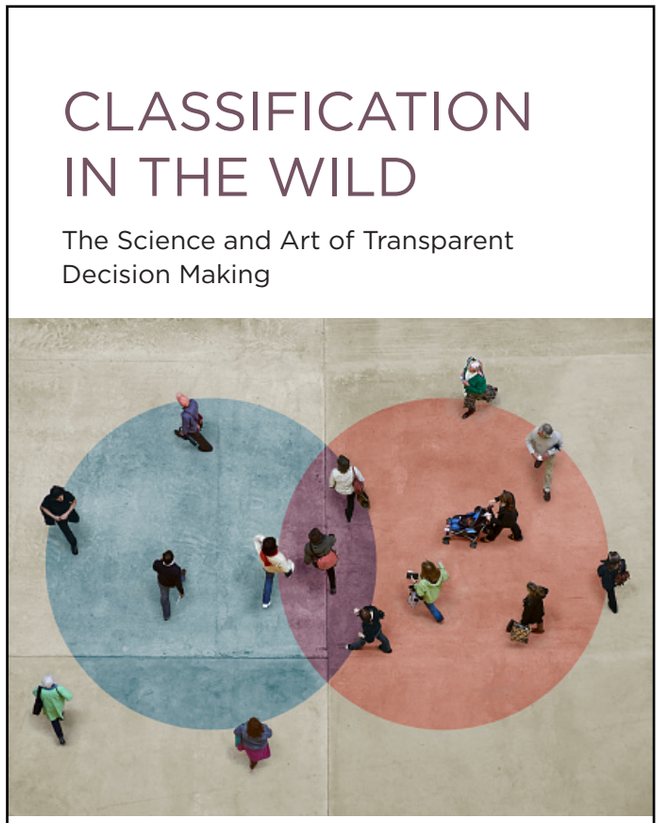
Fast and Frugal Classifiers

Özgür Şimşek
Department of Computer Science
University of Bath



Today

- Classification in the wild: Two examples
- Models of fast-and-frugal classification
- Building fast-and-frugal classifiers
- Building fast-and-frugal classifiers with software

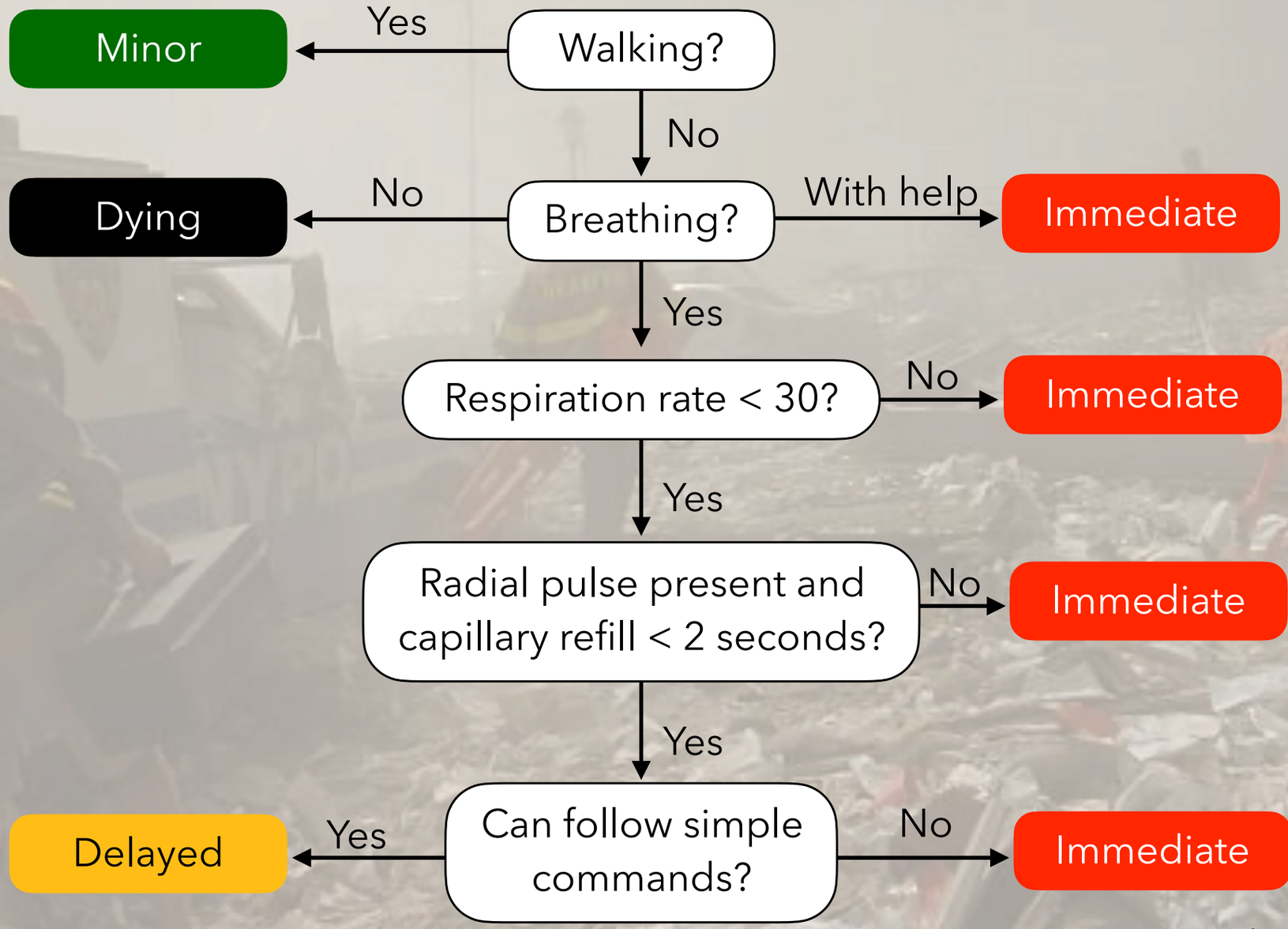
ffcr:
fast and frugal
classification in R

Classification in the wild:

Two examples

How to triage at a disaster site?

Simple Triage and Rapid Treatment (START)



Cook (Critical Care, 2001)
Super (START, 1984)

Who will be the next president
of the United States?

Keys to the White House

1. After the midterm elections, the incumbent party holds more seats in the House than it did after the previous midterm elections.
2. There is no serious contest for the incumbent-party nomination.
3. The incumbent-party candidate is the sitting president.
4. There is no significant third-party or independent campaign.
5. The economy is not in recession during the election campaign.
6. Real annual per capita economic growth during the term equals or exceeds mean growth during the two previous terms.
7. Incumbent administration effects major changes in national policy.
8. There is no sustained social unrest during the term.
9. Incumbent administration is untainted by major scandal.
10. Incumbent administration suffers no major failure in foreign or military affairs.
11. Incumbent administration achieves a major success in foreign or military affairs.
12. Incumbent-party candidate is charismatic or a national hero.
13. Challenging-party candidate is not charismatic or a national hero.

If 8 or more keys are “true”, predict win for the incumbent.

Allan Lichtman, 1981

“The wild”

Fundamental uncertainty –
Real-world situations where the future is not knowable and uncertainty cannot be meaningfully reduced to probability.

