Supplementary Material A: Tree-Based Models

Alexander M. Carnall

2023-04-23

PART 1: Data Import, Qualitative & Quantitative Description

```
library(tidyverse); library(flextable)

wineData = read.csv(paste0('https://archive.ics.uci.edu/ml/machine-learning-databases/',
                           'wine-quality/winequality-red.csv'),sep = ';') %>%
  mutate(category = case_when(quality < 6 ~ 'Low', quality >= 6 ~ 'High')) %>%
  mutate(quality = as.factor(quality)) %>% mutate(category = as.factor(category))

varDesc = data.frame(matrix(ncol = 4, nrow = ncol(wineData))) %>%
  setNames(., c('Attribute', 'Type', 'Purpose', 'Description')) %>%
  mutate(Attribute = names(wineData)) %>%
  mutate(Type = c(rep('numeric', 11), 'factor', 'factor')) %>%
  mutate(Purpose = c(rep('Predictor', 11), 'Response', 'Response')) %>%
  mutate(Description = c('Tartaric acid (g/dm³)', 'Acetic acid (g/dm³)', 'Citric acid (g/dm³)',
                         'Residual sugar (g/dm³)', 'Sodium chloride (g/dm³)',
                         'Free sulfur dioxide (g/dm³)', 'Total sulfur dioxide (g/dm³)',
                         'Density (g/dm³)', 'pH (negative log of H+ concentration)',
                         'Potassium sulphate (g/dm³)', '% Alcohol by volume',
                         'Median score from at least 3 expert ratings',
                         'Binary, Low (quality < 6), or High (quality >= 6)')) %>%
  flextable() %>% width(j = 1, width = 1.5) %>% width(j = 2:3, width = 1) %>%
  width(j = 4, width = 3) %>% bold(bold = TRUE, part = 'header') %>%
  fontsize(size = 8, part = 'all') %>% bold(j = 1, bold = TRUE, part = 'body') %>%
  align(j = 2:3, align = 'center', part = 'all') %>%
  add_header_row(values = 'Table 1: Description of Attributes in Data', colwidths = 4) %>%
  padding(padding.top = 0, padding.bottom = 0, part = 'body')

varDesc
```

Table 1: Description of Attributes in Data

Attribute	Type	Purpose	Description
fixed.acidity	numeric	Predictor	Tartaric acid (g/dm³)
volatile.acidity	numeric	Predictor	Acetic acid (g/dm³)
citric.acid	numeric	Predictor	Citric acid (g/dm³)
residual.sugar	numeric	Predictor	Residual sugar (g/dm³)

Table 1: Description of Attributes in Data

Attribute	Type	Purpose	Description
chlorides	numeric	Predictor	Sodium chloride (g/dm^3)
free.sulfur.dioxide	numeric	Predictor	Free sulfur dioxide (g/dm^3)
total.sulfur.dioxide	numeric	Predictor	Total sulfur dioxide (g/dm^3)
density	numeric	Predictor	Density (g/dm^3)
pH	numeric	Predictor	pH (negative log of H+ concentration)
sulphates	numeric	Predictor	Potassium sulphate (g/dm^3)
alcohol	numeric	Predictor	% Alcohol by volume
quality	factor	Response	Median score from at least 3 expert ratings
category	factor	Response	Binary, Low (quality < 6), or High (quality >= 6)

```
summary = do.call(cbind, lapply(wineData, summary)) %>% data.frame(check.names = FALSE) %>%
  mutate_if(is.numeric, round, 2) %>% rownames_to_column() %>% rename('value' = 'rowname') %>%
  mutate(category = c(sum(wineData$category == 'Low'), NaN, NaN, NaN, NaN,
                      sum(wineData$category == 'High'))) %>%
  rename('Quality (3:8)' = 'quality') %>% rename('Category (count)' = 'category')

summFlx = summary %>% flextable() %>% bold(bold = TRUE, part = 'header') %>%
  bold(j = 1, bold = TRUE, part = 'body') %>% fontsize(size = 8, part = 'all') %>%
  width(width = 6.5 / 14) %>% align(j  = 2:14, align = 'center', part = 'all') %>%
  rotate(j = 2:12, align = 'bottom', rotation = 'btlr', part = 'header') %>%
  add_header_row(values = 'Table 2: Summary Statistics for Attributes in Data', colwidths = 14)%>%
  padding(padding.top = 0, padding.bottom = 0, part = 'body')


summFlx
```

Table 2: Summary Statistics for Attributes in Data

value	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol	Quality (3:8)	Category (count)
Min.	4.60	0.12	0.00	0.90	0.01	1.00	6.00	0.99	2.74	0.33	8.40	10	744
1st Qu.	7.10	0.39	0.09	1.90	0.07	7.00	22.00	1.00	3.21	0.55	9.50	53	
Median	7.90	0.52	0.26	2.20	0.08	14.00	38.00	1.00	3.31	0.62	10.20	681	
Mean	8.32	0.53	0.27	2.54	0.09	15.87	46.47	1.00	3.31	0.66	10.42	638	
3rd Qu.	9.20	0.64	0.42	2.60	0.09	21.00	62.00	1.00	3.40	0.73	11.10	199	
Max.	15.90	1.58	1.00	15.50	0.61	72.00	289.00	1.00	4.01	2.00	14.90	18	855

PART 2: Visual Description of Response and Predictor Variables

```
library(cowplot)

qualHist = ggplot(wineData, aes(x = quality)) + coord_cartesian(ylim = c(0, 800)) +
    geom_histogram(bins = length(unique(wineData$quality)), stat = 'count',
                   binwidth = 1, boundary = -0.5, color = "black", fill = "white") +
  geom_text(stat = "count", aes(label = paste0('Count: ', ..count..),
                                fontface = 'bold'), size = 2.5, vjust = -2.1) +
  geom_text(stat = "count", aes(
    label = paste0('(', round((..count.. / sum(..count..)) * 100, 2), '%)'),
                                fontface = 'bold'), size = 2.5, vjust = -0.7) +
  labs(title = 'Figure 1: Quality Response Variable', x = 'Quality Score',
       y = 'Raw & Relative Frequencies') +
  theme(plot.title = element_text(size = 10), axis.title.x = element_text(size = 9),
        axis.title.y = element_text(size = 9), axis.text.y = element_text(size = 6),
        legend.position = 'none')

catHist = wineData %>% count(category) %>% mutate(pct = round(n / sum(n), 2)) %>% ggplot() +
  geom_bar(mapping = aes(x = factor(category, levels = c('Low', 'High')), y = n),
           fill = 'white', color = 'black', stat = 'identity') +
  geom_text(aes(label = paste0('Count: ', n), x = category, y = n),
            fontface = 'bold', vjust = -2.1, size = 2.5) +
  geom_text(aes(label = paste0('(', pct * 100, '%)'), x = category, y = n),
            fontface = 'bold', vjust = -0.7, size = 2.5) +
  labs(title = 'Figure 2: Category Response Variable', x = 'Category') + ylim(0, 1000) +
  theme(plot.title = element_text(size = 10),axis.title.x = element_text(size = 9),
        axis.title.y = element_blank(), axis.text.y = element_text(size = 6),
        legend.position = 'none')

resHist = plot_grid(qualHist, catHist, nrow = 1)

resHist
```

Figure 1: Quality Response Variable

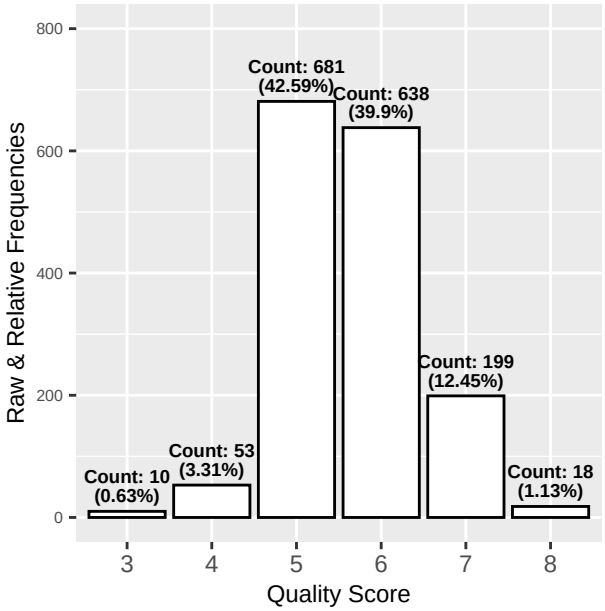
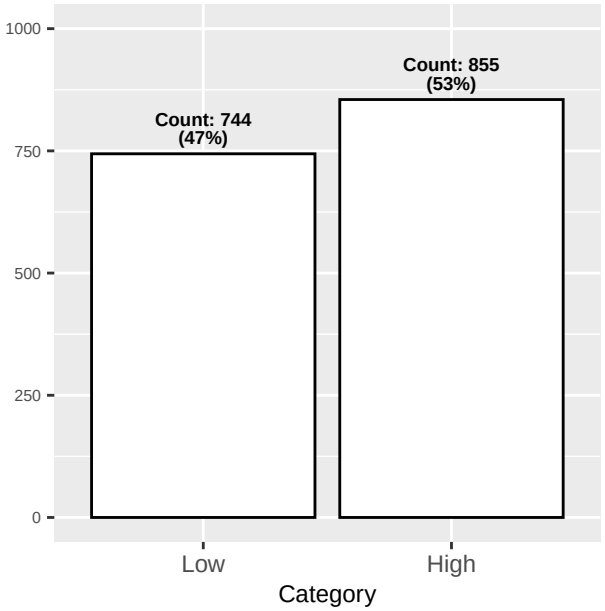
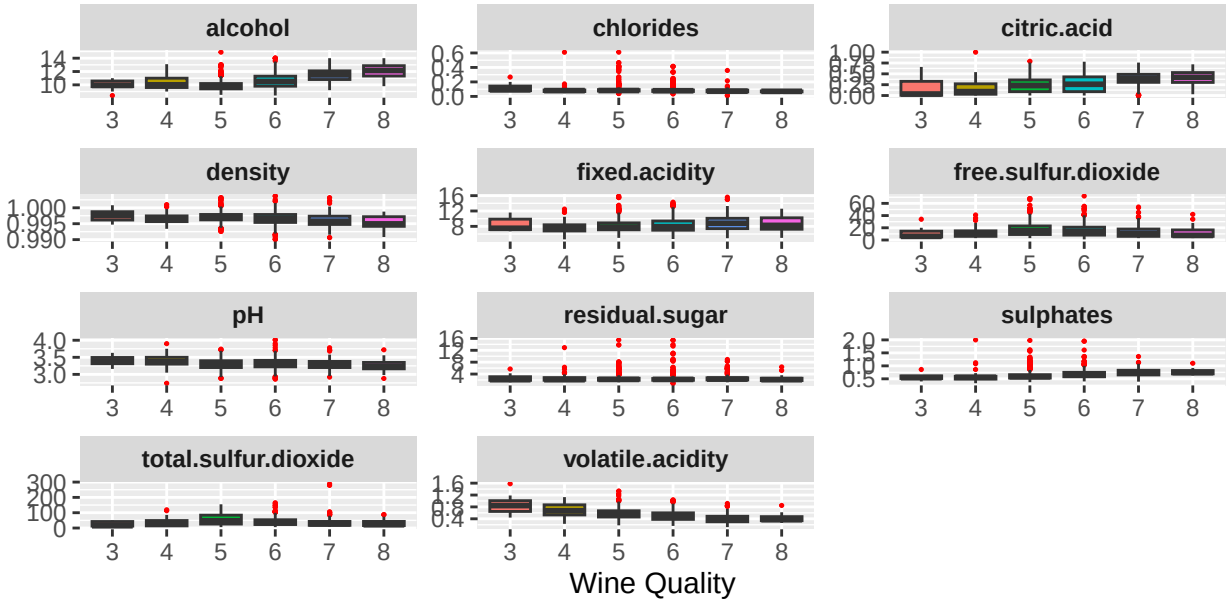


Figure 2: Category Response Variable

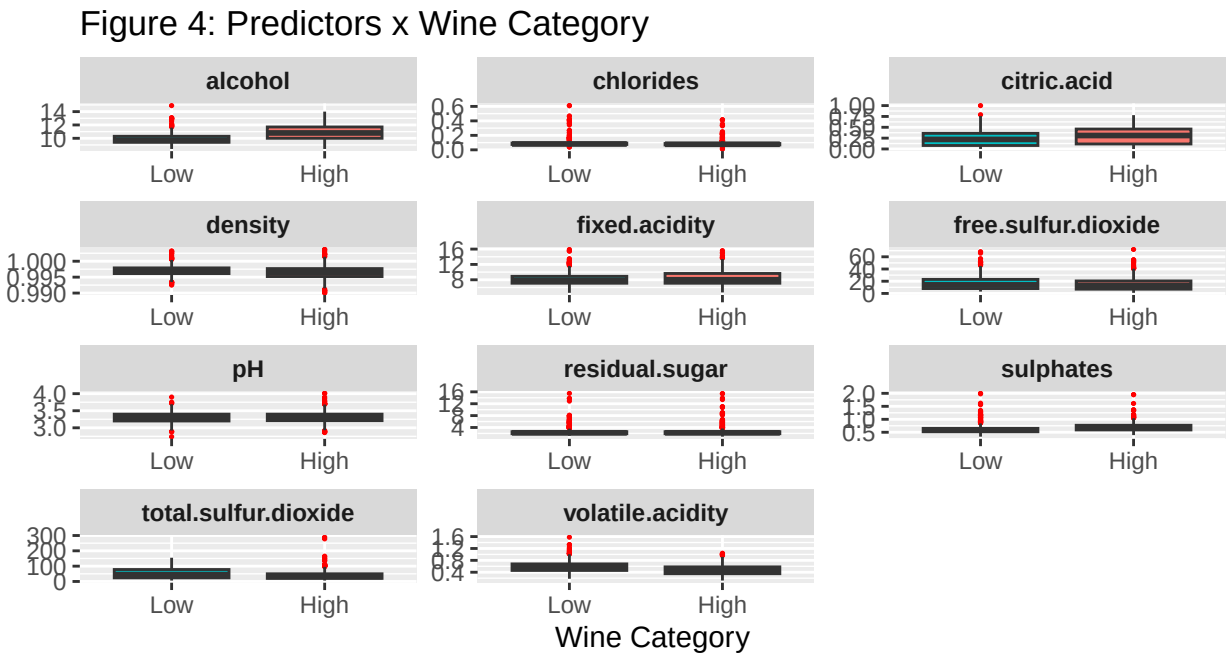



```
boxQuality = wineData[, c(1:12)] %>% pivot_longer(-quality) %>% ggplot() +
  geom_boxplot(aes(x = quality, y = value, fill = quality),
               outlier.size = 0.25, outlier.color = 'red') +
  facet_wrap(~name, scales = 'free', ncol = 3) +
  labs(x = 'Wine Quality', title = 'Figure 3: Predictors x Wine Quality') +
  theme(strip.text = element_text(size = 9, face = 'bold'), legend.position = 'none',
        axis.title.y = element_blank())

boxQuality
```

Figure 3: Predictors x Wine Quality



```
boxCategory = wineData[, c(1:11, 13)] %>% pivot_longer(-category) %>% ggplot() +
  geom_boxplot(aes(x = factor(category, levels = c('Low', 'High')), y = value, fill = category),
               outlier.size = 0.25, outlier.color = 'red') +
  facet_wrap(~name, scales = 'free', ncol = 3) +
  labs(x = 'Wine Category', title = 'Figure 4: Predictors x Wine Category') +
  theme(strip.text = element_text(size = 9, face = 'bold'), legend.position = 'none',
        axis.title.y = element_blank())

boxCategory
```




```
boxPlots = plot_grid(boxQuality, boxCategory, nrow = 1)
```

PART 3: Implementation for Tree-Based Methods

```
# Generating 'out of the box' trees for each response variable

library(tree)

q.Tree.Train = tree(quality ~ . -category, data = wineData)
c.Tree.Train = tree(category ~ . -quality, data = wineData)

q.Tree.Train
```

