Yoshita & Dhruvi Data 101 Final Pt2

Code

Hide

```
#Installing these packages took like.. four hours.


library(caret) # For data splitting and accuracy calculation
```

```
Warning: package 'caret' was built under R version 4.3.3Loading required package: ggplot2
Warning: package 'ggplot2' was built under R version 4.3.3Loading required package: lattice
```

Hide

```
library(e1071) # For Naive Bayes
```

```
Warning: package 'e1071' was built under R version 4.3.3
```

Hide

```
library(ggplot2) # For data visualization
library(dplyr)
```

```
Warning: package 'dplyr' was built under R version 4.3.3
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

Hide

```
library(corrplot)
```

```
Warning: package 'corrplot' was built under R version 4.3.3corrplot 0.92 loaded
```

Hide

```
library(rpart)
```

```
Warning: package 'rpart' was built under R version 4.3.3
```

Hide

```
library(rpart.plot)
```

```
Warning: package ‘rpart.plot’ was built under R version 4.3.3
```

Hide

```
apples <- read.csv("apple_quality.csv")
```

Hide

```
# Modified our Data to add extra features to show correlations + neg correlations. Got this from
the dplyr library at: # https://dplyr.tidyverse.org/reference/mutate.html

apples <- apples %>%
  mutate(
    SizeCrunchiness = Size * Crunchiness,      # New feature for positive correlation
    AcidJuice = Acidity * Juiciness ,    # Another feature for positive correlation
    SizeDSweet = Size / Sweetness,
  )


summary(apples)
```

```
      A_id             Size             Weight           Sweetness        Crunchiness         Ju
iciness
 Min.   :   0.0   Min.   :-7.1517   Min.   :-7.14985   Min.   :-6.8945   Min.   :-6.05506   Min.
:-5.9619
 1st Qu.: 999.8   1st Qu.:-1.8168   1st Qu.:-2.01177   1st Qu.:-1.7384   1st Qu.: 0.06276   1st
Qu.:-0.8013
 Median :1999.5   Median :-0.5137   Median :-0.98474   Median :-0.5048   Median : 0.99825   Medi
an : 0.5342
 Mean   :1999.5   Mean   :-0.5030   Mean   :-0.98955   Mean   :-0.4705   Mean   : 0.98548   Mean
: 0.5121
 3rd Qu.:2999.2   3rd Qu.: 0.8055   3rd Qu.: 0.03098   3rd Qu.: 0.8019   3rd Qu.: 1.89423   3rd
Qu.: 1.8360
 Max.   :3999.0   Max.   : 6.4064   Max.   : 5.79071   Max.   : 6.3749   Max.   : 7.61985   Max.
: 7.3644
    Ripeness          Acidity           Quality          SizeCrunchiness       AcidJuice
SizeDSweet
 Min.   :-5.8646   Min.   :-7.01054   Length:4000        Min.   :-19.33058   Min.   :-21.7504
Min.   :-193.9657
 1st Qu.:-0.7717   1st Qu.:-1.37742   Class :character   1st Qu.: -1.45504   1st Qu.: -0.9006
1st Qu.:  -1.2554
 Median : 0.5034   Median : 0.02261   Mode  :character   Median : -0.07887   Median :  0.2082
Median :  -0.2607
 Mean   : 0.4983   Mean   : 0.07688                      Mean   : -0.03640   Mean   :  1.0522
Mean   :  -0.0301
 3rd Qu.: 1.7662   3rd Qu.: 1.51049                      3rd Qu.:  1.12308   3rd Qu.:  2.4463
3rd Qu.:   0.7109
 Max.   : 7.2378   Max.   : 7.40474                      Max.   : 26.44163   Max.   : 32.6105
Max.   : 619.7229
```

Hide