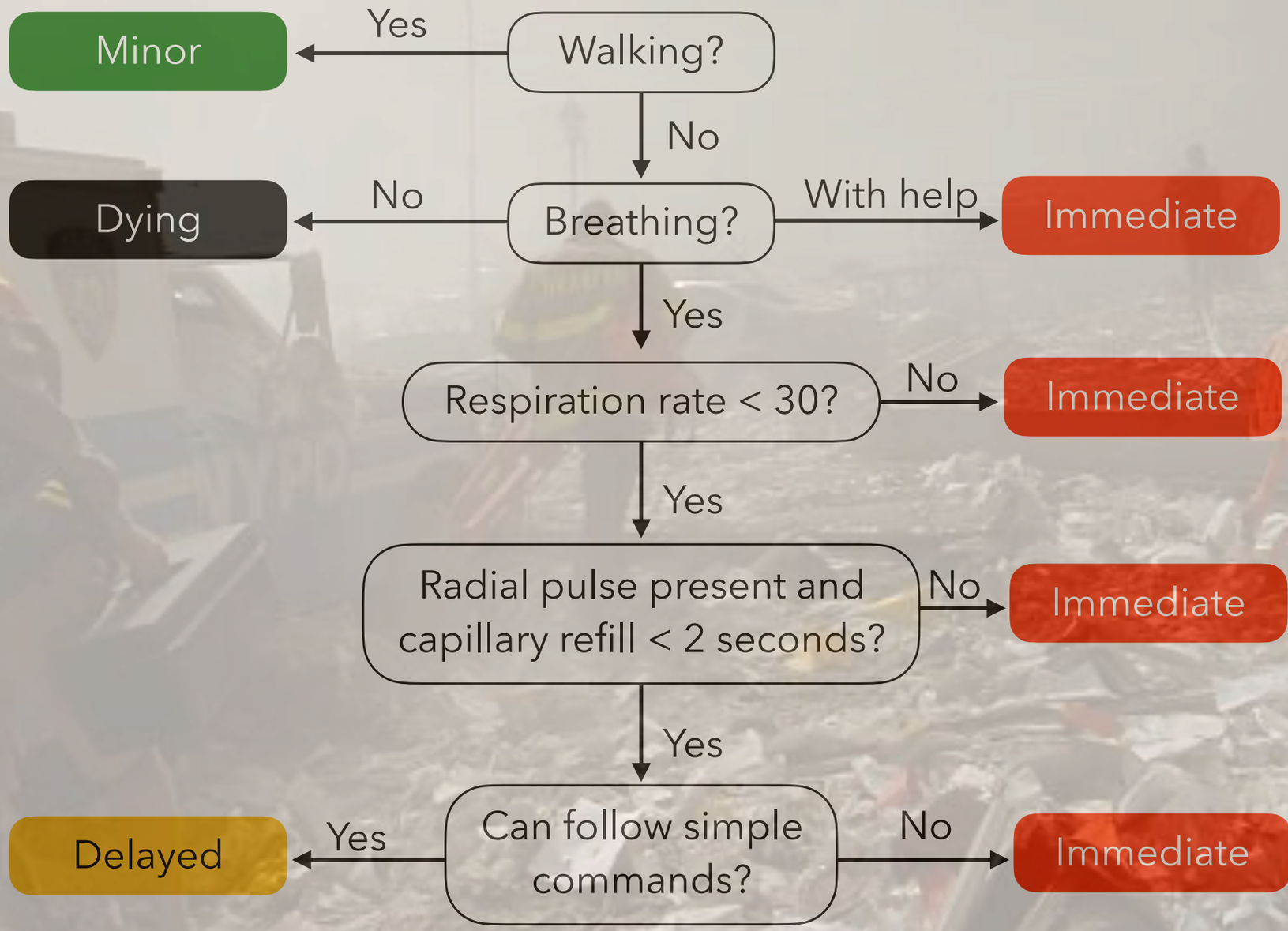
Fast-and-frugal classification

- Precise, formal models of classification
- Both descriptive and prescriptive
- Fast, accurate, transparent decisions
- Based on core competencies of humans

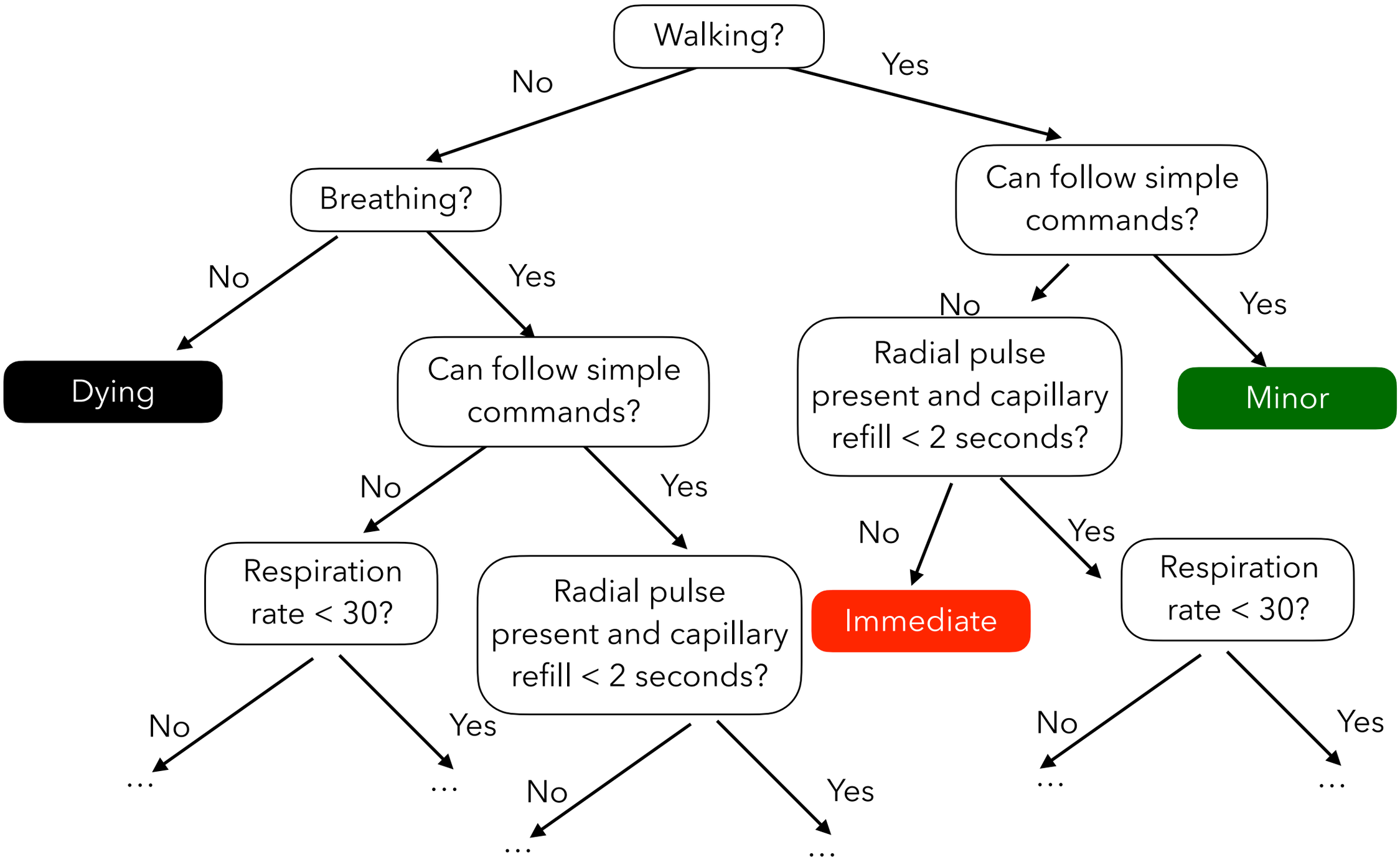
Ordering

Counting

START is a fast-and-frugal tree.



