ML modeling

Shrusti Ghela

7/14/2022

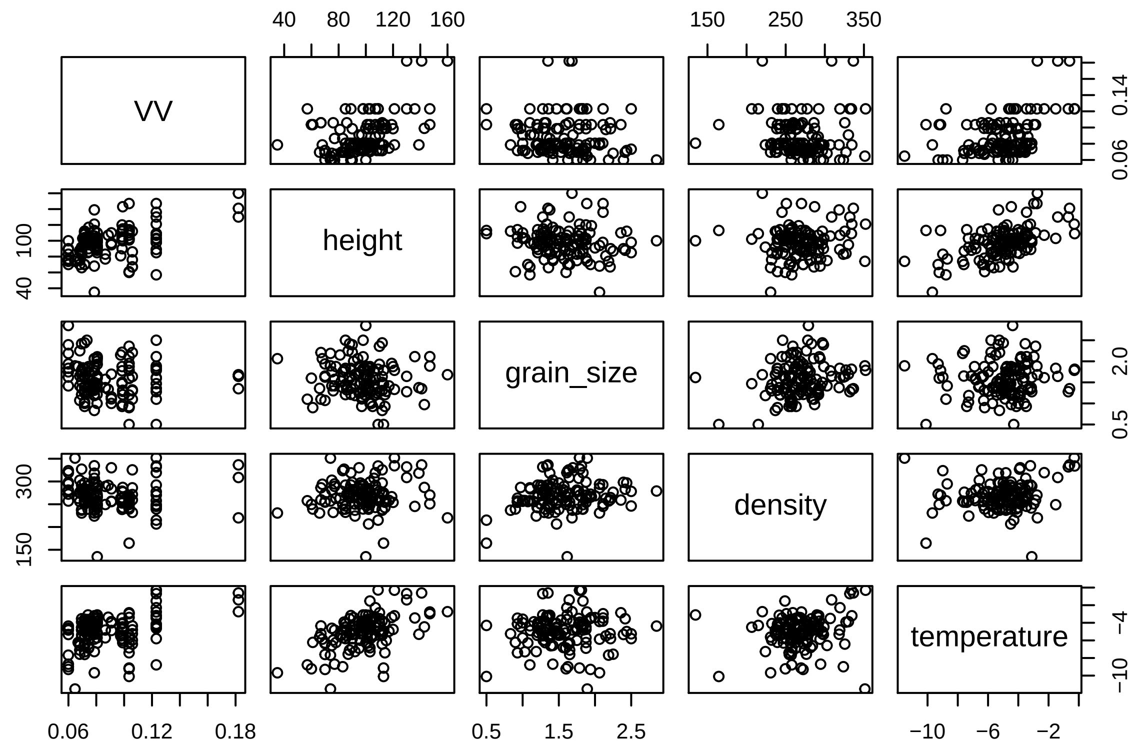
```r
data <- read.csv("~/Desktop/finaldata.csv")
```

```r
data <- as.data.frame(data)
```

```r
head(data)
```

```
##   X               pit_id         HH         HV         VH        VV height
## 1 0    COGM2S37_20200201 0.09027195 0.03370955 0.03457453 0.0716215    104
## 2 1 COGMSO_20200321_1006 0.15180993 0.05463321 0.05498413 0.1230603    136
## 3 2    COGM5S31_20200130 0.09027195 0.03370955 0.03457453 0.0716215    112
## 4 3 COGMCO_20200318_0825 0.14924057 0.04958111 0.04776910 0.1035636    119
## 5 4     COGM2C2_20200131 0.07149103 0.02265394 0.02551519 0.0681664     87
## 6 5 COGMSO_20200328_1630 0.15180993 0.05463321 0.05498413 0.1230603    147
##   grain_size  density temperature
## 1   1.277778 269.3182   -5.476923
## 2   2.111111 245.3077   -3.453333
## 3   1.305556 275.8750   -3.784615
## 4   1.950000 266.5694   -3.392857
## 5   2.250000 276.4286   -7.570000
## 6   1.886364 270.1786   -2.768750
```

```r
pairs(data[6:10])
```




```r
data$density <- scale(data$density, center=TRUE, scale=TRUE)
data$height <- scale(data$height,center=TRUE,scale=TRUE)
data$temperature <- scale(data$temperature,center=TRUE,scale=TRUE)
data$grain_size <- scale(data$grain_size,center=TRUE,scale=TRUE)
```

```r
data <- na.omit(data)
```

```r
library(dplyr)
```

```
##
## 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
```

```r
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
set.seed(1)
# Validation Set appraoch
training.samples <- data$VV %>%
  createDataPartition(p = 0.8, list = FALSE)      #creating 80-20 train-test split (because of less numb
train.data <- data[training.samples, ]
test.data <- data[-training.samples, ]

write.csv(train.data, "~/Desktop/train.data.csv")
write.csv(test.data, "~/Desktop/test.data.csv")
```

```r
#Linear regression for HH
lm.fit <- lm(HH ~ height + grain_size + density + temperature , data= train.data)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = HH ~ height + grain_size + density + temperature,
