Title: rpart Model

Cart algorithm pseudocode

https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/ (https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/)

Comparison of C5.0 & CART Classification algorithms using pruning technique

Prof. Nilima Patil, Prof.Rekha Lathi, Prof.Vidya Chitre

- C5.0 works by splitting based on the maximum information gain.
- Information Gain:
    - Let S be the sample:
    - C_i is Class $I; i = {1, 2,…, m_{I}(s1, s2, .., sm)} = -log_{2}())
    - S_i is the number of samples in class $i, P_{i} = , log_{2} is the binary logarithm
    - Let Attribute A have v distinct values.
    - Entropy = $E(A) = \sum_{j} = 1 \frac{(S1j+S2j+...+Smj)}{S} * I(s1j, ..., smj)$
    - Where S_{ij} is samples in Class i and subset j of Attribute A.
    - $I * (S1j, S2j, ..., Smj) = -\sum(P_{ij}log2(P_{ij}))$
    - Gain(A) = $I * (s1, s2, ..., sm) - E(A)$

C5.0 uses Shannon’s information gain

CART uses GINI index

CART Pseudocode:

Inputs:

1. R: a set of non-target attributes,
2. C: the target attribute,
3. S: training data.

Output:

1. Returns a decision tree

Start:

1. Initialize to empty tree;
2. If S is empty then { Return a single node failure value }
3. If S is made only for the values of the same target then { Return a single node of this value }
4. If R is empty then { Return a single node with value as the most common value of the target attribute values found in S }
5. Assign D = the attribute that has the largest Gain (D, S) from all the attributes of R.
6. Assign $d_j = 1, 2, ..., m$ = Attribute values of D
7. $S_j : j = 1, 2, ..., m$ = The subsets of S respectively constituted of djrecords attribute value D.

8. Return a tree whose root is D and the arcs are labeled by {d1, d2, …, dm} and going to sub-trees ID3 (R-{D}, C, S1), ID3 (R-{D} C, S2), .., ID3 (R-{D}, C, Sm)

End

---

https://techdifferences.com/difference-between-algorithm-and-pseudocode.html (https://techdifferences.com/difference-between-algorithm-and-pseudocode.html)

Difference Between Algorithm and Pseudocode

Comparison Chart

Pseudocode

Basis for comparison	**Algorithm**	**Pseudocode**
Comprehensibility	Hard to understand	Easy to interpret
Uses	Complicated programming language	Combination of programming language and natural language
Debugging	Moderate	Simpler
Ease of construction	Complex	Easier

Title: Cart Model Using Package 'rpart'

Libraries

```
Libraries = c("doMC", "caret", "rpart", "relaimpo", "beepr")

for(p in Libraries){  # Install Library if not present
    if(!require(p, character.only = TRUE)) { install.packages(p) }
    library(p, character.only = TRUE)
}
```

Import data & data handling

```
test_harness_paa <- read.csv("test_harness_paa.csv")
test_harness_paa <- test_harness_paa[, -c(2,3)]
Class <- as.factor(test_harness_paa$Class)
```

Partition data into training and testing sets

```
set.seed(1000)
index <- createDataPartition(test_harness_paa$Class, p = 0.8, list = FALSE)

training_set <- test_harness_paa[ index,]
test_set     <- test_harness_paa[-index,]

Class_test <- as.factor(test_set$Class)
```

CART Machine Model Building

```
set.seed(1000)
registerDoMC(cores = 3) # Start multi-processor mode
start_time <- Sys.time() # Start timer

# Create models
tcontrol <- trainControl(method = "repeatedcv",
                         number = 10, # 10X fold CV repeated 5X
                         repeats = 5,
                         allowParallel = TRUE)
# Train Model
model_list <- train(Class ~ .,
                    data = training_set,
                    methodList = "rpart",
                    trControl = tcontrol)

end_time <- Sys.time()   # End timer
end_time - start_time    # Display time
```

```
## Time difference of 7.167068 mins
```

```
registerDoSEQ() # Stop multi-processor mode
```