This is NOT a fast-and-frugal tree



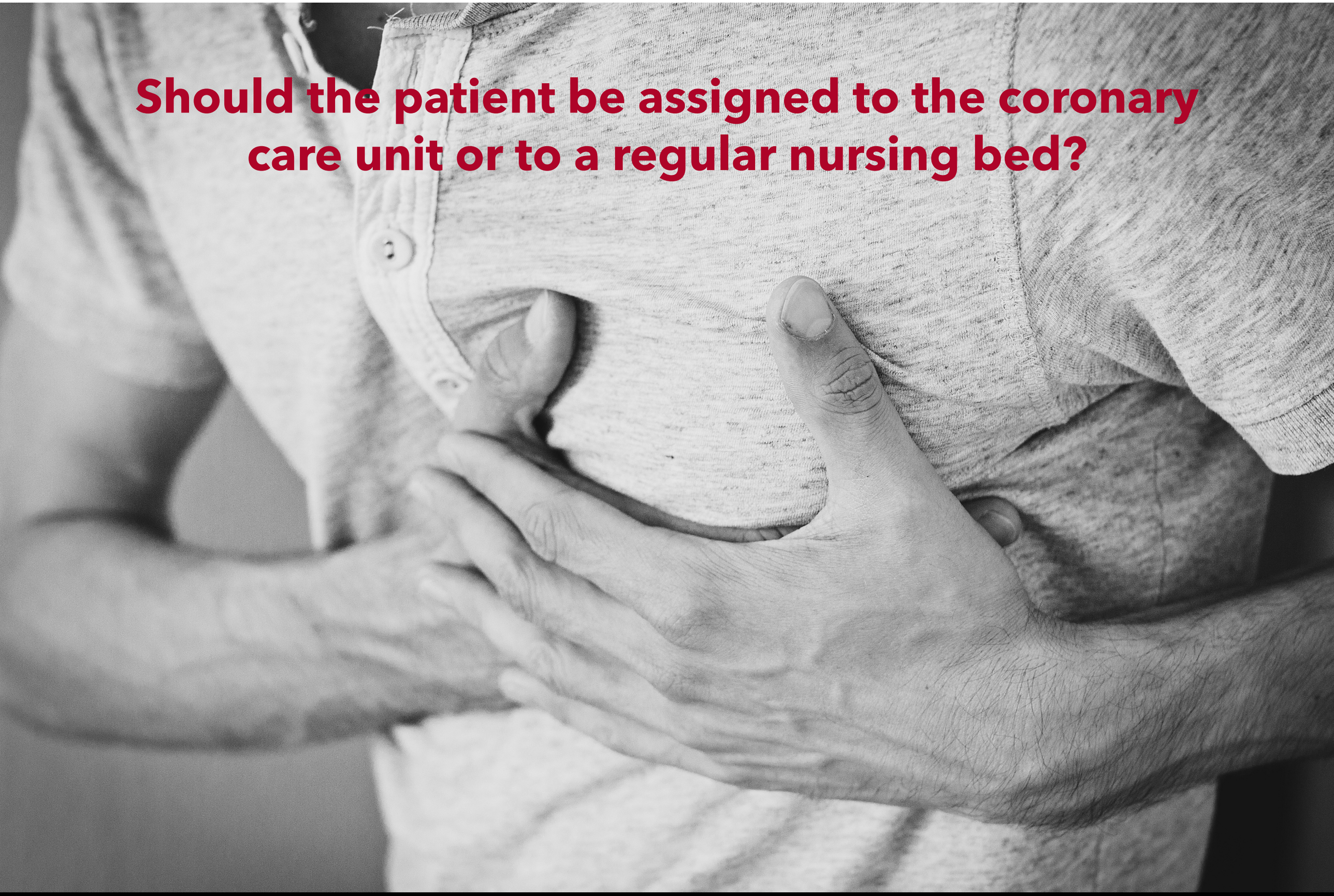
Keys to the White House is a Tallying Rule.

1. After the midterm elections, the incumbent party holds more seats in the House than it did after the previous midterm elections.
2. There is no serious contest for the incumbent-party nomination.
3. The incumbent-party candidate is the sitting president.
4. There is no significant third-party or independent campaign.
5. The economy is not in recession during the election campaign.
6. Real annual per capita economic growth during the term equals or exceeds mean growth during the two previous terms.
7. Incumbent administration effects major changes in national policy.
8. There is no sustained social unrest during the term.
9. Incumbent administration is untainted by major scandal.
10. Incumbent administration suffers no major failure in foreign or military affairs.
11. Incumbent administration achieves a major success in foreign or military affairs.
12. Incumbent-party candidate is charismatic or a national hero.
13. Challenging-party candidate is not charismatic or a national hero.

If 8 or more keys are “true”, predict win for the incumbent.

Allan Lichtman, 1981

Models of fast-and-frugal classification



Should the patient be assigned to the coronary care unit or to a regular nursing bed?

Should the patient be assigned to the coronary care unit or to a regular nursing bed?

Chest pain. Patient has reported pain in chest or left arm.

Chief complaint. Patient reports pressure, pain, or discomfort in chest as the most important symptom.

History. The patient has a known history of heart attack.

Nitroglycerin. The patient has a known history of nitroglycerin use for chest pain.

ST-change. Electrocardiogram shows ST segment (section between the end of the S wave and the beginning of the T wave) with elevation or depression of 1 mm or more.

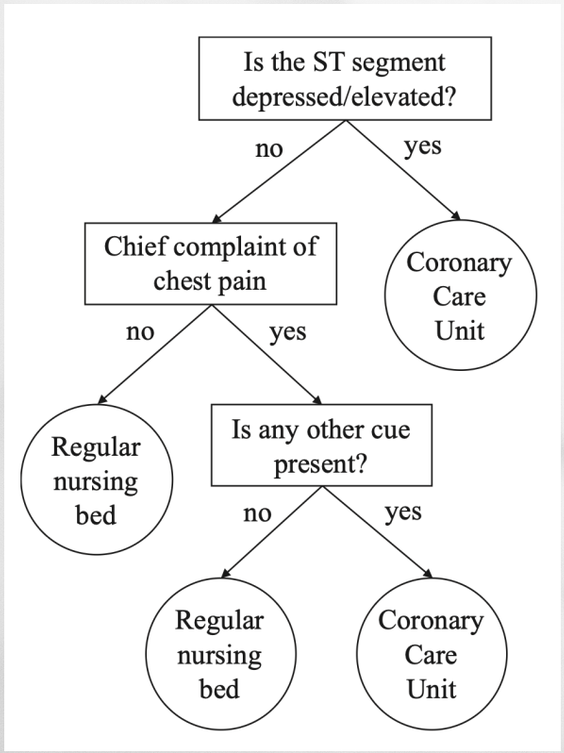
ST-barring. Electrocardiogram shows ST segment abnormally “straightened” or “barred” (but not depressed by more than 0.5 mm).

T peak/inversion. Electrocardiogram shows T waves with peaking or inversion of at least 1 mm.