```
head(apples)
```

	A... <int>	Size <dbl>	Weight <dbl>	Sweetness <dbl>	Crunchiness <dbl>	Juiciness <dbl>	Ripeness <dbl>	Acidity <dbl>	Q <c
1	0	-3.9700485	-2.512336	5.3463296	-1.01200871	1.8449004	0.32983980	-0.4915905	gc
2	1	-1.1952172	-2.839257	3.6640588	1.58823231	0.8532858	0.86753008	-0.7228094	gc
3	2	-0.2920239	-1.351282	-1.7384292	-0.34261593	2.8386355	-0.03803333	2.6216365	ba
4	3	-0.6571958	-2.271627	1.3248738	-0.09787472	3.6379705	-3.41376134	0.7907232	gc
5	4	1.3642168	-1.296612	-0.3846582	-0.55300577	3.0308744	-1.30384943	0.5019840	gc
6	5	-3.4253998	-1.409082	-1.9135112	-0.55577486	-3.8530715	1.91461592	-2.9815232	ba

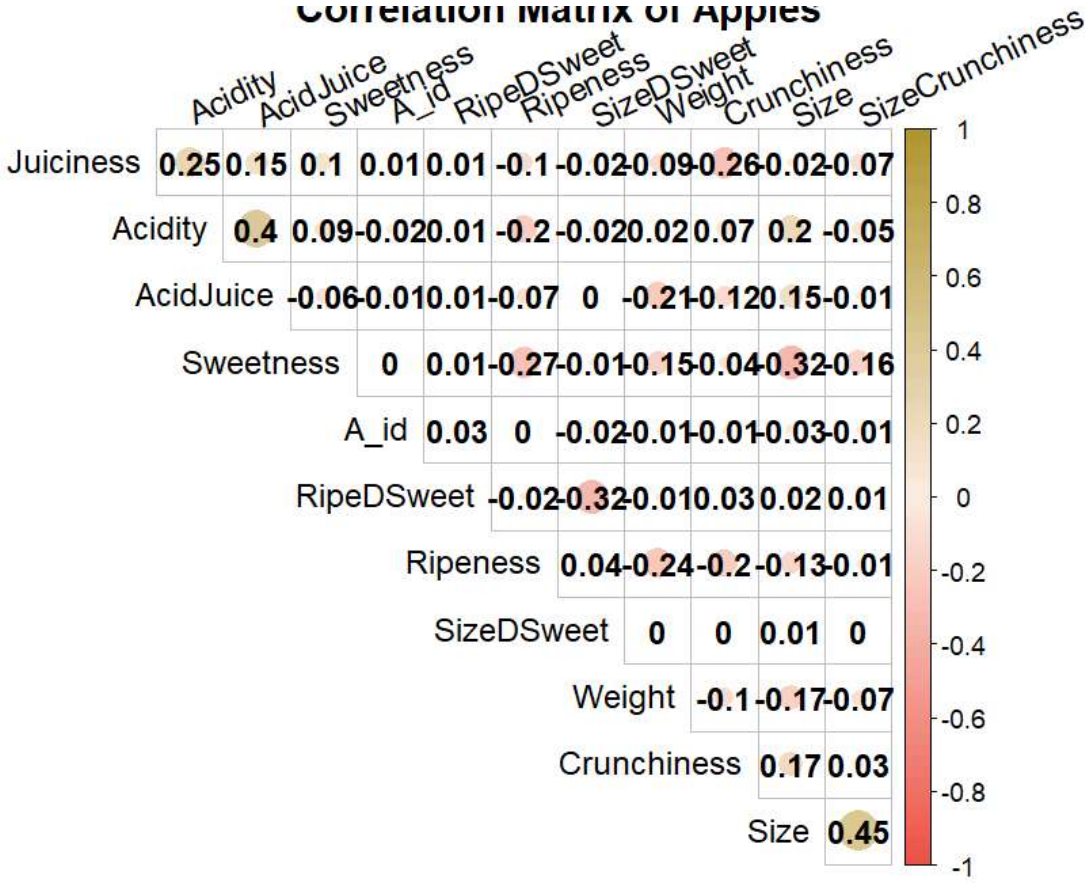
6 rows | 1-10 of 12 columns

Hide

```
cor_matrix <- cor(apples %>% select_if(is.numeric), use = "complete.obs")