##     data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.04217 -0.02231 -0.00624  0.01665  0.08636
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.110480   0.002991  36.932  < 2e-16 ***
## height       0.014677   0.003385   4.336 3.58e-05 ***
## grain_size  -0.003368   0.003089  -1.090   0.2783
## density     -0.005373   0.003048  -1.763   0.0811 .
## temperature  0.003768   0.003606   1.045   0.2987
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03001 on 96 degrees of freedom
## Multiple R-squared:  0.2894, Adjusted R-squared:  0.2598
## F-statistic: 9.775 on 4 and 96 DF,  p-value: 1.124e-06
```

```r
HH_lm.fit <- predict(lm.fit, test.data)
HH_lm.fit
```

```
##          5         11         13         15         16         23         32
## 0.09070494 0.12510268 0.12044519 0.08958814 0.11818304 0.11522718 0.09759574
##         36         40         53         56         58         64         68
## 0.10812614 0.09667161 0.11095613 0.08330467 0.11027569 0.12068552 0.10964494
##         72         87         91         95        100        102        104
## 0.08465593 0.11885455 0.08210549 0.12171834 0.13202268 0.11768610 0.12347501
##        106        107        122
## 0.11925569 0.11880822 0.10600479
```

```r
vs.error <- sqrt(mean((test.data$HH - predict(lm.fit, test.data))^2))
vs.error.1 <- sqrt(mean((train.data$HH - predict(lm.fit, train.data))^2))
```

```r
vs.error
```

```
## [1] 0.02338507
```

```r
vs.error.1
```

```
## [1] 0.02925332
```

```r
#Linear regression for HV
lm.fit <- lm(HH ~ height + grain_size + density + temperature , data= train.data)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = HH ~ height + grain_size + density + temperature,
##     data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.04217 -0.02231 -0.00624  0.01665  0.08636
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.110480   0.002991  36.932  < 2e-16 ***