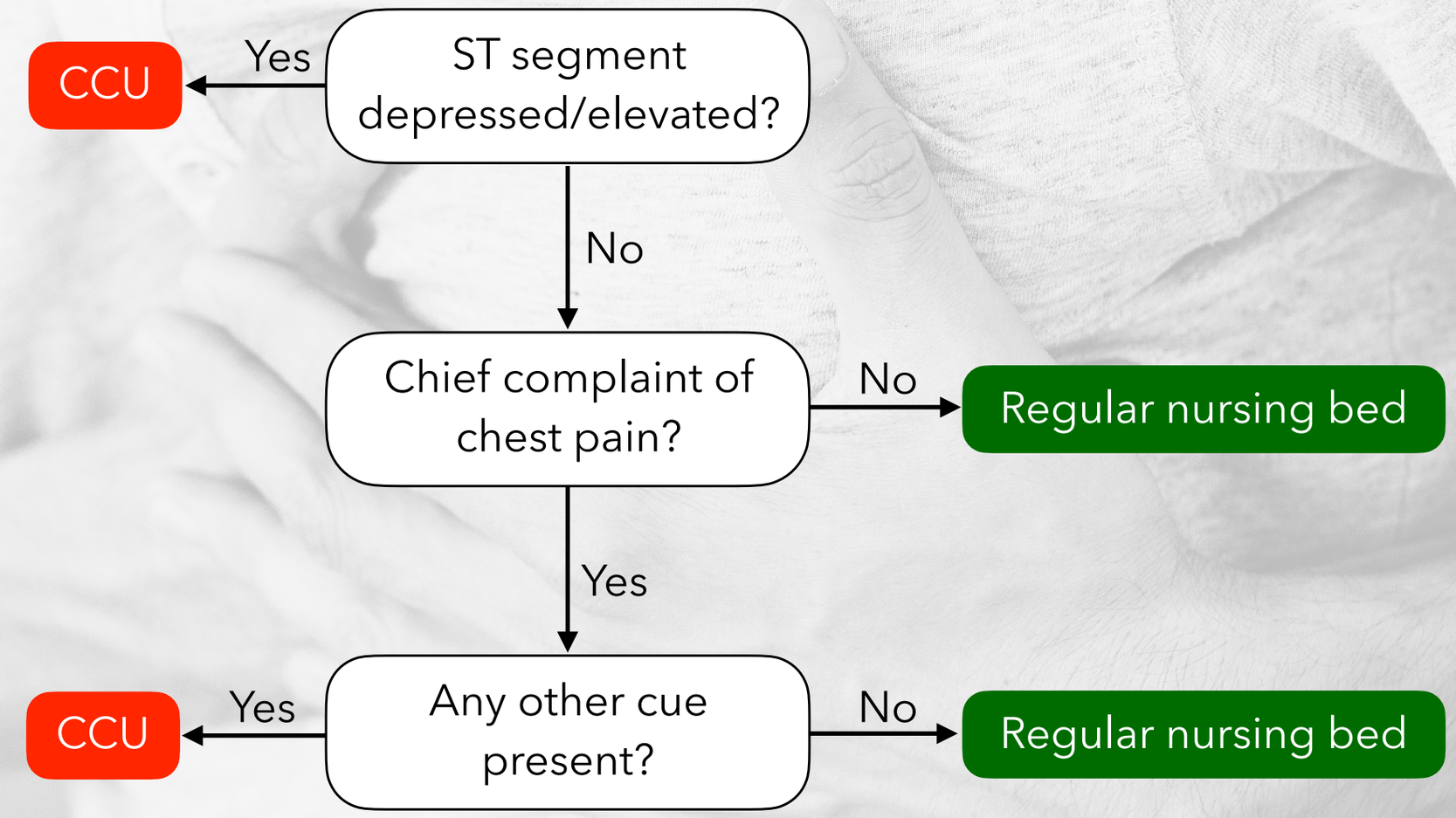
Cue values:
Absent (0), Present (1)

Posen et al. (1984) New England Journal of Medicine

Should the patient be assigned to the coronary care unit or to a regular nursing bed?



Fast-and-frugal-tree for coronary care



Green & Mehr (1997)

Should the patient be assigned to the coronary care unit or to a regular nursing bed?

Tallying for coronary care

Target class	Coronary Care Unit
Threshold	3
Reasons	Chest pain is present. Chief complaint is present. History is present. Nitroglycerin is present. ST- change is present. ST- barring is present. T peak/inversion is present.

Green & Mehr (1997)

Building fast-and-frugal classifiers

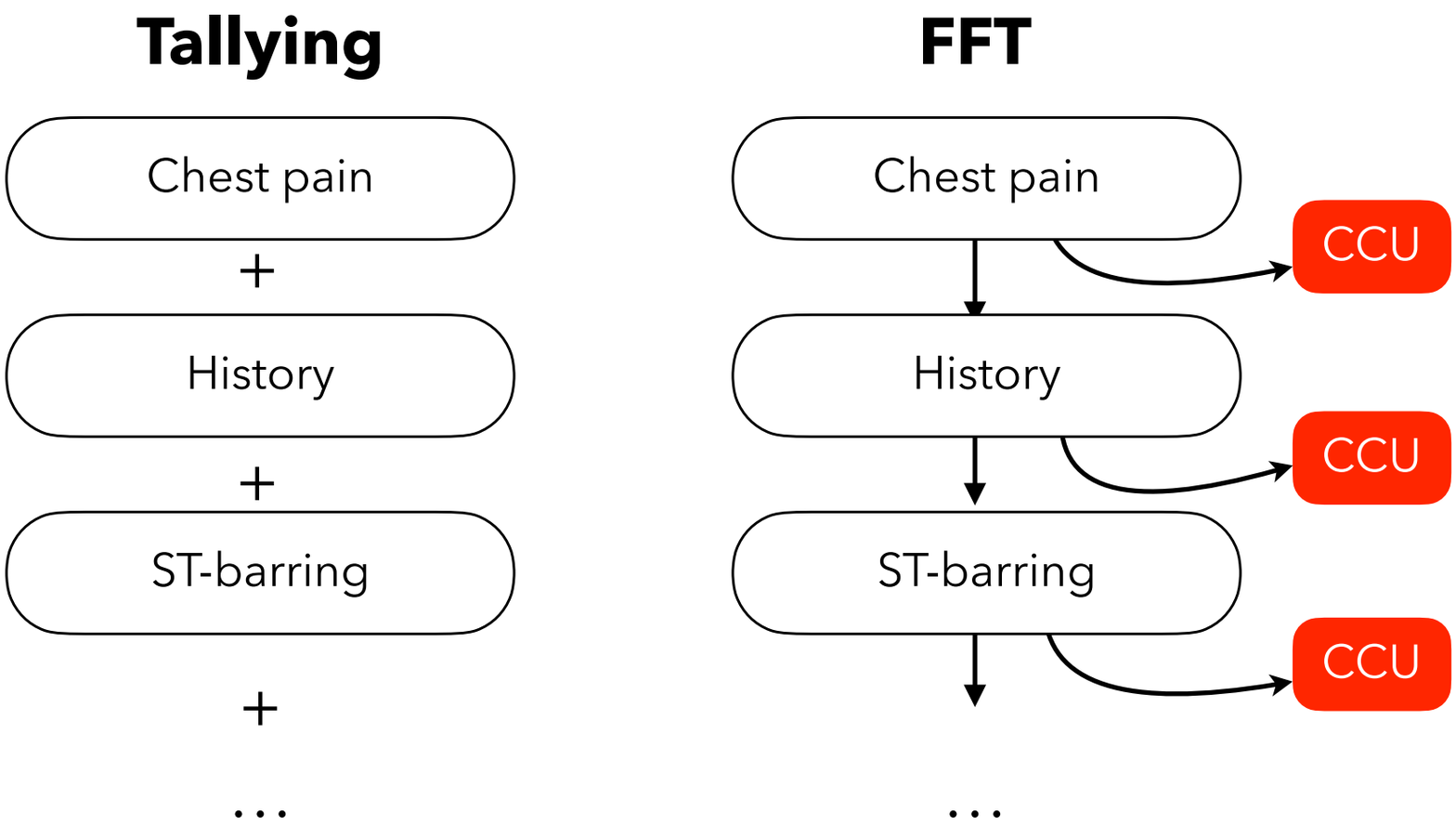
Building block of fast-and-frugal classifiers: Reasons

Cues
Chest pain
Chief complaint
History
Nitroglycerin
ST-change
ST-barring
T peak/inversion

→

Reasons
Presence of chest pain
is a reason for allocating to the CCU.

Integrating the reasons



Cues

Chest pain. Patient has reported pain in chest or left arm.

