Induction of Regression and Decision Trees

Feature selection for the first split

Martin Spott, Data Mining, Wirtschaftsinformatik, HTW Berlin
last revision: 12 May 2020

Regression Tree Induction

We want to predict the rent of a flat based on the quality of the area *(goodArea: 0=no, 1=yes)* and the size of the flat (*size: 0=small, 1=large)* based on a small data set:

```
# generate data
library(knitr)
options(scipen=999) # disable scientific notation for printing numbers
set.seed(10)

goodArea <- c(0,0,0,0,1,1,1,1)
size <- c(0,1,0,1,0,1,0,1)

rent <- floor(runif(8, 380, 420) + goodArea * runif(8, 50, 100)
              + size * runif(8, 100, 200))

# data table just for viewing
rentData <- data.frame(goodArea, size, rent)
kable(rentData)
```

goodArea	size	rent
0	0	400
0	1	518
0	0	397
0	1	591
1	0	439
1	1	630
1	0	458
1	1	597

As a baseline, we just the average rent as a prediction and check the residual sum of squares (RSS) as error measure.

```
mean(rent)
```

```
## [1] 503.75
```

```
rss_all <- sum((rent - mean(rent))^2)
rss_all
```

```
## [1] 60895.5
```

We split the data set on each of the features *goodArea* and *size* and compare the RSS. The split with the lower RSS is the better one.

```
# split on goodArea
# prediction for goodArea=0
```

```r
p_area0 <- mean(rent[goodArea == 0])
p_area0
```

```
## [1] 476.5
```

```r
# prediction for goodArea=1
p_area1 <- mean(rent[goodArea == 1])
p_area1
```

```
## [1] 531
```

```r
# RSS when splitting on goodArea
rss_area <- sum((rent[goodArea == 0] - p_area0)^2) +
              sum((rent[goodArea == 1] - p_area1)^2)
rss_area
```

```
## [1] 54955
```

```r
# split on size
# prediction for size=0
p_size0 <- mean(rent[size == 0])
p_size0
```

```
## [1] 423.5
```

```r
# prediction for size=1
p_size1 <- mean(rent[size == 1])
p_size1
```

```
## [1] 584
```

```r
# RSS when splitting on size
rss_size <- sum((rent[size == 0] - p_size0)^2) +
              sum((rent[size == 1] - p_size1)^2)
rss_size
```

```
## [1] 9375
```

Since $9375 = rss\_size < rss\_area = 54955$, split by feature *size* first. We can also see that both features improve on the baseline RSS of 60895.5.

Decision Tree Induction

We add a class *want_it* to the data frame that describes if we want to buy such a flat (1) or not (0).

```
rentData$want_it <- ifelse(rent > 450, 1, 0) # 1: I want it, 0: not interested
# view data
kable(rentData[,c(1,2,4)])
```

goodArea	size	want_it
0	0	0
0	1	1
0	0	0
0	1	1
1	0	0
1	1	1
1	0	1
1	1	1

As for a regression tree, we split the data set on each of the features *goodArea* and *size*, but this time compare the *Gini impurity* of the two splits. The split with the lower impurity is better.

Let us first compute the Gini impurity of *want_it* in the entire data set.

```
#   P(want_it = 0) = 3/8, P(want_it = 1) = 5/8
gini_all <- 3/8 * (1 - 3/8) + 5/8 * (1 - 5/8)
gini_all
```

```
## [1] 0.46875
```

