ML Assignment 2 - Decision Tree

PrabhTalwar

2023-01-09

```r
# Helper packages
library(dplyr) # for data wrangling
library(ggplot2) # for awesome plotting

# Modeling packages
library(rpart) # direct engine for decision tree application
library(caret) # meta engine for decision tree application
library(rsample)
library(randomForest) #For implementing random forest algorithm

# Model interpretability packages
library(rpart.plot) # for plotting decision trees
library(vip) # for feature importance
library(pdp) # for feature effects
```

```r
library(readr)
winequality_red <- read_csv("winequality-red.csv")
```

```
## Rows: 1599 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (12): fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlo...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
str(winequality_red)
```

```
## spc_tbl_ [1,599 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ fixed_acidity       : num [1:1599] 7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile_acidity    : num [1:1599] 0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric_acid         : num [1:1599] 0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual_sugar      : num [1:1599] 1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num [1:1599] 0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ..
##  $ free_sulfur_dioxide : num [1:1599] 11 25 15 17 11 13 15 15 9 17 ...
##  $ total_sulfur_dioxide: num [1:1599] 34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num [1:1599] 0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num [1:1599] 3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num [1:1599] 0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num [1:1599] 9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : num [1:1599] 5 5 5 6 5 5 5 7 7 5 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   fixed_acidity = col_double(),
##   ..   volatile_acidity = col_double(),
##   ..   citric_acid = col_double(),
##   ..   residual_sugar = col_double(),
##   ..   chlorides = col_double(),
##   ..   free_sulfur_dioxide = col_double(),
##   ..   total_sulfur_dioxide = col_double(),
##   ..   density = col_double(),
##   ..   pH = col_double(),
##   ..   sulphates = col_double(),
##   ..   alcohol = col_double(),
##   ..   quality = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

The above data set has 1599 rows and 12 columns where quality is our response variable. The data set has various chemical properties of the wine and will be building a decision tree to predict the quality of the wine.

Here, our response variable is read an integer whereas it should be a categorical/ factor variable for which we will use `as.factor` to convert numerical variable into factor variable.

Decision Tree

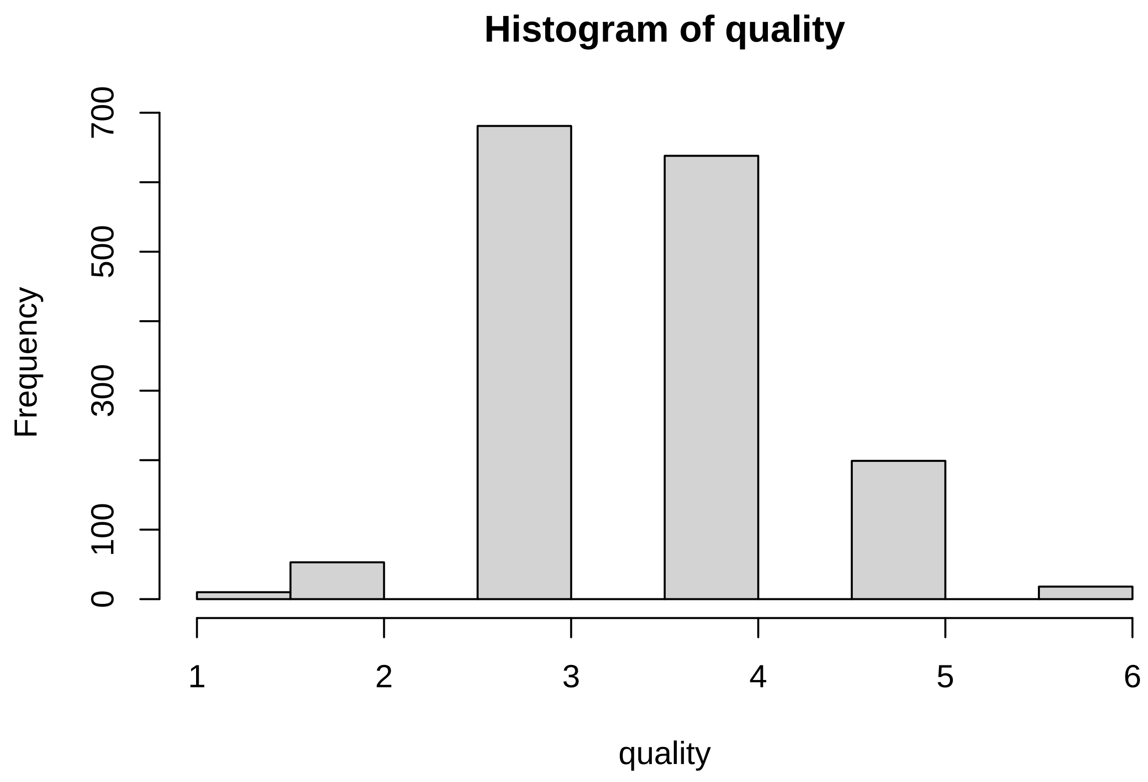
```
table(winequality_red$quality)
```