C5.0: Model Summary

```
model_list
```

```
## Random Forest
##
## 2800 samples
##   20 predictors
##    7 classes: 'Ctrl', 'Ery', 'Hcy', 'Hgb', 'Hhe', 'Lgb', 'Mgb'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 2520, 2520, 2520, 2520, 2520, 2520, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9078571  0.8925000
##   11    0.8931429  0.8753333
##   20    0.8748571  0.8540000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

C5.0: Predict new samples (Class_test)

```
Predicted_test_vals <- predict(model_list, test_set[, -1])

length(Predicted_test_vals)
```

```
## [1] 700
```

C5.0: Quick Summary

```
summary(Predicted_test_vals)
```

```
## Ctrl  Ery  Hcy  Hgb  Hhe  Lgb  Mgb
##   96   98  102   94  112  103   95
```

C5.0: Confusion Matrix

```
confusionMatrix(Predicted_test_vals,
                Class_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Ctrl Ery Hcy Hgb Hhe Lgb Mgb
##       Ctrl   90   1   1   4   0   0   0
##       Ery     0  92   3   2   0   0   1
##       Hcy     4   0  92   2   4   0   0
##       Hgb     0   2   2  81   4   0   5
##       Hhe     4   2   2   9  92   0   3
##       Lgb     0   0   0   2   0 100   1
##       Mgb     2   3   0   0   0   0  90
##
## Overall Statistics
##
##                Accuracy : 0.91
##                  95% CI : (0.8863, 0.9301)
##     No Information Rate : 0.1429
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.895
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Ctrl Class: Ery Class: Hcy Class: Hgb
## Sensitivity               0.9000     0.9200     0.9200     0.8100
## Specificity               0.9900     0.9900     0.9833     0.9783
## Pos Pred Value            0.9375     0.9388     0.9020     0.8617
## Neg Pred Value            0.9834     0.9867     0.9866     0.9686
## Prevalence                0.1429     0.1429     0.1429     0.1429
## Detection Rate            0.1286     0.1314     0.1314     0.1157
## Detection Prevalence      0.1371     0.1400     0.1457     0.1343
## Balanced Accuracy         0.9450     0.9550     0.9517     0.8942
##                      Class: Hhe Class: Lgb Class: Mgb
## Sensitivity              0.9200     1.0000     0.9000
## Specificity              0.9667     0.9950     0.9917
## Pos Pred Value           0.8214     0.9709     0.9474
## Neg Pred Value           0.9864     1.0000     0.9835
## Prevalence               0.1429     0.1429     0.1429
## Detection Rate           0.1314     0.1429     0.1286
## Detection Prevalence     0.1600     0.1471     0.1357
## Balanced Accuracy        0.9433     0.9975     0.9458
```

EOF

title: “Relative Importance from Linear Regression.rmd” author: “Matthew Curcio”

Title: Relative Importance from Linear Regression - relaimpo

Summary:

Import data & subset X values only

```
test_harness_paa <- read.csv("test_harness_paa.csv")
test_harness_paa = test_harness_paa[,-c(1,2,3)]
```

Build linear regression model

```
start_time <- Sys.time() # Start timer

lmMod <- lm(E ~ ., test_harness_paa)

Sys.time() - start_time # Display Time Difference
```

```
## Time difference of 0.0171802 secs
```

```
lmMod
```

```
##
## Call:
## lm(formula = E ~ ., data = test_harness_paa)
##
## Coefficients:
## (Intercept)            G            P            A            V
##      0.6329      -0.7342      -0.6232      -0.6426      -0.5464
##           L            I            M            C            F
##     -0.6089      -0.5732      -0.4976      -0.6255      -0.6730
##           Y            W            H            K            R
##     -0.5974      -0.5565      -0.5023      -0.4870      -0.4822
##           Q            N            D            S            T
##     -0.5771      -0.6687      -0.6494      -0.6986      -0.6377
```

Summary