Now compute the Gini impurities for the two splits.

```
# split on goodArea

# Gini impurity for goodArea == 0:
#   P(want_it = 0 | goodArea = 0) = P(want_it = 1  | goodArea = 0) = 1/2
gini_gA0 <- 1/2 * (1 - 1/2) + 1/2 * (1 - 1/2)
gini_gA0
```

```
## [1] 0.5
```

```
# Gini impurity for goodArea == 1:
#   P(want_it = 0 | goodArea = 1) = 1/4, P(want_it = 1 | goodArea = 1) = 3/4
gini_gA1 <- 1/4 * (1 - 1/4) + 3/4 * (1 - 3/4)
gini_gA1
```

```
## [1] 0.375
```

```
# compute Gini impurity for split on goodArea
# weighted average, both goodArea = 1 and goodArea = 0 have 4 data points out of 8
gini_gA <- 4/8 * gini_gA0 + 4/8 * gini_gA1
gini_gA
```

```
## [1] 0.4375
```

```
# split on size

# Gini impurity for size == 0:
#   P(want_it = 0 | size = 0) 3/4, P(want_it = 1 | size = 0) = 1/4
```

```r
gini_s0 <- 3/4 * (1 - 3/4) + 1/4 * (1 - 1/4)
gini_s0
```

```
## [1] 0.375
```

```r
# Gini impurity for size == 1:
#   P(want_it = 0 | size = 1) = 0, P(want_it = 1 | size = 1) = 1
gini_s1 <- 0 * (1 - 0) + 1 * (1 - 1)
gini_s1
```

```
## [1] 0
```

```r
# compute Gini impurity for split on size
# weighted average, both size = 1 and size = 0 have 4 data points out of 8
gini_s <- 4/8 * gini_s0 + 4/8 * gini_s1
gini_s
```

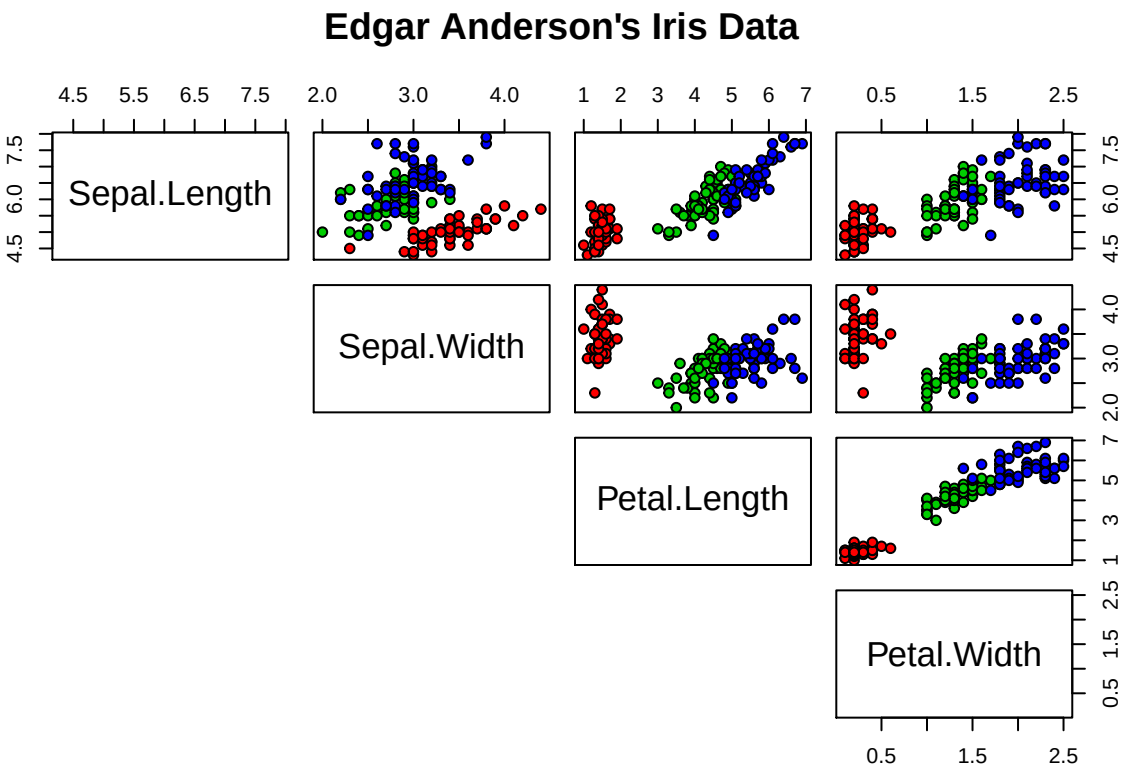
```
## [1] 0.1875
```

Gini impurity for split on *size* is smaller: $0.1875 = gini\_s < gini\_gA = 0.4375$. Therefore we split on *size* first. As for the regression tree, both splits would give a better separation of the two classes than we find in the overall data set, since $gini\_all = 0.46875$ is larger than the other two.

