Decision trees in R – `rpart()` implemetation

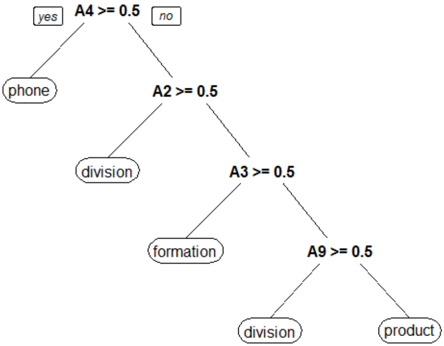
- library **rpart** (but there are also other libraries, e.g. **tree** and **party**)

```
Model=rpart(formula, data=, method=, control=)
```

- `?rpart`
- `formula` in the "traditional" format
  `TargetClass ~ Feature1 + Feature2 + ...`
- `data` specifies the input data frame
- `method` is `"class"` for classification task
- `control` other optional parameters

Visualisation using `rpart.plot()`

Visualisation of the model with library `rpart.plot`



Decision Trees – parameters

hypothesis parameters – parameters of the prediction function

- output of the learning algorithm, define the structure of the decision tree

learning parameters – parameters of the learning process

- "configuration" of the learning algorithm

Decision trees learning parameters

2 phases of decision tree learning:

- growing
- pruning

Learning parameters are used to control these two phases:

- when to stop growing
- how much to prune the tree

. . . to avoid overfitting and improve performance

Learning parameters in `rpart`

`rpart.control`

minsplit

- the minimum number of observations that must exist in a node in order for a split to be attempted

cp

- complexity parameter, influences the depth of the tree

... and others, see `?rpart.control`

Hint: Try to set different `cp` and `minsplit` values in your model learning, then observe and compare different resulting trees

“Complexity Parameter”

cp parameter is used to control the depth of the growing tree: Any split that does not decrease the **relative training error** by a factor of cp is not attempted

⇒ That means, the learning algorithm measures for each split how it improves the tree relative error and if the improvement is too small, the split will not be performed.

Relative error is the error relative to the misclassification error (without any splitting relative error is 100%)

cp parameter

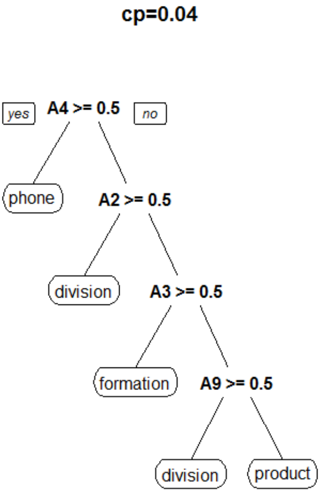
```
> M <- rpart(SENSE ~ A1+A2+A3+A4+A5+A6+A7+A8+A9+A10+A11, data=train,
             method="class", minsplit=5, cp=0.001)
> M$cptable
           CP nsplit rel error    xerror       xstd
1 0.093053735      0 1.0000000 1.0000000 0.01844043
2 0.057667104      1 0.9069463 0.9069463 0.01830335
3 0.048492792      2 0.8492792 0.8591088 0.01817412
4 0.040629096      3 0.8007864 0.8106160 0.01800131
5 0.009174312      4 0.7601573 0.7601573 0.01777550
6 0.003931848      9 0.7064220 0.7070773 0.01748535
7 0.001000000     10 0.7024902 0.7044561 0.01746957
```

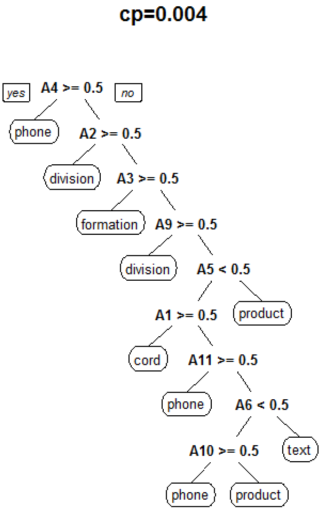
rel error relative error on training data

xerror relative error in x-fold **cross-validation**

xstd standard deviation of `xerror` on x validation folds

Models built with different cp value





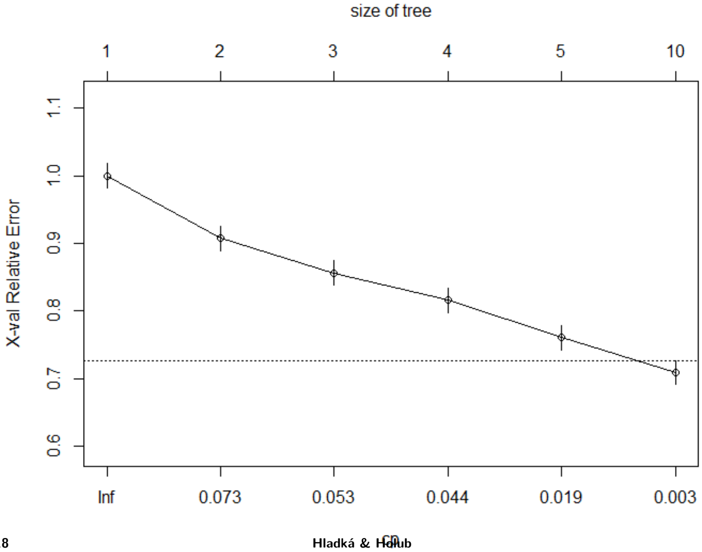
Useful functions

`plotcp(model)`	visualisation of the cross-validation error depending on cp value
`prune(model, cp=)`	prune the model based on cp value

```
> M5$cptable[which.min(M5$cptable[,"xerror"]),"CP"]
[1] 0.001
```

plotcp

- visualisation of the cross-validation error depending on cp value
- the horizontal line shows the minimal `xerror` + its standard deviation



How to choose the optimal cp value?

demo code cp-and-pruning.Forbes.R on course page

```
> m = rpart(profits ~ category + sales + assets + marketvalue,
            data=F[data.train, 1:8], cp=0.001)
> m$cptable
            CP nsplit rel error    xerror       xstd
1  0.543259557      0 1.0000000 1.0482897 0.03178559
2  0.027162978      1 0.4567404 0.4607646 0.02673551
3  0.007042254      3 0.4024145 0.4446680 0.02640028
4  0.006036217      6 0.3762575 0.4507042 0.02652763
5  0.005030181      8 0.3641851 0.4567404 0.02665301
6  0.004024145     15 0.3279678 0.4768612 0.02705703
7  0.003018109     19 0.3118712 0.4688129 0.02689795
8  0.002012072     21 0.3058350 0.4869215 0.02725122
9  0.001006036     23 0.3018109 0.5171026 0.02780383
10 0.001000000     25 0.2997988 0.5412475 0.02821490
```

How to choose the optimal cp value?