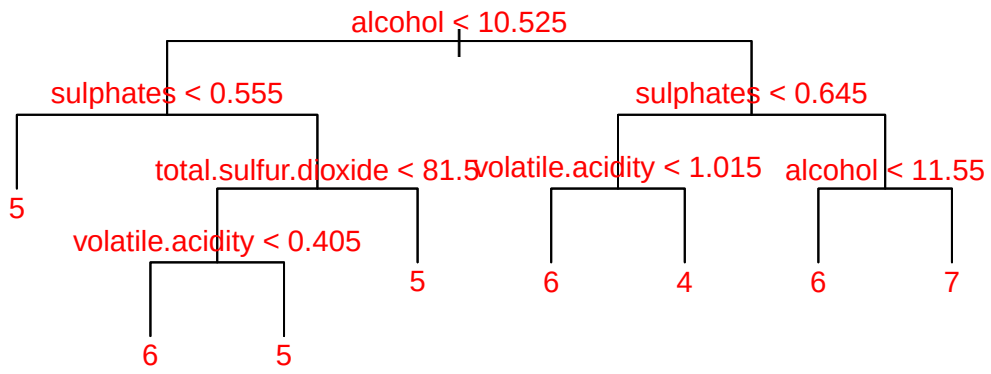
```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1599 3788.00 5 ( 0.006254 0.033146 0.425891 0.398999 0.124453 0.011257 )
##    2) alcohol < 10.525 983 1902.00 5 ( 0.007121 0.034588 0.584944 0.333672 0.037640 0.002035 )
```

```
##      4) sulphates < 0.555 320  495.50 5 ( 0.009375 0.059375 0.734375 0.193750 0.003125 0.000000 ) *
##      5) sulphates > 0.555 663 1328.00 5 ( 0.006033 0.022624 0.512821 0.401207 0.054299 0.003017 )
##       10) total.sulfur.dioxide < 81.5 544 1133.00 6 ( 0.007353 0.023897 0.448529 0.452206 0.064338 0
##         20) volatile.acidity < 0.405 136  271.70 6 ( 0.000000 0.000000 0.286765 0.573529 0.125000 0.0
##         21) volatile.acidity > 0.405 408  819.30 5 ( 0.009804 0.031863 0.502451 0.411765 0.044118 0.0
##       11) total.sulfur.dioxide > 81.5 119  138.50 5 ( 0.000000 0.016807 0.806723 0.168067 0.008403 0
##    3) alcohol > 10.525 616 1513.00 6 ( 0.004870 0.030844 0.172078 0.503247 0.262987 0.025974 )
##      6) sulphates < 0.645 272  662.80 6 ( 0.011029 0.066176 0.257353 0.518382 0.143382 0.003676 )
##       12) volatile.acidity < 1.015 262  599.20 6 ( 0.000000 0.053435 0.255725 0.538168 0.148855 0.003
##       13) volatile.acidity > 1.015 10   21.78 4 ( 0.300000 0.400000 0.300000 0.000000 0.000000 0.000
##      7) sulphates > 0.645 344  761.40 6 ( 0.000000 0.002907 0.104651 0.491279 0.357558 0.043605 )
##       14) alcohol < 11.55 206  439.50 6 ( 0.000000 0.004854 0.160194 0.563107 0.252427 0.019417 ) *
##       15) alcohol > 11.55 138  274.40 7 ( 0.000000 0.000000 0.021739 0.384058 0.514493 0.079710 ) *
```

```
summary(q.Tree.Train)
```

```
##
## Classification tree:
## tree(formula = quality ~ . - category, data = wineData)
## Variables actually used in tree construction:
## [1] "alcohol"              "sulphates"            "total.sulfur.dioxide"
## [4] "volatile.acidity"
## Number of terminal nodes:  8
## Residual mean deviance:  1.923 = 3060 / 1591
## Misclassification error rate: 0.4084 = 653 / 1599
```

```
plot(q.Tree.Train, type = 'uniform')
text(q.Tree.Train, pretty = 1, cex = 0.9, col = 'red')
title('Figure 5: Predicting Wine Quality')
```



```
c.Tree.Train
```

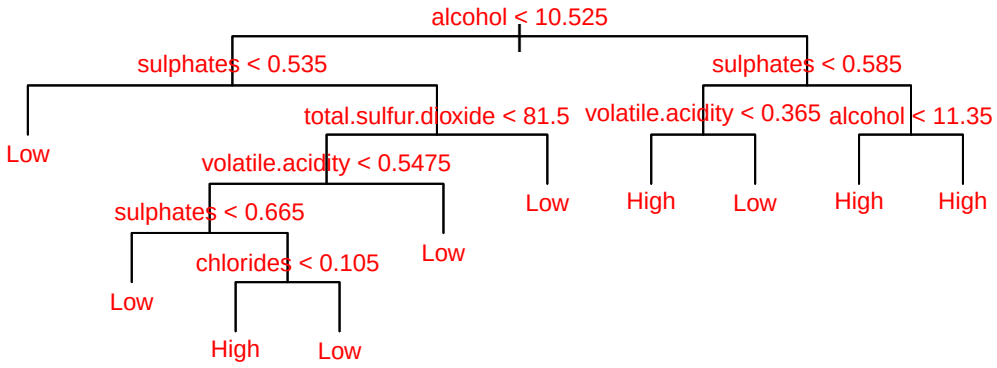
```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1599 2209.00 High ( 0.53471 0.46529 )
##    2) alcohol < 10.525 983 1299.00 Low ( 0.37335 0.62665 )
##      4) sulphates < 0.535 230  196.20 Low ( 0.15217 0.84783 ) *
##      5) sulphates > 0.535 753 1033.00 Low ( 0.44090 0.55910 )
##       10) total.sulfur.dioxide < 81.5 610  845.60 High ( 0.50328 0.49672 )
##         20) volatile.acidity < 0.5475 324  432.10 High ( 0.61420 0.38580 )
##           40) sulphates < 0.665 178  246.70 Low ( 0.48876 0.51124 ) *
##           41) sulphates > 0.665 146  158.50 High ( 0.76712 0.23288 )
##             82) chlorides < 0.105 114   88.78 High ( 0.86842 0.13158 ) *
##             83) chlorides > 0.105 32   43.23 Low ( 0.40625 0.59375 ) *
##         21) volatile.acidity > 0.5475 286  379.20 Low ( 0.37762 0.62238 ) *
##       11) total.sulfur.dioxide > 81.5 143  132.50 Low ( 0.17483 0.82517 ) *
##    3) alcohol > 10.525 616  629.60 High ( 0.79221 0.20779 )
##      6) sulphates < 0.585 166  225.40 High ( 0.58434 0.41566 )
##       12) volatile.acidity < 0.365 37   15.56 High ( 0.94595 0.05405 ) *
##       13) volatile.acidity > 0.365 129  178.60 Low ( 0.48062 0.51938 ) *
##      7) sulphates > 0.585 450  349.60 High ( 0.86889 0.13111 )
##       14) alcohol < 11.35 230  232.90 High ( 0.79565 0.20435 ) *
##       15) alcohol > 11.35 220   93.14 High ( 0.94545 0.05455 ) *
```

```
summary(c.Tree.Train)
```

```
##
## Classification tree:
## tree(formula = category ~ . - quality, data = wineData)
## Variables actually used in tree construction:
## [1] "alcohol"              "sulphates"            "total.sulfur.dioxide"
## [4] "volatile.acidity"     "chlorides"
## Number of terminal nodes:  10
## Residual mean deviance:  1.011 = 1607 / 1589
## Misclassification error rate: 0.2539 = 406 / 1599
```

```
plot(c.Tree.Train, type = 'uniform')
text(c.Tree.Train, pretty = 0, cex = 0.75, col = 'red')
title('Figure 6: Predicting Wine Category')
```

Figure 6: Predicting Wine Category




```
#Estimating training error for each 'out of the box' tree

q.Tree.Pred.Base = predict(q.Tree.Train, wineData, type = 'class')

q.Tree.Base.Table = table(q.Tree.Pred.Base, wineData$quality)

q.Tree.Base.Table
```

```
##
## q.Tree.Pred.Base   3   4   5   6   7   8
##                3   0   0   0   0   0   0
##                4   3   4   3   0   0   0
##                5   7  34 536 250  20   0
##                6   0  15 139 335 108   7
##                7   0   0   3  53  71  11
##                8   0   0   0   0   0   0
```

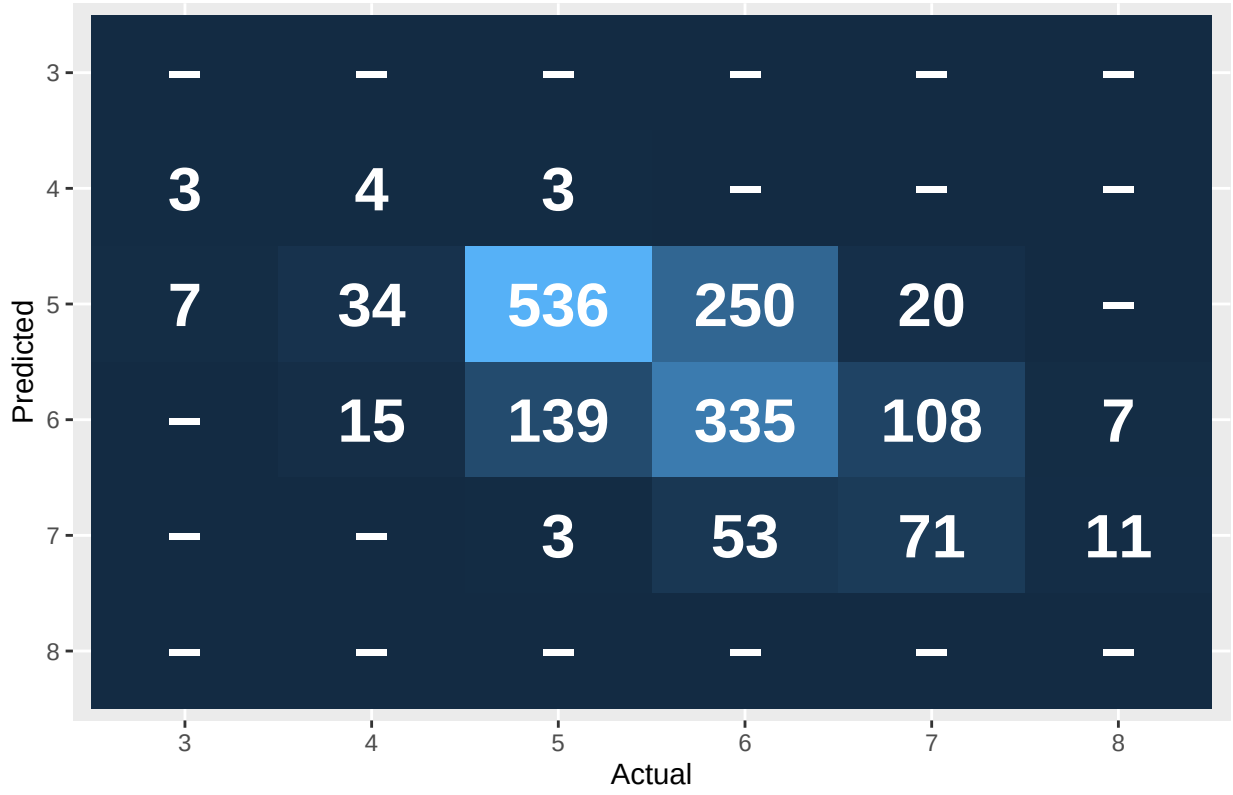
```
q.Tree.Base =
  1 - sum(diag(table(q.Tree.Pred.Base, wineData$quality))) /
  sum(table(q.Tree.Pred.Base, wineData$quality))

q.Tree.Base
```

```
## [1] 0.4083802
```

```
q.Tree.Base.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')), color = 'white',
            size = 8, fontface = 'bold') +
```

```
  labs(title = paste0('Figure 7: Confusion Matrix (Quality), Training Error Rate = ',
                      round(q.Tree.Base * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

Figure 7: Confusion Matrix (Quality), Training Error Rate = 40.84%




```
c.Tree.Pred.Base = predict(c.Tree.Train, wineData, type = 'class')

c.Tree.Base.Table = table(c.Tree.Pred.Base, wineData$category)

c.Tree.Base.Table
```

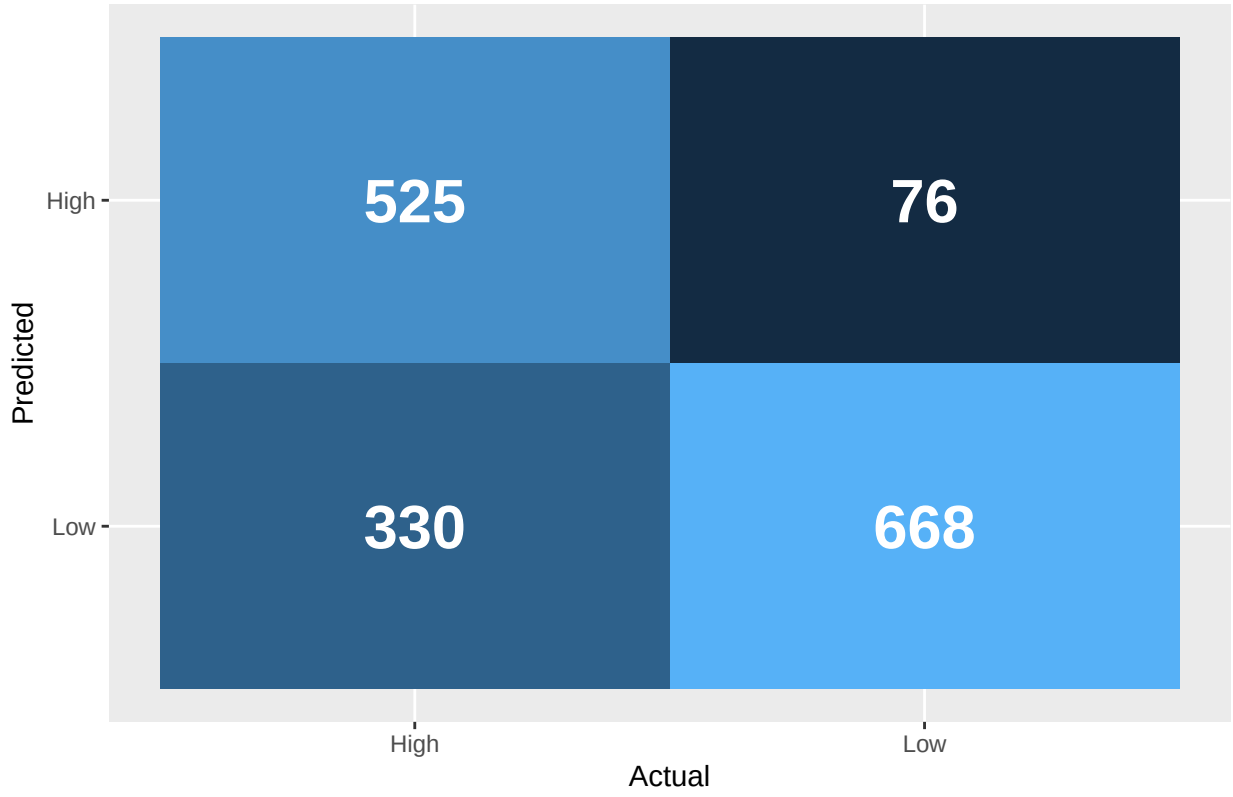
```
##
## c.Tree.Pred.Base High Low
##             High  525  76
##             Low   330 668
```

```
c.Tree.Base =
  1 - sum(diag(table(c.Tree.Pred.Base, wineData$category))) /
  sum(table(c.Tree.Pred.Base, wineData$category))

c.Tree.Base
```

```
## [1] 0.2539087
```

```
c.Tree.Base.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')),
            color = 'white', size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 8: Confusion Matrix (Category), Training Error Rate = ',
                      round(c.Tree.Base * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

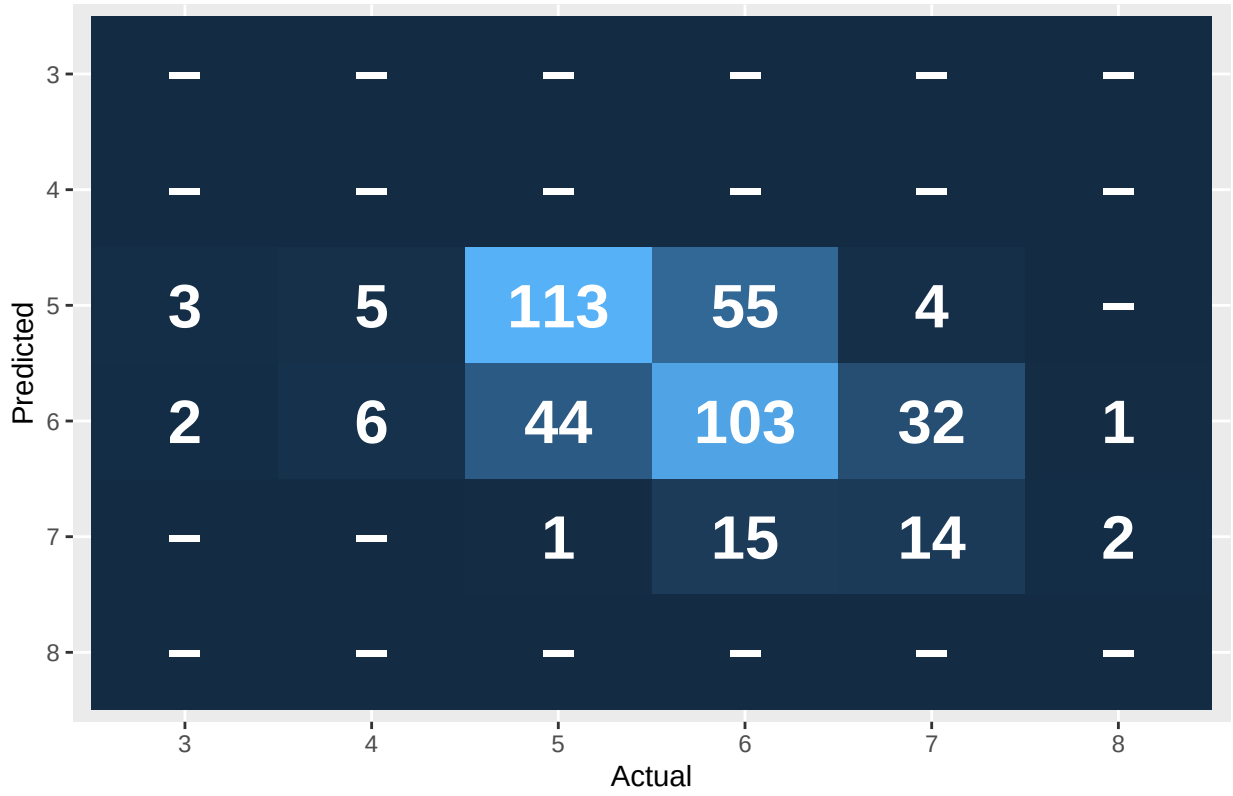
Figure 8: Confusion Matrix (Category), Training Error Rate = 25.39%




```
#Generating Estimates of Test Error for 'out-of-the-box' models

set.seed(422)

trainIndx = sample(1:nrow(wineData), 0.75 * nrow(wineData))

q.Tree.Test = tree(quality ~ . -category, data = wineData[trainIndx, ])
q.Tree.Pred = predict(q.Tree.Test, wineData[-trainIndx, ], type = 'class')
q.Tree.Error = 1 - sum(diag(table(q.Tree.Pred, wineData[-trainIndx, 'quality']))) /
  sum(table(q.Tree.Pred, wineData[-trainIndx, 'quality']))

c.Tree.Test = tree(category ~ . -quality, data = wineData[trainIndx, ])
c.Tree.Pred = predict(c.Tree.Test, wineData[-trainIndx, ], type = 'class')
c.Tree.Error = 1 - sum(diag(table(c.Tree.Pred, wineData[-trainIndx, 'category']))) /
  sum(table(c.Tree.Pred, wineData[-trainIndx, 'category']))
```