Classification Tree for the Iris Data Set

```
# Decision Trees
data("iris")

# plot iris data with class as colour
pairs(iris[1:4], main = "Edgar Anderson's Iris Data", pch = 21,
      bg = c("red", "green3", "blue")[unclass(iris$Species)], lower.panel=NULL)
```




```
# summary information of iris data set
summary(iris)
```

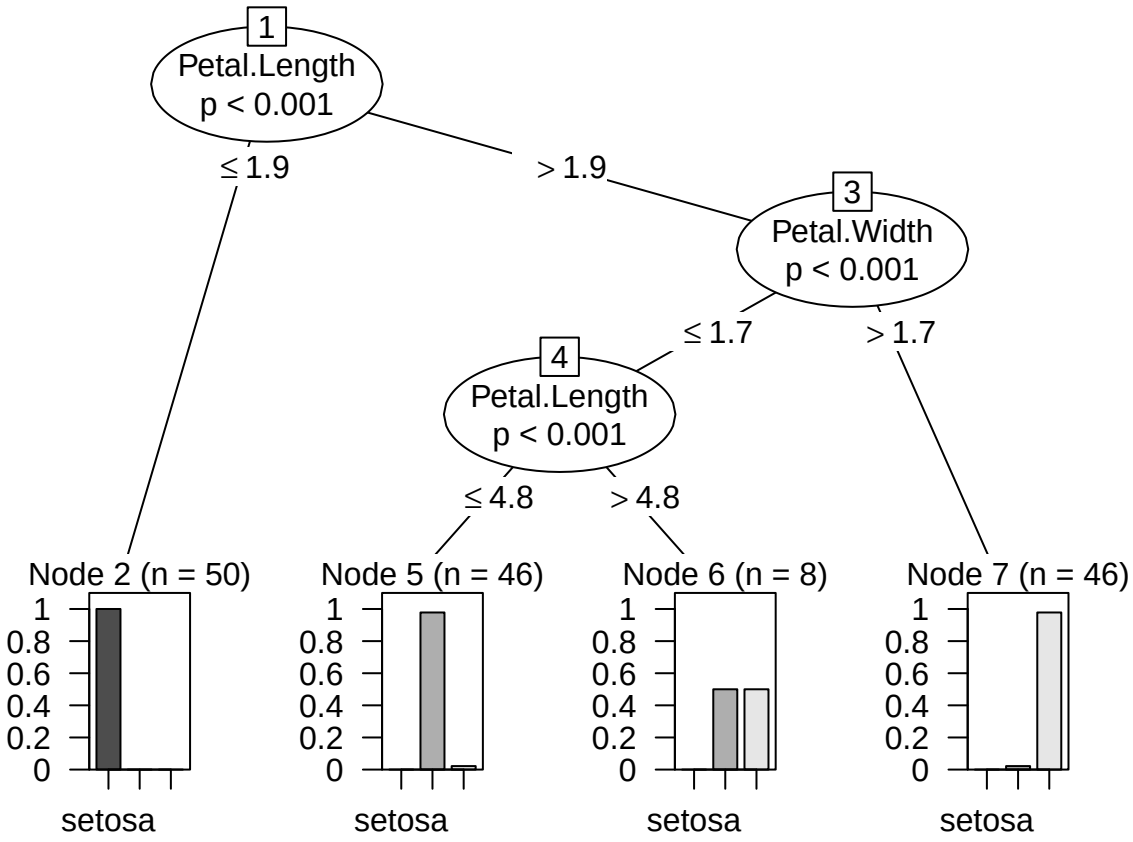
```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
# induce some trees
# conditional inference tree (http://www.rdatamining.com/examples/decision-tree)
library(party)
```