Chief complaint. Patient reports pressure, pain, or discomfort in chest as the most important symptom.

History. The patient has a known history of heart attack.

Nitroglycerin. The patient has a known history of nitroglycerin use for chest pain.

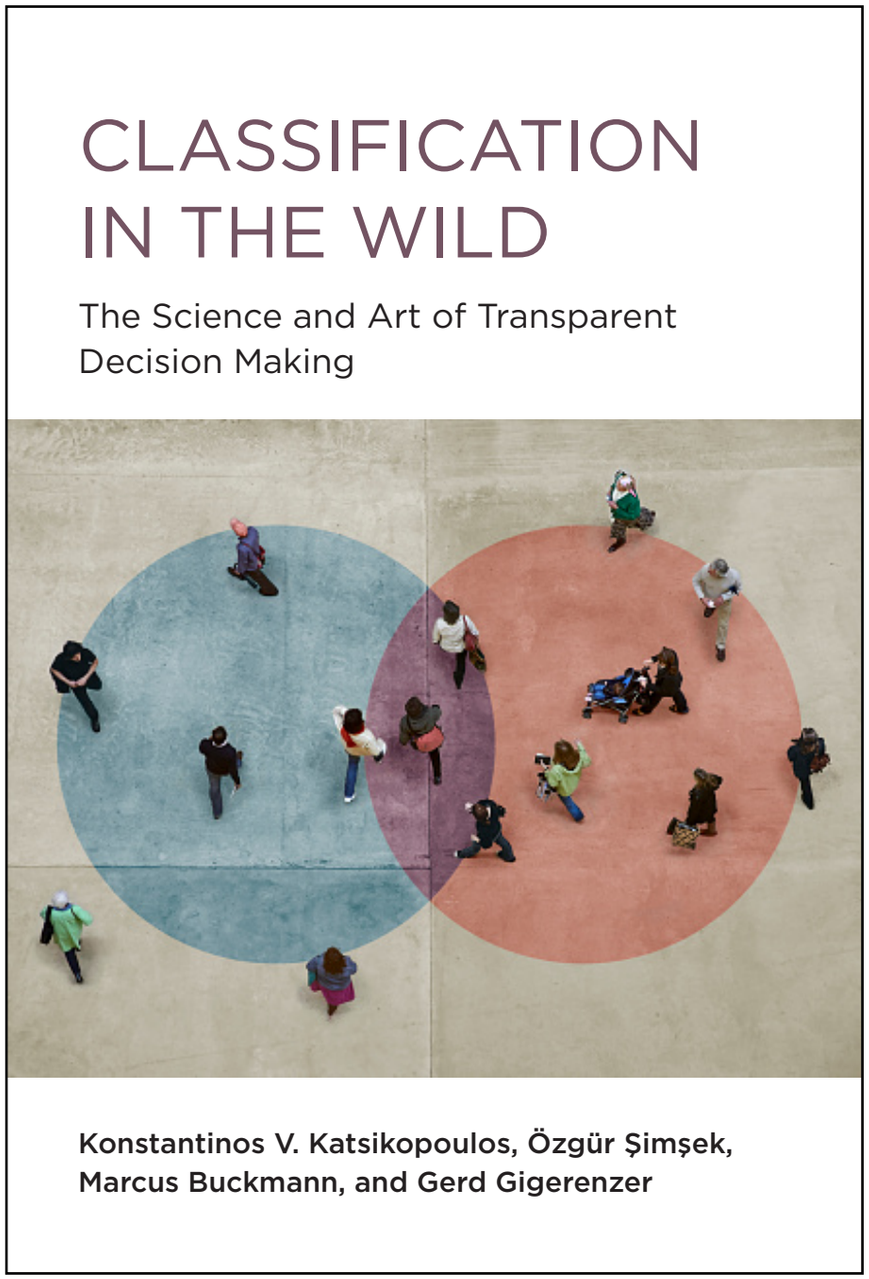
ST-change. Electrocardiogram shows ST segment (section between the end of the S wave and the beginning of the T wave) with elevation or depression of 1 mm or more.

ST-barring. Electrocardiogram shows ST segment abnormally "straightened" or "barred" (but not depressed by more than 0.5 mm).

T peak/inversion. Electrocardiogram shows T waves with peaking or inversion of at least 1 mm.

Cue values: Absent (0), Present (1)

Building fast-and-frugal classifiers with software



+

ffcr:
fast and frugal
classification in R

An R package for constructing fast-and-frugal trees and tallying models from data

https://github.com/marcusbuckmann/ffcr

Product Solutions Resources Open Source Enterprise Pricing

Search or jump to... / Sign in Sign up

marcusbuckmann / ffcr Public

Notifications Fork 0 Star 2

Code Issues Pull requests Actions Projects Security Insights

master 1 Branch 0 Tags Go to file Code

marcusbuckmann Merge branch 'master' of https://github.com/marcusbuckmann/ffcr 2f8f849 · 2 years ago 23 Commits

R	fixed bug, where splitting failed because a split create a ...	2 years ago
data	first commit	4 years ago
inst/liver	fixed tallying intercet bug	4 years ago
man	- documentation updated	3 years ago
src-i386	forgot files	3 years ago
src-x64	forgot files	3 years ago
src	- documentation updated	3 years ago
tests	utf-8 to ascii	4 years ago
vignettes	- new RCPP version	3 years ago
.Rbuildignore	fixed tallying intercet bug	4 years ago
.Rmd	fixed tallying intercet bug	4 years ago
.gitignore	first commit	4 years ago
DESCRIPTION	- documentation updated	3 years ago
NAMESPACE	major improvements in documentation	4 years ago
README.Rmd	fixed tallying intercet bug	4 years ago
README.md	Update README.md	2 years ago
ffcr.Rproj	fixed tallying intercet bug	4 years ago
ffcr_1.0.zip	updated binary - documentation	3 years ago

About

R package to train two families of transparent classification models: fast-and-frugal trees and tallying models

Readme
Activity
2 stars
3 watching
0 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

R 93.4% C++ 5.7% TeX 0.9%

Installing ffcr

```
install.packages("devtools")
devtools::install_github(“marcusbuckmann/ffcr”)
```

What you can do with ffcr

Construct fast-and-frugal trees and tallying models from data using different learning algorithms

Visualise the models learned

Obtain metrics of how well they perform

Make new predictions using them

Constructing a fast and frugal tree

```
my_fft <- fftree(data = liver)
```

```
Fast-and-frugal Tree object
Trained with : “recursive” method.

Call:
fftree(data = data, formula = formula)

Formula:
diagnosis ~ age + sex + totalBilirubin + directBilirubin + alkaline +
    alamine + aspartate + proteins + albumin + albuminGlobulin

Tree:
Reason / Prediction / (Proportion of class 'Liver disease') / (Number of
objects classified)

  totalBilirubin > 1.65: Liver disease (0.91) (213)
    alkaline > 211.5: Liver disease (0.76) (132)
      age <= 25.5: No liver disease (0.30) (27)
        alamine > 90.5: Liver disease (1.00) (5)
          proteins <= 4.45: Liver disease (0.88) (8)
            albuminGlobulin > 1.825: No liver disease (0.00) (3)
              albuminGlobulin <= 1.825: Liver disease (0.53) (191)

...
```

Constructing a fast and frugal tree

```
my_fft <- fftree(data = liver)
```

```
...

Fitted values:
    Observed          Predicted         N
    Liver disease     Liver disease     406
    No liver disease  Liver disease     143
    Liver disease     No liver disease    8
    No liver disease  No liver disease   22

Fitting performance:
    Accuracy             0.74
    Sensitivity          0.98
    Specificity          0.13
    Balanced accuracy    0.56
    F1 score             0.84

    Depth     6.00
    Features  6.00
    Frugality 3.08
```