```
summary(lmMod)
```

```
##
## Call:
## lm(formula = E ~ ., data = test_harness_paa)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.188702 -0.004501  0.000722  0.005394  0.071789
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.632913   0.007520   84.16   <2e-16 ***
## G           -0.734190   0.011802  -62.21   <2e-16 ***
## P           -0.623236   0.013372  -46.61   <2e-16 ***
## A           -0.642622   0.009395  -68.40   <2e-16 ***
## V           -0.546390   0.011803  -46.29   <2e-16 ***
## L           -0.608898   0.013486  -45.15   <2e-16 ***
## I           -0.573217   0.013862  -41.35   <2e-16 ***
## M           -0.497552   0.019531  -25.48   <2e-16 ***
## C           -0.625512   0.024549  -25.48   <2e-16 ***
## F           -0.673005   0.014999  -44.87   <2e-16 ***
## Y           -0.597408   0.013951  -42.82   <2e-16 ***
## W           -0.556483   0.025425  -21.89   <2e-16 ***
## H           -0.502319   0.016455  -30.53   <2e-16 ***
## K           -0.487028   0.012271  -39.69   <2e-16 ***
## R           -0.482225   0.014268  -33.80   <2e-16 ***
## Q           -0.577072   0.014372  -40.15   <2e-16 ***
## N           -0.668668   0.012642  -52.89   <2e-16 ***
## D           -0.649428   0.013296  -48.84   <2e-16 ***
## S           -0.698571   0.012684  -55.07   <2e-16 ***
## T           -0.637749   0.014421  -44.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01372 on 3480 degrees of freedom
## Multiple R-squared:  0.7827, Adjusted R-squared:  0.7815
## F-statistic: 659.7 on 19 and 3480 DF,  p-value: < 2.2e-16
```

Calculate relative importance - relimp

```
start_time <- Sys.time() # Start timer

relImportance <- calc.relimp(lmMod, type = "lmg")

Sys.time() - start_time # Display Time Difference
```

```
## Time difference of 2.926216 mins
```

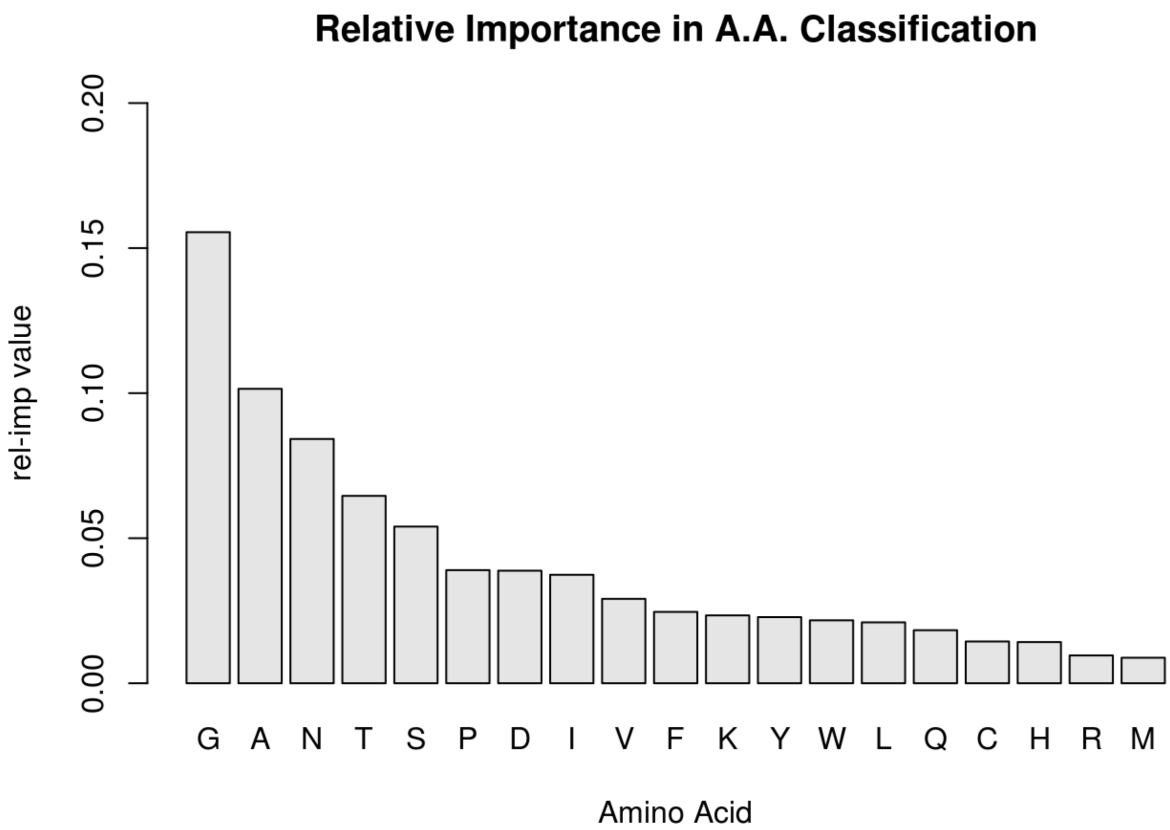
Sort & print table

