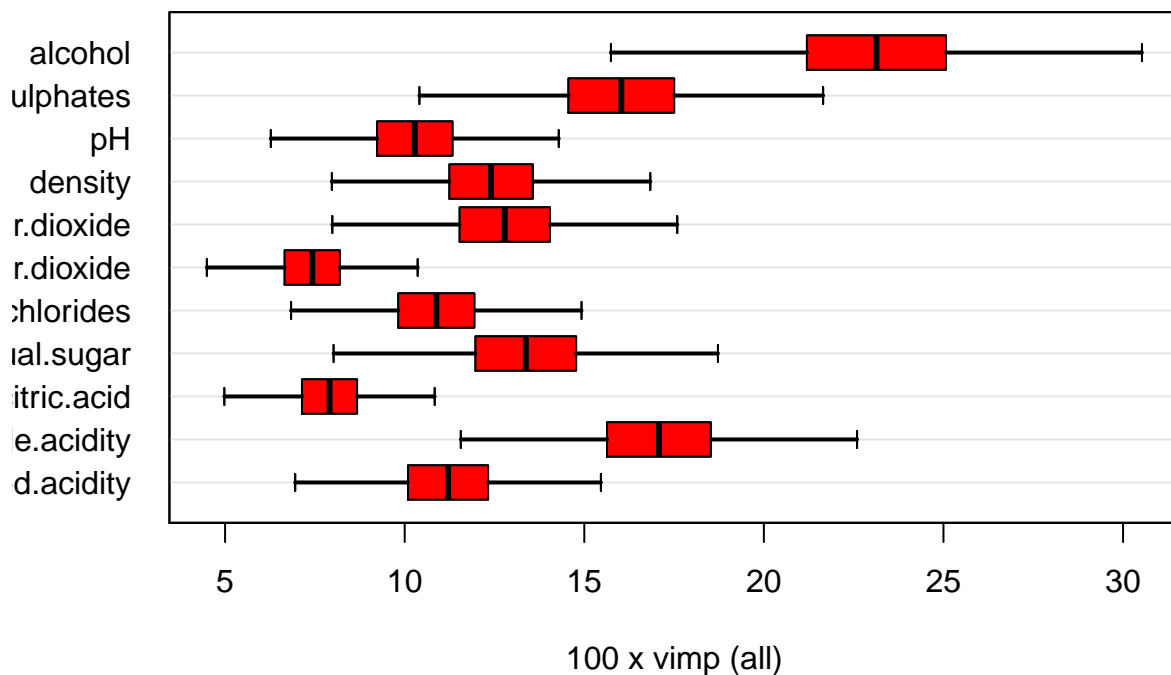
```
## Confusion matrix:
##
##      predicted
## observed 3 4  5  6  7  8 class.error
##      3 0 1  6  2  0  0      1.0000
##      4 0 1 27 16  1  0      0.9778
##      5 0 2 442 94  4  0      0.1845
##      6 0 2 105 370 34  0      0.2759
##      7 0 0  6  70 81  1      0.4873
##      8 0 0  0  9  5  2      0.8750
##
##      (OOB) Misclassification rate: 0.3005464
```

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

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

	lower	mean	upper	pvalue	signif
fixed.acidity	7.968889	11.207219	14.44555	5.883303e-12	TRUE
volatile.acidity	12.880739	17.078132	21.27552	7.643822e-16	TRUE
citric.acid	5.678804	7.907006	10.13521	1.761114e-12	TRUE
residual.sugar	9.299919	13.371125	17.44233	6.087257e-11	TRUE
chlorides	7.803877	10.880255	13.95663	2.077257e-12	TRUE
free.sulfur.dioxide	5.193364	7.426327	9.65929	3.553201e-11	TRUE
total.sulfur.dioxide	9.132848	12.786412	16.43998	3.459830e-12	TRUE
density	9.031748	12.405096	15.77844	2.848544e-13	TRUE
pH	7.231599	10.281104	13.33061	1.950054e-11	TRUE
sulphates	11.753794	16.029724	20.30565	1.009243e-13	TRUE
alcohol	17.509637	23.134183	28.75873	3.768791e-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: 254.034
## 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: 0.06901203
## Normalized Brier score: 0.4968866
## AUC: 0.84581745
## Requested performance error: 0.31132075, 1, 1, 0.1942446, 0.2992126, 0.56097561, 1
##
## Confusion matrix:
##
##      predicted
## observed 3 4 5 6 7 8 class.error
##      3 0 0 1 0 0 0 1.0000
##      4 0 0 6 2 0 0 1.0000
##      5 0 1 11 2 1 0 0.1942
##      6 0 0 36 8 2 0 0.2992
##      7 0 0 3 19 18 1 0.5610
##      8 0 0 0 1 1 0 1.0000
##
## Misclassification error: 0.3113208

```

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      0.6576191  0.3546296  0.5091676
##   2      0.6570003  0.3558905  0.5088165
##   3      0.6569815  0.3559688  0.5079349
##
## 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.
##   4.579  5.255   5.542   5.636   6.032   6.822

# 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
## 5 102   1  36   0   0   0
## 7   2  11  28   0   0   0
## 6  42   5  80   0   0   0
## 4   5   0   3   0   0   0
## 8   0   1   1   0   0   0
## 3   1   0   0   0   0   0
##
## Overall Statistics
##
##               Accuracy : 0.6069
##               95% CI : (0.5509, 0.661)
##   No Information Rate : 0.478
##   P-Value [Acc > NIR] : 2.631e-06
##
##               Kappa : 0.3426
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##               Class: 5 Class: 7 Class: 6 Class: 4 Class: 8 Class: 3
## Sensitivity      0.6711  0.61111  0.5405      NA      NA      NA

```

```
## Specificity          0.7771  0.90000  0.7235  0.97484 0.993711 0.996855
## Pos Pred Value      0.7338  0.26829  0.6299      NA      NA      NA
## Neg Pred Value      0.7207  0.97473  0.6440      NA      NA      NA
## Prevalence          0.4780  0.05660  0.4654  0.00000 0.000000 0.000000
## Detection Rate      0.3208  0.03459  0.2516  0.00000 0.000000 0.000000
## Detection Prevalence 0.4371  0.12893  0.3994  0.02516 0.006289 0.003145
## Balanced Accuracy    0.7241  0.75556  0.6320      NA      NA      NA
```

```
# ROC plot
```

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

```
modelName3 <- 'Partial Least Squares'
```