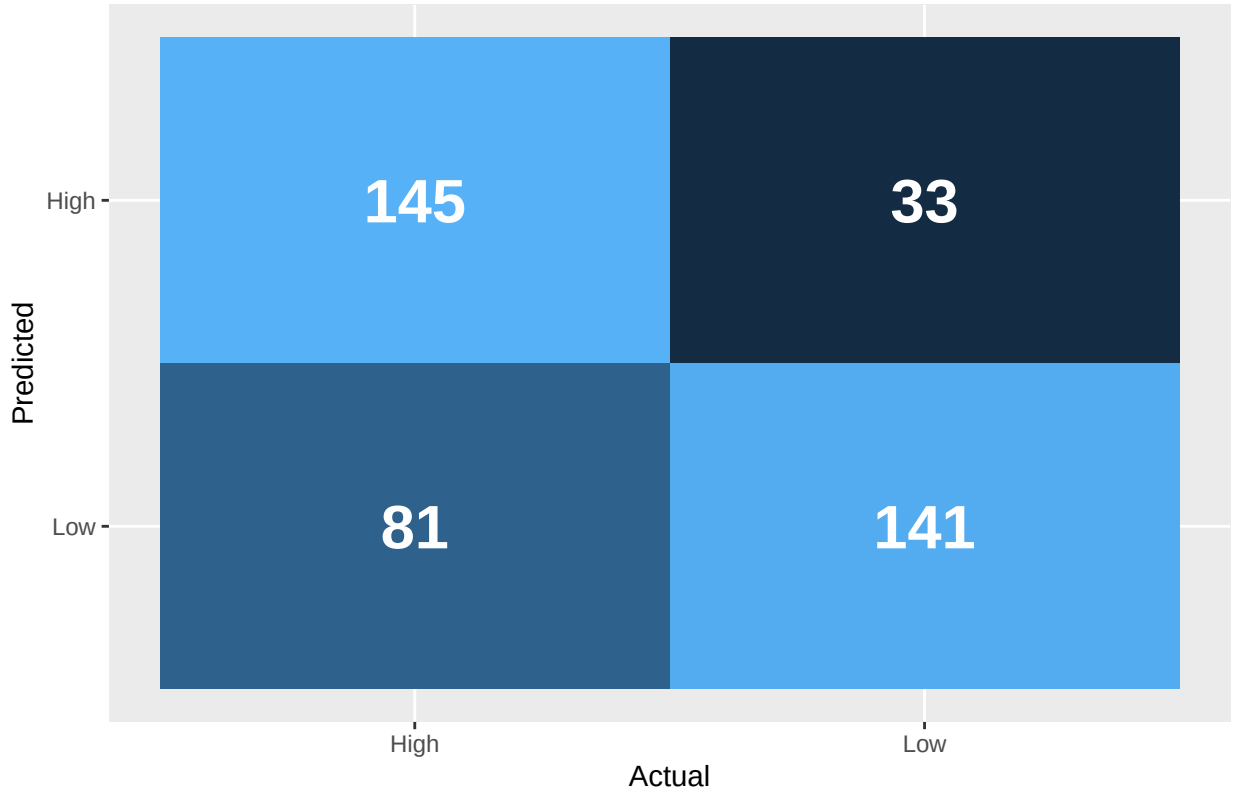
```
q.Tree.Test.Table = table(q.Tree.Pred, wineData[-trainIndx, 'quality'])
q.Tree.Test.Rate = 1 - sum(diag(table(q.Tree.Pred, wineData[-trainIndx, 'quality']))) /
  sum(table(q.Tree.Pred, wineData[-trainIndx, 'quality']))

c.Tree.Test.Table = table(c.Tree.Pred, wineData[-trainIndx, 'category'])
c.Tree.Test.Rate = 1 - sum(diag(table(c.Tree.Pred, wineData[-trainIndx, 'category']))) /
  sum(table(c.Tree.Pred, wineData[-trainIndx, 'category']))

q.Tree.Test.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')), color = 'white',
            size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 9: Confusion Matrix (Quality), Test Error Rate = ',
                      round(q.Tree.Test.Rate * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

Figure 9: Confusion Matrix (Quality), Test Error Rate = 42.5%



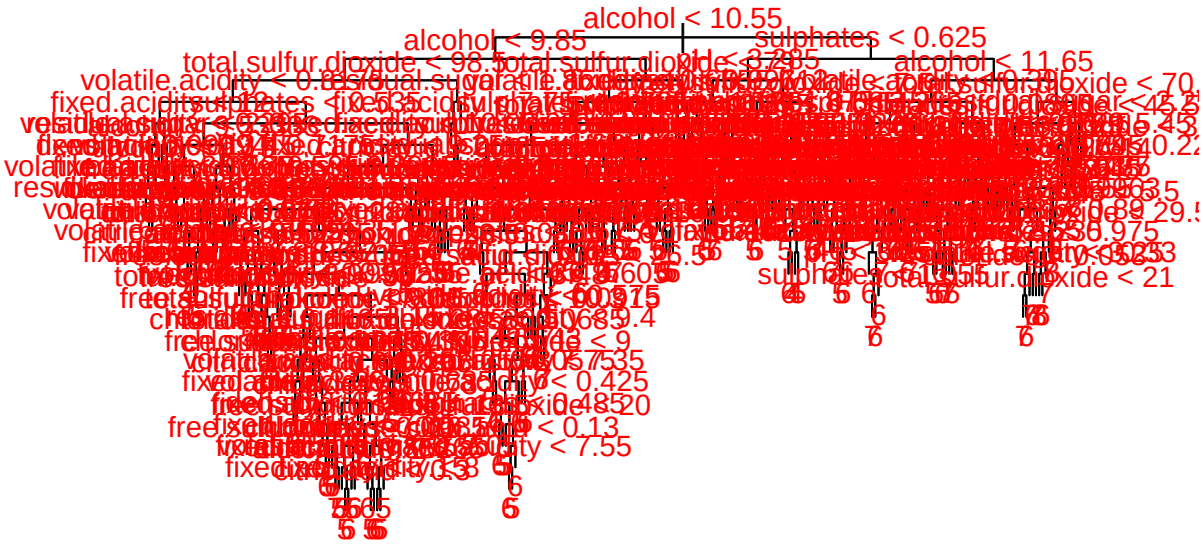

```
c.Tree.Test.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')), color = 'white',
            size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 10: Confusion Matrix (Category), Test Error Rate = ',
                      round(c.Tree.Test.Rate * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

Figure 10: Confusion Matrix (Category), Test Error Rate = 28.5%



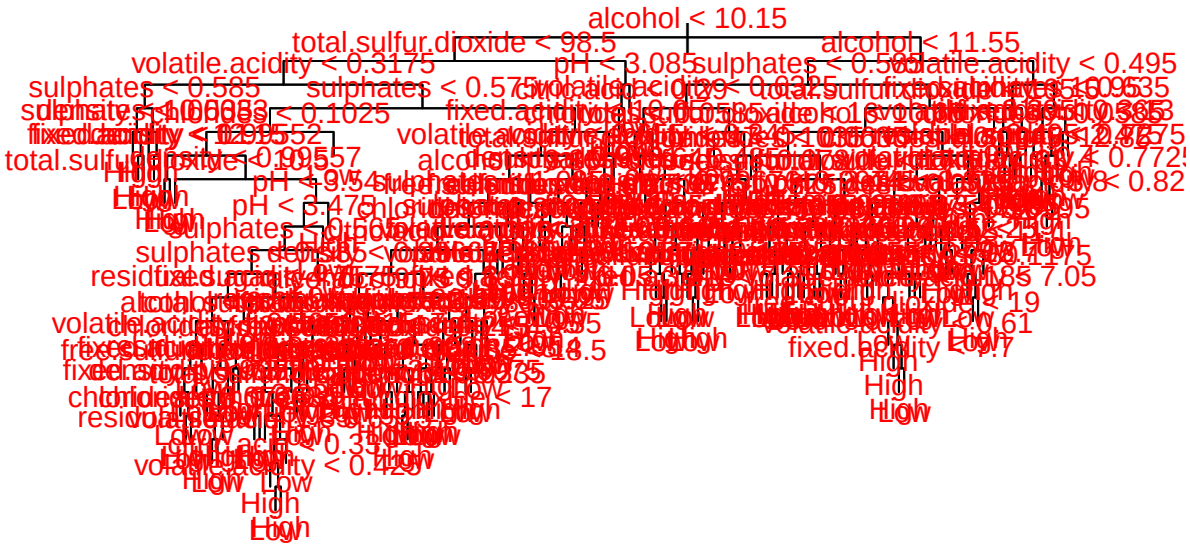

```
# Grow Tree0 and Perform Tree Pruning guided by misclassification error.

q.Tree.0 = tree(quality ~ . -category, data = wineData[trainIndx, ],
                minsize = 2, mindev = 0)
plot(q.Tree.0, type = 'uniform')
text(q.Tree.0, pretty = 1, cex = 0.9, col = 'red')
title('Figure 7: Maximal Complexity Tree (Pred. Quality)')
```

Figure 7: Maximal Complexity Tree (Pred. Quality)



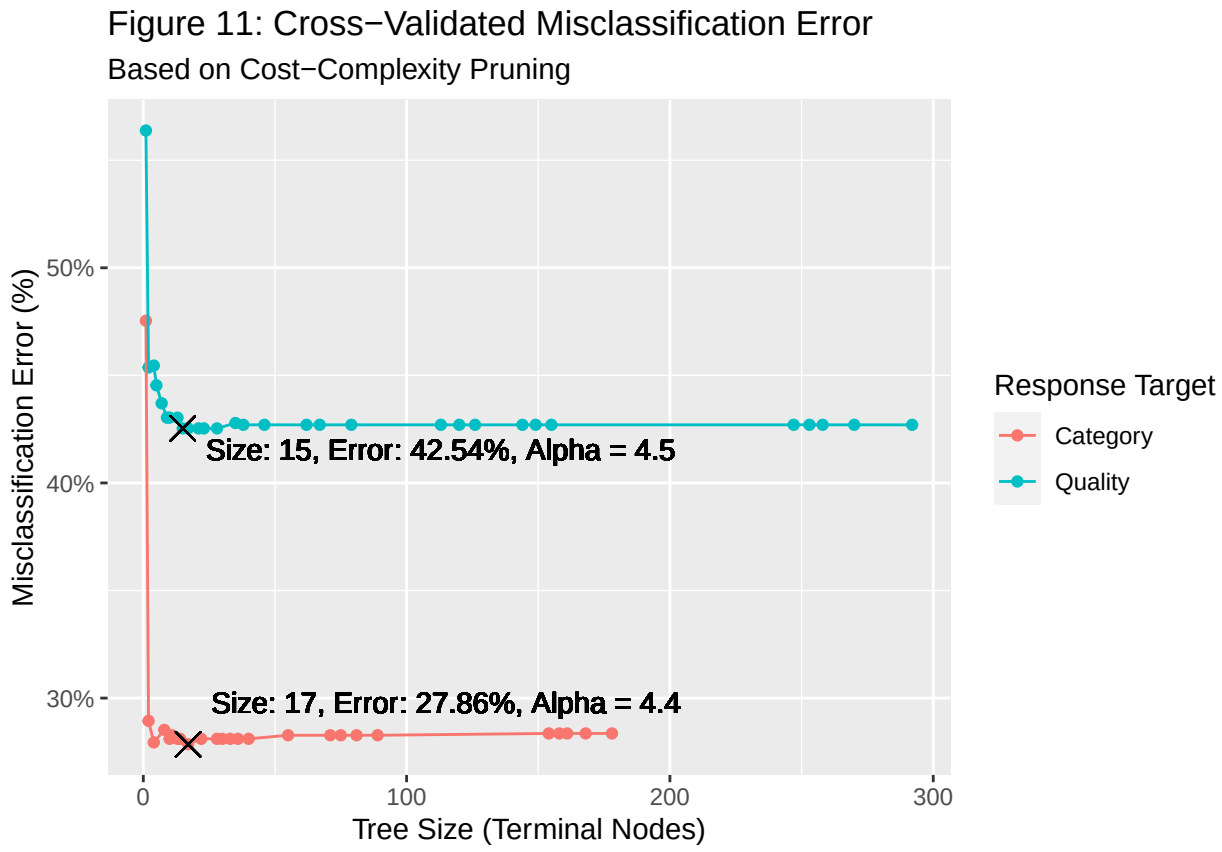

```
c.Tree.0 = tree(category ~ . -quality, data = wineData[trainIndx, ],
                minsize = 2, mindev = 0)
plot(c.Tree.0, type = 'uniform')
text(c.Tree.0, pretty = 1, cex = 0.9, col = 'red')
title('Figure 8: Maximal Complexity Tree (Pred. Category)')
```

Figure 8: Maximal Complexity Tree (Pred. Category)




```
set.seed(422)

q.Tree.CV = cv.tree(q.Tree.0, FUN = prune.misclass)
c.Tree.CV = cv.tree(c.Tree.0, FUN = prune.misclass)

q.CV.Tree.Params = data.frame(q.Tree.CV$size, q.Tree.CV$dev, q.Tree.CV$k) %>%
  setNames(., c('size', 'misclass', 'alpha')) %>%
  mutate(error = misclass / length(trainIndx)) %>%
  mutate(target = 'Quality')

c.CV.Tree.Params = data.frame(c.Tree.CV$size, c.Tree.CV$dev, c.Tree.CV$k) %>%
  setNames(., c('size', 'misclass', 'alpha')) %>%
  mutate(error = misclass / length(trainIndx)) %>%
  mutate(target = 'Category')

q.min.error = q.CV.Tree.Params %>% filter(error == min(error)) %>%
  filter(size == min(size))
c.min.error = c.CV.Tree.Params %>% filter(error == min(error)) %>%
  filter(size == min(size))

rbind(q.CV.Tree.Params, c.CV.Tree.Params) %>% ggplot() +
  geom_point(aes(x = size, y = error, color = target)) +
  geom_line(aes(x = size, y = error, color = target)) +
  geom_point(aes(x = q.min.error$size, y = q.min.error$error), shape = 4, size = 4) +
  geom_text(aes(label = paste0('Size: ', q.min.error$size, ', Error: ',
                               round(q.min.error$error * 100, 2), '%, Alpha = ',
```

```
                              round(q.min.error$alpha, 2))),
          x = q.min.error$size, y = q.min.error$error, hjust = -0.05, vjust = 1.5) +
  geom_point(aes(x = c.min.error$size, y = c.min.error$error), shape = 4, size = 4) +
  geom_text(aes(label = paste0('Size: ', c.min.error$size, ', Error: ',
                              round(c.min.error$error * 100, 2), '%, Alpha = ',
                              round(c.min.error$alpha, 2))),
          x = c.min.error$size, y = c.min.error$error, hjust = -0.05, vjust = -1.5) +
  scale_y_continuous(labels = scales::percent) +
  labs(x = 'Tree Size (Terminal Nodes)', y = 'Misclassification Error (%)',
       title = 'Figure 11: Cross-Validated Misclassification Error',
       subtitle = 'Based on Cost-Complexity Pruning', color = 'Response Target')
```




```
# Estimate test error for pruned trees based on optimal parameters

q.Tree.Prune = prune.misclass(q.Tree.0, best = as.numeric(q.min.error$size))

q.Prune.Pred = predict(q.Tree.Prune, wineData[-trainIndx, ], type = 'class')

table(q.Prune.Pred, wineData[-trainIndx, 'quality'])
```

```
##
## q.Prune.Pred   3   4   5   6   7   8
##            3   0   0   0   0   0   0
##            4   0   0   0   0   0   0
##            5   5   6 113  65   6   0
```

```
##            6   0   4  43  90  34   2
##            7   0   1   2  18  10   1
##            8   0   0   0   0   0   0
```

```
q.Prune.Test.Error =
  1 - sum(diag(table(q.Prune.Pred, wineData[-trainIndx, 'quality']))) /
  sum(table(q.Prune.Pred, wineData[-trainIndx, 'quality']))

q.Prune.Test.Error
```

```
## [1] 0.4675
```

```
c.Tree.Prune = prune.misclass(c.Tree.0, best = as.numeric(c.min.error$size))

c.Prune.Pred = predict(c.Tree.Prune, wineData[-trainIndx, ], type = 'class')

table(c.Prune.Pred, wineData[-trainIndx, 'category'])
```