```
rel_imp <- sort(round(relImportance$lmg, 4), decreasing = TRUE)
rel_imp
```

```
##      G      A      N      T      S      P      D      I      V      F
## 0.1555 0.1015 0.0842 0.0646 0.0540 0.0390 0.0388 0.0374 0.0291 0.0246
##      K      Y      W      L      Q      C      H      R      M
## 0.0234 0.0228 0.0217 0.0210 0.0183 0.0144 0.0142 0.0096 0.0088
```

Barplot of Relative Importance

```
barplot(rel_imp,
        main = "Relative Importance in A.A. Classification",
        col = "grey90",
        border = TRUE,
        ylab = "rel-imp value",
        xlab = "Amino Acid",
        ylim = c(0, 0.2))
```



kable Table

```
knitr::kable(rel_imp,
             format = "html",
             col.names = "Relative Importance",
             caption = "Relative Importance in A.A. Classification")
```

Relative Importance in A.A. Classification

	Relative Importance
G	0.1555
A	0.1015
N	0.0842
T	0.0646
S	0.0540
P	0.0390
D	0.0388
I	0.0374
V	0.0291
F	0.0246

	Relative Importance
K	0.0234
Y	0.0228
W	0.0217
L	0.0210
Q	0.0183
C	0.0144
H	0.0142
R	0.0096
M	0.0088

EOF

“Feature-Selection-Variable-Importance-Using-caret-rpart.rmd”

Title: Variable Importance Using `caret` & `rpart`

Import data & data handling

Train an `rpart` model and compute variable importance.

```
set.seed(1000)
registerDoMC(cores = 3) # Start multi-processor mode
start_time <- Sys.time() # Start timer

rpart_model <- train(Class ~ .,
                     data = test_harness_paa,
                     method = "rpart")

rpart_importance <- varImp(rpart_model)

Sys.time() - start_time # Display Time Difference
```

```
## Time difference of 5.409642 secs
```

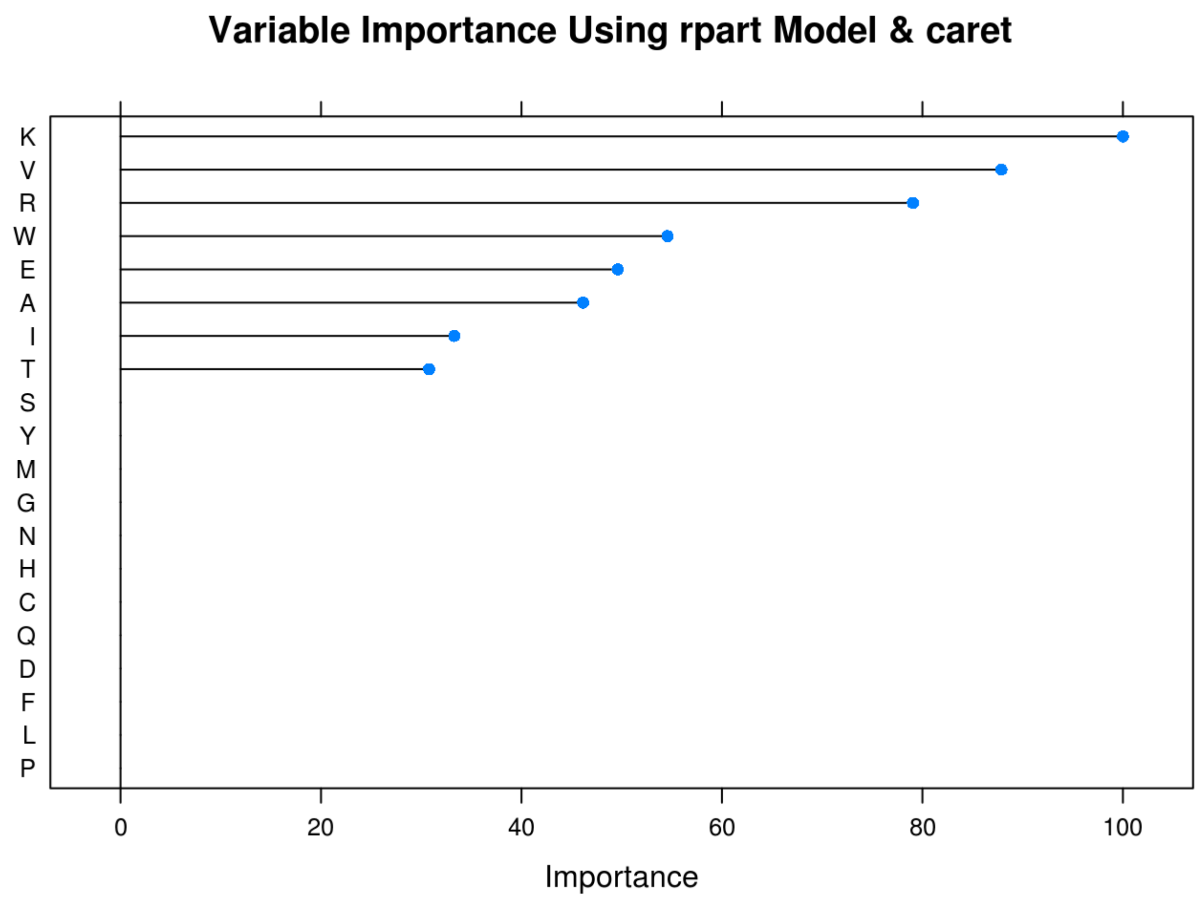
```
registerDoSEQ()