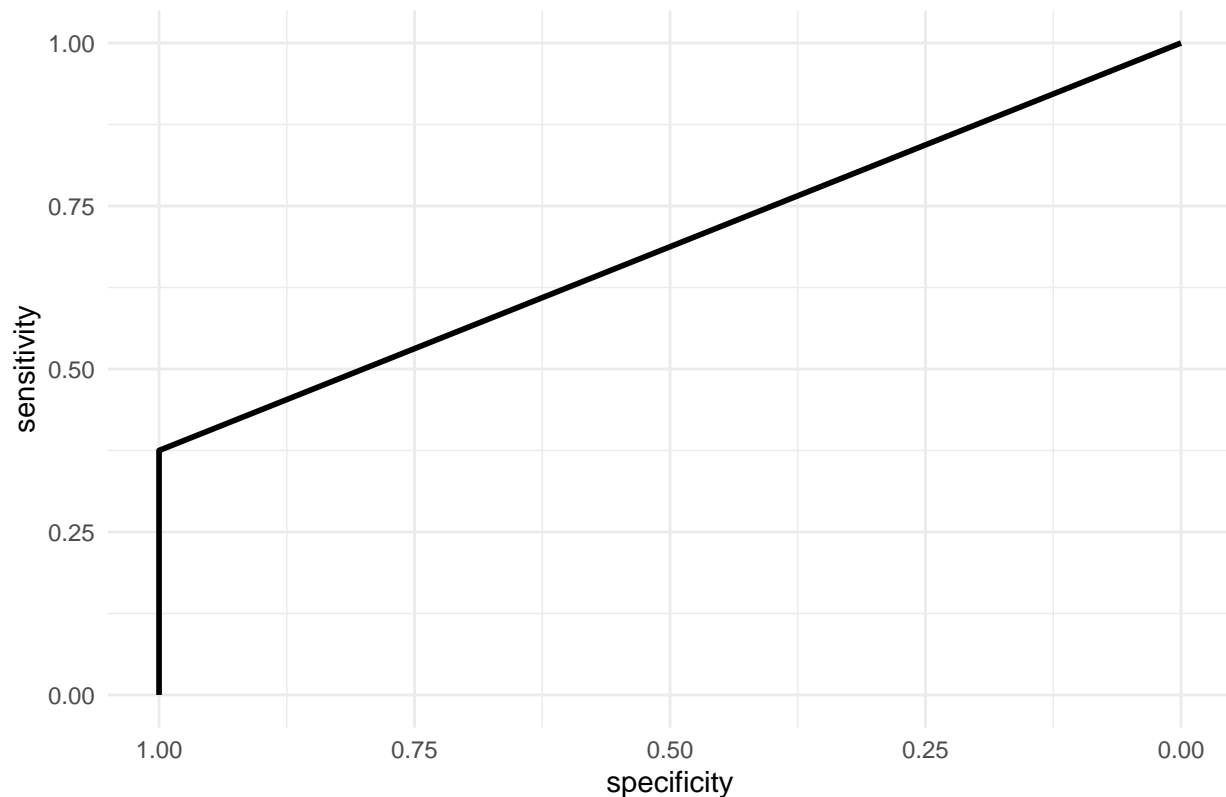
```
roc3 <- roc(df3$quality, df3$predicted_int)
```

```
auc3 <- round(auc(df3$quality, df3$predicted_int), 4)
```

```
ggroc(roc3, colours = 'red', size = 1) +
```

```
  ggtitle(paste0(modelName3, ' - ROC Curve ', '(AUC = ', auc3 , ')')) + theme_minimal()
```

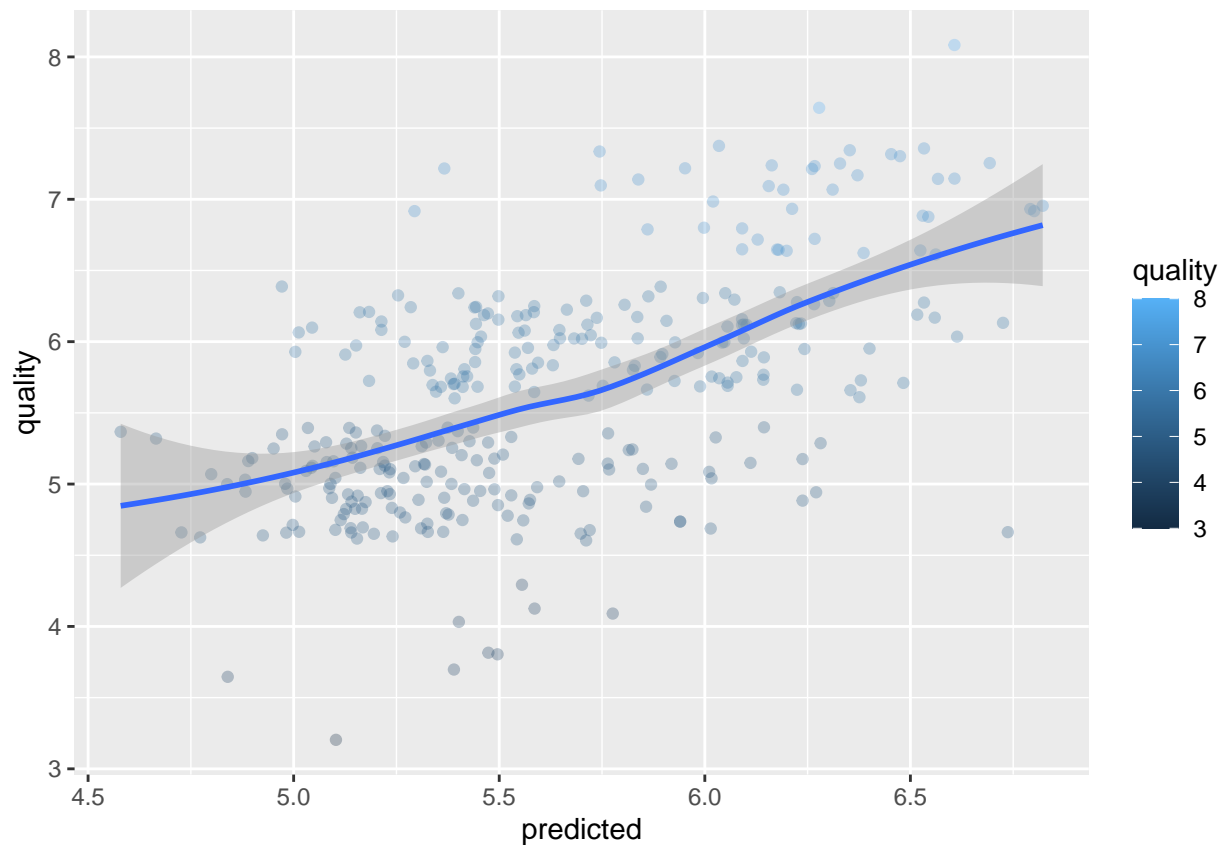
Partial Least Squares – ROC Curve (AUC = 0.6875)



```
# Scatter plot of predicted
```

```
ggplot(df3, aes(x = predicted, y = quality, colour = quality ))+
```

```
  geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```



Mars Tuning

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

```
mars_wine
```

```
## Selected 14 of 16 terms, and 5 of 5 predictors
## Termination condition: Reached nk 21
## Importance: alcohol, sulphates, volatile.acidity, total.sulfur.dioxide, ...
## Number of terms at each degree of interaction: 1 13 (additive model)
## GCV 0.4248782    RSS 521.5673    GRSq 0.3631477    RSq 0.388757
```