# Custom color palette from CGPT to go between the apple colors
col <- colorRampPalette(c("#ef5449", "#ffefe4", "#b29831"))(200)


corrplot(cor_matrix, method = "circle", type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 25,
         addCoef.col = "black",
         col = col,
         title = "Correlation Matrix of Apples",
         diag = FALSE)
```

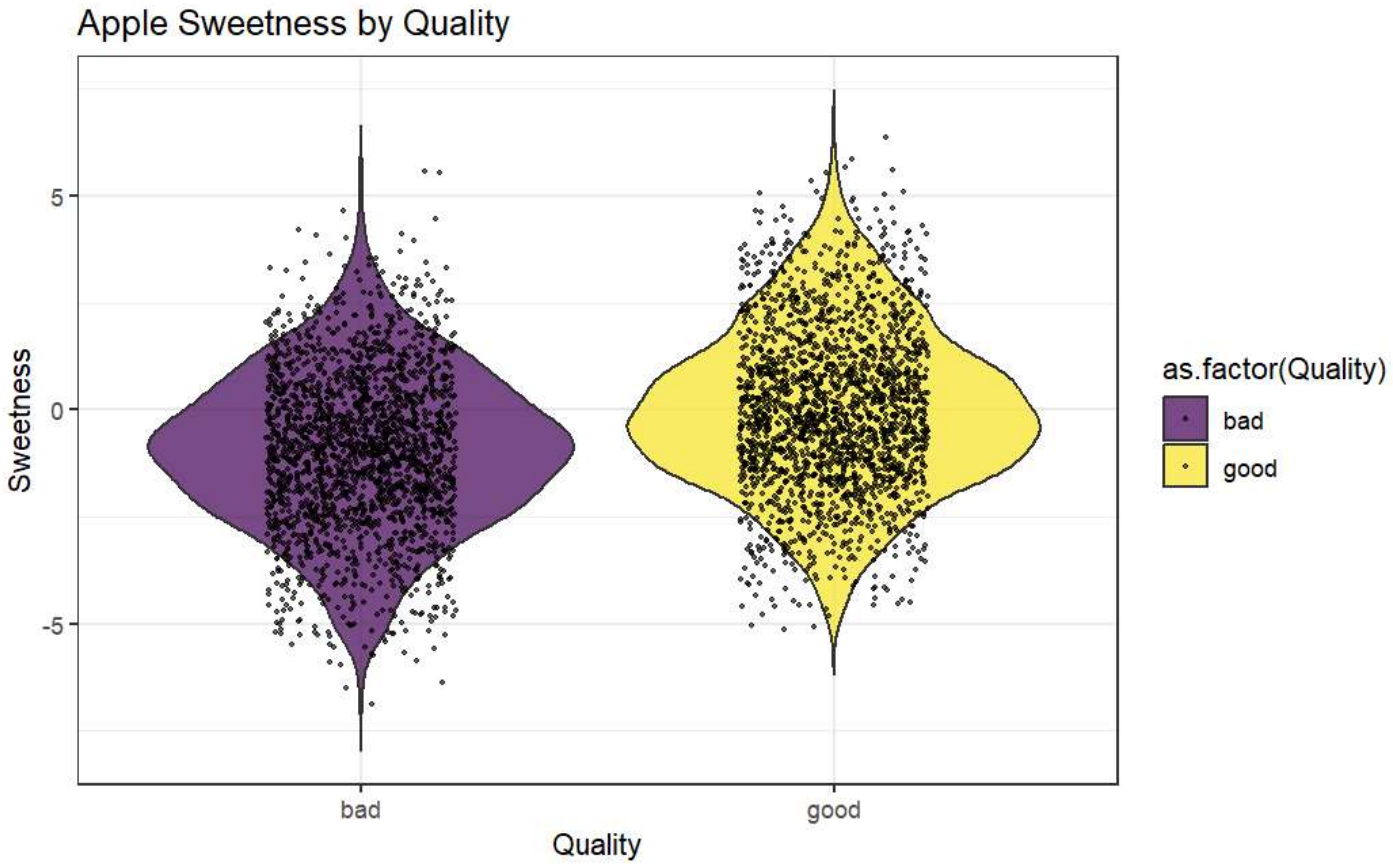


Hide