## height       0.014677   0.003385   4.336 3.58e-05 ***
## grain_size  -0.003368   0.003089  -1.090   0.2783
## density     -0.005373   0.003048  -1.763   0.0811 .
## temperature  0.003768   0.003606   1.045   0.2987
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03001 on 96 degrees of freedom
## Multiple R-squared:  0.2894, Adjusted R-squared:  0.2598
## F-statistic: 9.775 on 4 and 96 DF,  p-value: 1.124e-06
```

```r
HV_lm.fit <- predict(lm.fit, test.data)
vs.error <- sqrt(mean((test.data$HV - predict(lm.fit, test.data))^2))
vs.error.1 <- sqrt(mean((train.data$HV - predict(lm.fit, train.data))^2))
```

```r
vs.error
```

```
## [1] 0.0715948
```

```r
vs.error.1
```

```
## [1] 0.07241566
```

```r
#Linear regression for VV
lm.fit <- lm(VV ~ height + grain_size + density + temperature , data= train.data)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = VV ~ height + grain_size + density + temperature,
##     data = train.data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.027247 -0.015094 -0.007389  0.008932  0.073664
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.088924   0.002170  40.980  < 2e-16 ***
## height       0.008582   0.002456   3.495 0.000719 ***
## grain_size  -0.002040   0.002241  -0.910 0.364935
## density     -0.002298   0.002211  -1.039 0.301321
## temperature  0.004490   0.002616   1.717 0.089270 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02177 on 96 degrees of freedom
## Multiple R-squared:  0.2508, Adjusted R-squared:  0.2196
## F-statistic: 8.034 on 4 and 96 DF,  p-value: 1.248e-05
```

```r
VV_lm.fit <- predict(lm.fit, test.data)
vs.error <- sqrt(mean((test.data$VV - predict(lm.fit, test.data))^2))
vs.error.1 <- sqrt(mean((train.data$VV - predict(lm.fit, train.data))^2))
```

```r
vs.error
```