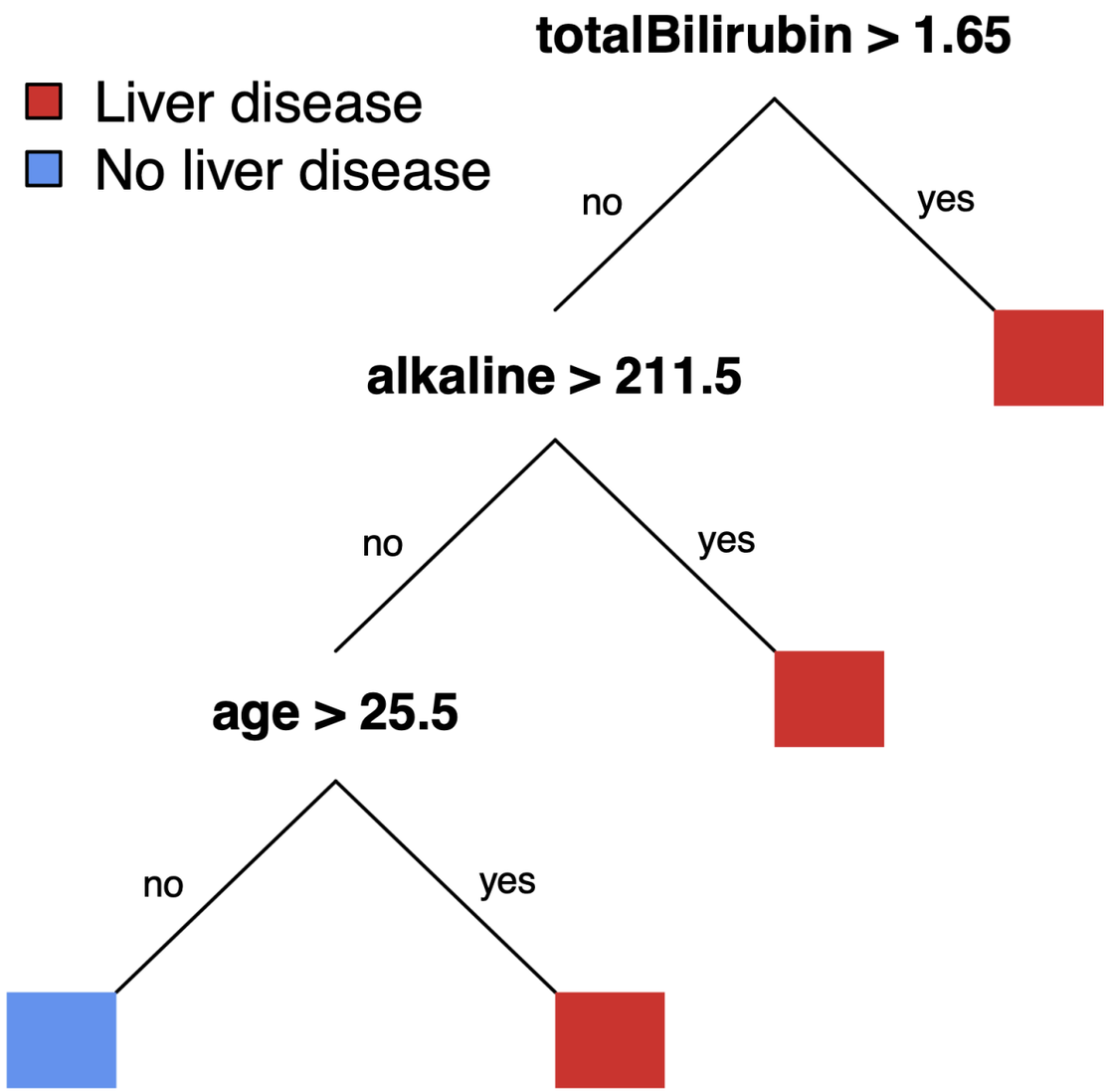
Constructing a fast and frugal tree

```
my_fft <- fftree(data = liver, cv = TRUE)
```

```
...

Cross-validation performance:
    Accuracy             0.69
    Sensitivity          0.83
    Specificity          0.33
    Balanced accuracy    0.58
    F1 score             0.79

    Depth     5.80
    Features  5.80
    Frugality 3.06
```

Visualising the tree

```
plot(another_fft)
```



Make new predictions

```
predict(my_fft, newdata = liver[1:10,])
```

```
 [1] "Liver disease"
 [2] "Liver disease"
 [3] "Liver disease"
 [4] "Liver disease"
 [5] "Liver disease"
 [6] "Liver disease"
 [7] "Liver disease"
 [8] "Liver disease"
 [9] "No liver disease"
[10] "Liver disease"
```

Constructing a fast and frugal tree: options

```
my_fft <- fftree(data = liver)
```

```
my_fft <- fftree(
  data = liver,
  formula = stats::as.formula(data.frame(data)),          # specifies predicted, predictors
  method = “greedy", # “basic” “cross-entropy”           # tree induction method
  max_depth = 6,                                          # maximum number of nodes in the tree
  split_function = “gini", # “entropy” “median”          # how to split numeric features into binary
  weights = c(1, 1),                                      # weights of instances in the two classes
  pruning = FALSE,                                        # If TRUE, tree is pruned using cross-validation
  cv = FALSE,                                             # If TRUE, 10-fold cross validation estimates of predictive performance
  use_features_once = TRUE,                               # If TRUE, an attribute is used only once
  cross_entropy_parameters = cross_entropy_control())
```

Constructing a tallying model

```
my_tally <- tally(data = liver)
```

```
Tallying object
Trained with : “basic” method.

Call:
tally(data = data, formula = formula)

Formula:
diagnosis ~ age + sex + totalBilirubin + directBilirubin + alkaline +
    alamine + aspartate + proteins + albumin + albuminGlobulin

Reasons:
   + totalBilirubin   >     1.65
   + directBilirubin  >     0.85
   + alkaline         >   211.50
   + alamine          >    32.50
   + aspartate        >    47.50
   + proteins         <=    3.65
   ____________________________________
Predict Liver disease if at least 1 reasons hold.

...
```