```
##
##   3   4   5   6   7   8
##  10  53 681 638 199  18
```

```
winequality_red$quality <- as.factor(winequality_red$quality)
str(winequality_red$quality) # converted into factor
```

```
##  Factor w/ 6 levels "3","4","5","6",..: 3 3 3 4 3 3 3 5 5 3 ...
```

```
# converting into numerical to build the histogram
quality <- as.numeric(winequality_red$quality)
hist(quality)
```



```r
levels(winequality_red$quality)
```

```
## [1] "3" "4" "5" "6" "7" "8"
```

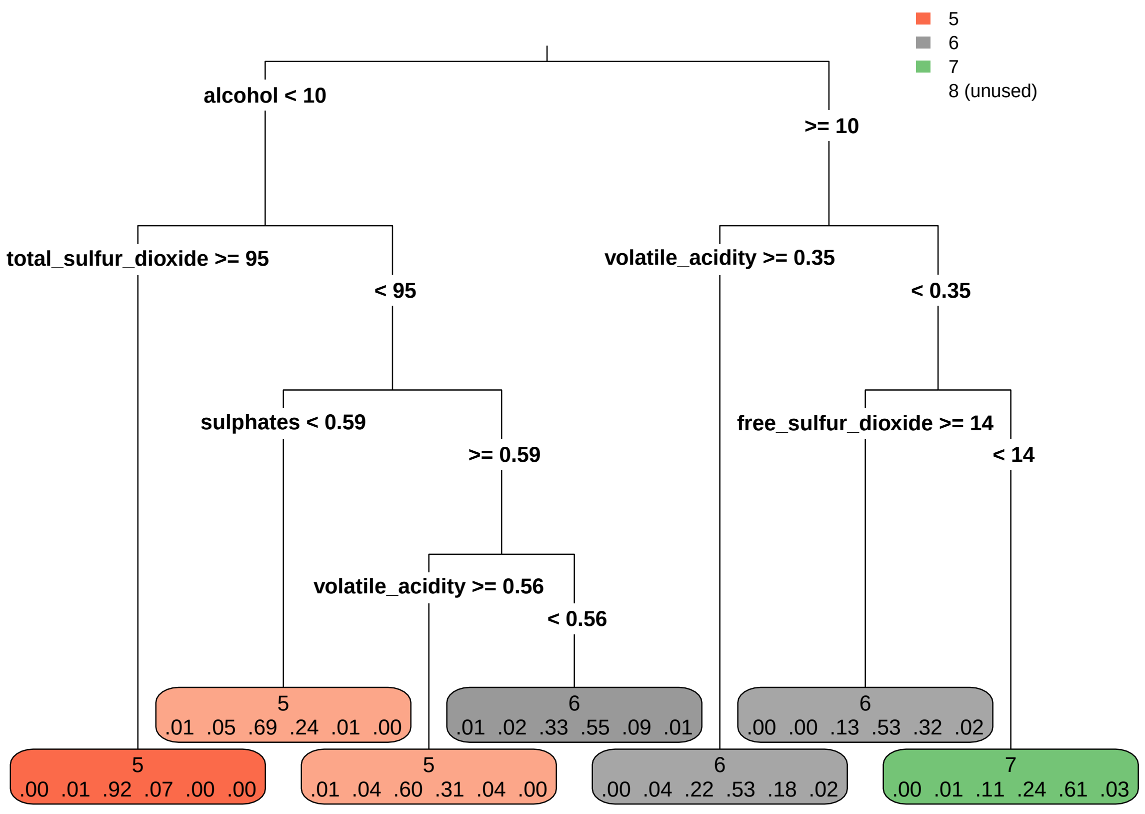
```r
# Using rsample package
set.seed(123) # for reproducibility

# Splitting the data in a 80-20 split
split_wine <- initial_split(winequality_red, prop = 0.8)
train_wine <- training(split_wine)
test_wine <- testing(split_wine)
```