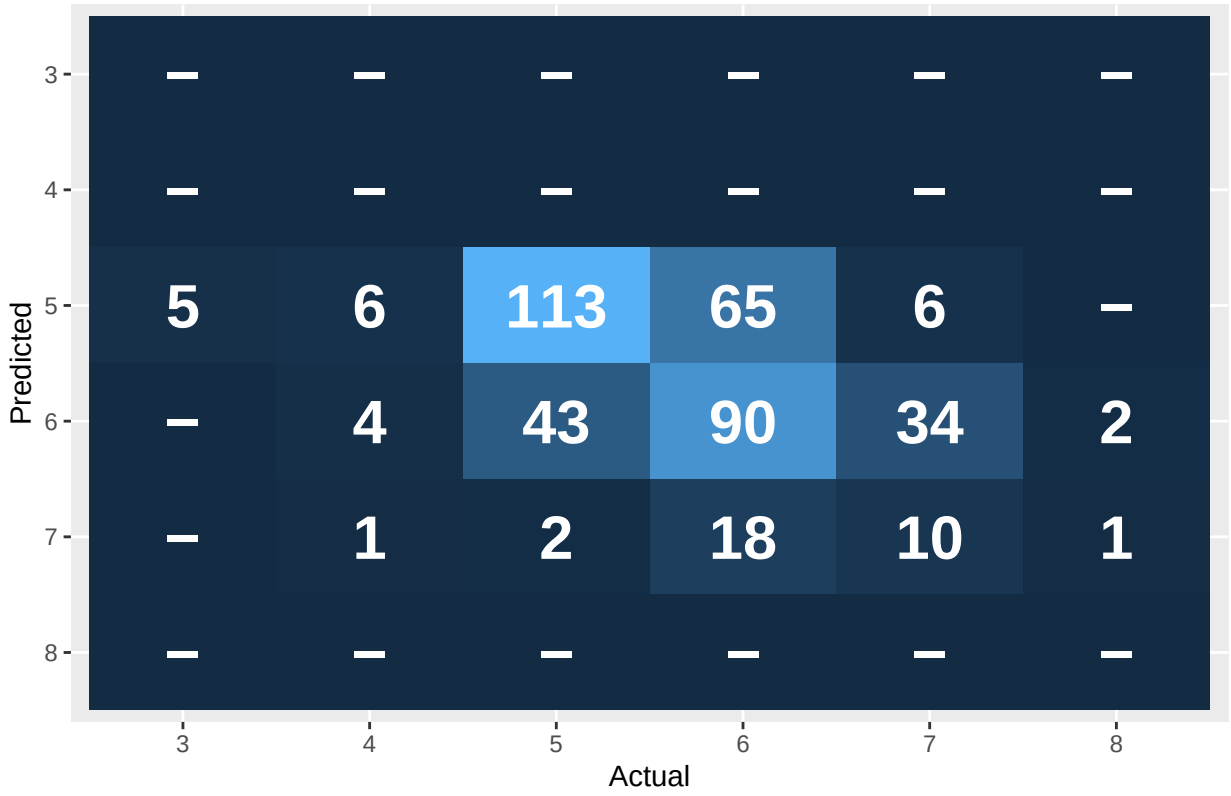
```
##
## c.Prune.Pred High Low
##         High  166  43
##         Low    60 131
```

```
c.Prune.Test.Error =
  1 - sum(diag(table(c.Prune.Pred, wineData[-trainIndx, 'category']))) /
  sum(table(c.Prune.Pred, wineData[-trainIndx, 'category']))

c.Prune.Test.Error
```

```
## [1] 0.2575
```

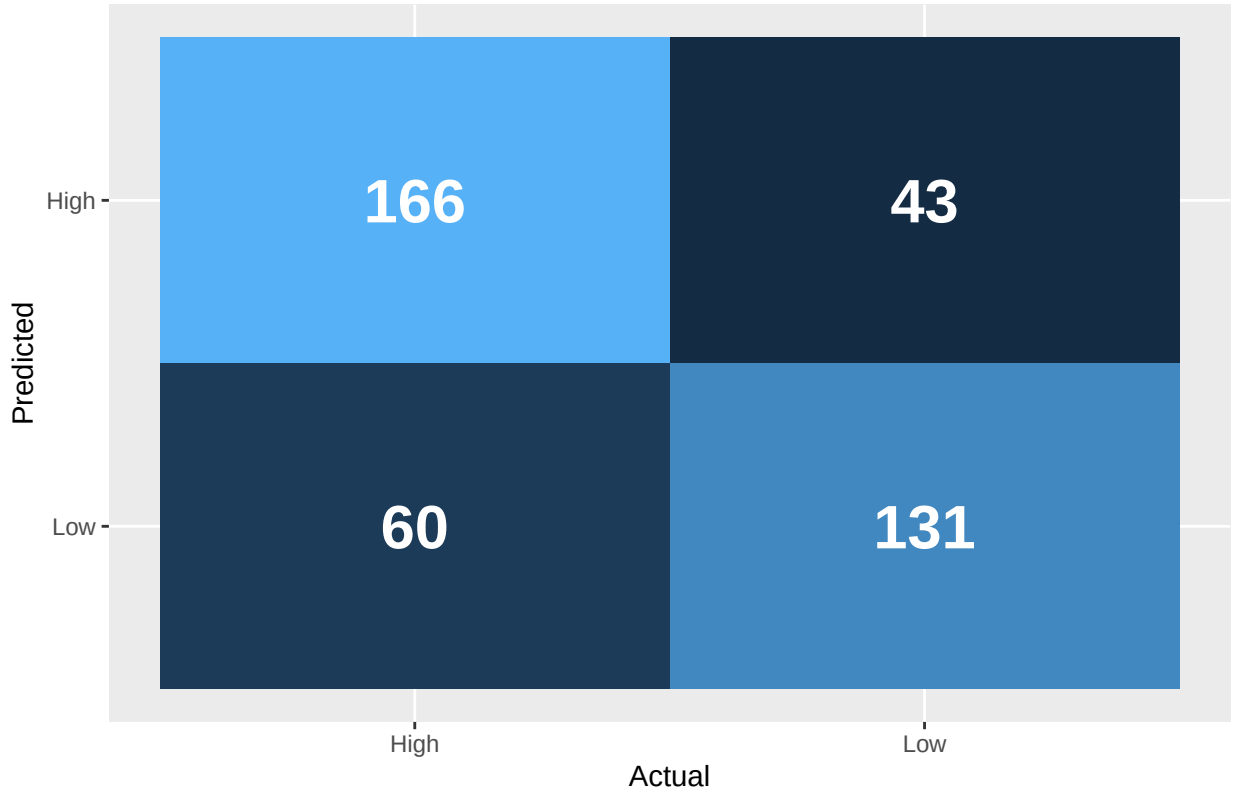
```
q.Tree.Prune.Table = table(q.Prune.Pred, wineData[-trainIndx, 'quality'])
q.Prune.Test.Rate = 1 - sum(diag(table(q.Prune.Pred, wineData[-trainIndx, 'quality']))) /
  sum(table(q.Prune.Pred, wineData[-trainIndx, 'quality']))

c.Tree.Prune.Table = table(c.Prune.Pred, wineData[-trainIndx, 'category'])
c.Prune.Test.Rate = 1 - sum(diag(table(c.Prune.Pred, wineData[-trainIndx, 'category']))) /
  sum(table(c.Prune.Pred, wineData[-trainIndx, 'category']))

q.Tree.Prune.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')),
            color = 'white', size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 12: Confusion Matrix (Quality), Pruned Test Error Rate = ',
                      round(q.Prune.Test.Rate * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

Figure 12: Confusion Matrix (Quality), Pruned Test Error Rate = 46.75%



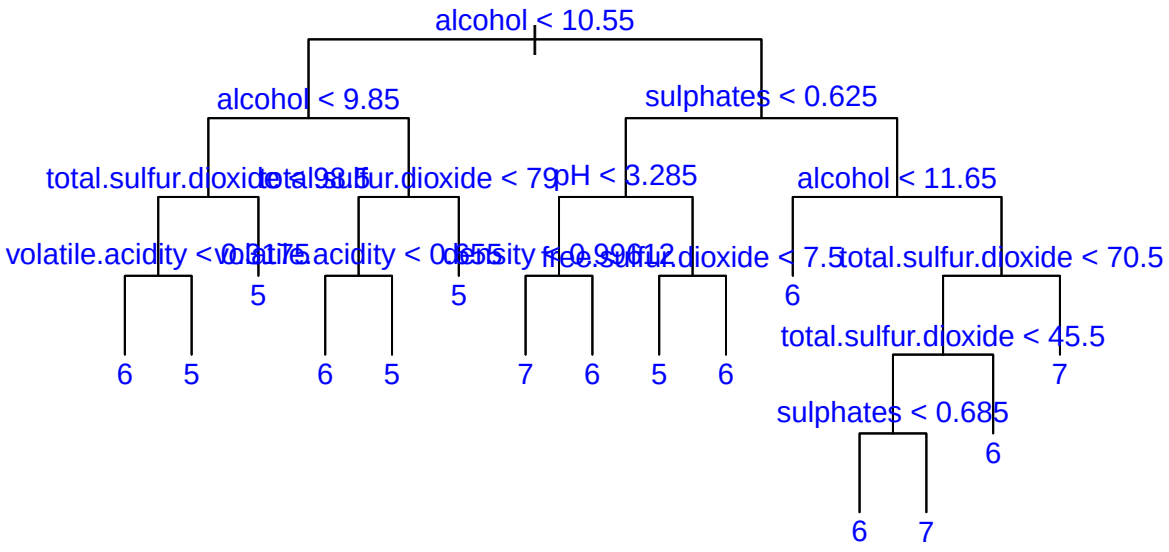

```
c.Tree.Prune.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')), color = 'white',
            size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 13: Confusion Matrix (Category), Pruned Test Error Rate = ',
                      round(c.Prune.Test.Rate * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

Figure 13: Confusion Matrix (Category), Pruned Test Error Rate = 25.75%



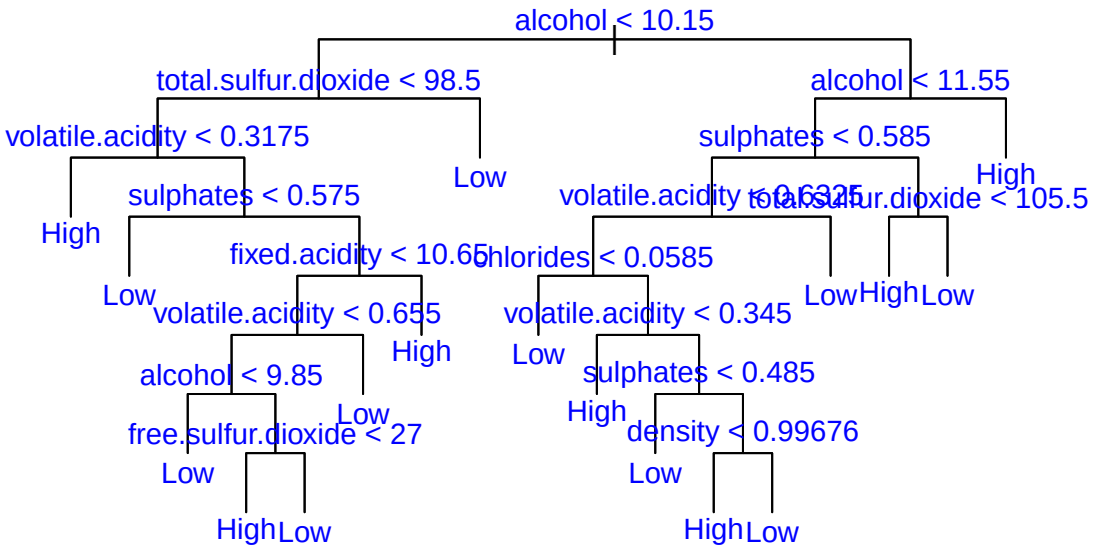

```
# Plot Pruned Trees - Omitted from Project Report

plot(q.Tree.Prune, type = 'uniform')
text(q.Tree.Prune, pretty = 1, cex = 0.9, col = 'blue')
title('Figure 14: Pruned Tree (Pred. Quality)')
```

Figure 14: Pruned Tree (Pred. Quality)



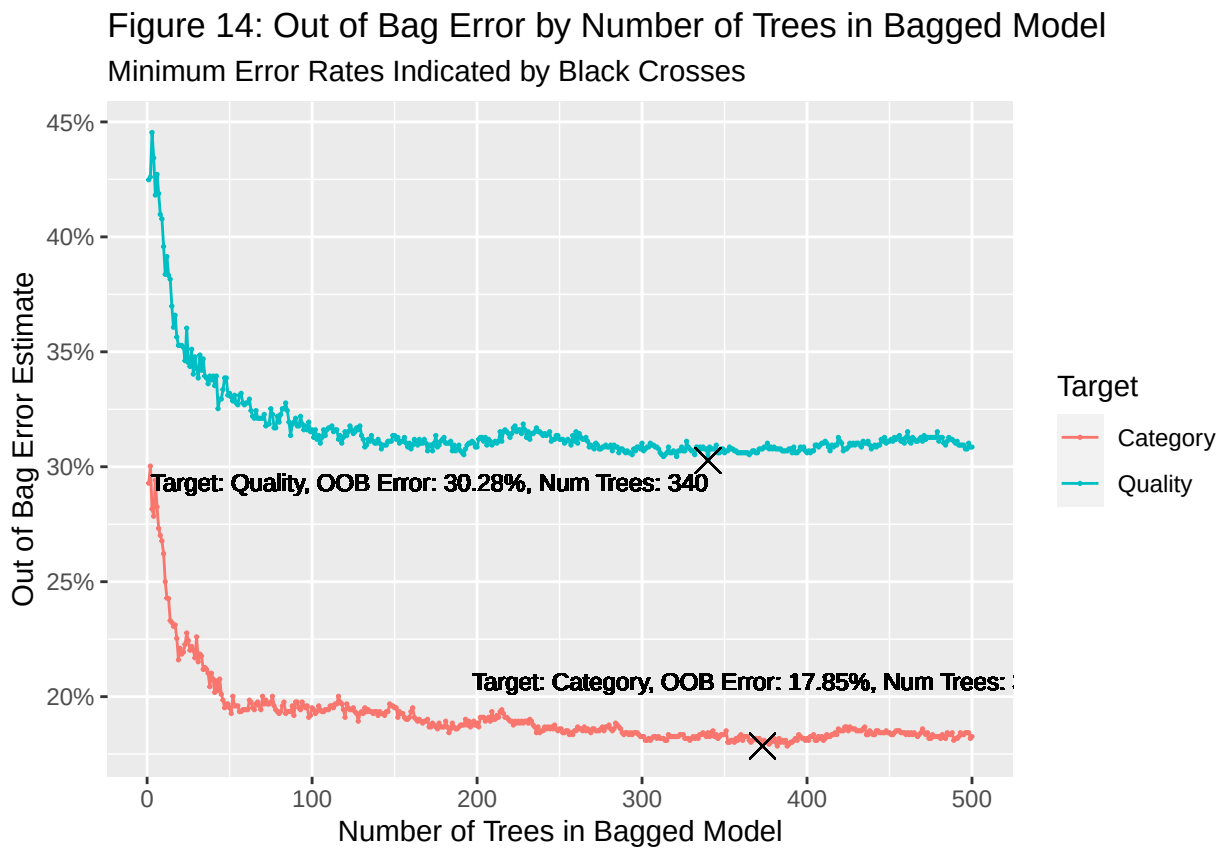

```
plot(c.Tree.Prune, type = 'uniform')
text(c.Tree.Prune, pretty = 1, cex = 0.9, col = 'blue')
title('Figure 15: Pruned Tree (Pred. Category)')
```

Figure 15: Pruned Tree (Pred. Category)




```
# Construction of Bagged Models Using OOB Error estimate as a function of B

library(randomForest)

set.seed(888)

q.Bag.Model = randomForest(quality ~ . -category, wineData[trainIndx, ],
                           mtry = 11, importance = TRUE, ntrees = 500)


c.Bag.Model = randomForest(category ~ . -quality, wineData[trainIndx, ],
                           mtry = 11, importance = TRUE, ntrees = 500)

bag.Error.Data = q.Bag.Model$err.rate %>% data.frame() %>% select(OOB) %>%
  mutate(Target = 'Quality') %>% mutate(Trees = 1:500) %>%
  rbind(c.Bag.Model$err.rate %>% data.frame() %>% select(OOB) %>%
          mutate(Target = 'Category') %>% mutate(Trees = 1:500))

q.bag.min.error = bag.Error.Data %>% filter(Target == 'Quality') %>%
  filter(OOB == min(OOB)) %>% filter(Trees == min(Trees))

c.bag.min.error = bag.Error.Data %>% filter(Target == 'Category') %>%
  filter(OOB == min(OOB)) %>% filter(Trees == min(Trees))

bag.Error.Data %>% ggplot() +
  geom_point(aes(x = Trees, y = OOB, color = Target), size = 0.25) +
```

```
  geom_line(aes(x = Trees, y = OOB, color = Target)) +
  geom_point(aes(x = q.bag.min.error$Trees, y = q.bag.min.error$OOB),
             shape = 4, size = 4) +
  geom_text(aes(label = paste0('Target: ', q.bag.min.error$Target,
                               ', OOB Error: ', round(q.bag.min.error$OOB * 100, 2),
                               '%, Num Trees: ', q.bag.min.error$Trees)),
            x = q.bag.min.error$Trees, y = q.bag.min.error$OOB,
            vjust = 1.75, hjust = 'inward', size = 3) +
  geom_point(aes(x = c.bag.min.error$Trees, y = c.bag.min.error$OOB),
             shape = 4, size = 4) +
  geom_text(aes(label = paste0('Target: ', c.bag.min.error$Target,
                               ', OOB Error: ', round(c.bag.min.error$OOB * 100, 2),
                               '%, Num Trees: ', c.bag.min.error$Trees)),
            x = c.bag.min.error$Trees, y = c.bag.min.error$OOB,
            vjust = -3.25, size = 3) +
  scale_y_continuous(labels = scales::percent) +
  labs(x = 'Number of Trees in Bagged Model', y = 'Out of Bag Error Estimate',
       title = 'Figure 14: Out of Bag Error by Number of Trees in Bagged Model',
       subtitle = 'Minimum Error Rates Indicated by Black Crosses')
```




```
# Generating Test Error Rate using Optimal Parameter for each target.

set.seed(345)

q.Bag.Test.Model = randomForest(quality ~ . -category, wineData[trainIndx, ],
```

```
                                mtry = 11, ntrees = as.numeric(q.bag.min.error$Trees))

c.Bag.Test.Model = randomForest(category ~ . -quality, wineData[trainIndx, ],
                                mtry = 11, ntrees = as.numeric(c.bag.min.error$Trees))

q.Bag.Test.Preds = predict(q.Bag.Test.Model, wineData[-trainIndx, ])

c.Bag.Test.Preds = predict(c.Bag.Test.Model, wineData[-trainIndx, ])

table(q.Bag.Test.Preds, wineData[-trainIndx, 'quality'])
```

```
##
## q.Bag.Test.Preds   3   4   5   6   7   8
##                3   0   0   0   0   0   0
##                4   0   0   0   0   0   0
##                5   5   6 126  43   4   0
##                6   0   4  32 114  25   0
##                7   0   1   0  15  21   2
##                8   0   0   0   1   0   1
```

```
q.Bag.Test.Error =
  1 - sum(diag(table(q.Bag.Test.Preds, wineData[-trainIndx, 'quality']))) /
  sum(table(q.Bag.Test.Preds, wineData[-trainIndx, 'quality']))

q.Bag.Test.Error
```

```
## [1] 0.345
```

```
table(c.Bag.Test.Preds, wineData[-trainIndx, 'category'])
```

```
##
## c.Bag.Test.Preds High Low
##             High  179  35
##             Low    47 139
```