```
summary(mars_wine)
```

```
## Call: earth(formula=quality~volatile.acidity+chlorides+total.sulfur.di...),
##           data=wine_train)
##
##
##               coefficients
## (Intercept)      29.408320
## h(0.84-volatile.acidity)  0.830692
## h(volatile.acidity-0.84) -1.983507
## h(chlorides-0.041)      38.221144
## h(0.152-chlorides)      39.984404
## h(chlorides-0.152)     -39.519916
## h(total.sulfur.dioxide-9) -0.212396
```



```

## h(total.sulfur.dioxide-92)      -0.007246
## h(135-total.sulfur.dioxide)    -0.210830
## h(total.sulfur.dioxide-135)    0.228526
## h(0.82-sulphates)              -1.809642
## h(alcohol-9)                   -0.593061
## h(alcohol-9.6)                 0.453324
## h(12.5-alcohol)                -0.466959
##
## Selected 14 of 16 terms, and 5 of 5 predictors
## Termination condition: Reached nk 21
## Importance: alcohol, sulphates, volatile.acidity, total.sulfur.dioxide, ...
## Number of terms at each degree of interaction: 1 13 (additive model)
## GCV 0.4248782   RSS 521.5673   GRSq 0.3631477   RSq 0.388757

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    0.7233527  0.2262431  0.5666630
##  1         3    0.6868415  0.3022130  0.5295100
##  1         4    0.6657077  0.3447624  0.5160514
##  1         5    0.6661297  0.3445123  0.5154858
##  1         6    0.6657229  0.3452116  0.5156309
##  1         7    0.6664653  0.3442757  0.5154168
##  1         8    0.6650382  0.3471788  0.5139120
##  1         9    0.6678134  0.3427006  0.5146854
##  1        10    0.6678225  0.3431591  0.5147597
##  1        11    0.6690708  0.3411141  0.5152562
##  1        12    0.6724844  0.3351686  0.5172197
##  1        13    0.6725114  0.3353018  0.5171558
##  1        14    0.6738001  0.3332975  0.5179456
##  1        15    0.6739969  0.3329714  0.5180914
##  1        16    0.6741257  0.3327127  0.5181518
##  1        17    0.6741257  0.3327127  0.5181518
##  1        18    0.6741257  0.3327127  0.5181518

```

##	1	19	0.6741257	0.3327127	0.5181518
##	1	20	0.6741257	0.3327127	0.5181518
##	1	21	0.6741257	0.3327127	0.5181518
##	1	22	0.6741257	0.3327127	0.5181518
##	1	23	0.6741257	0.3327127	0.5181518
##	1	24	0.6741257	0.3327127	0.5181518
##	1	25	0.6741257	0.3327127	0.5181518
##	1	26	0.6741257	0.3327127	0.5181518
##	1	27	0.6741257	0.3327127	0.5181518
##	1	28	0.6741257	0.3327127	0.5181518
##	1	29	0.6741257	0.3327127	0.5181518
##	1	30	0.6741257	0.3327127	0.5181518
##	1	31	0.6741257	0.3327127	0.5181518
##	1	32	0.6741257	0.3327127	0.5181518
##	1	33	0.6741257	0.3327127	0.5181518
##	1	34	0.6741257	0.3327127	0.5181518
##	1	35	0.6741257	0.3327127	0.5181518
##	1	36	0.6741257	0.3327127	0.5181518
##	1	37	0.6741257	0.3327127	0.5181518
##	1	38	0.6741257	0.3327127	0.5181518
##	2	2	0.7231748	0.2268181	0.5654967
##	2	3	0.6868942	0.3017161	0.5284748
##	2	4	0.6631585	0.3501376	0.5110999
##	2	5	0.6624569	0.3546497	0.5108027
##	2	6	0.6620991	0.3563575	0.5088160
##	2	7	0.6618622	0.3567854	0.5087871
##	2	8	0.6615411	0.3577201	0.5081466
##	2	9	0.6666280	0.3504647	0.5100786
##	2	10	0.6716259	0.3427631	0.5125885
##	2	11	0.6701642	0.3454127	0.5114122
##	2	12	0.6715120	0.3432874	0.5122353
##	2	13	0.6729682	0.3412621	0.5127732
##	2	14	0.6751924	0.3396589	0.5130240
##	2	15	0.6752857	0.3394232	0.5132838
##	2	16	0.6760605	0.3382298	0.5136306
##	2	17	0.6759477	0.3383992	0.5135739
##	2	18	0.6759477	0.3383992	0.5135739
##	2	19	0.6759477	0.3383992	0.5135739
##	2	20	0.6759477	0.3383992	0.5135739
##	2	21	0.6759477	0.3383992	0.5135739
##	2	22	0.6759477	0.3383992	0.5135739
##	2	23	0.6759477	0.3383992	0.5135739
##	2	24	0.6759477	0.3383992	0.5135739
##	2	25	0.6759477	0.3383992	0.5135739
##	2	26	0.6759477	0.3383992	0.5135739
##	2	27	0.6759477	0.3383992	0.5135739
##	2	28	0.6759477	0.3383992	0.5135739
##	2	29	0.6759477	0.3383992	0.5135739
##	2	30	0.6759477	0.3383992	0.5135739
##	2	31	0.6759477	0.3383992	0.5135739
##	2	32	0.6759477	0.3383992	0.5135739
##	2	33	0.6759477	0.3383992	0.5135739
##	2	34	0.6759477	0.3383992	0.5135739
##	2	35	0.6759477	0.3383992	0.5135739

```

##      2      36      0.6759477  0.3383992  0.5135739
##      2      37      0.6759477  0.3383992  0.5135739
##      2      38      0.6759477  0.3383992  0.5135739
##
## 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.    :4.045
##  1st Qu.:5.262
##  Median :5.493
##  Mean    :5.631
##  3rd Qu.:5.985
##  Max.    :7.158

# 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
##  5 104    0  33   2   0   0
##  7   3   14  24   0   0   0
##  6  44    9  74   0   0   0
##  4   6    0   2   0   0   0
##  8   0    1   1   0   0   0
##  3   0    0   0   1   0   0
##
## Overall Statistics
##
##               Accuracy : 0.6038
##               95% CI   : (0.5477, 0.6579)
##      No Information Rate : 0.4937
##      P-Value [Acc > NIR] : 5.208e-05
##
##               Kappa    : 0.3461
##
##  McNemar's Test P-Value : NA
##
## Statistics by Class:
##