```r
wine_dt <- rpart(formula = quality ~ ., data = train_wine, method = "class")
wine_dt
```

```
## n= 1279
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1279 747 5 (0.0063 0.034 0.42 0.4 0.13 0.013)
##    2) alcohol< 10.35 683 268 5 (0.0088 0.035 0.61 0.31 0.034 0.0029)
##      4) total_sulfur_dioxide>=94.5 90   7 5 (0 0.011 0.92 0.067 0 0) *
##      5) total_sulfur_dioxide< 94.5 593 261 5 (0.01 0.039 0.56 0.35 0.039 0.0034)
##       10) sulphates< 0.585 271  84 5 (0.015 0.052 0.69 0.24 0.0074 0) *
##       11) sulphates>=0.585 322 177 5 (0.0062 0.028 0.45 0.44 0.065 0.0062)
##         22) volatile_acidity>=0.555 139  55 5 (0.0072 0.043 0.6 0.31 0.036 0) *
##         23) volatile_acidity< 0.555 183  83 6 (0.0055 0.016 0.33 0.55 0.087 0.011) *
##    3) alcohol>=10.35 596 302 6 (0.0034 0.034 0.2 0.49 0.25 0.023)
##      6) volatile_acidity>=0.345 457 214 6 (0.0044 0.042 0.22 0.53 0.18 0.024) *
##      7) volatile_acidity< 0.345 139  72 7 (0 0.0072 0.12 0.37 0.48 0.022)
##       14) free_sulfur_dioxide>=13.5 60  28 6 (0 0 0.13 0.53 0.32 0.017) *
##       15) free_sulfur_dioxide< 13.5 79  31 7 (0 0.013 0.11 0.24 0.61 0.025) *
```

```r
rpart.plot(wine_dt, type = 3, extra = 4, tweak = 0.9)
```



In the above decision tree the leaves display the Probability per class. For example, the leaf to the most left tells that the quality of wine with alcohol and total sulfur dioxide is 5 with giving probability for each class 3(0.00), 4(0.01), 5(0.92), 6(0.07), 7(0.00) and 8(0.00).

```r
# Prediction
predict_train <- predict(wine_dt, train_wine, type = "class")

# Classification on training data
table_wine <- table(train_wine$quality, predict_train)
table_wine
```

```
##    predict_train
##       3    4    5    6    7    8
##   3   0   0   5   3   0   0
##   4   0   0  21  22   1   0
##   5   0   0 354 169   9   0
##   6   0   0 113 375  19   0
##   7   0   0   7 117  48   0
##   8   0   0   0  14   2   0
```

```r
confusionMatrix(predict_train, train_wine$quality)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8
##          3   0   0   0   0   0   0
##          4   0   0   0   0   0   0
##          5   5  21 354 113   7   0
##          6   3  22 169 375 117  14
##          7   0   1   9  19  48   2
##          8   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6075
##                  95% CI : (0.5801, 0.6344)
##     No Information Rate : 0.4159
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3588
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000   0.0000   0.6654   0.7396  0.27907  0.00000
## Specificity          1.000000   1.0000   0.8046   0.5790  0.97200  1.00000
## Pos Pred Value            NaN      NaN   0.7080   0.5357  0.60759      NaN
## Neg Pred Value       0.993745   0.9656   0.7715   0.7720  0.89667  0.98749
## Prevalence           0.006255   0.0344   0.4159   0.3964  0.13448  0.01251
## Detection Rate       0.000000   0.0000   0.2768   0.2932  0.03753  0.00000
## Detection Prevalence 0.000000   0.0000   0.3909   0.5473  0.06177  0.00000
## Balanced Accuracy    0.500000   0.5000   0.7350   0.6593  0.62553  0.50000
```

```r
accuracy_Test <- sum(diag(table_wine)) / sum(table_wine)
accuracy_Test
```

```
## [1] 0.6075059
```

The accuracy for the training data is 0.6075.