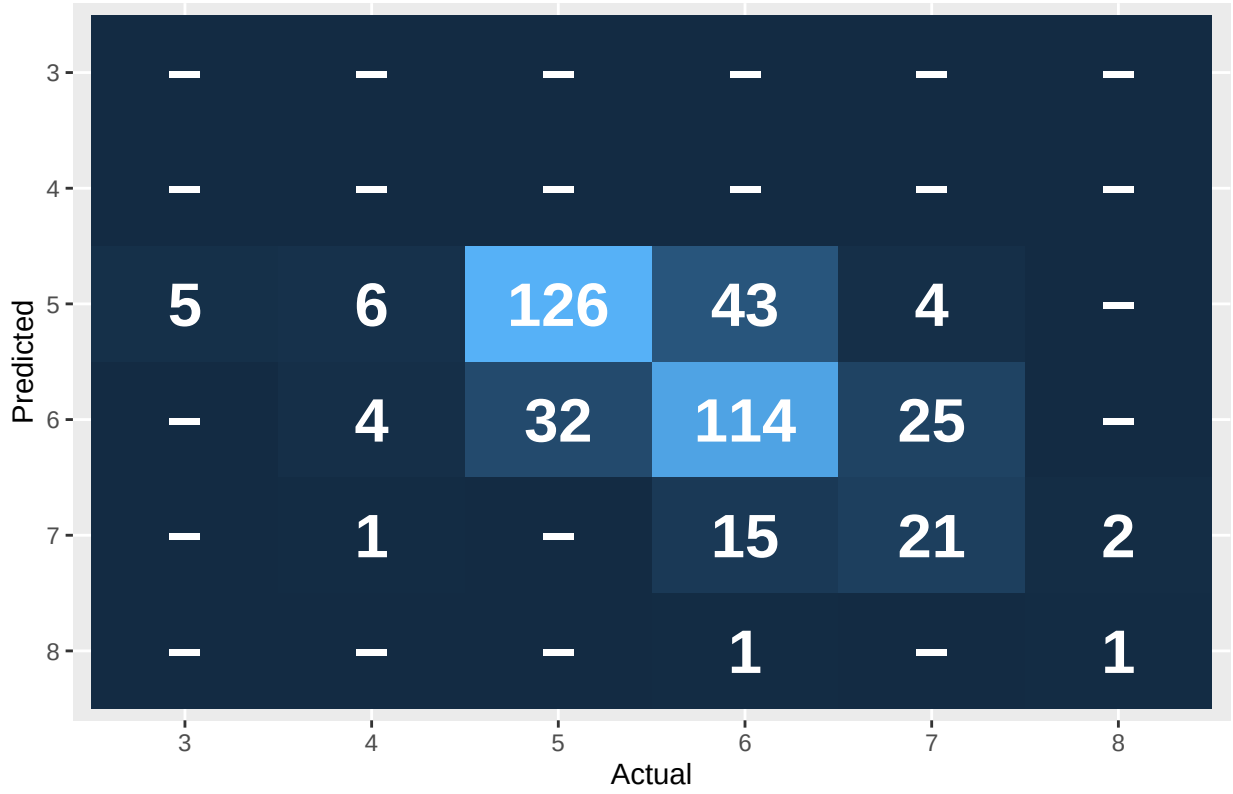
```
c.Bag.Test.Error =
  1 - sum(diag(table(c.Bag.Test.Preds, wineData[-trainIndx, 'category']))) /
  sum(table(c.Bag.Test.Preds, wineData[-trainIndx, 'category']))

c.Bag.Test.Error
```

```
## [1] 0.205
```

```
q.Bag.Test.Table = table(q.Bag.Test.Preds, wineData[-trainIndx, 'quality'])
c.Bag.Test.Table = table(c.Bag.Test.Preds, wineData[-trainIndx, 'category'])

q.Bag.Test.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')),
```

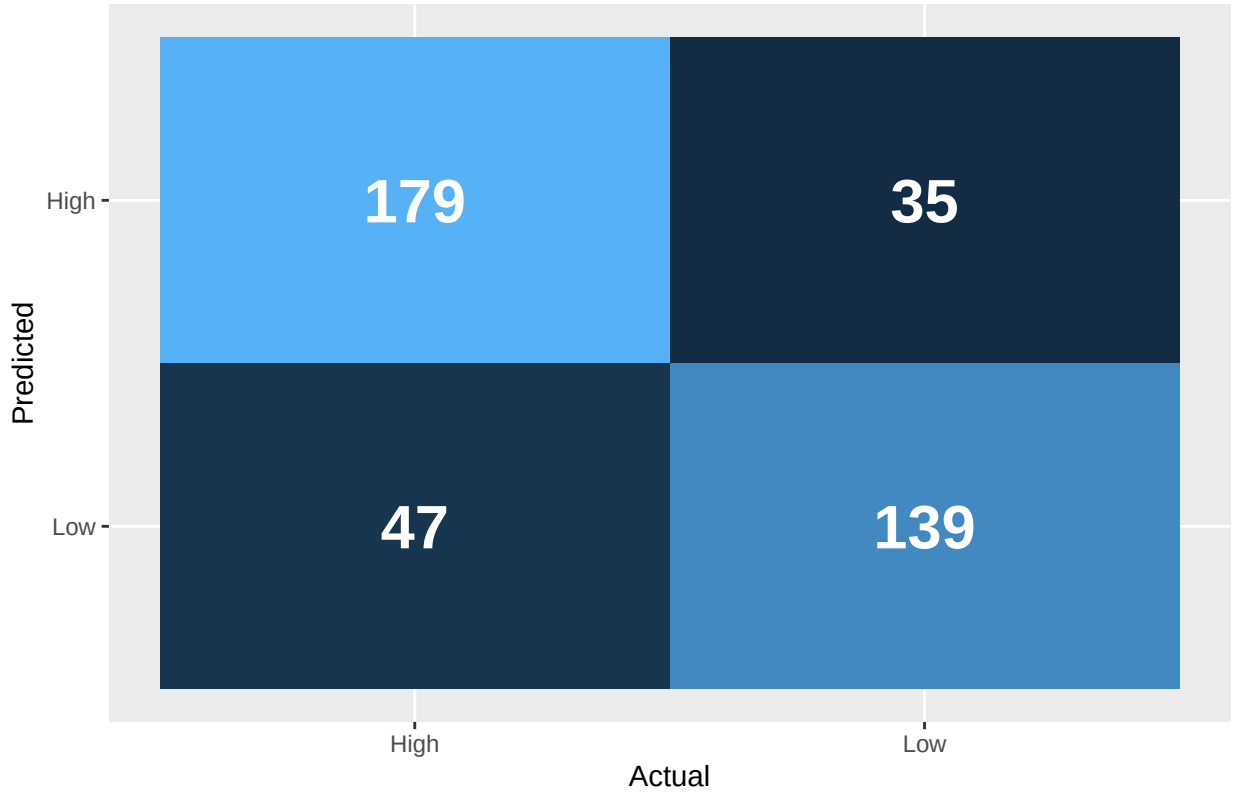
```
            color = 'white', size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 15: Confusion Matrix (Quality), Bagged Test Error Rate = ',
                      round(q.Bag.Test.Error * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

Figure 15: Confusion Matrix (Quality), Bagged Test Error Rate = 34.5%




```
c.Bag.Test.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')),
            color = 'white', size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 16: Confusion Matrix (Category), Bagged Test Error Rate = ',
                      round(c.Bag.Test.Error * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

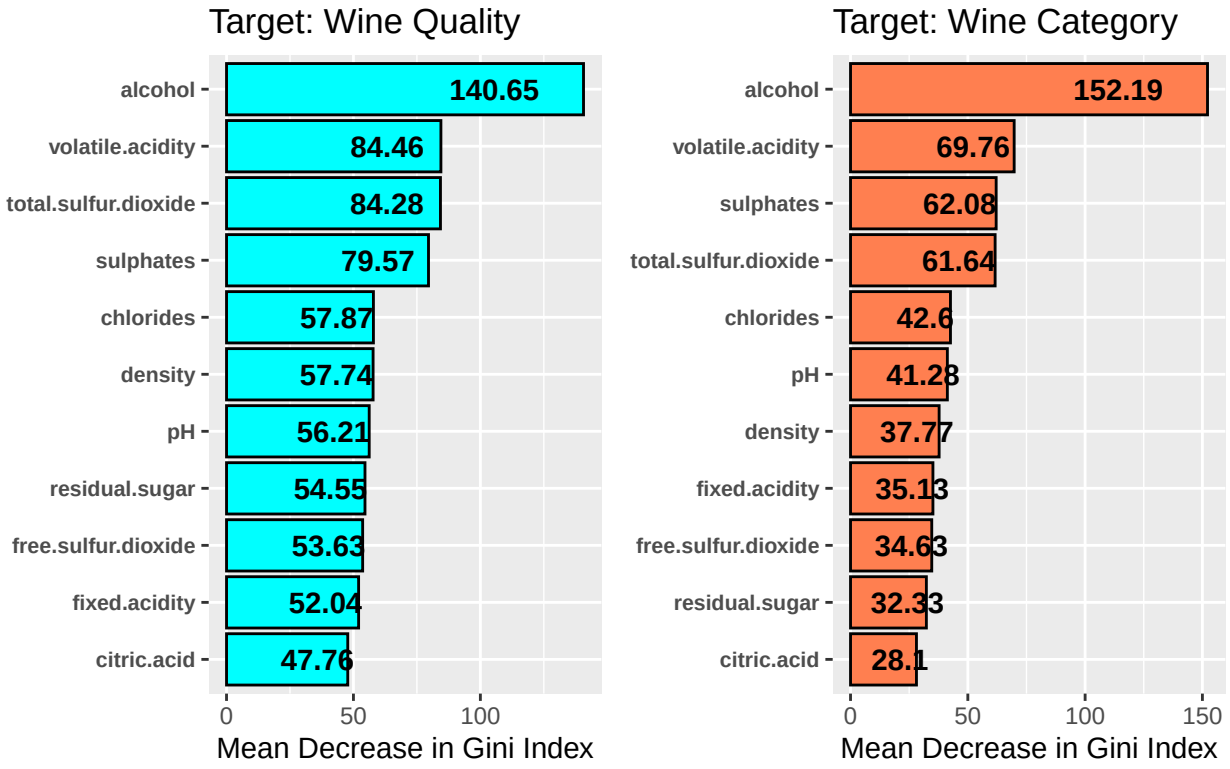
Figure 16: Confusion Matrix (Category), Bagged Test Error Rate = 20.5%




```
# Generate variable importance plots for the bagged models.

qualImp = importance(q.Bag.Test.Model) %>% data.frame() %>%
  rownames_to_column() %>%
  mutate_if(is.numeric, round, 2) %>%
  ggplot(aes(x = MeanDecreaseGini, y = reorder(rowname, MeanDecreaseGini))) +
  geom_bar(stat = 'identity', fill = 'cyan', color = 'black') +
  geom_text(aes(label = MeanDecreaseGini, fontface = 'bold'),
            position = position_stack(vjust = 0.75)) +
  labs(title = 'Target: Wine Quality',
       x = 'Mean Decrease in Gini Index') +
  theme(axis.text.y = element_text(size = 8, face = 'bold'),
        axis.title.y = element_blank())

catImp = importance(c.Bag.Test.Model) %>% data.frame() %>%
  rownames_to_column() %>%
  mutate_if(is.numeric, round, 2) %>%
  ggplot(aes(x = MeanDecreaseGini, y = reorder(rowname, MeanDecreaseGini))) +
  geom_bar(stat = 'identity', fill = 'coral', color = 'black') +
  geom_text(aes(label = MeanDecreaseGini, fontface = 'bold'),
            position = position_stack(vjust = 0.75)) +
  labs(title = 'Target: Wine Category',
       x = 'Mean Decrease in Gini Index') +
  theme(axis.text.y = element_text(size = 8, face = 'bold'),
        axis.title.y = element_blank())
```

```
grdTitl = ggdraw() + draw_label('Figure 17: Predictor Importance x Target (Bagged Models)')
impGrid = plot_grid(qualImp, catImp, nrow = 1)

plot_grid(grdTitl, impGrid, ncol = 1, rel_heights=c(0.1, 1))
```

Figure 17: Predictor Importance x Target (Bagged Models)

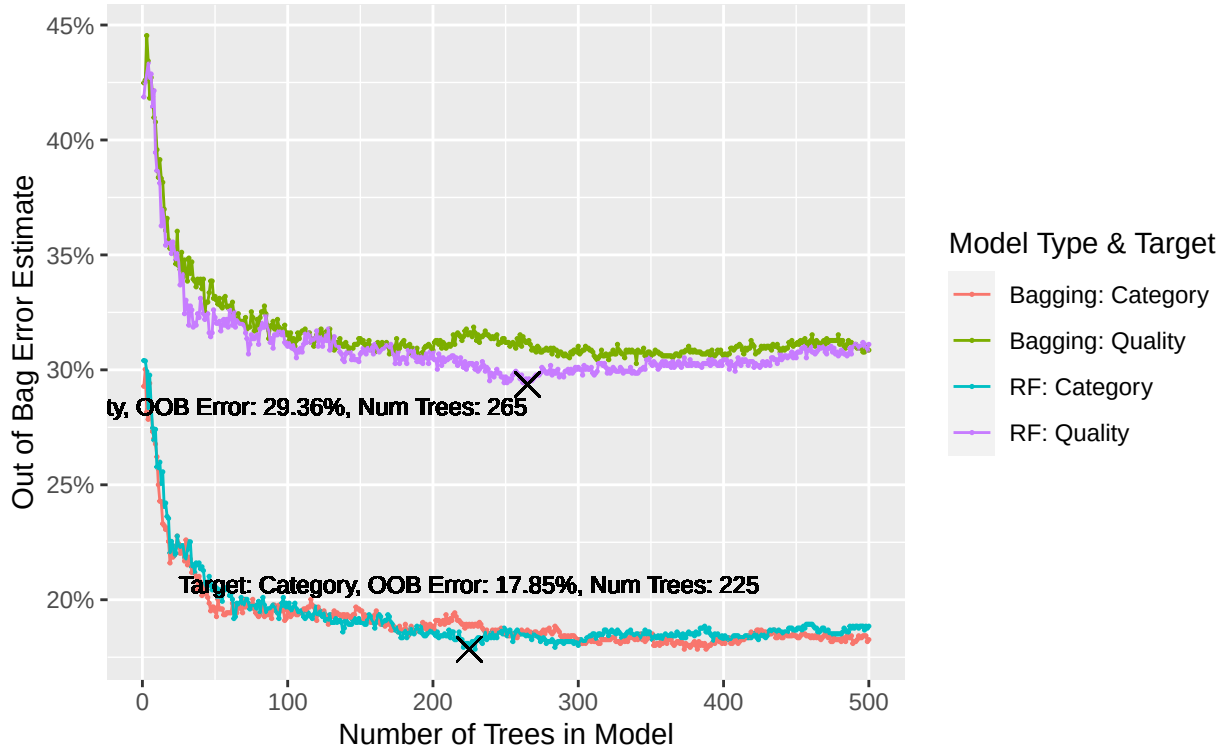



```
# Construction of Random Forest Models and Search for B

set.seed(456)

q.RF.Model = randomForest(quality ~ . -category, wineData[trainIndx, ],
                          mtry = sqrt(11), ntrees = 500)

c.RF.Model = randomForest(category ~ . -quality, wineData[trainIndx, ],
                          mtry = sqrt(11), ntrees = 500)

rf.Error.Data = q.RF.Model$err.rate %>% data.frame() %>% select(OOB) %>%
  mutate(Target = 'Quality') %>% mutate(Trees = 1:500) %>%
  rbind(c.RF.Model$err.rate %>% data.frame() %>% select(OOB) %>%
          mutate(Target = 'Category') %>% mutate(Trees = 1:500)) %>%
  mutate(Model = ifelse(Target == 'Quality', 'RF: Quality', 'RF: Category'))

q.RF.min.error = rf.Error.Data %>% filter(Target == 'Quality') %>%
  filter(OOB == min(OOB)) %>% filter(Trees == min(Trees))

c.RF.min.error = rf.Error.Data %>% filter(Target == 'Category') %>%
```

```
  filter(OOB == min(OOB)) %>% filter(Trees == min(Trees))

all.Error.Data = bag.Error.Data %>%
  mutate(Model = ifelse(Target == 'Quality', 'Bagging: Quality',
                        'Bagging: Category')) %>%
  rbind(rf.Error.Data) %>% ggplot() +
  geom_point(aes(x = Trees, y = OOB, color = Model), size = 0.25) +
  geom_line(aes(x = Trees, y = OOB, color = Model)) +
  geom_point(aes(x = q.RF.min.error$Trees, y = q.RF.min.error$OOB),
             shape = 4, size = 4) +
  geom_text(aes(label = paste0('Target: ', q.RF.min.error$Target,
                               ', OOB Error: ', round(q.RF.min.error$OOB * 100, 2),
                               '%, Num Trees: ', q.RF.min.error$Trees)),
            x = q.RF.min.error$Trees, y = q.RF.min.error$OOB,
            vjust = 1.75, hjust = 'inward', size = 3, nudge_x = -0.5) +
  geom_point(aes(x = c.RF.min.error$Trees, y = c.RF.min.error$OOB),
             shape = 4, size = 4) +
  geom_text(aes(label = paste0('Target: ', c.RF.min.error$Target,
                               ', OOB Error: ', round(c.RF.min.error$OOB * 100, 2),
                               '%, Num Trees: ', c.RF.min.error$Trees)),
            x = c.RF.min.error$Trees, y = c.RF.min.error$OOB,
            vjust = -3.25, size = 3, nudge_x = -3) +
  scale_y_continuous(labels = scales::percent) +
  labs(x = 'Number of Trees in Model', y = 'Out of Bag Error Estimate',
       title = 'Figure 18: OOB Error Estimate by Number of Trees',
       subtitle = 'Bagged vs. Random Forest Models (Crosses for RF Minimums)',
       color = 'Model Type & Target')

all.Error.Data
```