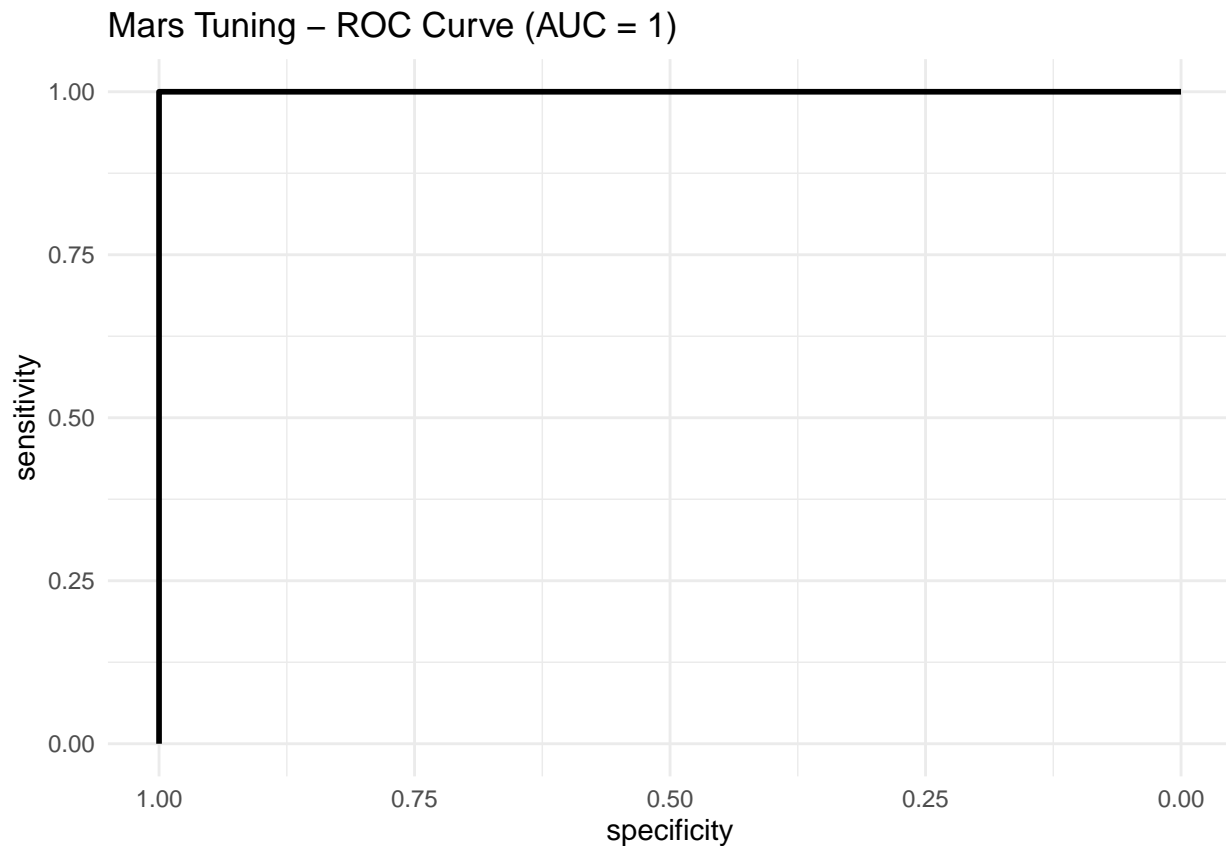
```

```
##
##          Class: 5 Class: 7 Class: 6 Class: 4 Class: 8 Class: 3
## Sensitivity      0.6624 0.58333 0.5522 0.000000      NA      NA
## Specificity      0.7826 0.90816 0.7120 0.974603 0.993711 0.996855
## Pos Pred Value   0.7482 0.34146 0.5827 0.000000      NA      NA
## Neg Pred Value   0.7039 0.96390 0.6859 0.990323      NA      NA
## Prevalence       0.4937 0.07547 0.4214 0.009434 0.000000 0.000000
## Detection Rate   0.3270 0.04403 0.2327 0.000000 0.000000 0.000000
## Detection Prevalence 0.4371 0.12893 0.3994 0.025157 0.006289 0.003145
## Balanced Accuracy 0.7225 0.74575 0.6321 0.487302      NA      NA
```

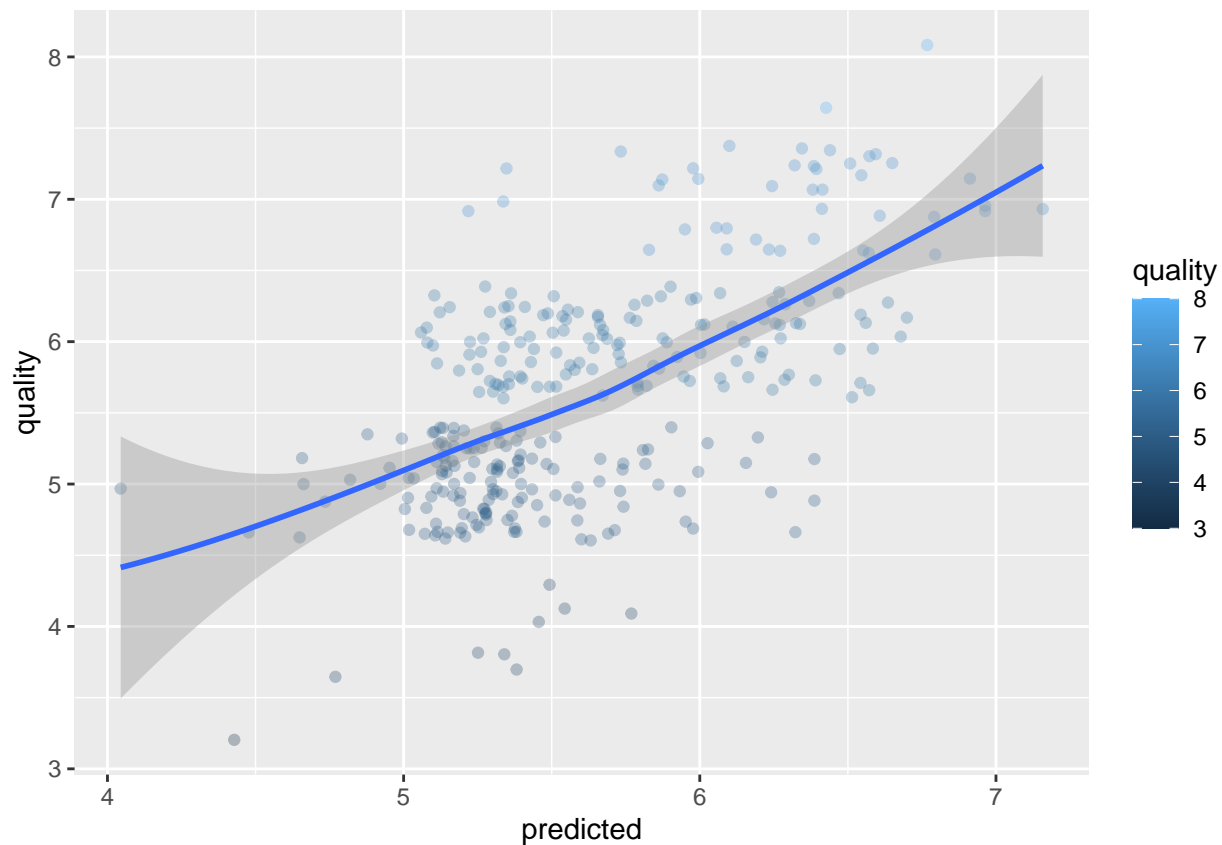
```
# ROC plot
df4$predicted_int = round(as.numeric(as.character(df4$predicted)), digits = 0)