```
# The closer to 1 it is the more it is correlated, the closer to -1, the less it is.
```

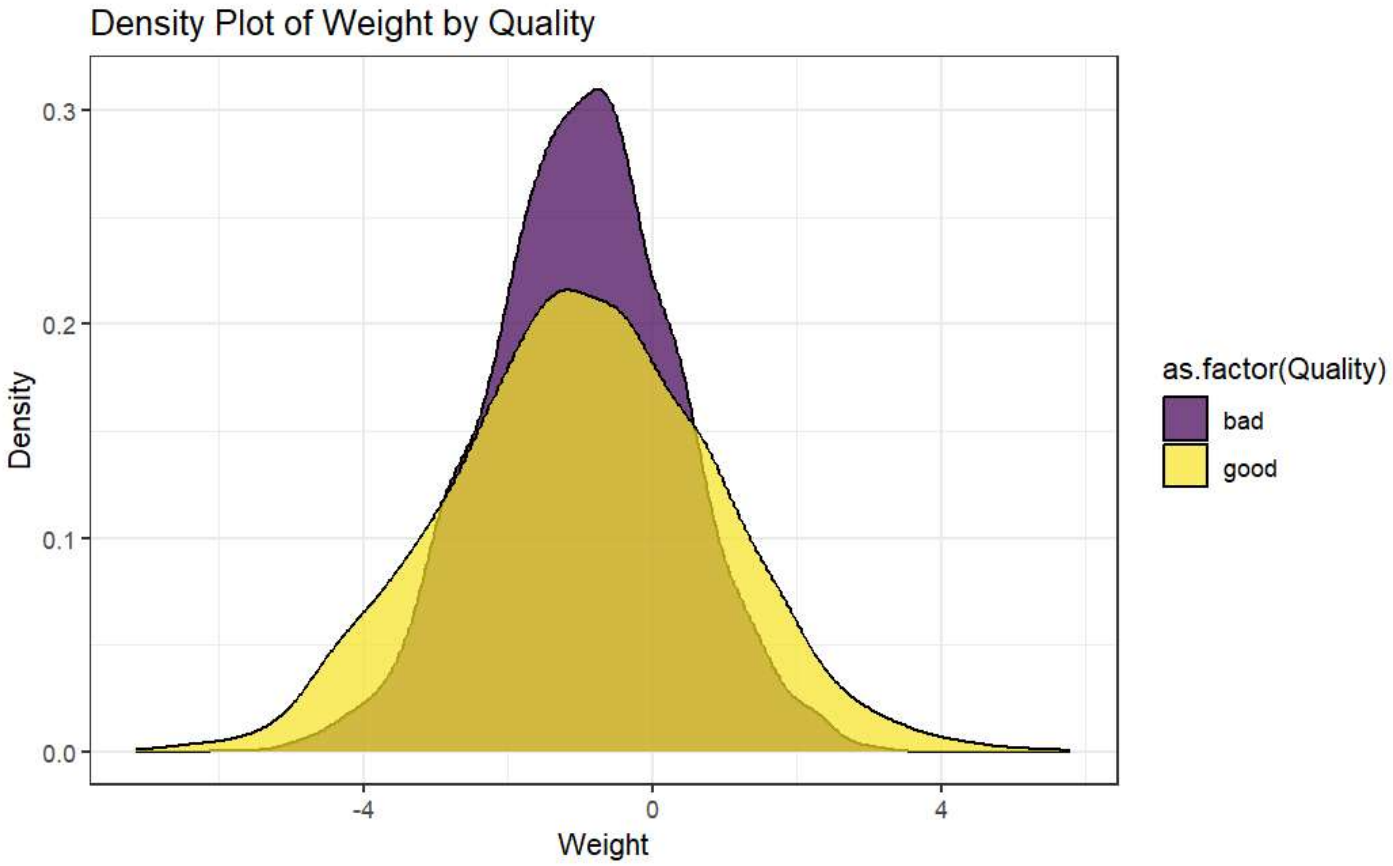
Hide