```
## [1] 0.01597983
```

```r
vs.error.1
```

```
## [1] 0.02122012
```

```r
#Linear regression for VH
lm.fit <- lm(VH ~ height + grain_size + density + temperature , data= train.data)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = VH ~ height + grain_size + density + temperature,
##     data = train.data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.014962 -0.007514 -0.001954  0.004688  0.034316
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0412734  0.0010004  41.257  < 2e-16 ***
## height       0.0043415  0.0011321   3.835 0.000225 ***
## grain_size  -0.0006025  0.0010330  -0.583 0.561089
## density     -0.0013169  0.0010194  -1.292 0.199515
## temperature  0.0020163  0.0012058   1.672 0.097751 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01003 on 96 degrees of freedom
## Multiple R-squared:  0.2698, Adjusted R-squared:  0.2394
## F-statistic: 8.867 on 4 and 96 DF,  p-value: 3.892e-06
```

```r
VH_lm.fit <- predict(lm.fit, test.data)
vs.error <- sqrt(mean((test.data$VH - predict(lm.fit, test.data))^2))
vs.error.1 <- sqrt(mean((train.data$VH - predict(lm.fit, train.data))^2))
```

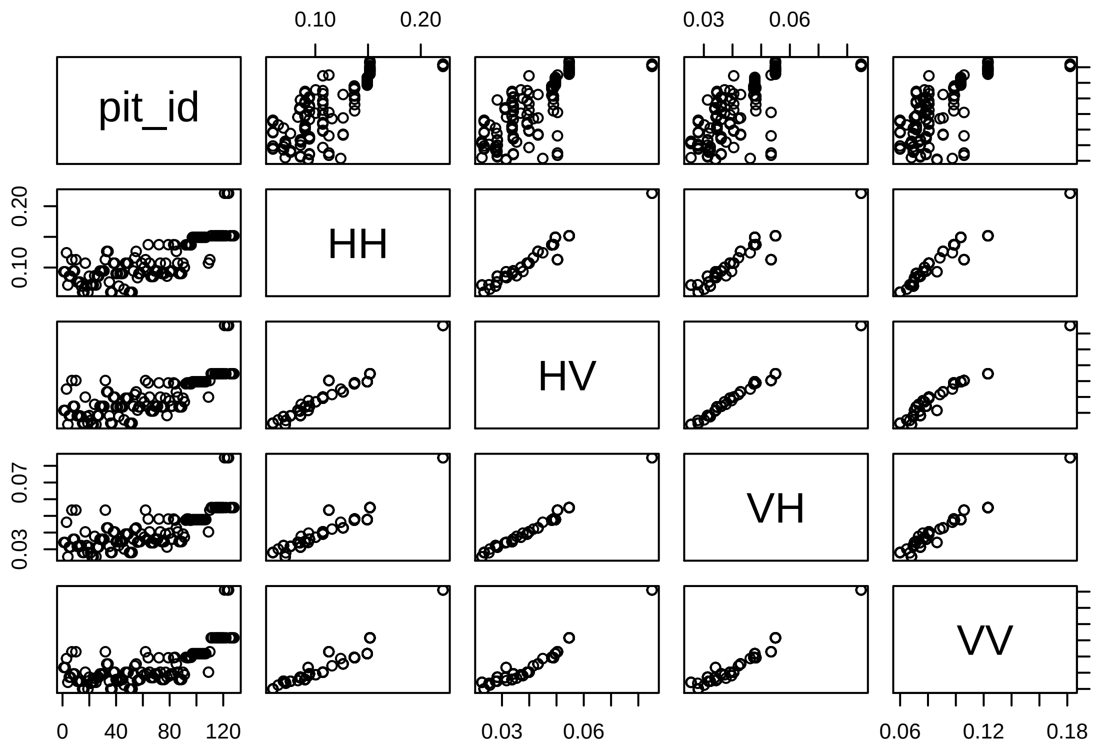
```r
vs.error
```

```
## [1] 0.008213999
```

```r
vs.error.1
```

```
## [1] 0.009782923
```

```r
pairs(data[2:6])
```




```r
library(rpart)

mytree <- rpart(
  VV ~ height + grain_size + density + temperature,
  data = train.data,
  method = "anova"
)

mytree
```

```
## n= 101
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 101 0.060703670 0.08906841