```r
# Prediction
predict_test <- predict(wine_dt, test_wine, type = "class")
```

```
# Classification on test data
table_wine_test <- table(test_wine$quality, predict_test)
table_wine_test
```

```
##    predict_test
##      3  4  5  6  7  8
##   3  0  0  1  1  0  0
##   4  0  0  6  3  0  0
##   5  0  0 96 50  3  0
##   6  0  0 34 89  8  0
##   7  0  0  0 22  5  0
##   8  0  0  0  2  0  0
```

```
accuracy_Test1 <- sum(diag(table_wine_test)) / sum(table_wine_test)
accuracy_Test1
```

```
## [1] 0.59375
```

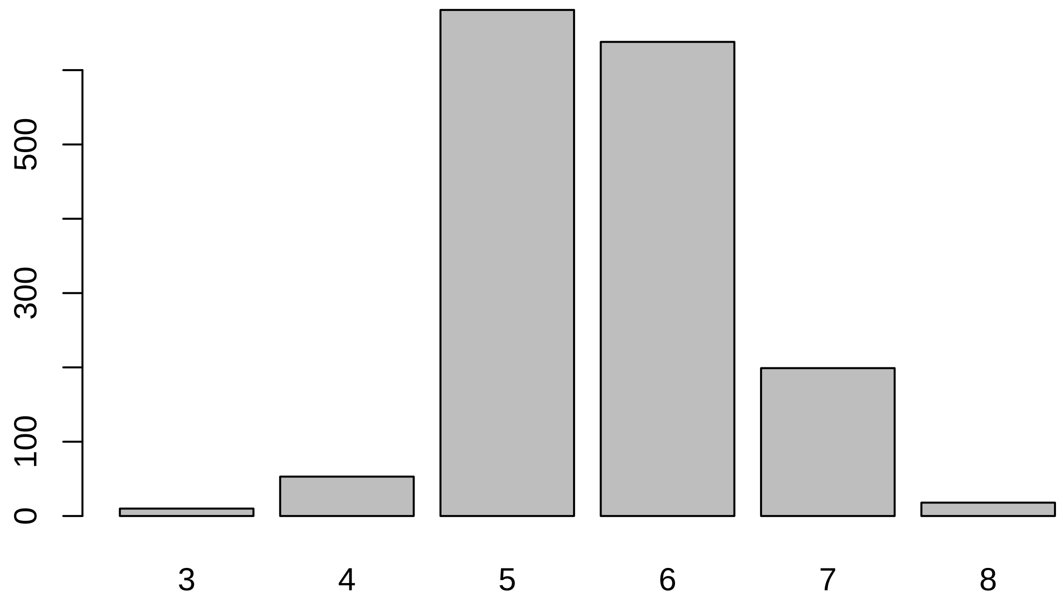
The accuracy for the testing data is 0.5938. Here, we can say that our model is not overfitted.

Random Forest

```
Wine_Data <- read_csv("winequality-red.csv")
```

```
## Rows: 1599 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (12): fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlo...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
barplot(table(Wine_Data$quality))
```



```
str(Wine_Data$quality)
```

```
##  num [1:1599] 5 5 5 6 5 5 5 7 7 5 ...
```

```
Wine_Data$quality <- as.numeric(Wine_Data$quality)

str(Wine_Data$quality)
```

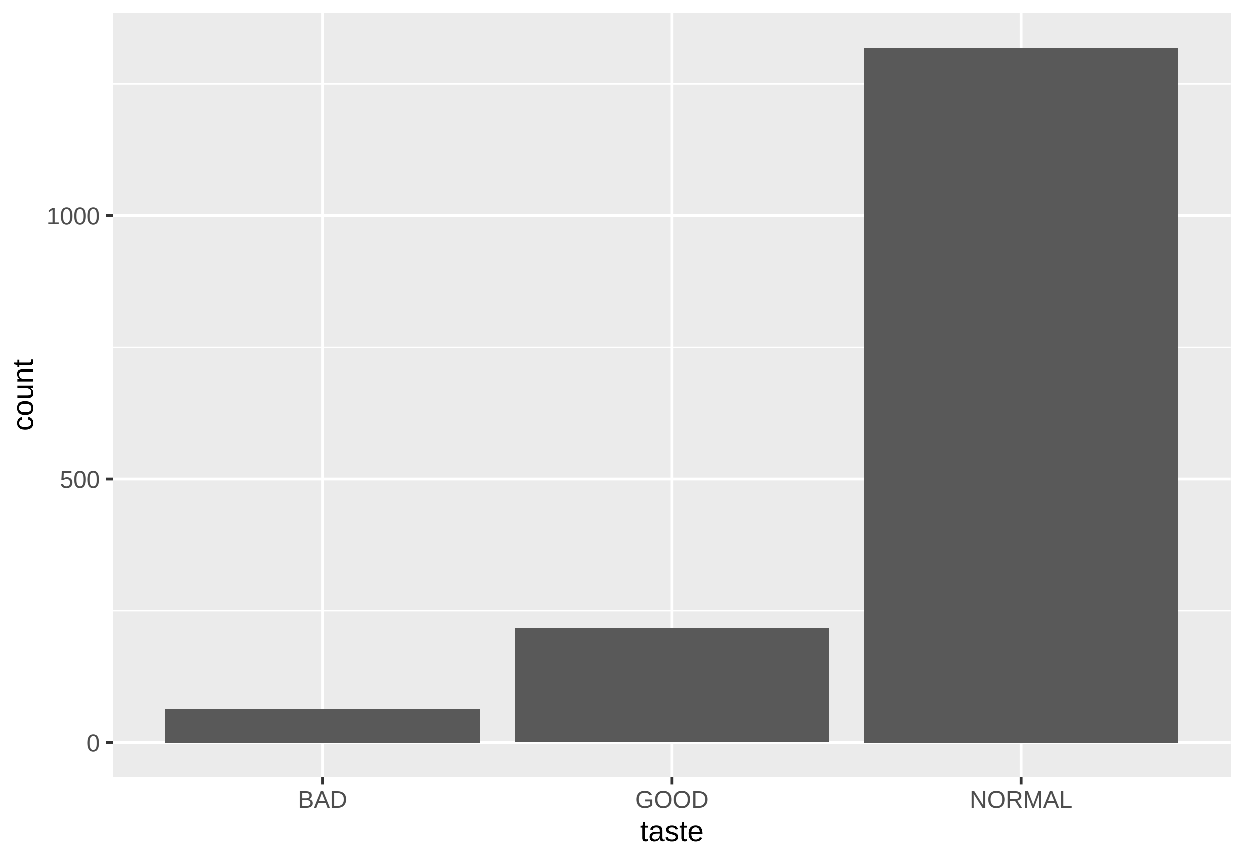
```
##  num [1:1599] 5 5 5 6 5 5 5 7 7 5 ...
```