```
ggplot(apples, aes(x = as.factor(Quality), y = Sweetness, fill = as.factor(Quality))) +
  geom_violin(trim = FALSE, alpha = 0.7) +
  geom_jitter(width = 0.2, color = "black", size = 0.5, alpha = 0.6) +  # Add jittered points
  scale_fill_viridis_d() +  labs(title = "Apple Sweetness by Quality", x = "Quality", y = "Sweet
ness") +
  theme_bw()
```



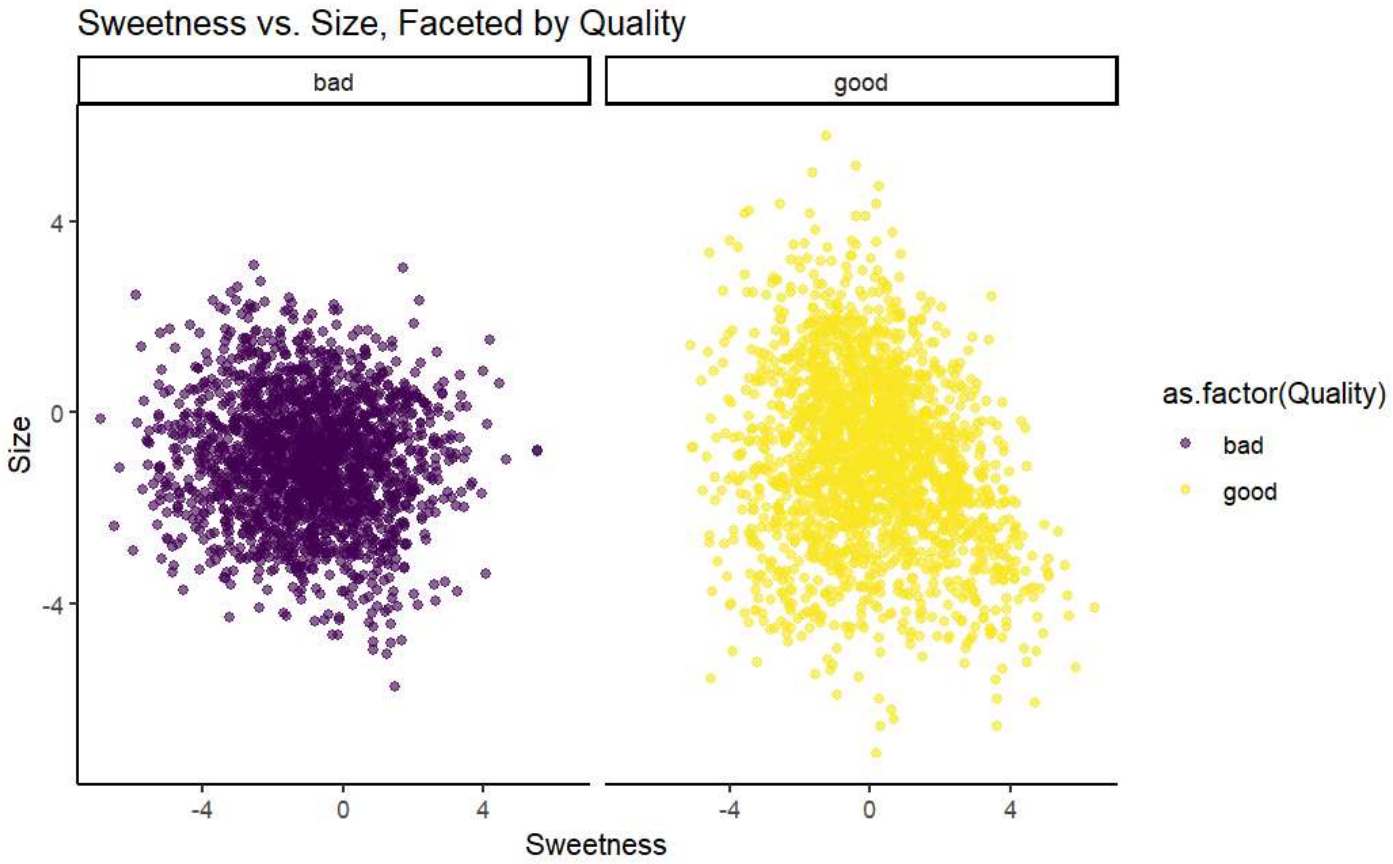
Hide