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  0.7611645  0.3095018  0.4418281
```

```

## 2 0.7059464 0.3221223 0.4923622
## 3 0.6978592 0.3201862 0.4995266
## 4 0.6918078 0.3179160 0.5105364
## 5 0.6774652 0.3336540 0.5054629
## 6 0.6667031 0.3468976 0.5008060
## 7 0.6586993 0.3588673 0.4951389
## 8 0.6537894 0.3649030 0.4946392
## 9 0.6532109 0.3637033 0.4973384
## 10 0.6544274 0.3609880 0.4999216
## 11 0.6521244 0.3649248 0.4998247
## 12 0.6522693 0.3644235 0.4995611
## 13 0.6522334 0.3645066 0.4987153
## 14 0.6512748 0.3666665 0.4973378
## 15 0.6501829 0.3685282 0.4972328
## 16 0.6516641 0.3656231 0.4991521
## 17 0.6496509 0.3694799 0.4970775
## 18 0.6488745 0.3707600 0.4977312
## 19 0.6482618 0.3717198 0.4976589
## 20 0.6489652 0.3706964 0.4986046
## 21 0.6504049 0.3680175 0.4996562
## 22 0.6505565 0.3679427 0.5007805
## 23 0.6502980 0.3687124 0.5005675
## 24 0.6486853 0.3721803 0.4990811
## 25 0.6497087 0.3704544 0.4998703
## 26 0.6481956 0.3733048 0.4989776
## 27 0.6476405 0.3746132 0.4985409
## 28 0.6483439 0.3730679 0.5003046
## 29 0.6487146 0.3726818 0.5006378
## 30 0.6486947 0.3729127 0.5005301
## 31 0.6490613 0.3723327 0.5007166
## 32 0.6489346 0.3727020 0.5010819
## 33 0.6484185 0.3741388 0.5012523
## 34 0.6482329 0.3744969 0.5015616
## 35 0.6484952 0.3741138 0.5016705
## 36 0.6494802 0.3724373 0.5022773
## 37 0.6501885 0.3711916 0.5023928
## 38 0.6498152 0.3721481 0.5021523
## 39 0.6498068 0.3722171 0.5024646
## 40 0.6491849 0.3732611 0.5022056
## 41 0.6495999 0.3726338 0.5021190
## 42 0.6485574 0.3745594 0.5013595
## 43 0.6486343 0.3742573 0.5016012
## 44 0.6483522 0.3749460 0.5016836
## 45 0.6480208 0.3756090 0.5012867
## 46 0.6477996 0.3759857 0.5015260
## 47 0.6474121 0.3767748 0.5013182
## 48 0.6471637 0.3774581 0.5016746
## 49 0.6471179 0.3776970 0.5017991
## 50 0.6472295 0.3776650 0.5016013
##

```