```
class(Wine_Data$quality)
```

```
## [1] "numeric"
```

```
Wine_Data$taste <- ifelse(Wine_Data$quality < 5, "BAD", "GOOD")
Wine_Data$taste[Wine_Data$quality == 5] <- "NORMAL"
Wine_Data$taste[Wine_Data$quality == 6] <- "NORMAL"
Wine_Data$taste <- as.factor(Wine_Data$taste)

ggplot(data = Wine_Data, aes(x = taste))+
  geom_bar()
```



```r
# Using rsample package
set.seed(123) # for reproducibility

# Splitting the data in a 80-20 split
split_wine1 <- initial_split(Wine_Data, prop = 0.8)
train_wine1 <- training(split_wine1)
test_wine1 <- testing(split_wine1)
```

```r
wine_rf <- randomForest(taste ~ . -quality, data = train_wine1)
wine_rf
```

```
##
## Call:
##  randomForest(formula = taste ~ . - quality, data = train_wine1)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 13.29%
## Confusion matrix:
##        BAD GOOD NORMAL class.error
## BAD      0    1     51  1.00000000
## GOOD     0  101     87  0.46276596
## NORMAL   1   30   1008  0.02983638
```

```
# Prediction
prediction_train <- predict(wine_rf, newdata = train_wine1)

# Classification on training data
predtable <- table(prediction_train, train_wine1$taste)
predtable
```

```
##
## prediction_train  BAD GOOD NORMAL
##           BAD      52    0      0
##           GOOD      0  188      0
##           NORMAL    0    0   1039
```

```
wine_rf$confusion
```

```
##        BAD GOOD NORMAL class.error
## BAD      0    1     51  1.00000000
## GOOD     0  101     87  0.46276596
## NORMAL   1   30   1008  0.02983638
```

```
accuarcy <- sum(diag(predtable))/nrow(train_wine1)
accuarcy
```

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

The accuracy for the training data is 1.

```
# Prediction
prediction <- predict(wine_rf, newdata = test_wine1)

# Classification on test data
predtable1 <- table(prediction, test_wine1$taste)
predtable1
```

```
##
## prediction BAD GOOD NORMAL
##     BAD      0    0      0
##     GOOD     0   15      8
##     NORMAL  11   14    272
```

```
wine_rf$confusion
```

```
##        BAD GOOD NORMAL class.error
## BAD      0    1     51  1.00000000
## GOOD     0  101     87  0.46276596
## NORMAL   1   30   1008  0.02983638
```

```
accuarcy1 <- sum(diag(predtable1))/nrow(test_wine1)
accuarcy1
```

```
## [1] 0.896875
```

The accuracy for the training data is 0.896875.