```
ggplot(apples, aes(x = Weight, fill = as.factor(Quality))) +
  geom_density(alpha = 0.7) +
  scale_fill_viridis_d() +
  labs(title = "Density Plot of Weight by Quality", x = "Weight", y = "Density") +
  theme_bw()
```



Hide

```
ggplot(apples, aes(x = Sweetness, y = Weight)) +
  geom_point(aes(color = as.factor(Quality)), alpha = 0.6) +
  facet_wrap(~Quality) +
  scale_color_viridis_d() +
  labs(title = "Sweetness vs. Size, Faceted by Quality", x = "Sweetness", y = "Size") +
  theme_classic()
```



Hide

```
set.seed(777)
#My favourite number <3

indexes <- createDataPartition(apples$Quality, p = 0.7, list = FALSE)
trainer <- apples[indexes, ]
tester <- apples[-indexes, ]



rmodel <- rpart(Quality ~ ., data = trainer,control = rpart.control(cp = 0.0038, minsplit = 60,
minbucket = 15, maxdepth = 7))

print(rmodel)
```

```
n= 2801

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 2801 1398 good (0.49910746 0.50089254)
    2) RipeDSweet< -0.2483992 1393  491 bad (0.64752333 0.35247667)
      4) SizeDSweet>=-0.6551097 1010  244 bad (0.75841584 0.24158416)
        8) Ripeness>=0.544325 736   92 bad (0.87500000 0.12500000)
         16) Weight< 0.6423137 642   57 bad (0.91121495 0.08878505) *
         17) Weight>=0.6423137 94   35 bad (0.62765957 0.37234043)
           34) Acidity>=0.02763898 54    8 bad (0.85185185 0.14814815) *
           35) Acidity< 0.02763898 40   13 good (0.32500000 0.67500000) *
        9) Ripeness< 0.544325 274  122 good (0.44525547 0.55474453)
         18) Juiciness< -0.4230122 81   10 bad (0.87654321 0.12345679) *
         19) Juiciness>=-0.4230122 193   51 good (0.26424870 0.73575130) *
      5) SizeDSweet< -0.6551097 383  136 good (0.35509138 0.64490862)
       10) Acidity>=2.196335 87   28 bad (0.67816092 0.32183908)
         20) Size< 1.305088 61    3 bad (0.95081967 0.04918033) *
         21) Size>=1.305088 26    1 good (0.03846154 0.96153846) *
       11) Acidity< 2.196335 296   77 good (0.26013514 0.73986486)
         22) Juiciness< -1.561125 43   12 bad (0.72093023 0.27906977) *
         23) Juiciness>=-1.561125 253   46 good (0.18181818 0.81818182)
           46) Crunchiness>=0.5670633 154   41 good (0.26623377 0.73376623)
             92) Juiciness< 1.601745 104   37 good (0.35576923 0.64423077)
              184) Size< 0.6354964 38   11 bad (0.71052632 0.28947368) *
              185) Size>=0.6354964 66   10 good (0.15151515 0.84848485) *
             93) Juiciness>=1.601745 50    4 good (0.08000000 0.92000000) *
           47) Crunchiness< 0.5670633 99    5 good (0.05050505 0.94949495) *
    3) RipeDSweet>=-0.2483992 1408  496 good (0.35227273 0.64772727)
      6) Juiciness< -1.545194 186   53 bad (0.71505376 0.28494624)
       12) Weight< 1.231498 165   36 bad (0.78181818 0.21818182) *
       13) Weight>=1.231498 21    4 good (0.19047619 0.80952381) *
      7) Juiciness>=-1.545194 1222  363 good (0.29705401 0.70294599)
       14) SizeDSweet< -1.700434 299  148 good (0.49498328 0.50501672)
         28) Size< -1.720563 158   38 bad (0.75949367 0.24050633)
           56) Acidity>=0.2444499 84    4 bad (0.95238095 0.04761905) *
           57) Acidity< 0.2444499 74   34 bad (0.54054054 0.45945946)
            114) Weight< -1.12806 27    2 bad (0.92592593 0.07407407) *
            115) Weight>=-1.12806 47   15 good (0.31914894 0.68085106) *
         29) Size>=-1.720563 141   28 good (0.19858156 0.80141844) *
       15) SizeDSweet>=-1.700434 923  215 good (0.23293608 0.76706392)
         30) Weight>=-2.236457 660  200 good (0.30303030 0.69696970)
           60) RipeDSweet< 0.1336296 179   89 good (0.49720670 0.50279330)
            120) AcidJuice>=-1.681743 135   53 bad (0.60740741 0.39259259)
              240) Weight< 1.103678 120   41 bad (0.65833333 0.34166667) *
              241) Weight>=1.103678 15    3 good (0.20000000 0.80000000) *
            121) AcidJuice< -1.681743 44    7 good (0.15909091 0.84090909) *
           61) RipeDSweet>=0.1336296 481  111 good (0.23076923 0.76923077)
            122) Ripeness< 0.1844071 311   99 good (0.31832797 0.68167203)
              244) Acidity>=3.587452 16    2 bad (0.87500000 0.12500000) *
              245) Acidity< 3.587452 295   85 good (0.28813559 0.71186441) *
```