Constructing a tallying model

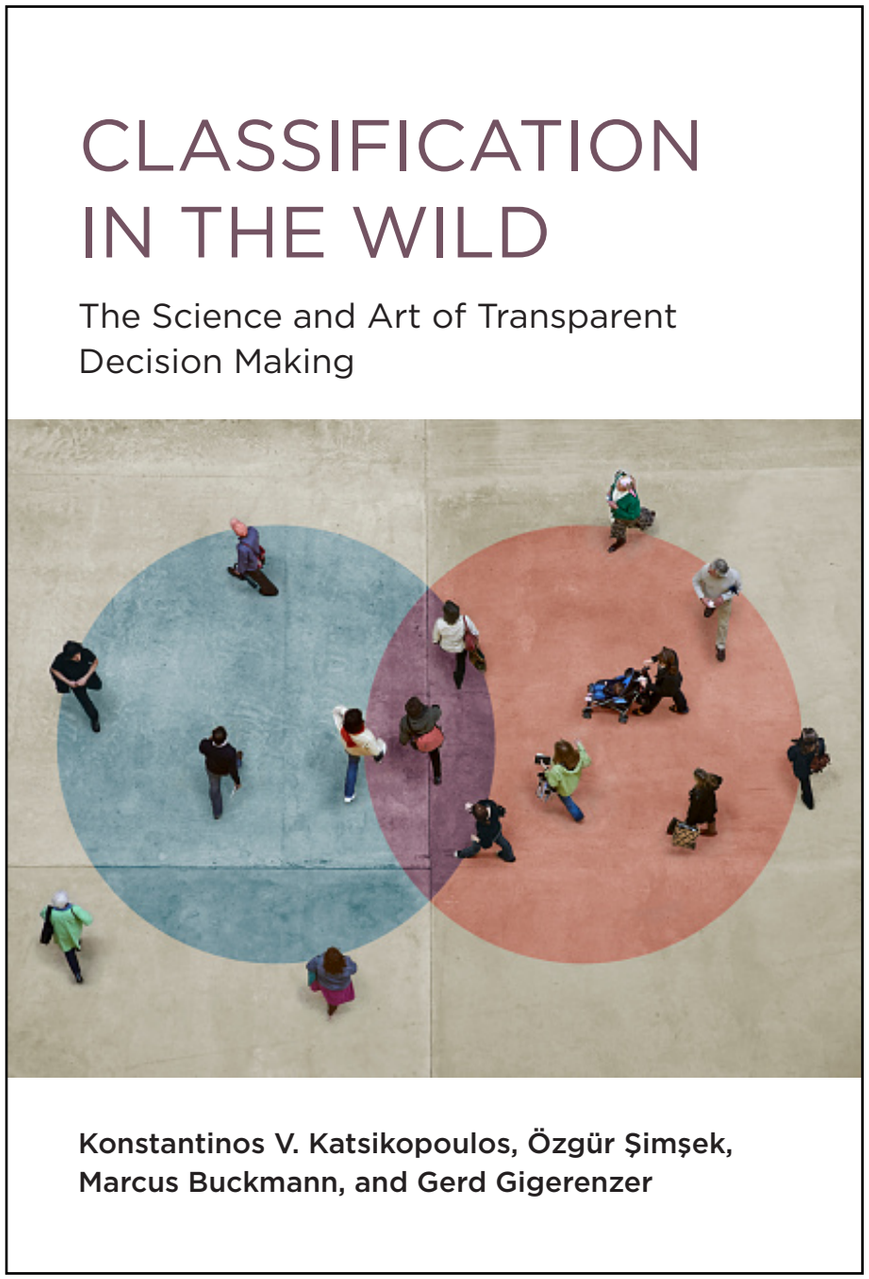
```
my_tally <- tally(data = liver)
```

```
...

Fitted values:
    Observed          Predicted         N
    Liver disease     Liver disease     342
    No liver disease  Liver disease      86
    Liver disease     No liver disease   72
    No liver disease  No liver disease   79

Fitting performance:
    Accuracy             0.73
    Sensitivity          0.83
    Specificity          0.48
    Balanced accuracy    0.65
    F1 score             0.81
```

Constructing a tallying model: options

```
my_tally <- tally(data = liver)
```

```
my_tally <- tally(
  data = liver,
  formula = stats::as.formula(data.frame(data)),          # specifies predicted, predictors
  method = “basic", # “cross-entropy”                     # induction method
  max_size = 6,                                           # maximum number of features in the model
  split_function = “gini", # “entropy” “median”           # how to split numeric features into binary
  weights = c(1, 1),                                      # weights of instances in the two classes
  cv = FALSE,                                             # If TRUE, 10-fold cross validation estimates of predictive performance
  cross_entropy_parameters = cross_entropy_control())
```



+

ffcr:
fast and frugal
classification in R

An R package for constructing fast-and-frugal trees and tallying models from data

https://github.com/marcusbuckmann/ffcr

Additional slides

Basic Tallying

Target class The class that is observed less frequently.

Reasons One reason for each cue.
Cue direction = the one that performs better than chance in training data.

Threshold The threshold that works best in training data.

Basic Fast-and-Frugal Tree

d = maximum tree depth

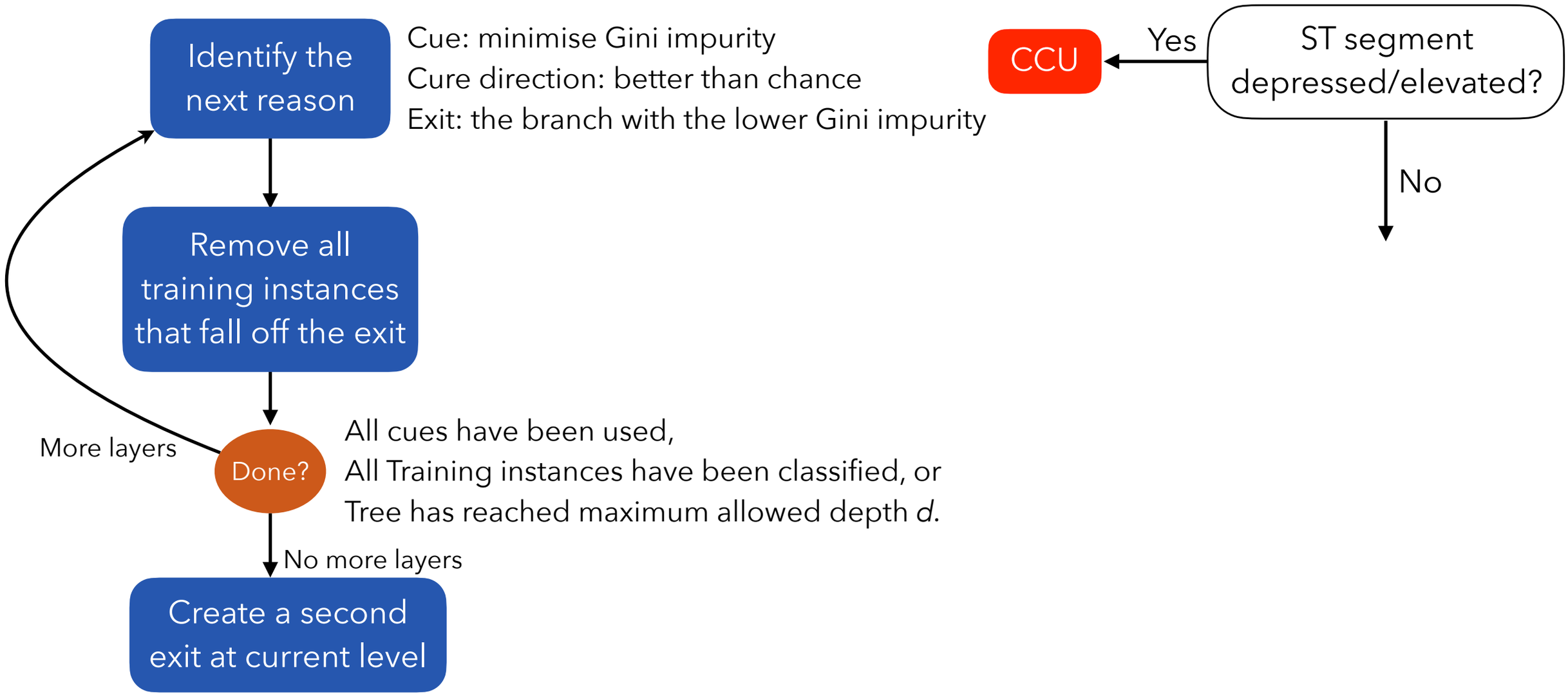
Cue order
Order cues in increasing order of their Gini impurity
Take only the d cues with the lowest impurity.

Cue directions
Set each cue direction to the value (positive, negative) that performs better than chance in the training set, as measured by hit rate and false alarms rate.

Exit structure
Among the 2^d+1 possible exit structures, pick the one that gives the best trade-off between hit rate and false alarm rate in the training data

Greedy Fast-and-Frugal Tree

Build the tree layer by layer.