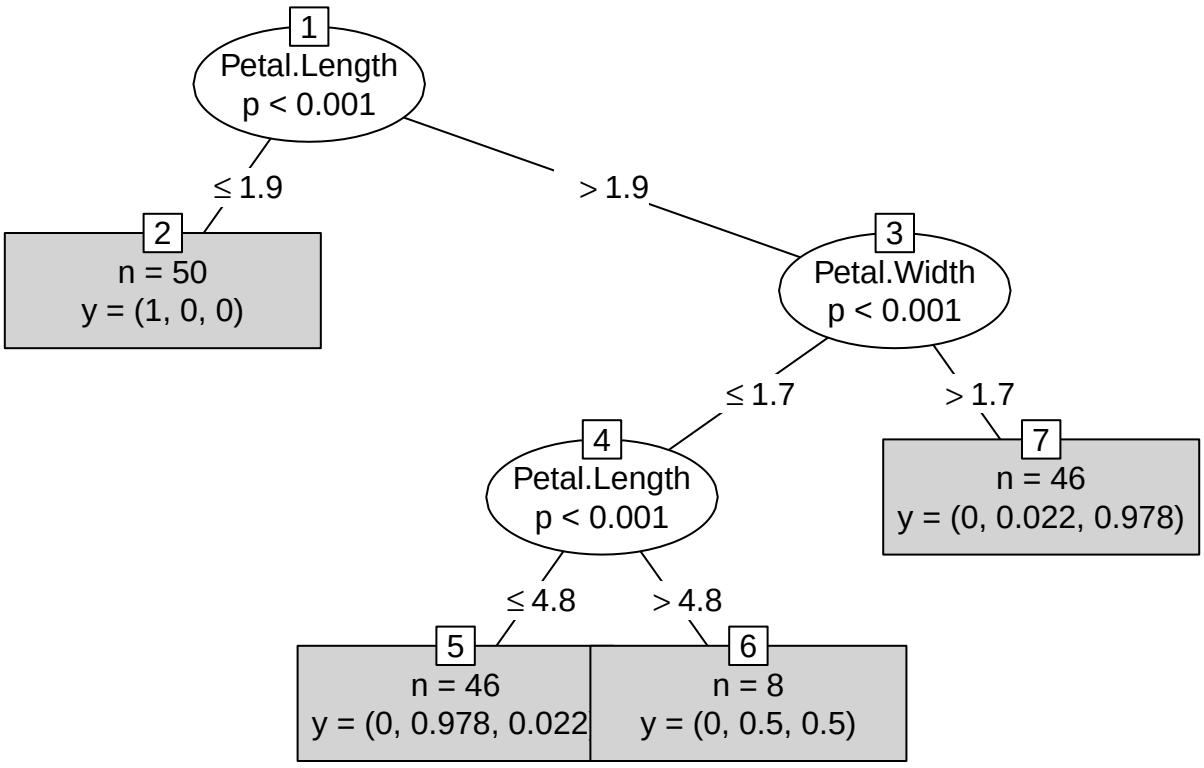
```
iris_ctree <- ctree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data=iris)
print(iris_ctree)
```

```
##
##   Conditional inference tree with 4 terminal nodes
##
## Response:  Species
## Inputs:  Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
## Number of observations:  150
##
## 1) Petal.Length <= 1.9; criterion = 1, statistic = 140.264
##   2)*  weights = 50
## 1) Petal.Length > 1.9
##   3) Petal.Width <= 1.7; criterion = 1, statistic = 67.894
##     4) Petal.Length <= 4.8; criterion = 0.999, statistic = 13.865
##       5)*  weights = 46
##     4) Petal.Length > 4.8
##       6)*  weights = 8
##   3) Petal.Width > 1.7
##     7)*  weights = 46
```

```
plot(iris_ctree)
```