Figure 18: OOB Error Estimate by Number of Trees

Bagged vs. Random Forest Models (Crosses for RF Minimums)




```
# Generate Test Error Estimates for Random Forest Models

set.seed(777)

q.RF.Test.Model = randomForest(quality ~ . -category, wineData[trainIndx, ],
                               mtry = sqrt(11),
                              ntrees = as.numeric(q.RF.min.error$Trees))

c.RF.Test.Model = randomForest(category ~ . -quality, wineData[trainIndx, ],
                               mtry = sqrt(11),
                              ntrees = as.numeric(c.RF.min.error$Trees))

q.RF.Test.Preds = predict(q.RF.Test.Model, wineData[-trainIndx, ])

c.RF.Test.Preds = predict(c.RF.Test.Model, wineData[-trainIndx, ])

table(q.RF.Test.Preds, wineData[-trainIndx, 'quality'])
```

```
##
## q.RF.Test.Preds   3   4   5   6   7   8
##               3   0   0   0   0   0   0
##               4   0   0   0   0   0   0
##               5   4   7 127  44   4   0
##               6   1   3  31 116  26   0
##               7   0   1   0  13  20   2
##               8   0   0   0   0   0   1
```

```
q.RF.Test.Error =
  1 - sum(diag(table(q.RF.Test.Preds, wineData[-trainIndx, 'quality']))) /
  sum(table(q.RF.Test.Preds, wineData[-trainIndx, 'quality']))

q.RF.Test.Error
```

```
## [1] 0.34
```

```
table(c.RF.Test.Preds, wineData[-trainIndx, 'category'])
```

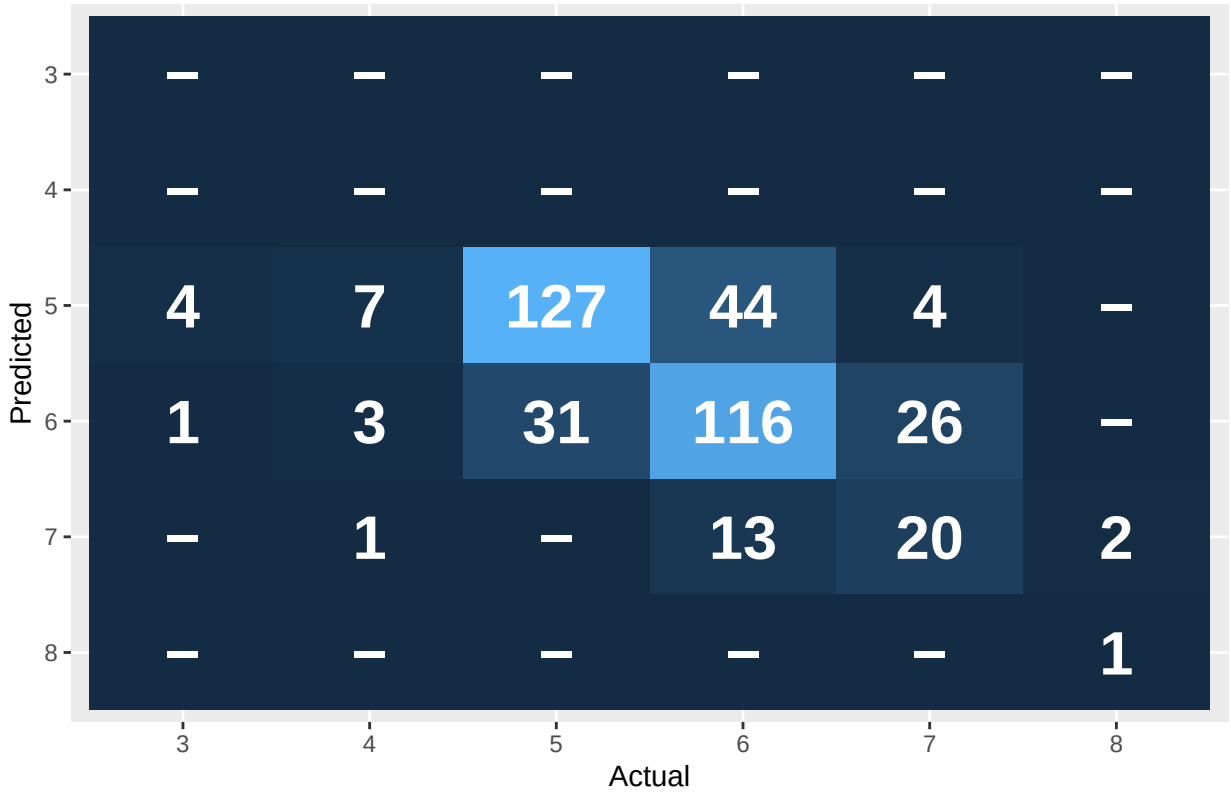
```
##
## c.RF.Test.Preds High Low
##            High  182  38
##            Low    44 136
```

```
c.RF.Test.Error =
  1 - sum(diag(table(c.RF.Test.Preds, wineData[-trainIndx, 'category']))) /
  sum(table(c.RF.Test.Preds, wineData[-trainIndx, 'category']))

c.RF.Test.Error
```

```
## [1] 0.205
```

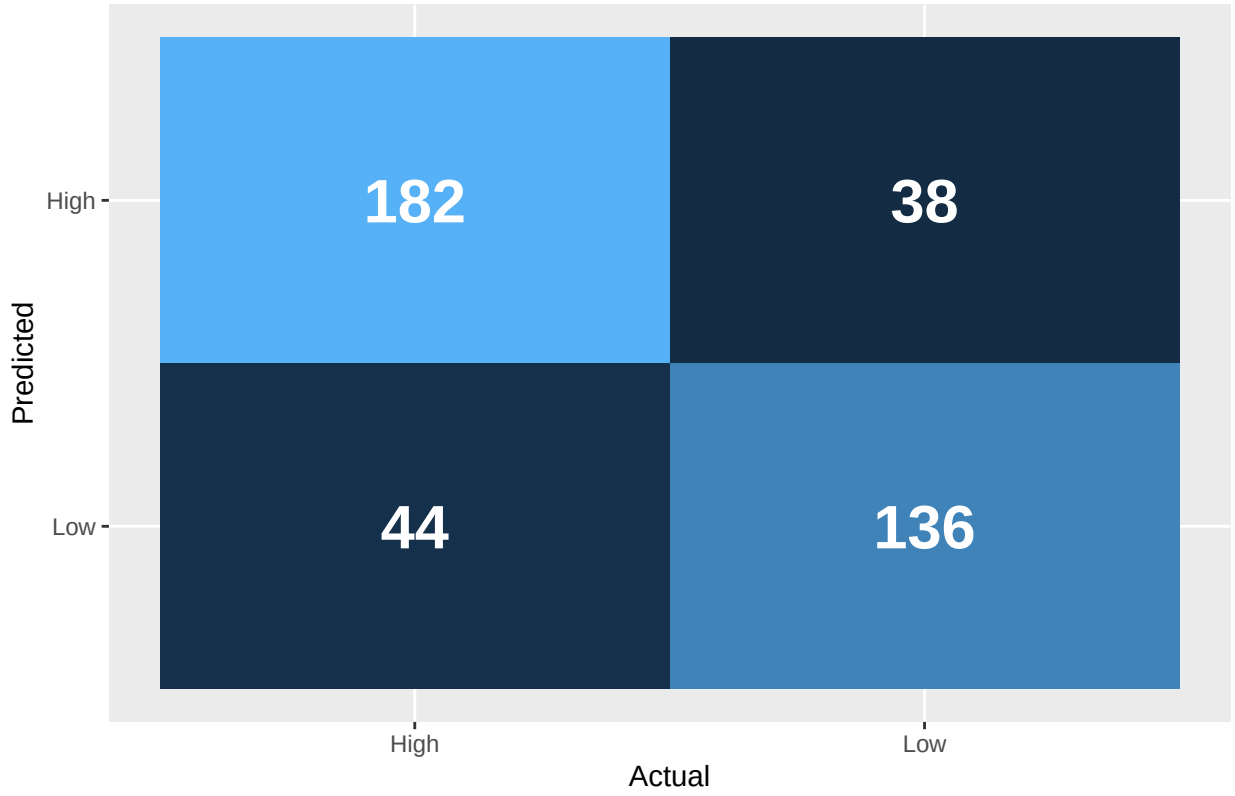
```
q.Tree.RF.Table = table(q.RF.Test.Preds, wineData[-trainIndx, 'quality'])
c.Tree.RF.Table = table(c.RF.Test.Preds, wineData[-trainIndx, 'category'])

q.Tree.RF.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')),
            color = 'white', size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 19: Confusion Matrix (Quality), RF Test Error Rate = ',
                      round(q.RF.Test.Error * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

Figure 19: Confusion Matrix (Quality), RF Test Error Rate = 34%




```
c.Tree.RF.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')),
            color = 'white', size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 20: Confusion Matrix (Category), RF Test Error Rate = ',
                      round(c.RF.Test.Error * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

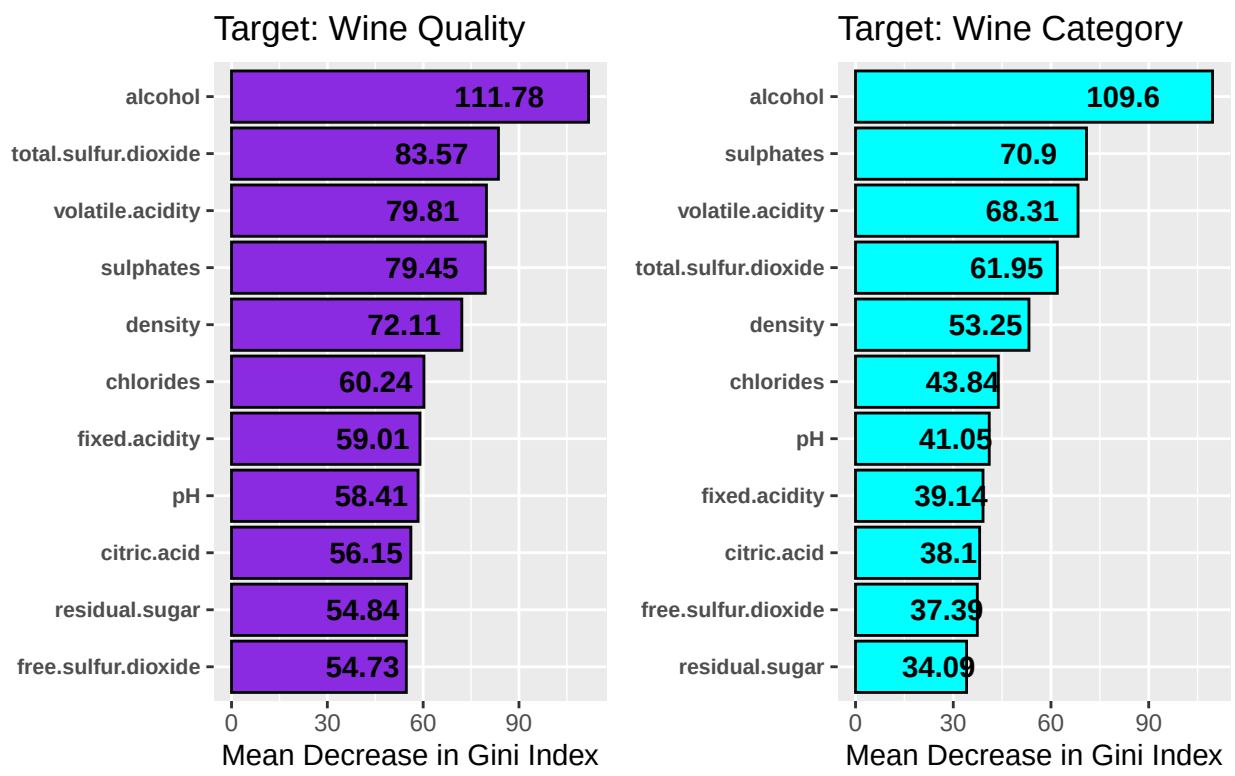
Figure 20: Confusion Matrix (Category), RF Test Error Rate = 20.5%



And assess the variable importance.

```
# Variable Importance Plots for Random Forest Models.

qualImpRF = importance(q.RF.Test.Model) %>% data.frame() %>%
  rownames_to_column() %>%
  mutate_if(is.numeric, round, 2) %>%
  ggplot(aes(x = MeanDecreaseGini,
             y = reorder(rowname, MeanDecreaseGini))) +
  geom_bar(stat = 'identity', fill = 'blueviolet', color = 'black') +
  geom_text(aes(label = MeanDecreaseGini, fontface = 'bold'),
            position = position_stack(vjust = 0.75)) +
  labs(title = 'Target: Wine Quality',
       x = 'Mean Decrease in Gini Index') +
  theme(axis.text.y = element_text(size = 8, face = 'bold'),
        axis.title.y = element_blank())

catImpRF = importance(c.RF.Test.Model) %>% data.frame() %>%
  rownames_to_column() %>%
  mutate_if(is.numeric, round, 2) %>%
  ggplot(aes(x = MeanDecreaseGini,
             y = reorder(rowname, MeanDecreaseGini))) +
  geom_bar(stat = 'identity', fill = 'cyan', color = 'black') +
  geom_text(aes(label = MeanDecreaseGini, fontface = 'bold'),
            position = position_stack(vjust = 0.75)) +
  labs(title = 'Target: Wine Category',
```

```
       x = 'Mean Decrease in Gini Index') +
  theme(axis.text.y = element_text(size = 8, face = 'bold'),
        axis.title.y = element_blank())

grdTitl2 = ggdraw() +
  draw_label('Figure 21: Predictor Importance x Target (RF Models)')
impGrid2 = plot_grid(qualImpRF, catImpRF, nrow = 1)

plot_grid(grdTitl2, impGrid2, ncol = 1, rel_heights=c(0.1, 1))
```




```
# Perform grid search for interaction depths and lambda values for
# up to 1000 trees.

start = Sys.time()

library(gbm)

set.seed(333)

boostOutput = data.frame(matrix(ncol = 7, nrow = 0)) %>%
  setNames(., c('interaction', 'numTrees', 'Target', 'Error', 'Value',
                'group', 'tune'))

boostWine = wineData %>% mutate(binCategory =
                                  ifelse(category == 'High', 1, 0)) %>%
```

```
  select(-category)

for (depth in seq(1, 5, 1)) {

  for (shrink in seq(0.001, 0.30, 0.01)) {

    q.Boost.Model = gbm(quality ~ . -binCategory, boostWine[trainIndx, ],
                        distribution = 'multinomial', n.trees = 1000,
                        interaction.depth = depth, shrinkage = shrink,
                        cv.folds = 5)

    c.Boost.Model = gbm(binCategory ~ . -quality, boostWine[trainIndx, ],
                        distribution = 'bernoulli', n.trees = 1000,
                        interaction.depth = depth, shrinkage = shrink,
                        cv.folds = 5)

    idpth = paste0('d = ', depth)
    shrnk = shrink

    q.Err = data.frame(q.Boost.Model$train.error, q.Boost.Model$cv.error) %>%
      setNames(., c('Train.Error', 'CV.Error')) %>% mutate(interaction = idpth) %>%
      mutate(numTrees = 1:1000) %>% mutate(Target = 'Quality') %>%
      mutate(lambda = shrink) %>%
      pivot_longer(cols = c('Train.Error', 'CV.Error'),
                   names_to = 'Error', values_to = 'Value')

    c.Err = data.frame(c.Boost.Model$train.error, c.Boost.Model$cv.error) %>%
      setNames(., c('Train.Error', 'CV.Error')) %>% mutate(interaction = idpth) %>%
      mutate(numTrees = 1:1000) %>% mutate(Target = 'Category') %>%
      mutate(lambda = shrink) %>%
      pivot_longer(cols = c('Train.Error', 'CV.Error'),
                   names_to = 'Error', values_to = 'Value')

    boostError = rbind(q.Err, c.Err) %>% data.frame() %>%
      mutate(group = paste(Target, Error)) %>%
      mutate(tune = paste(interaction, ', ', lambda))

    boostOutput = rbind(boostOutput, boostError)

  }
}

print(Sys.time() - start)
```

```
## Time difference of 1.145736 hours
```

```
# Generate table for presentation of grid-search results for parameter values

q.Tune = boostOutput %>% filter(Target == 'Quality' & Error == 'CV.Error') %>%
  filter(Value == min(Value, na.rm = TRUE)) %>%
  select(Target, numTrees, interaction, lambda, Value) %>%
  setNames(., c('Target', 'Trees', 'Interaction', 'Lambda', 'CV.Deviance'))
```

```
c.Tune = boostOutput %>% filter(Target == 'Category' & Error == 'CV.Error') %>%
  filter(Value == min(Value, na.rm = TRUE)) %>%
  select(Target, numTrees, interaction, lambda, Value) %>%
  setNames(., c('Target', 'Trees', 'Interaction', 'Lambda', 'CV.Deviance'))

tuneOpt = rbind(q.Tune, c.Tune) %>% mutate(iDepth =
                                             as.numeric(gsub("[^[:digit:]]", "",
                                                             Interaction)))

tuneFlex = tuneOpt %>% select(-iDepth) %>% flextable() %>%
  bold(bold = TRUE, part = 'header') %>%
  width(width = 6.5 / 5) %>%
  align(j = 2:5, align = 'center', part = 'all') %>%
  add_header_row(values = 'Table 3: Optimal Parameters for Boosted Model (Based on CV)',
                 colwidths = 5)

tuneFlex
```