##    2) temperature< 1.110414 93 0.026818170 0.08424122
##      4) density>=-0.4689942 67 0.015646570 0.08049126
##        8) grain_size>=0.5147475 22 0.003415090 0.07420710
##         16) density>=0.02454802 13 0.000737969 0.06912607 *
##         17) density< 0.02454802 9 0.001856716 0.08154638 *
##        9) grain_size< 0.5147475 45 0.010937950 0.08356352
##         18) height< -0.6347133 8 0.002771827 0.07731662 *
##         19) height>=-0.6347133 37 0.007786428 0.08491420
##           38) grain_size< 0.0107652 27 0.004429123 0.08180180 *
```

```
##           39) grain_size>=0.0107652 10 0.002389573 0.09331767 *
##      5) density< -0.4689942 26 0.007801535 0.09390459
##       10) height< 0.1866562 15 0.003543639 0.08559345 *
##       11) height>=0.1866562 11 0.001808872 0.10523800 *
##    3) temperature>=1.110414 8 0.006526357 0.14518440 *
```

```r
library(rattle)
```

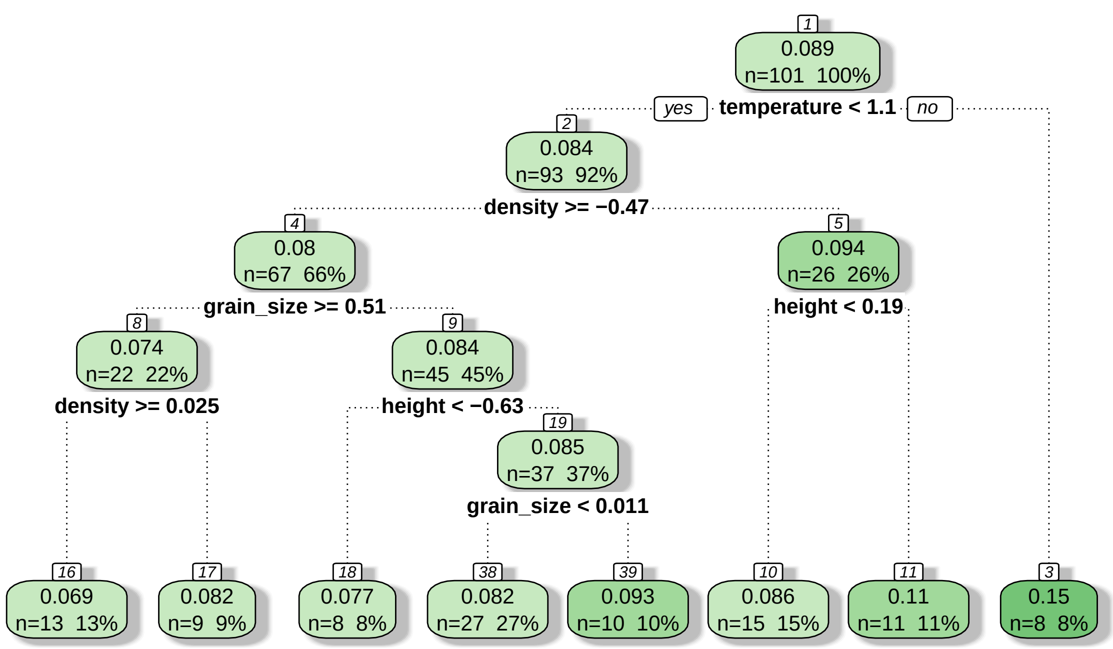
```
## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(rpart.plot)
library(RColorBrewer)

# plot mytree
fancyRpartPlot(mytree, caption = NULL)
```



```r
VV_tree <- predict(mytree, test.data)
vs.error <- sqrt(mean((test.data$VV - predict(mytree, test.data))^2))
vs.error.1 <- sqrt(mean((train.data$VV - predict(mytree, train.data))^2))
```

```r
vs.error
```

```
## [1] 0.01948291
```

```r
vs.error.1
```

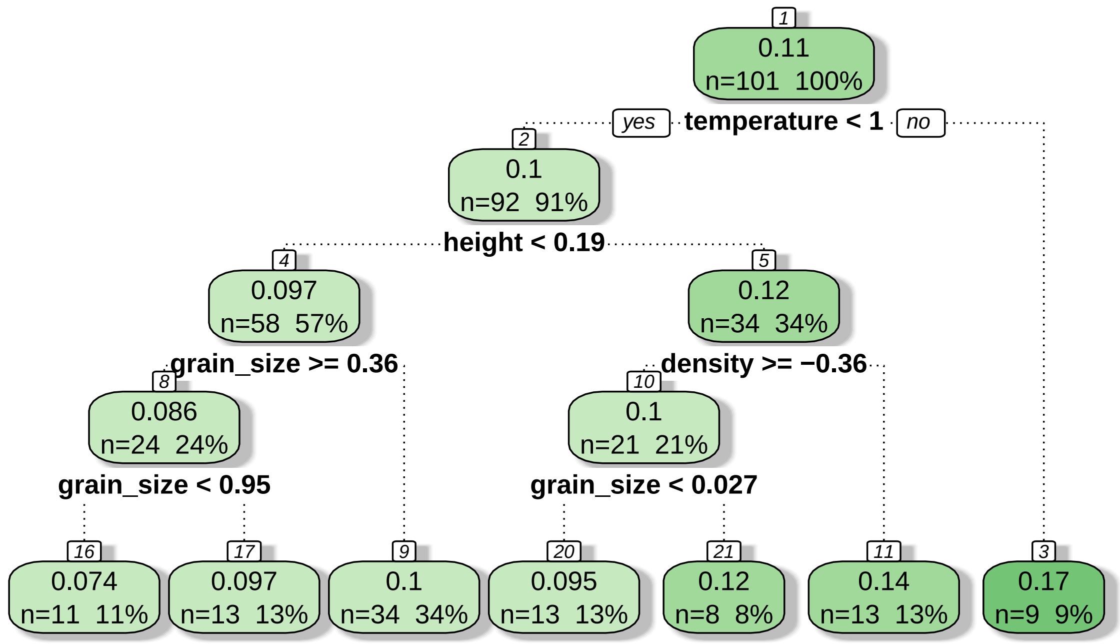
```
## [1] 0.01543561
```

```r
mytree <- rpart(
  HH ~ height + grain_size + density + temperature,
  data = train.data,
  method = "anova"
)

mytree
```

```
## n= 101
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 101 0.121633100 0.11078110
##    2) temperature< 1.022113 92 0.071658680 0.10453680
##      4) height< 0.1866562 58 0.039879230 0.09723997
##        8) grain_size>=0.3556577 24 0.015120950 0.08649578
##         16) grain_size< 0.9457146 11 0.001395961 0.07364013 *
##         17) grain_size>=0.9457146 13 0.010368780 0.09737364 *
##        9) grain_size< 0.3556577 34 0.020032130 0.10482410 *
##      5) height>=0.1866562 