```

## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 49.

```

```

# 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.
##      4.878  5.265   5.551   5.651   6.000   6.796

# 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)

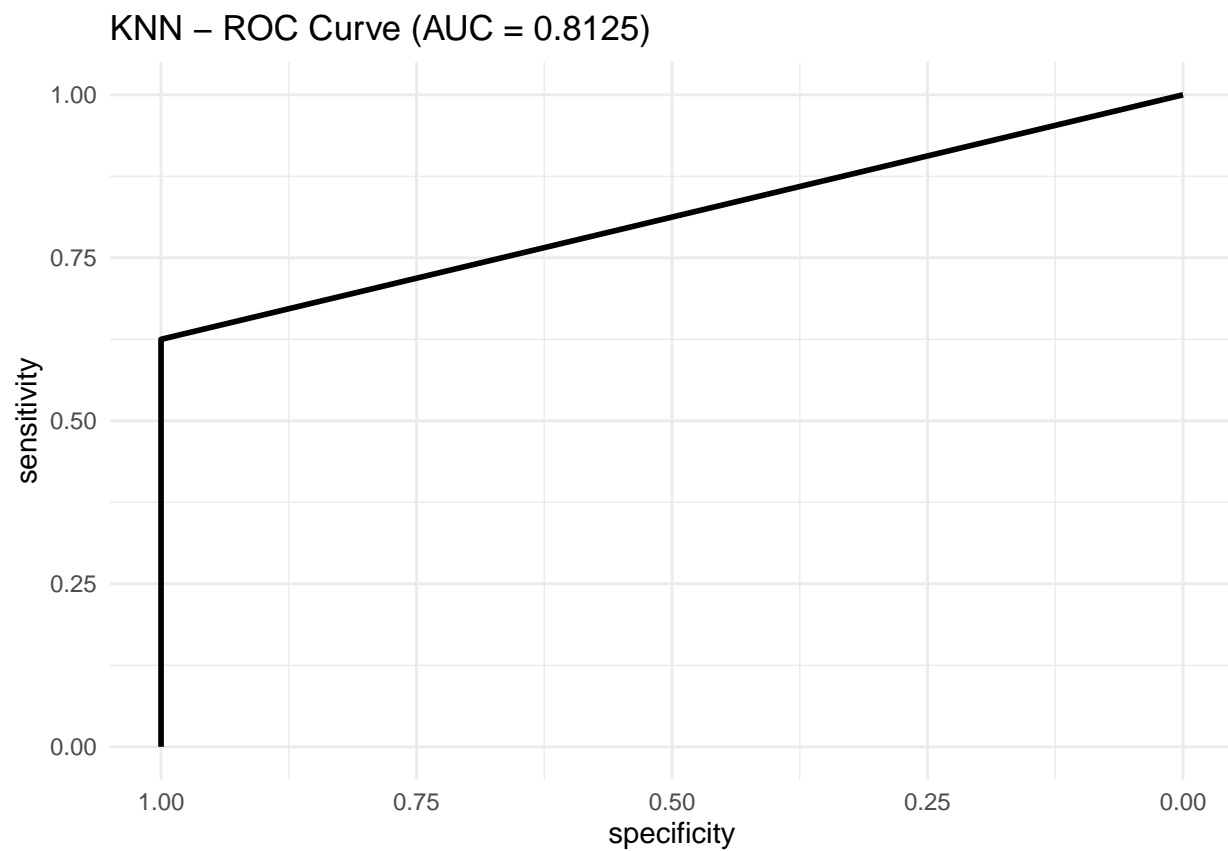
## Confusion Matrix and Statistics
##
##
##          5    7    6    4    8    3
## 5 104    0   35    0    0    0
## 7   2   12   27    0    0    0
## 6  40    4   83    0    0    0
## 4   3    0    5    0    0    0
## 8   0    2    0    0    0    0
## 3   1    0    0    0    0    0
##
## Overall Statistics
##
##              Accuracy : 0.6258
##              95% CI   : (0.5701, 0.6792)
##    No Information Rate : 0.4717
##    P-Value [Acc > NIR] : 2.392e-08
##
##              Kappa   : 0.3744
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##              Class: 5 Class: 7 Class: 6 Class: 4 Class: 8 Class: 3
## Sensitivity          0.6933  0.66667   0.5533      NA      NA      NA
## Specificity          0.7917  0.90333   0.7381  0.97484  0.993711  0.996855
## Pos Pred Value       0.7482  0.29268   0.6535      NA      NA      NA
## Neg Pred Value       0.7430  0.97834   0.6492      NA      NA      NA
## Prevalence           0.4717  0.05660   0.4717  0.00000  0.000000  0.000000
## Detection Rate       0.3270  0.03774   0.2610  0.00000  0.000000  0.000000
## Detection Prevalence 0.4371  0.12893   0.3994  0.02516  0.006289  0.003145
## Balanced Accuracy     0.7425  0.78500   0.6457      NA      NA      NA

```

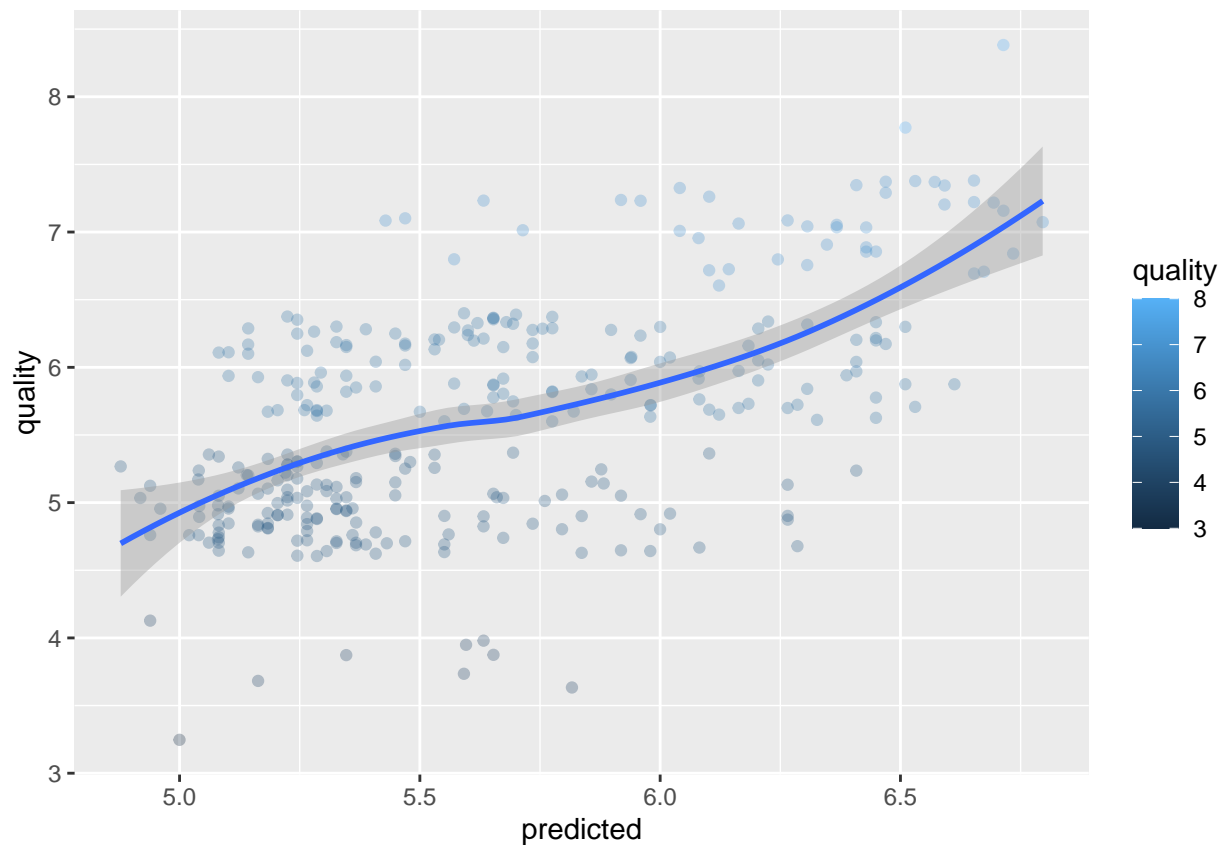
```
# ROC plot
df5$predicted_int = round(as.numeric(as.character(df5$predicted)), digits = 0)

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 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, 1152, 1151, 1154, 1153, ...
## Resampling results across tuning parameters:
##
##    C      Accuracy   Kappa
## 0.25 0.5877289 0.3089678
```

```

##    0.50  0.5892976  0.3141981
##    1.00  0.5861540  0.3116337
##    2.00  0.5877290  0.3191442
##    4.00  0.5923682  0.3275874
##
## Tuning parameter 'sigma' was held constant at a value of 0.5592836
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.5592836 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
##    0    1 162 133   22    0

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

## Confusion Matrix and Statistics
##
##
##      3    4    5    6    7    8
## 3    0    0    1    0    0    0
## 4    0    0    6    2    0    0
## 5    0    1 105   31    2    0
## 6    0    0  48   72    7    0
## 7    0    0    2   27   12    0
## 8    0    0    0    1    1    0
##
## Overall Statistics
##
##               Accuracy : 0.5943
##               95% CI : (0.5381, 0.6488)
##      No Information Rate : 0.5094
##      P-Value [Acc > NIR] : 0.001434
##
##               Kappa : 0.3254
##
##  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  0.6481  0.5414  0.54545      NA
## Specificity          0.996855 0.974763  0.7821  0.7027  0.90203 0.993711
## Pos Pred Value          NA 0.000000  0.7554  0.5669  0.29268      NA
## Neg Pred Value          NA 0.996774  0.6816  0.6806  0.96390      NA
## Prevalence           0.000000 0.003145  0.5094  0.4182  0.06918 0.000000
## Detection Rate        0.000000 0.000000  0.3302  0.2264  0.03774 0.000000
## Detection Prevalence 0.003145 0.025157  0.4371  0.3994  0.12893 0.006289

```

```
## Balanced Accuracy          NA 0.487382  0.7151  0.6220  0.72374      NA
```