```
            123) Ripeness>=0.1844071 170   12 good (0.07058824 0.92941176) *
         31) Weight< -2.236457 263   15 good (0.05703422 0.94296578) *
```

Hide

```
predictions <- predict(rmodel, tester, type = "class")
tester$Quality <- as.factor(tester$Quality)
predictions <- factor(predictions, levels = levels(tester$Quality))


conr <- confusionMatrix(predictions, tester$Quality)
print(conr)
```

```
Confusion Matrix and Statistics

          Reference
Prediction bad good
      bad  466  107
      good 132  494

               Accuracy : 0.8007
                 95% CI : (0.7769, 0.8229)
    No Information Rate : 0.5013
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6013

 Mcnemar's Test P-Value : 0.1206

            Sensitivity : 0.7793
            Specificity : 0.8220
         Pos Pred Value : 0.8133
         Neg Pred Value : 0.7891
             Prevalence : 0.4987
         Detection Rate : 0.3887
   Detection Prevalence : 0.4779
      Balanced Accuracy : 0.8006

       'Positive' Class : bad
```

Hide

```
# I got an accuracy rate of about 70%, which I thought was good, but not good enough yet. So we wanted to try K-Fold cross validation as shown by CGPT to get more accuracy. But we got a lower number, so instead we decided to add modifications to the rpart model instead.


train_control <- trainControl(method = "cv", number = 10)
grid <- expand.grid(.cp = seq(0.01, 0.1, by = 0.01))
model <- train(Quality ~ ., data = trainer, method = "rpart",
               trControl = train_control,
               tuneGrid = grid)
print(model)
```

```
CART

2801 samples
  12 predictor
   2 classes: 'bad', 'good'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2521, 2521, 2522, 2521, 2521, 2520, ...
Resampling results across tuning parameters:

  cp    Accuracy   Kappa
  0.01  0.7808136  0.5615655
  0.02  0.7615394  0.5229894
  0.03  0.7279664  0.4556641
  0.04  0.7069037  0.4136706
  0.05  0.7069037  0.4136706
  0.06  0.6804801  0.3606659
  0.07  0.6722658  0.3442374
  0.08  0.6497415  0.2992550
  0.09  0.6500923  0.3000347
  0.10  0.6422401  0.2844948