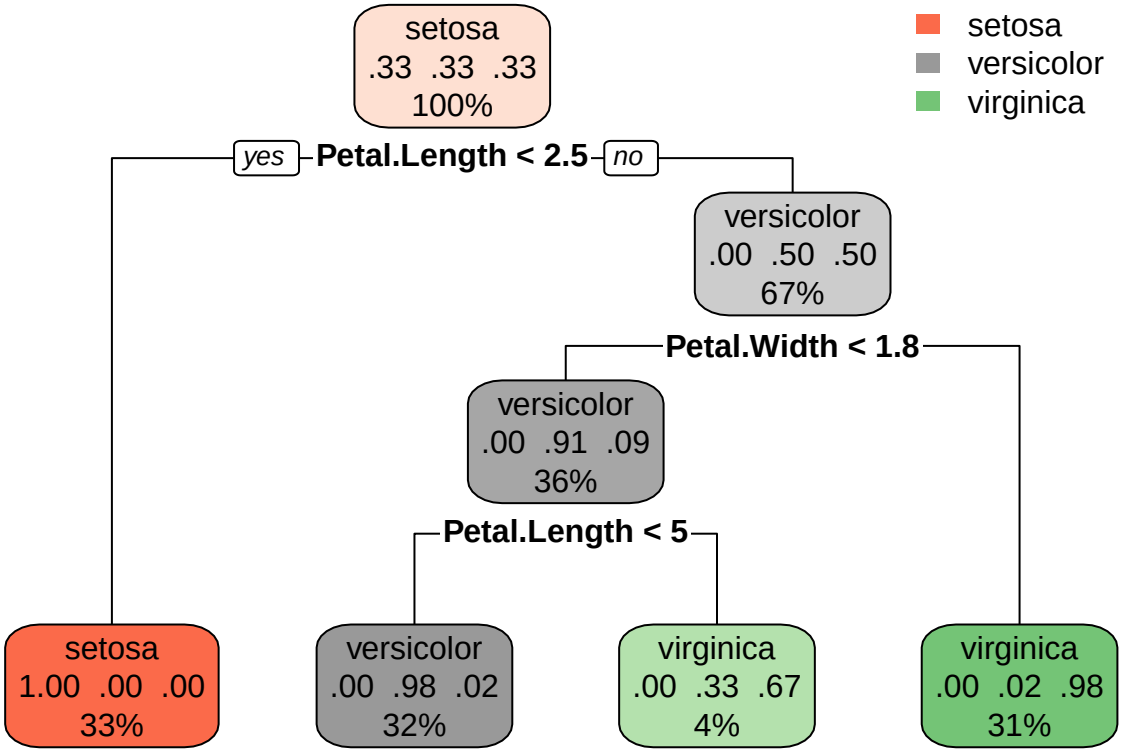



```
plot(iris_ctree, type="simple")
```




```
# rpart
library("rpart")
library("rpart.plot")

tree <- rpart(Species ~ ., data = iris, method = "class",
              control = rpart.control(maxdepth = 4, minsplit = 5))
rpart.plot(tree)
```

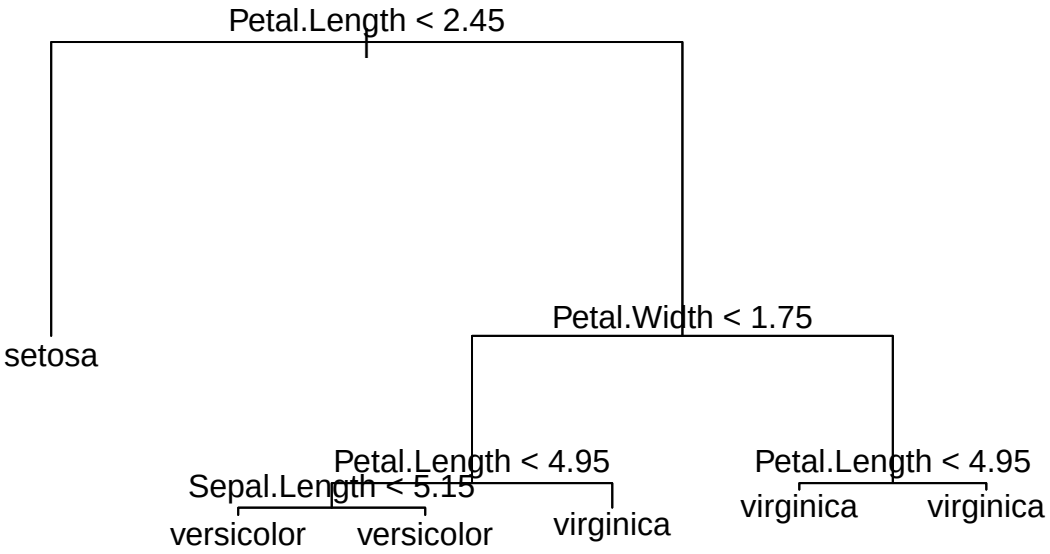



```
# tree
library(tree)

tree_iris <- tree(Species ~ ., data = iris)
summary(tree_iris)
```

```
##
## Classification tree:
## tree(formula = Species ~ ., data = iris)
## Variables actually used in tree construction:
## [1] "Petal.Length" "Petal.Width"  "Sepal.Length"
## Number of terminal nodes:  6
## Residual mean deviance:  0.1253 = 18.05 / 144
## Misclassification error rate: 0.02667 = 4 / 150
```

```
plot(tree_iris)
text(tree_iris, pretty=0)
```