```
# ROC plot
```

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

```
modelName6 <- 'SVM'
```

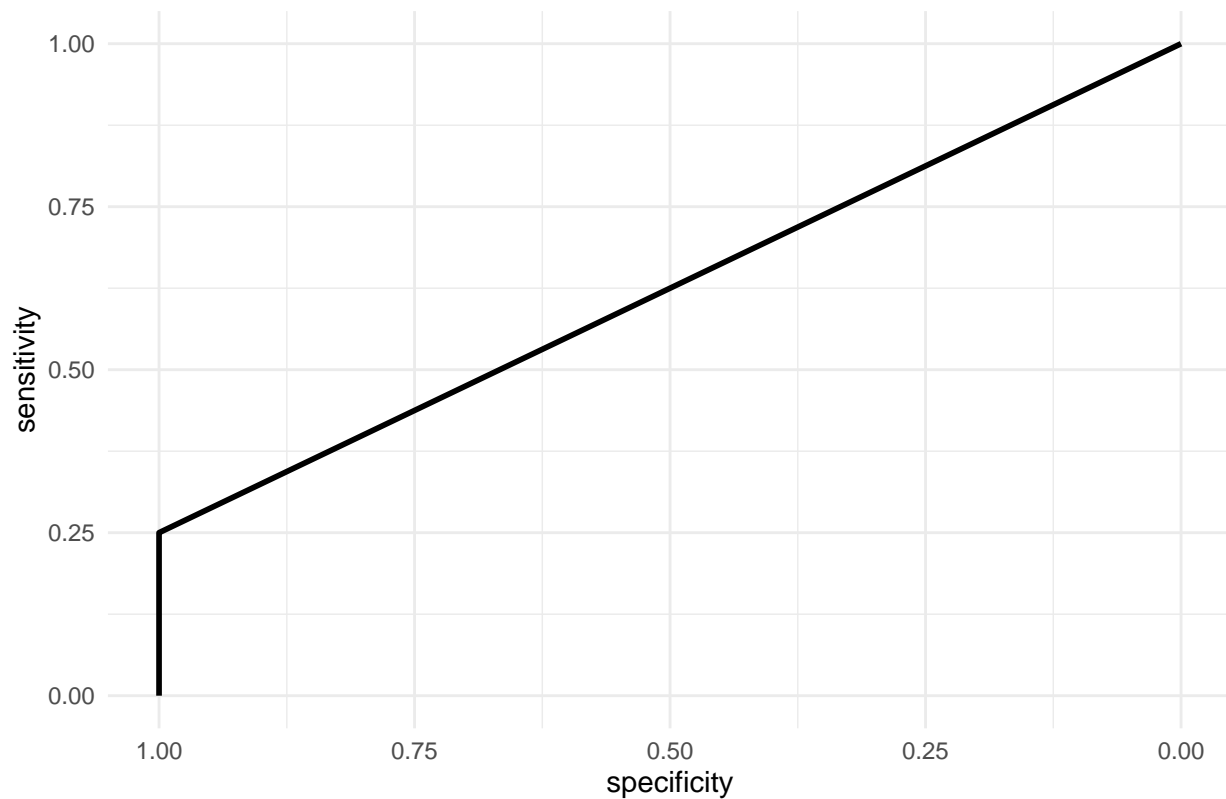
```
roc6 <- roc(df6$quality, df6$predicted_int)
```

```
auc6 <- round(auc(df6$quality, df6$predicted_int), 4)
```

```
ggroc(roc6, colours = 'red', size = 1) +
```

```
  ggtitle(paste0(modelName6, ' - ROC Curve ', '(AUC = ', auc6 , ')')) + theme_minimal()
```

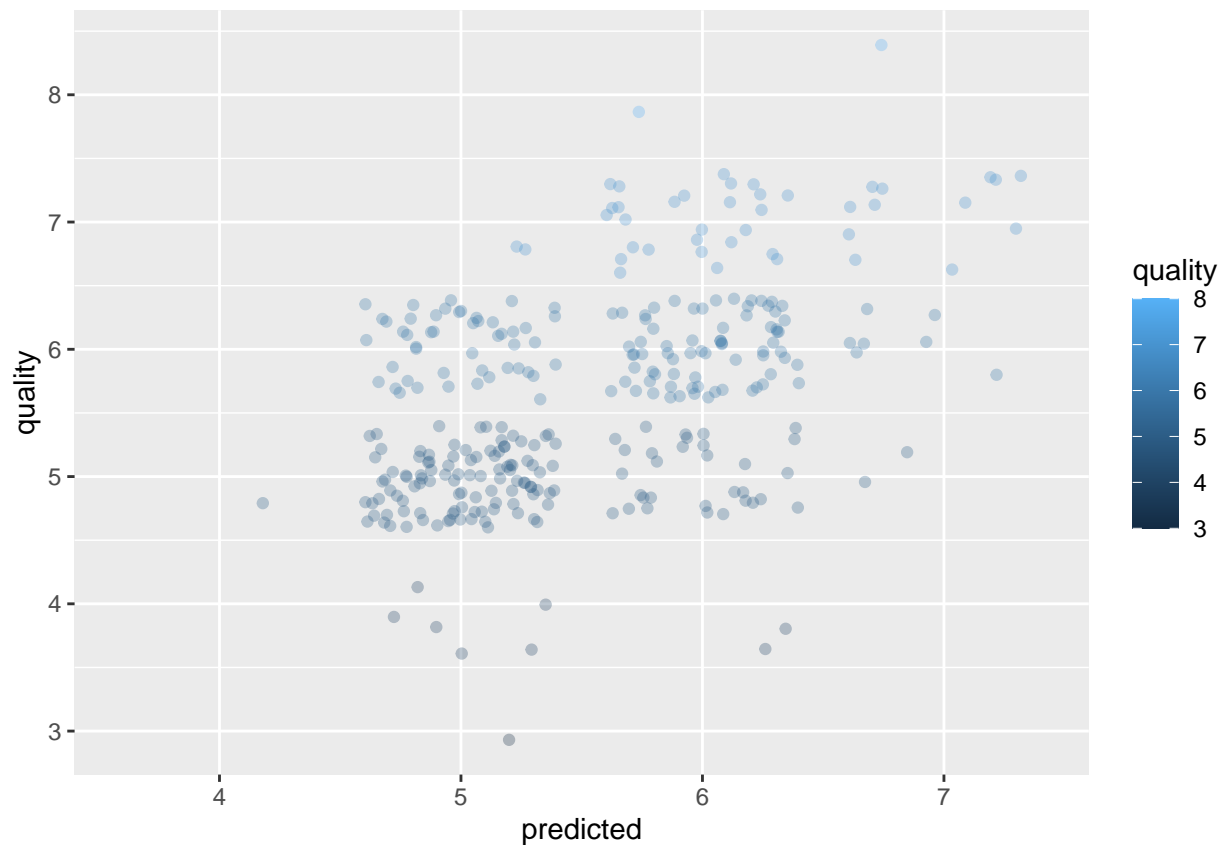
SVM – ROC Curve (AUC = 0.625)