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.01.
```

Hide

```
bayes <- naiveBayes(Quality ~ ., data = trainer, laplace =2, usekernel = TRUE)
bayespredict <- predict(bayes,tester)

bayespredict <- factor(bayespredict, levels = levels(tester$Quality))
conBayes <- caret::confusionMatrix(bayespredict, tester$Quality)

print(conBayes)
```

```
Confusion Matrix and Statistics

          Reference
Prediction bad good
      bad  536  267
      good  62  334

               Accuracy : 0.7256
                 95% CI : (0.6994, 0.7507)
    No Information Rate : 0.5013
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4517

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8963
            Specificity : 0.5557
         Pos Pred Value : 0.6675
         Neg Pred Value : 0.8434
             Prevalence : 0.4987
         Detection Rate : 0.4470
   Detection Prevalence : 0.6697
      Balanced Accuracy : 0.7260

       'Positive' Class : bad
```

Hide

```
raccuracy <- conr$overall['Accuracy']
naccuracy <- conBayes$overall['Accuracy']


cat("Accuracy of rpart model:", raccuracy, "\n")
```

```
Accuracy of rpart model: 0.8006672
```

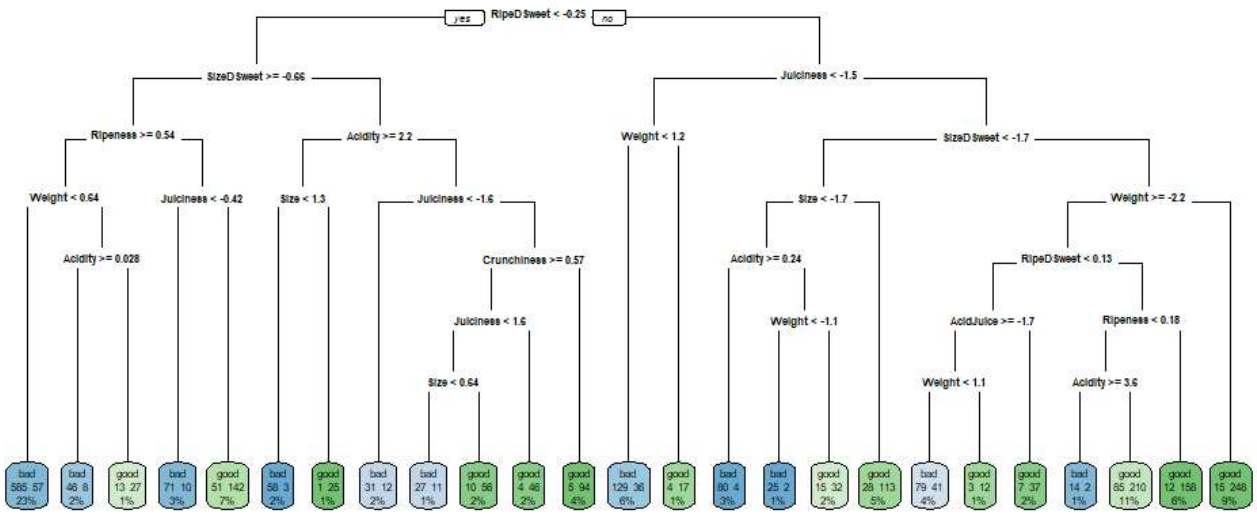
Hide

```
cat("Accuracy of Naïve Bayes model:", naccuracy, "\n")
```

```
Accuracy of Naïve Bayes model: 0.7256047
```

Hide

```
rpart.plot(rmodel, type = 0, extra = 101)
```



Hide

```
importance <- varImp(model)$importance
print(importance)
```

	Overall <dbl>
Acidity	23.70687
AcidJuice	17.67466
Crunchiness	12.16460
Juiciness	73.07352
RipeDSweet	37.16836
Ripeness	79.96798
Size	100.00000
SizeCrunchiness	24.04799
SizeDSweet	43.67450
Sweetness	76.44047

1-10 of 12 rows Previous **1** 2 Next

Hide

NA