print(rpart_importance)
```

```
## rpart variable importance
##
##   Overall
## K  100.00
## V   87.87
## R   79.06
## W   54.57
## E   49.60
## A   46.15
## I   33.30
## T   30.78
## F    0.00
## H    0.00
## P    0.00
## Y    0.00
## N    0.00
## M    0.00
## Q    0.00
## G    0.00
## L    0.00
## D    0.00
## C    0.00
## S    0.00
```

```
relImportance <- calc.relimp(lmMod, type = "lmg")
```

Plot rpart Model Results

```
plot(rpart_importance,
     top = 20,
     main = 'Variable Importance Using rpart Model & caret')
```



EOF TAKEN FROM mitchell-dectrees.pdf

CART Pseudocode:

Input:
1. $(X,\ Y)$ is a labeled pair,

where:

$\langle X\rangle\ \ is\ a\ matrix\ \ \in R^n,\ \ \langle Y\rangle\ \ is\ a\ vector\ \ \in R^n,$

$Y \leftarrow\ \ label\ y_i \in \{1, 2, 3, \ldots, C\}$ such that $f(x_i, x_j, \ldots) =\ \ label\ y_i,$

Output:
1. node is a class which has properties values, childs, and next.
2. root is first/top node in the decision tree.

Declare:
1. root = CART $(X,\ Y,\ root)$
2. Entropy: H$(x) =\ -p_{(+)}log_2p_{(+)}\ -\ p_{(-)}log_2p_{(-)}$

---

1. Input: $X =$ Matrix(m, n), $y_i =$ Vector(y), root

2. initialize node as a new node instance
3. if row x_i has only single classification C, then
   - insert label C into node
   - return node
4. if $x_i = null$, then
   - insert dominant label in x_i into node
   - return node
5. best_feature is an feature with maximum Entropy in x_i
6. insert feature best_feature into node
7. for (v_i in values of best_feature)
   - For example, protein category has 7 values: {Ctrl, Ery, Hcy, Hhe, Hmb, Lgb, Mgb}
   - insert value v_i as branch of node
   - create $v_i[Rows, \quad]$ with rows that only contains value v_i
8. if ($v_i[Rows, \quad] = null$, then)
   - node branch ended by a leaf with value which is dominant label in x_i

   else
   - new_y = list of features y_i with best_feature removed
   - nextNode = next node connected by this branch
   - nextNode = CART ($v_i[Rows, \quad]$, new_y, label, nextNode)
   - return node

     End