34 0.023423290 0.11698440
##       10) density>=-0.355763 21 0.009070171 0.10264340
##         20) grain_size< 0.02738653 13 0.002000204 0.09473375 *
##         21) grain_size>=0.02738653 8 0.004935017 0.11549660 *
##       11) density< -0.355763 13 0.003057463 0.14015050 *
##    3) temperature>=1.022113 9 0.009718551 0.17461130 *
```

```r
# plot mytree
fancyRpartPlot(mytree, caption = NULL)
```




```
HH_tree <- predict(mytree, test.data)
vs.error <- sqrt(mean((test.data$HH - predict(mytree, test.data))^2))
vs.error.1 <- sqrt(mean((train.data$HH - predict(mytree, train.data))^2))
```

```
vs.error
```

```
## [1] 0.03472694
```

```
vs.error.1
```

```
## [1] 0.02258276
```

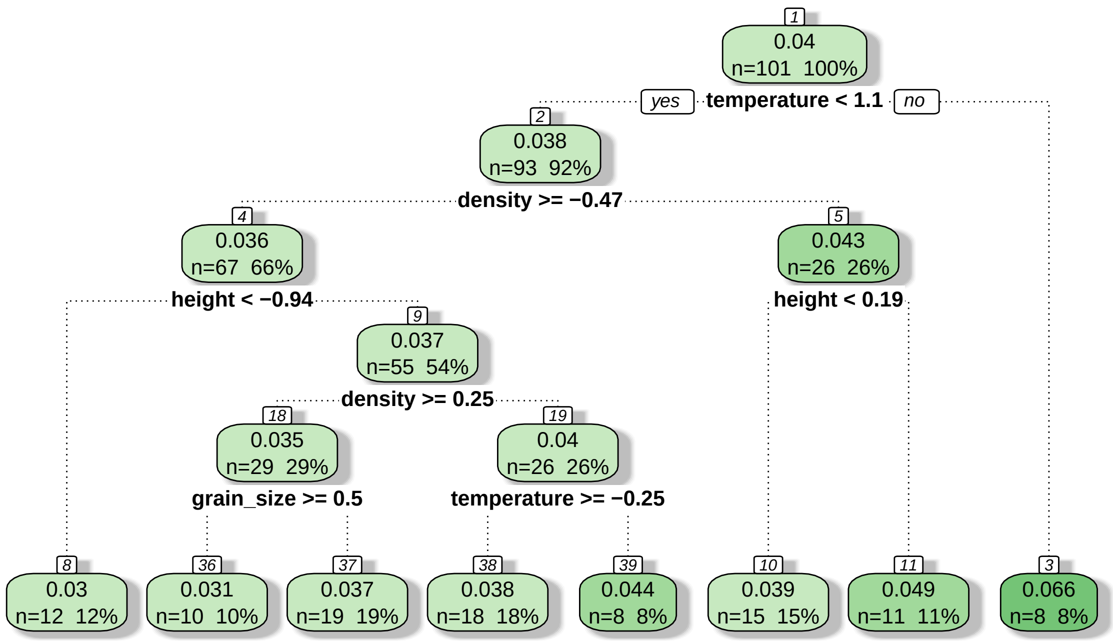
```
mytree <- rpart(
  HV ~ height + grain_size + density + temperature,
  data = train.data,
  method = "anova"
)

mytree
```

```
## n= 101
##
## node), split, n, deviance, yval
##       * denotes terminal node
```

```
##
##  1) root 101 0.0159877200 0.04016172
##    2) temperature< 1.110414 93 0.0084387360 0.03793611
##      4) density>=-0.4689942 67 0.0053468900 0.03589493
##        8) height< -0.9427269 12 0.0009298805 0.02951490 *
##        9) height>=-0.9427269 55 0.0038219790 0.03728694
##         18) density>=0.2470216 29 0.0019083400 0.03508933
##           36) grain_size>=0.5019253 10 0.0003716354 0.03072198 *
##           37) grain_size< 0.5019253 19 0.0012455790 0.03738794 *
##         19) density< 0.2470216 26 0.0016173700 0.03973812
##           38) temperature>=-0.2460669 18 0.0009372667 0.03786979 *
##           39) temperature< -0.2460669 8 0.0004759000 0.04394186 *
##      5) density< -0.4689942 26 0.0020933520 0.04319606
##       10) height< 0.1866562 15 0.0010524150 0.03867337 *
##       11) height>=0.1866562 11 0.0003157219 0.04936338 *
##    3) temperature>=1.110414 8 0.0017331620 0.06603440 *
```