```
# Scatter plot of predicted
```

```
ggplot(df6, aes(x = predicted, y = quality, colour = quality ))+
```

```
  geom_point(alpha = 0.3, position = position_jitter()) + stat_smooth()
```



Penalized Logistic Regression Tuning

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

glmnetTune

## 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, 1152, 1153, ...
## Resampling results across tuning parameters:
##
##  alpha  lambda  Accuracy  Kappa
##  0.0    0.0100  0.5885118  0.3162331
```

```
## 0.0 0.0575 0.5822860 0.2987068
## 0.0 0.1050 0.5768173 0.2840738
## 0.0 0.1525 0.5791549 0.2860009
## 0.0 0.2000 0.5815109 0.2891512
## 0.1 0.0100 0.5893053 0.3205317
## 0.1 0.0575 0.5799362 0.2926301
## 0.1 0.1050 0.5799484 0.2875440
## 0.1 0.1525 0.5814985 0.2890215
## 0.1 0.2000 0.5728985 0.2741068
## 0.2 0.0100 0.5877184 0.3160999
## 0.2 0.0575 0.5783858 0.2886395
## 0.2 0.1050 0.5775800 0.2825011
## 0.2 0.1525 0.5705670 0.2701028
## 0.2 0.2000 0.5690290 0.2667256
## 0.4 0.0100 0.5869371 0.3144909
## 0.4 0.0575 0.5783613 0.2855434
## 0.4 0.1050 0.5689984 0.2672190
## 0.4 0.1525 0.5697922 0.2676438
## 0.4 0.2000 0.5580733 0.2471184
## 0.6 0.0100 0.5877306 0.3139489
## 0.6 0.0575 0.5666546 0.2647442
## 0.6 0.1050 0.5705612 0.2690615
## 0.6 0.1525 0.5479108 0.2294916
## 0.6 0.2000 0.5510361 0.2335781
## 0.8 0.0100 0.5861681 0.3104682
## 0.8 0.0575 0.5705427 0.2702880
## 0.8 0.1050 0.5549422 0.2420563
## 0.8 0.1525 0.5549425 0.2407836
## 0.8 0.2000 0.5370280 0.2087613
## 1.0 0.0100 0.5830552 0.3047040
## 1.0 0.0575 0.5728866 0.2735412
## 1.0 0.1050 0.5541673 0.2397724
## 1.0 0.1525 0.5471541 0.2268064
## 1.0 0.2000 0.4387409 0.0288173
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.1 and lambda = 0.01.
```

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

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

```
## 3 4 5 6 7 8
## 0 1 162 133 22 0
```

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

```
## Confusion Matrix and Statistics
##
##
```

```
##      3  4  5  6  7  8
##  3  0  1  0  0  0  0
##  4  0  0  4  4  0  0
##  5  0  1 107 30  1  0
##  6  0  0  48 70  9  0
##  7  0  0  2 27 12  0
##  8  0  0  0  1  1  0
```

```
##
```

```
## Overall Statistics
```

```
##
```

```
##           Accuracy : 0.5943
```

```
##           95% CI : (0.5381, 0.6488)
```

```
##      No Information Rate : 0.5063
```

```
##      P-Value [Acc > NIR] : 0.0009891
```

```
##
```

```
##           Kappa : 0.3278
```

```
##
```

```
## 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  0.6646  0.5303  0.52174      NA
```

```
## Specificity          0.996855 0.974684  0.7962  0.6935  0.90169 0.993711
```

```
## Pos Pred Value           NA 0.000000  0.7698  0.5512  0.29268      NA
```

```
## Neg Pred Value           NA 0.993548  0.6983  0.6754  0.96029      NA
```

```
## Prevalence             0.000000 0.006289  0.5063  0.4151  0.07233 0.000000
```

```
## Detection Rate          0.000000 0.000000  0.3365  0.2201  0.03774 0.000000
```

```
## Detection Prevalence    0.003145 0.025157  0.4371  0.3994  0.12893 0.006289
```

```
## Balanced Accuracy           NA 0.487342  0.7304  0.6119  0.71172      NA
```

```
# ROC plot
```

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

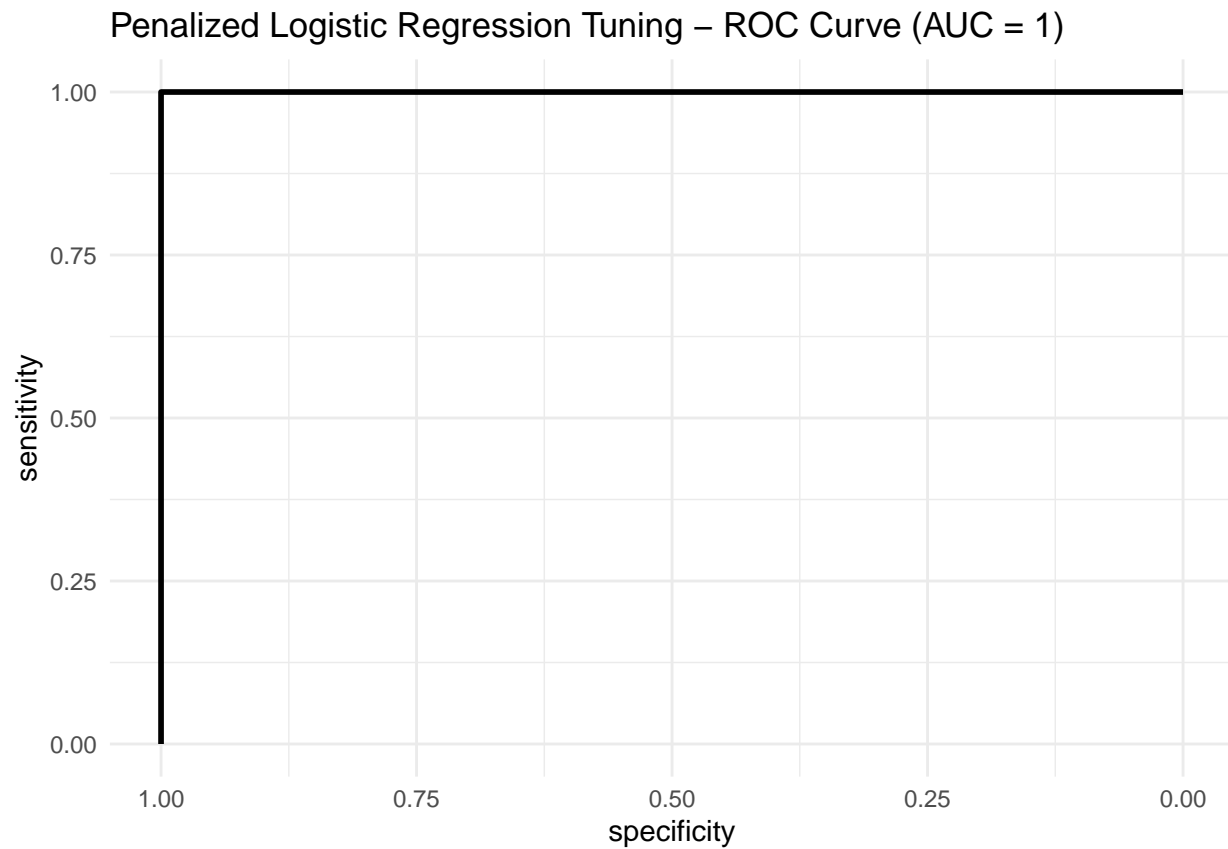
```
modelName7 <- 'Penalized Logistic Regression Tuning'
```

```
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()
```