Gini impurity

Gini impurity is one of various measures used to evaluate, how good a feature is for the next split and how to split the range of values. Rather than just measuring the accuracy of the prediction by adding the split, Gini impurity measures how well a split separates the different classes. Predicting all classes with the same probability is the worst split (*most impure split*), the winning class having a probability of 1 and all others 0 is the best split (*purest split*).

```
# quadratic entropy (Gini impurity)
# three classes with equal probability
3 * 1/3 * (1 - 1/3)
```

```
## [1] 0.6666667
```

```
# three classes with "one winner"
.8 * (1 - .8) + .1 * (1 - .1) + .1 * (1 - .1)
```

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

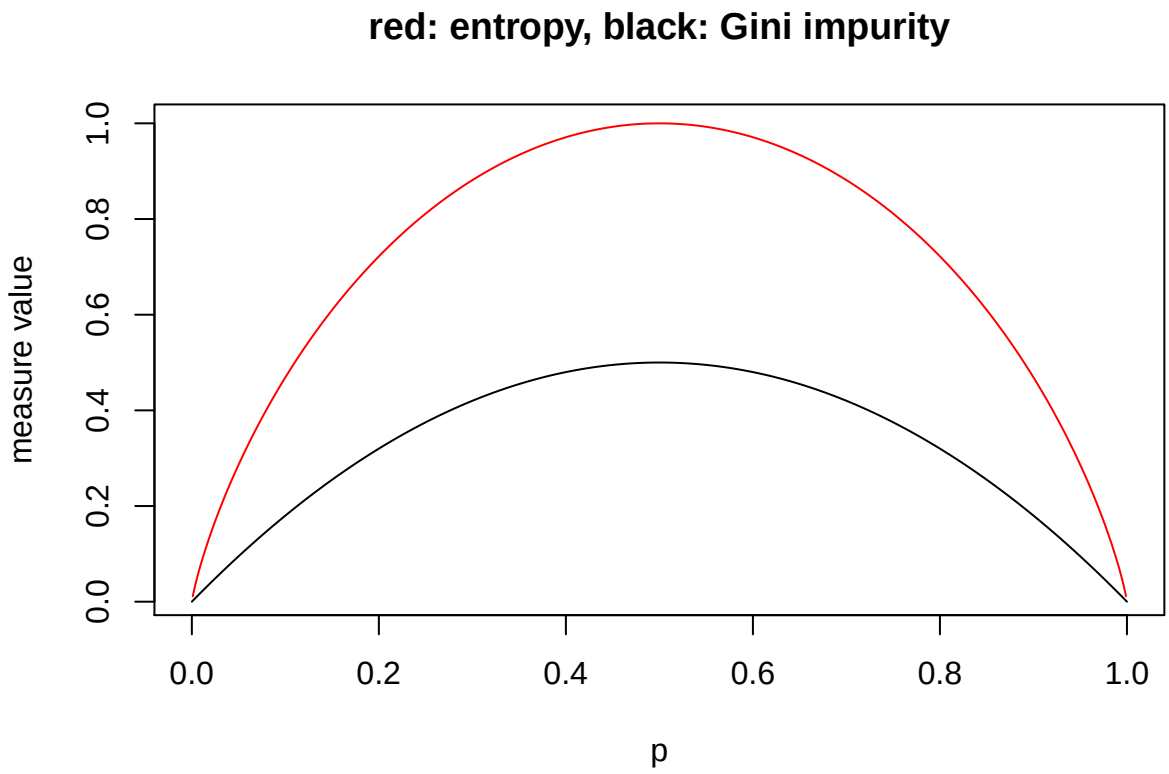
```
# three classes with "one even stronger winner"
.9 * (1 - .9) + .05 * (1 - .05) + .05 * (1 - .05)
```

```
## [1] 0.185
```

```
# three classes with "one winner taking all"
1 * (1 - 1) + 0 * (1 - 0) + 0 * (1 - 0)
```

```
## [1] 0
```

```
# Compare Gini impurity and entropy values for binary classification
x <- seq(0,1,0.001)
plot(x, -x * log2(x) - (1-x) * log2(1-x), type="l", col="red",
     main="red: entropy, black: Gini impurity", xlab="p", ylab="measure value")
lines(x, 2 * x * (1 - x))
```



Entropy is another measure giving similar results.