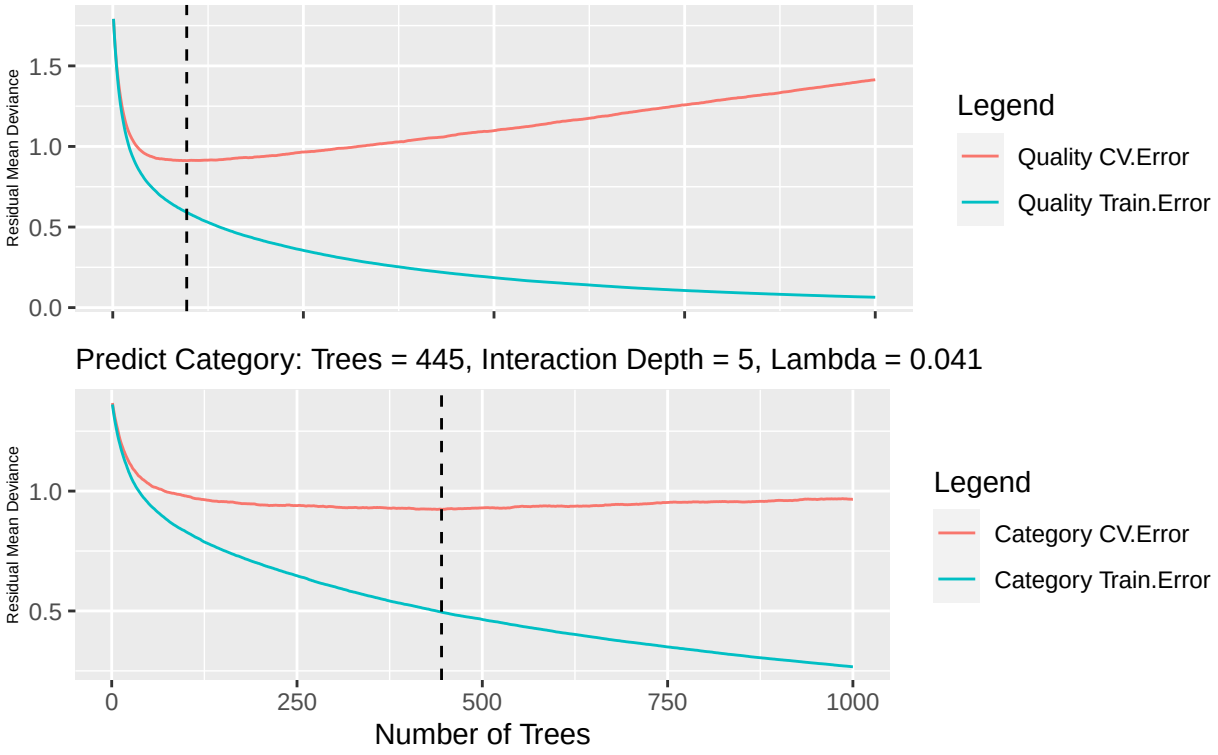
Table 3: Optimal Parameters for Boosted Model (Based on CV)

Target	Trees	Interaction	Lambda	CV.Deviance
Quality	97	d = 5	0.041	0.9117986
Category	445	d = 5	0.041	0.9236878

```
# Generate plots of training vs. cross-validated residual mean deviance
# with optimal hyperparameter values identified

tuneOpt2 = tuneOpt %>% mutate(combo = paste(Target, Interaction, Lambda))

q.Params = tuneOpt %>% filter(Target == 'Quality')
c.Params = tuneOpt %>% filter(Target == 'Category')

paramPlots = boostOutput %>% mutate(combo = paste(Target, interaction, lambda)) %>%
  filter(combo %in% tuneOpt2$combo)

q.ParamPlot = paramPlots %>% filter(Target == 'Quality') %>% ggplot() +
  geom_line(aes(x = numTrees, y = Value, color = group)) +
  labs(y = 'Residual Mean Deviance',
       title = 'Figure 22: Optimal Boosting Tuning Parameters (Top: Quality, Bottom: Category)',
       subtitle = paste0('Predict Quality: Trees = ', q.Params$Trees,
       ', Interaction Depth = ', q.Params$iDepth,
       ', Lambda = ', q.Params$Lambda),
       color = 'Legend') +
  geom_vline(xintercept = q.Params$Trees, linetype = 'dashed', linewidth = 0.5) +
  theme(axis.title.x = element_blank(), axis.text.x = element_blank(),
        axis.title.y = element_text(size = 6))

c.ParamPlot = paramPlots %>% filter(Target == 'Category') %>% ggplot() +
  geom_line(aes(x = numTrees, y = Value, color = group)) +
  labs(y = 'Residual Mean Deviance', x = 'Number of Trees',
       subtitle = paste0('Predict Category: Trees = ', c.Params$Trees,
```

```
       ', Interaction Depth = ', c.Params$iDepth,
       ', Lambda = ', c.Params$Lambda),
       color = 'Legend') +
  geom_vline(xintercept = c.Params$Trees, linetype = 'dashed', linewidth = 0.5) +
  theme(axis.title.y = element_text(size = 6))

paramGrid = plot_grid(q.ParamPlot, c.ParamPlot, nrow = 2)

paramGrid
```

Figure 22: Optimal Boosting Tuning Parameters (Top: Quality, Bottom: Categ




```
# Train boosted model on optimal parameters from grid search and generate
# test error values from the held-out observations.

set.seed(579)

qParams = tuneOpt %>% filter(Target == 'Quality')
cParams = tuneOpt %>% filter(Target == 'Category')

q.Boost.Test = gbm(quality ~ . -binCategory, boostWine[trainIndx, ],
                   distribution = 'multinomial',
                   n.trees = as.numeric(qParams$Trees),
                   interaction.depth = as.numeric(qParams$iDepth),
                   shrinkage = as.numeric(qParams$Lambda))

c.Boost.Test = gbm(binCategory ~ . -quality, boostWine[trainIndx, ],
```

```
                  distribution = 'bernoulli',
                  n.trees = as.numeric(cParams$Trees),
                  interaction.depth = as.numeric(cParams$iDepth),
                  shrinkage = as.numeric(cParams$Lambda))

q.Boost.Pred = predict(q.Boost.Test, boostWine[-trainIndx, ], type = 'response')
q.Boost.Clas = factor(colnames(q.Boost.Pred)[apply(q.Boost.Pred,1,which.max)], levels = c(3:8))

c.Boost.Pred = predict(c.Boost.Test, boostWine[-trainIndx, ], type = 'response')
c.Boost.Clas = ifelse(c.Boost.Pred > 0.5, 1, 0)

table(q.Boost.Clas, boostWine[-trainIndx, 'quality'])
```

```
##
## q.Boost.Clas   3   4   5   6   7   8
##            3   0   0   0   0   0   0
##            4   0   1   1   0   0   0
##            5   4   8 121  53   8   0
##            6   1   1  35 101  27   1
##            7   0   1   1  19  15   2
##            8   0   0   0   0   0   0
```

```
q.Boost.Errs =
  1 - sum(diag(table(q.Boost.Clas, boostWine[-trainIndx, 'quality']))) /
  sum(table(q.Boost.Clas, boostWine[-trainIndx, 'quality']))

q.Boost.Errs
```

```
## [1] 0.405
```

```
table(c.Boost.Clas, boostWine[-trainIndx, 'binCategory'])
```

```
##
## c.Boost.Clas   0   1
##            0 139  57
##            1  35 169
```

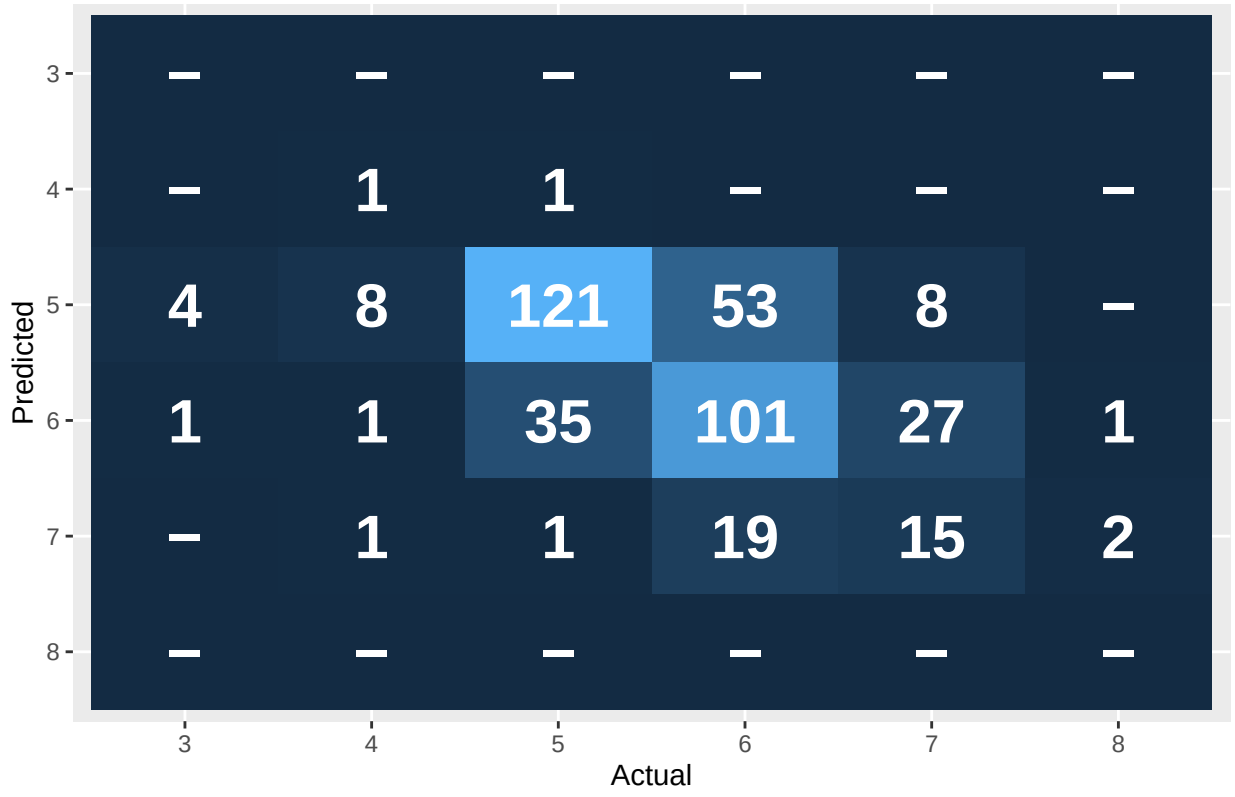
```
c.Boost.Errs =
  1 - sum(diag(table(c.Boost.Clas, boostWine[-trainIndx, 'binCategory']))) /
  sum(table(c.Boost.Clas, boostWine[-trainIndx, 'binCategory']))

c.Boost.Errs
```

```
## [1] 0.23
```

```
q.Boost.Table = table(q.Boost.Clas, boostWine[-trainIndx, 'quality'])
c.Boost.Table = table(c.Boost.Clas, boostWine[-trainIndx, 'binCategory'])

q.Boost.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
```

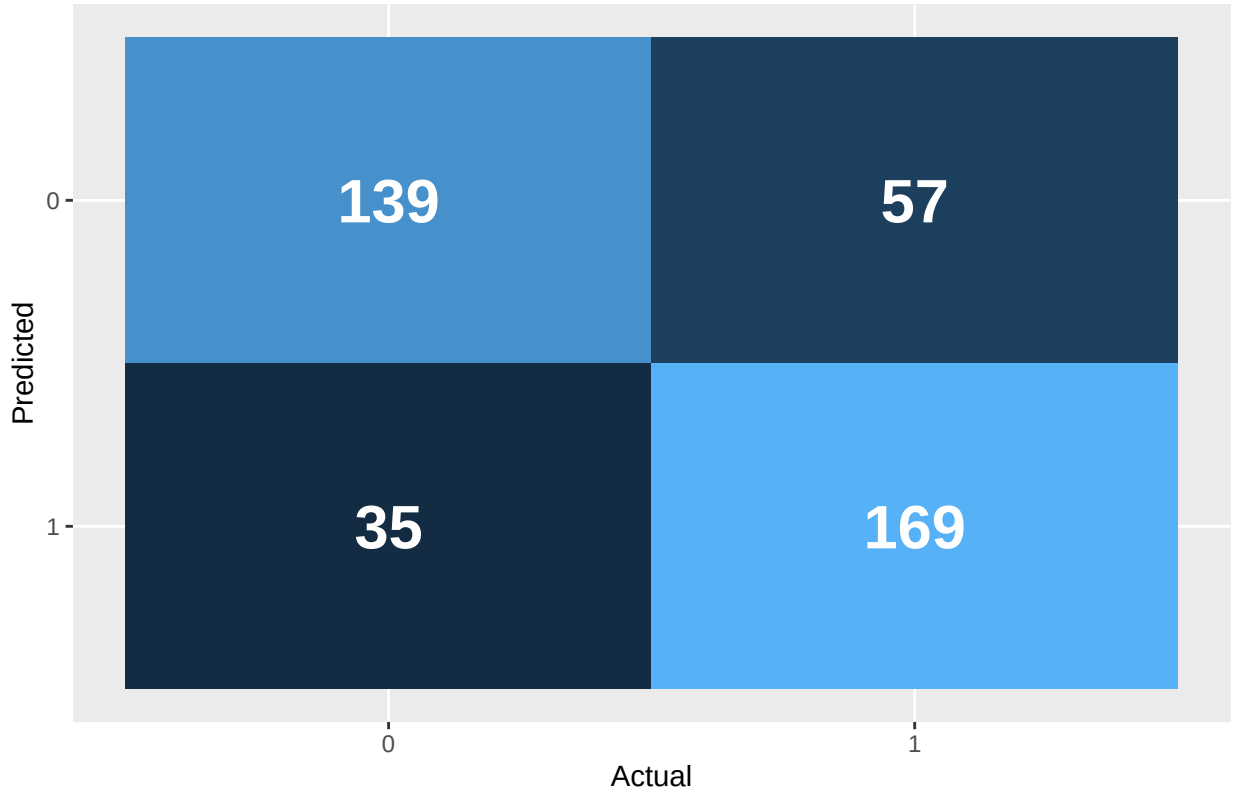
```
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')),
            color = 'white', size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 23: Confusion Matrix (Quality), Boosting Test Error Rate = ',
                      round(q.Boost.Errs * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

Figure 23: Confusion Matrix (Quality), Boosting Test Error Rate = 40.5%