```r
# plot mytree
fancyRpartPlot(mytree, caption = NULL)
```




```r
HV_tree <- predict(mytree, test.data)
vs.error <- sqrt(mean((test.data$HV - predict(mytree, test.data))^2))
vs.error.1 <- sqrt(mean((train.data$HV - predict(mytree, train.data))^2))
```

```
vs.error
```

```
## [1] 0.0120355
```

```
vs.error.1
```

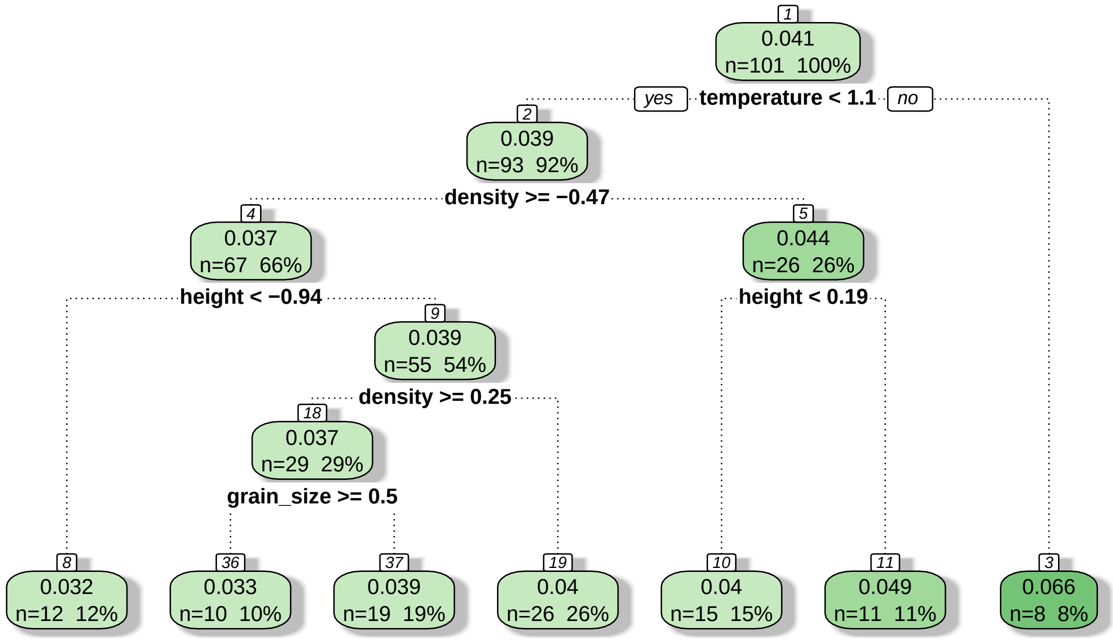
```
## [1] 0.008361605
```

```
mytree <- rpart(
  VH ~ height + grain_size + density + temperature,
  data = train.data,
  method = "anova"
)

mytree
```

```
## n= 101
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 101 0.0132375900 0.04136295
##    2) temperature< 1.110414 93 0.0062139700 0.03922799
##      4) density>=-0.4689942 67 0.0039269740 0.03748687
##        8) height< -0.9427269 12 0.0007122880 0.03228576 *
##        9) height>=-0.9427269 55 0.0028192410 0.03862166
##         18) density>=0.2470216 29 0.0013960000 0.03697434
##           36) grain_size>=0.5019253 10 0.0002106235 0.03349532 *
##           37) grain_size< 0.5019253 19 0.0010006370 0.03880540 *
##         19) density< 0.2470216 26 0.0012567680 0.04045906 *
##      5) density< -0.4689942 26 0.0015604910 0.04371470
##       10) height< 0.1866562 15 0.0007899759 0.04000904 *
##       11) height>=0.1866562 11 0.0002836552 0.04876788 *
##    3) temperature>=1.110414 8 0.0016718700 0.06618191 *
```

```
# plot mytree
fancyRpartPlot(mytree, caption = NULL)
```




```
VH_tree <- predict(mytree, test.data)
vs.error <- sqrt(mean((test.data$VH - predict(mytree, test.data))^2))
vs.error.1 <- sqrt(mean((train.data$VH - predict(mytree, train.data))^2))
```

```
vs.error
```

```
## [1] 0.01065279
```

```
vs.error.1
```

```
## [1] 0.00765973
```