```
c.Boost.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')),
            color = 'white', size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 24: Confusion Matrix (Category), Boosting Test Error Rate = ',
                      round(c.Boost.Errs * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none')
```

Figure 24: Confusion Matrix (Category), Boosting Test Error Rate = 23%



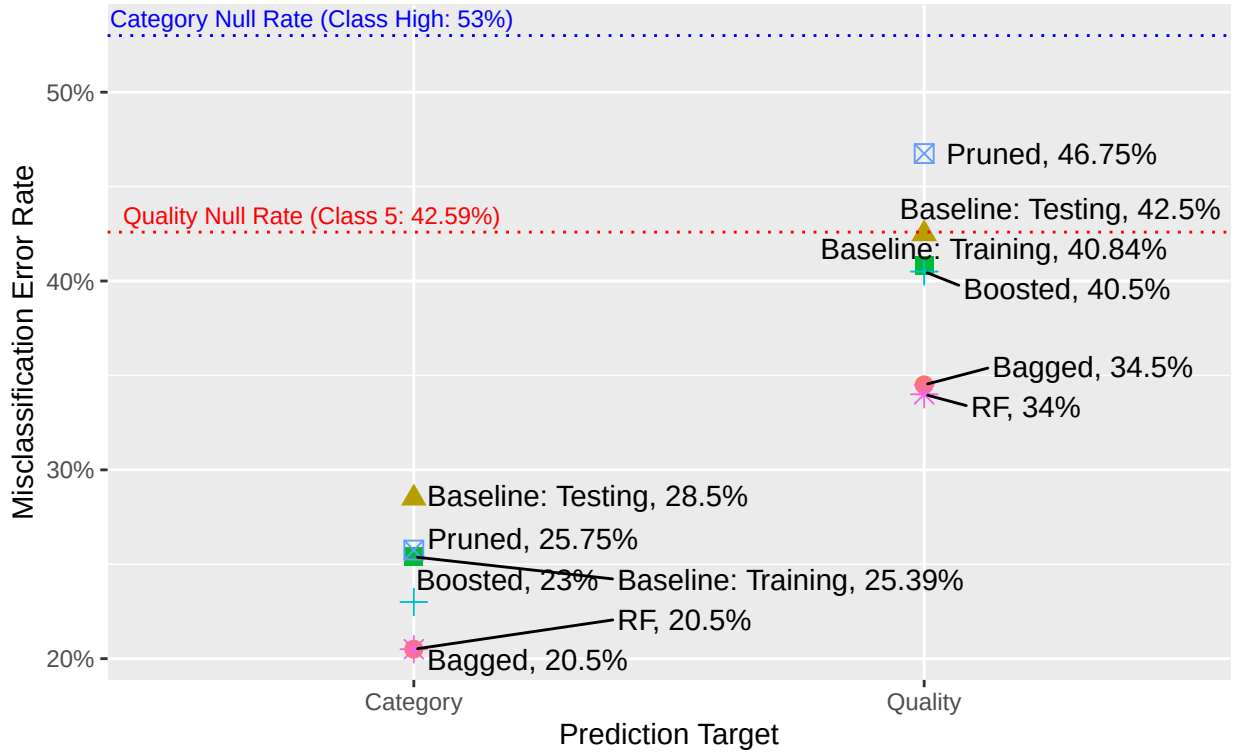

```
# Generate table ranking predictor variable importance across
# all ensemble methods

q.Bag.Inf = importance(q.Bag.Test.Model) %>% data.frame() %>%
  rownames_to_column() %>% setNames(., c('Predictor', 'Gini')) %>%
  arrange(-Gini) %>% mutate(Rank = 1:11) %>% select(Predictor, Rank)

c.Bag.Inf = importance(c.Bag.Test.Model) %>% data.frame() %>%
  rownames_to_column() %>% setNames(., c('Predictor', 'Gini')) %>%
  arrange(-Gini) %>% mutate(Rank = 1:11) %>% select(Predictor, Rank)

q.RF.Inf = importance(q.RF.Test.Model) %>% data.frame() %>%
  rownames_to_column() %>% setNames(., c('Predictor', 'Gini')) %>%
  arrange(-Gini) %>% mutate(Rank = 1:11) %>% select(Predictor, Rank)

c.RF.Inf = importance(c.RF.Test.Model) %>% data.frame() %>%
  rownames_to_column() %>% setNames(., c('Predictor', 'Gini')) %>%
  arrange(-Gini) %>% mutate(Rank = 1:11) %>% select(Predictor, Rank)

q.Boost.Inf = relative.influence(q.Boost.Test, n.trees = qParams$Trees) %>%
  data.frame() %>% rownames_to_column() %>%
  setNames(., c('Predictor', 'q.Boost.Influence')) %>%
  arrange(-q.Boost.Influence) %>% mutate(Rank = 1:11) %>%
  select(Predictor, Rank)

c.Boost.Inf = relative.influence(c.Boost.Test, n.trees = cParams$Trees) %>%
```

```
  data.frame() %>% rownames_to_column() %>%
  setNames(., c('Predictor', 'c.Boost.Influence')) %>%
  arrange(-c.Boost.Influence) %>% mutate(Rank = 1:11) %>%
  select(Predictor, Rank)

allPredictrs = q.Bag.Inf %>% merge(c.Bag.Inf, by = 'Predictor') %>%
  merge(q.RF.Inf, by = 'Predictor') %>% merge(c.RF.Inf, by = 'Predictor') %>%
  merge(q.Boost.Inf, by = 'Predictor') %>% merge(c.Boost.Inf, by = 'Predictor') %>%
  setNames(., c('Predictor', 'Bag (Q)', 'Bag (C)', 'RF (Q)', 'RF (C)',
                'Boost (Q)', 'Boost (C)')) %>%
  mutate(`Mean Rank` = rowMeans(.[,2:7])) %>% mutate_if(is.numeric, round, 2) %>%
  arrange(`Mean Rank`) %>% flextable() %>% width(width = 1) %>%
  align(j = 2:8, align = 'center', part = 'all') %>%
  bold(bold = TRUE, part = 'header') %>%
  padding(padding.top = 0, padding.bottom = 0, part = 'all') %>%
  add_header_row(values = 'Table 4: Rank of Predictor Impact Across Models',
                 colwidths = 8)
```

```
# Generate dataframe with all test error values to report in graph form

library(ggrepel)

allTestErrors = data.frame(
  Model = c('Baseline: Training', 'Baseline: Training',
            'Baseline: Testing', 'Baseline: Testing',
            'Pruned', 'Pruned', 'Bagged', 'Bagged',
            'RF', 'RF', 'Boosted', 'Boosted'),
  Target = c(rep(c('Quality', 'Category'), 6)),
  Rates = c(q.Tree.Base, c.Tree.Base, q.Tree.Test.Rate,
            c.Tree.Test.Rate, q.Prune.Test.Error, c.Prune.Test.Error,
            q.Bag.Test.Error, c.Bag.Test.Error, q.RF.Test.Error,
            c.RF.Test.Error, q.Boost.Errs, c.Boost.Errs)
)

ggplot(data = allTestErrors, aes(x = Target, y = Rates)) +
  geom_point(aes(shape = Model, color = Model), size = 3) +
  geom_text_repel(aes(label = paste0(Model, ', ',
                                     round(Rates * 100, 2), '%')),
                  nudge_x = 0.25) +
  labs(x = 'Prediction Target', y = 'Misclassification Error Rate',
       title = 'Figure 25: Model Performance Summary',
       subtitle = 'Includes all tree-based models tested in project') +
  scale_y_continuous(labels = scales::percent) +
  geom_hline(yintercept = 0.4259, linetype = 'dotted', col = 'red') +
  annotate('text', x = 0.80, y = 0.4259,
           label = 'Quality Null Rate (Class 5: 42.59%)', color = 'red',
           vjust= -0.5, size = 3) +
  geom_hline(yintercept = 0.53, linetype = 'dotted', col = 'blue') +
  annotate('text', x = 0.80, y = 0.53,
           label = 'Category Null Rate (Class High: 53%)', color = 'blue',
           vjust= -0.5, size = 3) + theme(legend.position = 'none')
```

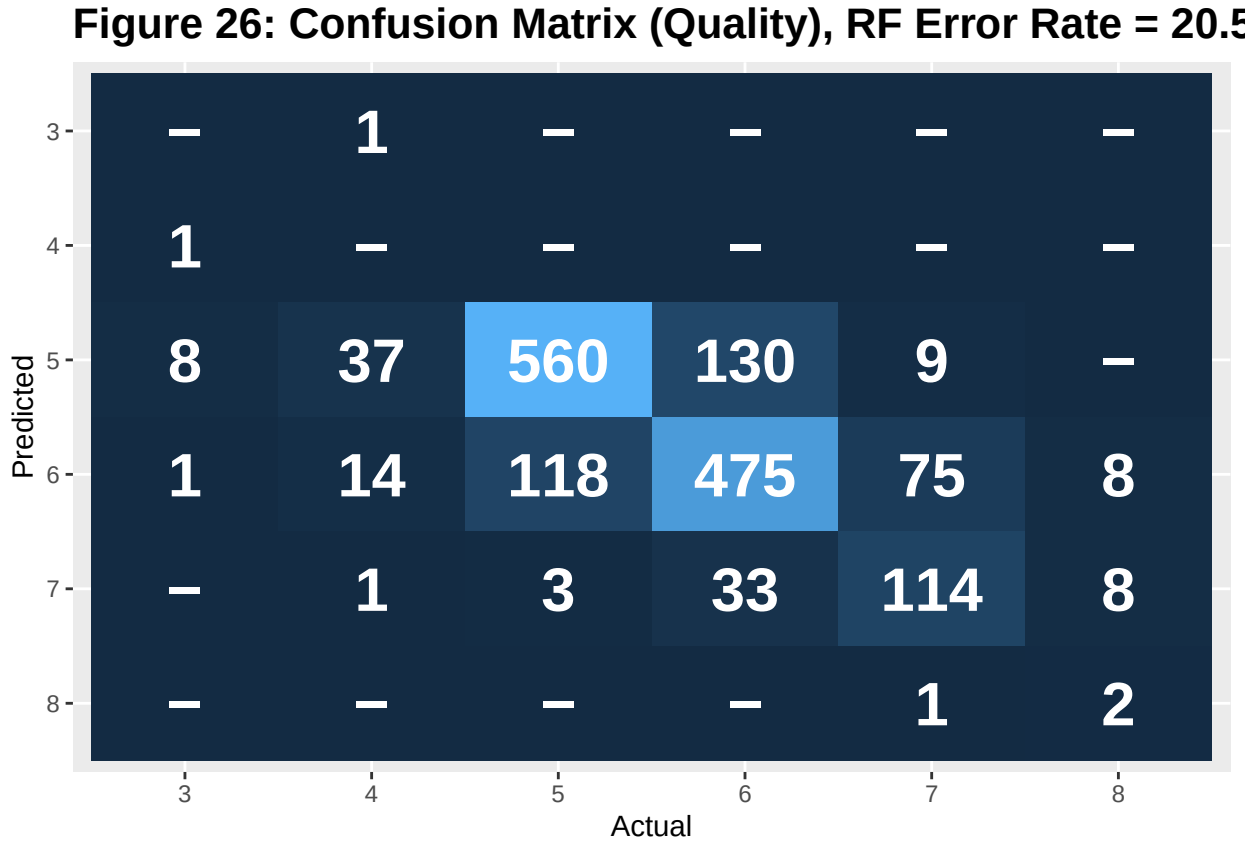
Figure 25: Model Performance Summary

Includes all tree−based models tested in project



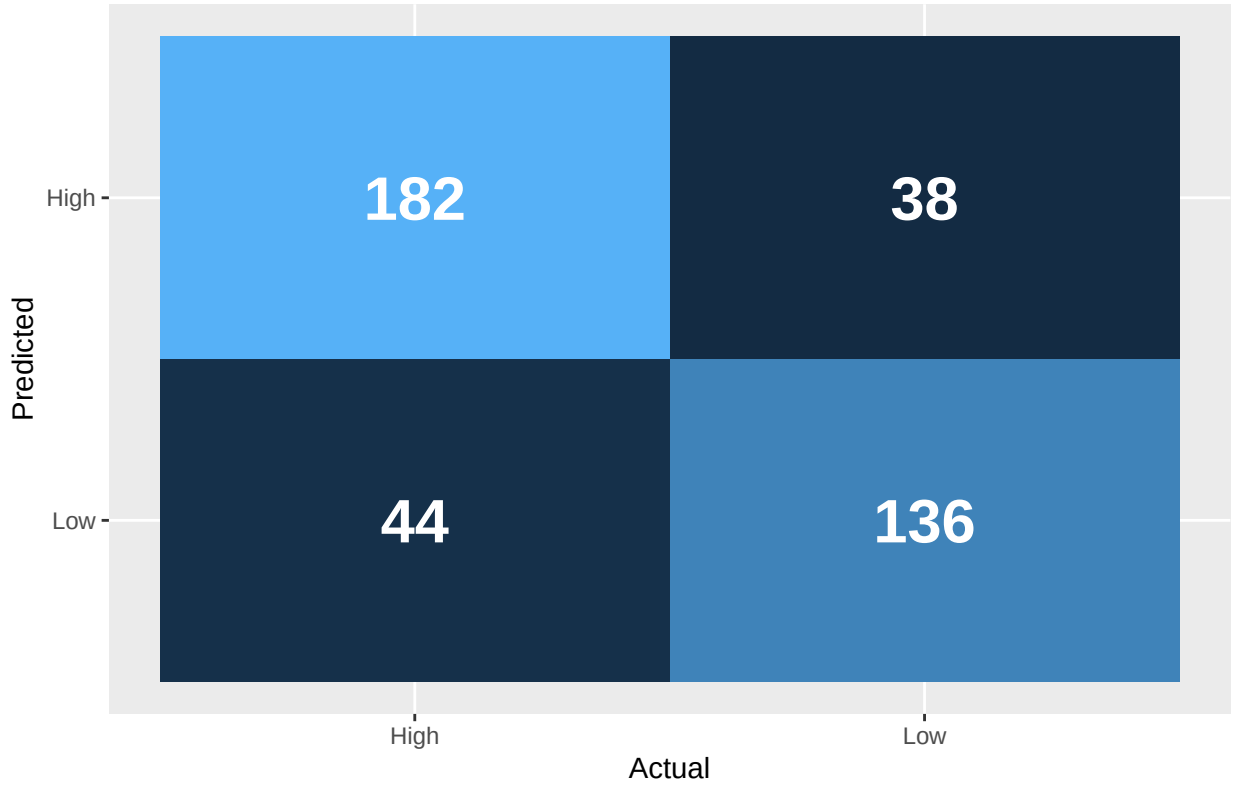

```
# Fit Final Model to all data and generate confusion matrices and
# misclassification error rates

q.RF.Final.Model = randomForest(
  quality ~ . -category, wineData, mtry = sqrt(11),
  ntrees = as.numeric(q.RF.min.error$Trees))

q.RF.Final.Table = table(q.RF.Final.Model$predicted,
                         wineData$quality)

q.RF.Final.Error = 1 - sum(diag(q.RF.Final.Table)) /
  sum(q.RF.Final.Table)

c.RF.Final.Model = randomForest(
  category ~ . -quality, wineData, mtry = sqrt(11),
  ntrees = as.numeric(c.RF.min.error$Trees))

c.RF.Final.Table = table(c.RF.Final.Model$predicted,
                         wineData$category)

c.RF.Final.Error = 1 - sum(diag(c.RF.Final.Table)) /
  sum(c.RF.Final.Table)


q.RF.Final.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
```

```
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')),
            color = 'white', size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 26: Confusion Matrix (Quality), RF Error Rate = ',
                      round(c.RF.Test.Error * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none',
                                plot.title = element_text(face = 'bold', size = 16))
```




```
c.Tree.RF.Table %>% data.frame() %>%
  setNames(., c('Predicted', 'Actual', 'Frequency')) %>%
  ggplot(aes(x = Actual, y = fct_rev(Predicted), fill = Frequency)) + geom_tile() +
  geom_text(aes(label = ifelse(Frequency > 0, Frequency, '-')),
            color = 'white', size = 8, fontface = 'bold') +
  labs(title = paste0('Figure 27: Confusion Matrix (Category), RF Error Rate = ',
                      round(c.RF.Final.Error * 100, 2), '%'),
       y = 'Predicted') + theme(legend.position = 'none',
                                plot.title = element_text(face = 'bold', size = 16))
```

Figure 27: Confusion Matrix (Category), RF Error Rate =




```
# Generate plots of variable importance from final model

q.Final.Imp = importance(q.RF.Final.Model) %>% data.frame() %>%
  rownames_to_column() %>%
  mutate_if(is.numeric, round, 2) %>%
  ggplot(aes(x = MeanDecreaseGini,
             y = reorder(rowname, MeanDecreaseGini))) +
  geom_bar(stat = 'identity', fill = 'aquamarine', color = 'black') +
  geom_text(aes(label = MeanDecreaseGini, fontface = 'bold'),
            position = position_stack(vjust = 0.75)) +
  labs(title = 'Target: Wine Quality',
       x = 'Mean Decrease in Gini Index') +
  theme(axis.text.y = element_text(size = 8, face = 'bold'),
        axis.title.y = element_blank())

c.Final.Imp = importance(c.RF.Final.Model) %>% data.frame() %>%
  rownames_to_column() %>%
  mutate_if(is.numeric, round, 2) %>%
  ggplot(aes(x = MeanDecreaseGini,
             y = reorder(rowname, MeanDecreaseGini))) +
  geom_bar(stat = 'identity', fill = 'deeppink', color = 'black') +
  geom_text(aes(label = MeanDecreaseGini, fontface = 'bold'),
            position = position_stack(vjust = 0.75)) +
  labs(title = 'Target: Wine Category',
       x = 'Mean Decrease in Gini Index') +
  theme(axis.text.y = element_text(size = 8, face = 'bold'),
```

```
        axis.title.y = element_blank())

grdTitl3 = ggdraw() +
  draw_label('Figure 28: Predictor Importance x Target (Final RF)')
impGrid3 = plot_grid(q.Final.Imp, c.Final.Imp, nrow = 1)

plot_grid(grdTitl3, impGrid3, ncol = 1, rel_heights=c(0.1, 1))
```