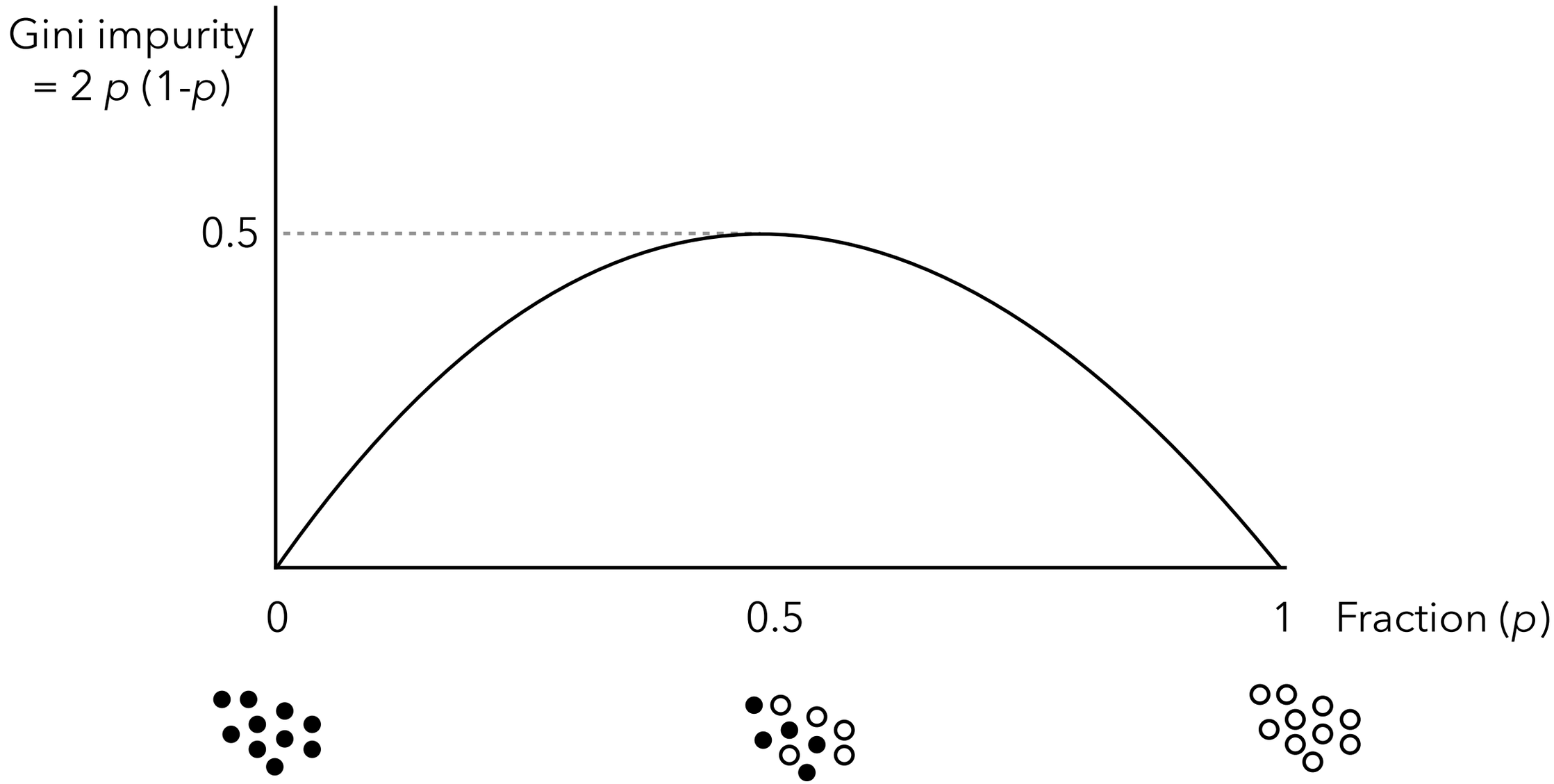
Best-Fit (Cross Entropy)

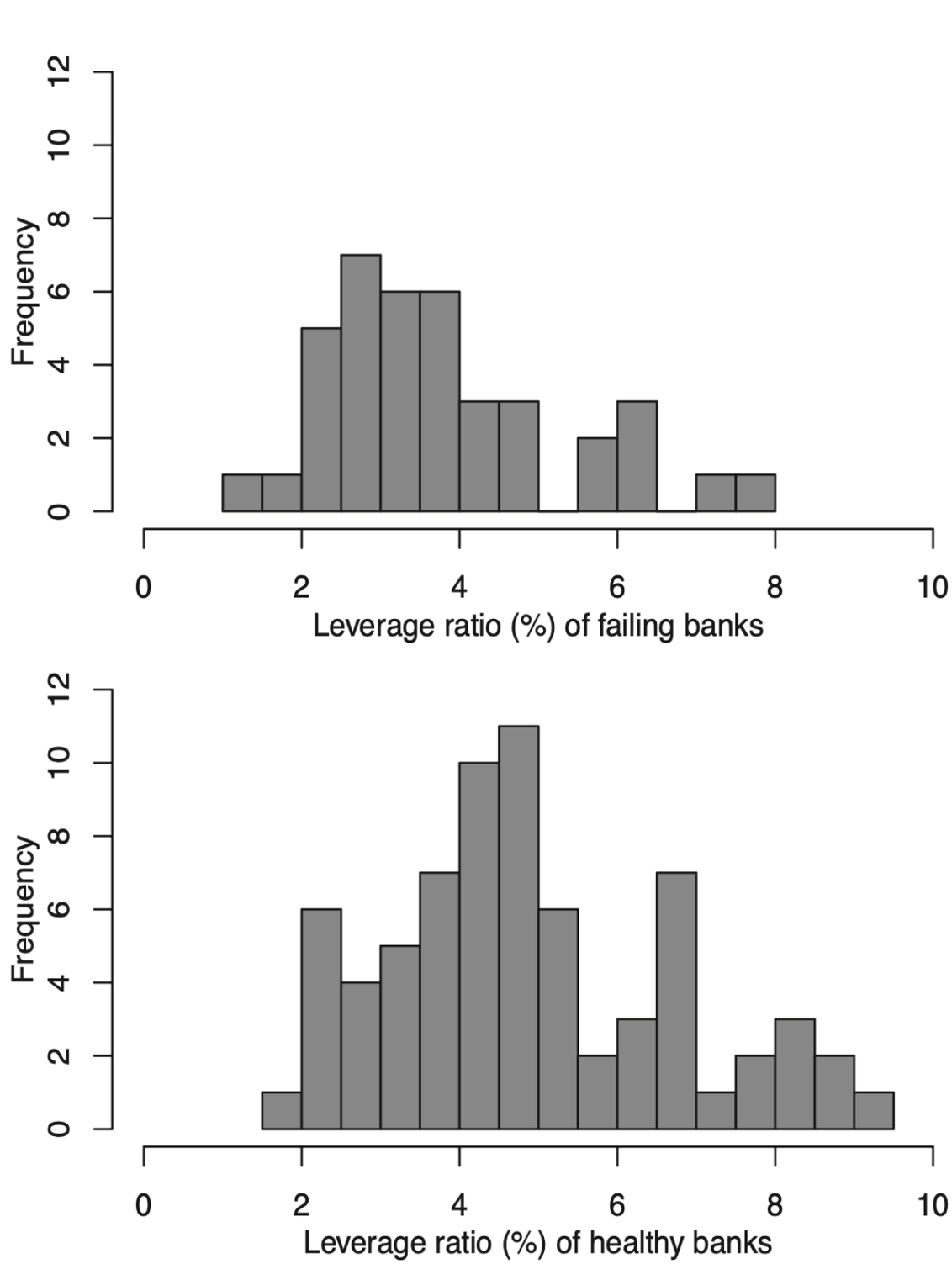
Enumerate all possible fast and frugal trees (or tallying models).
Pick the one that performs best in the training set.

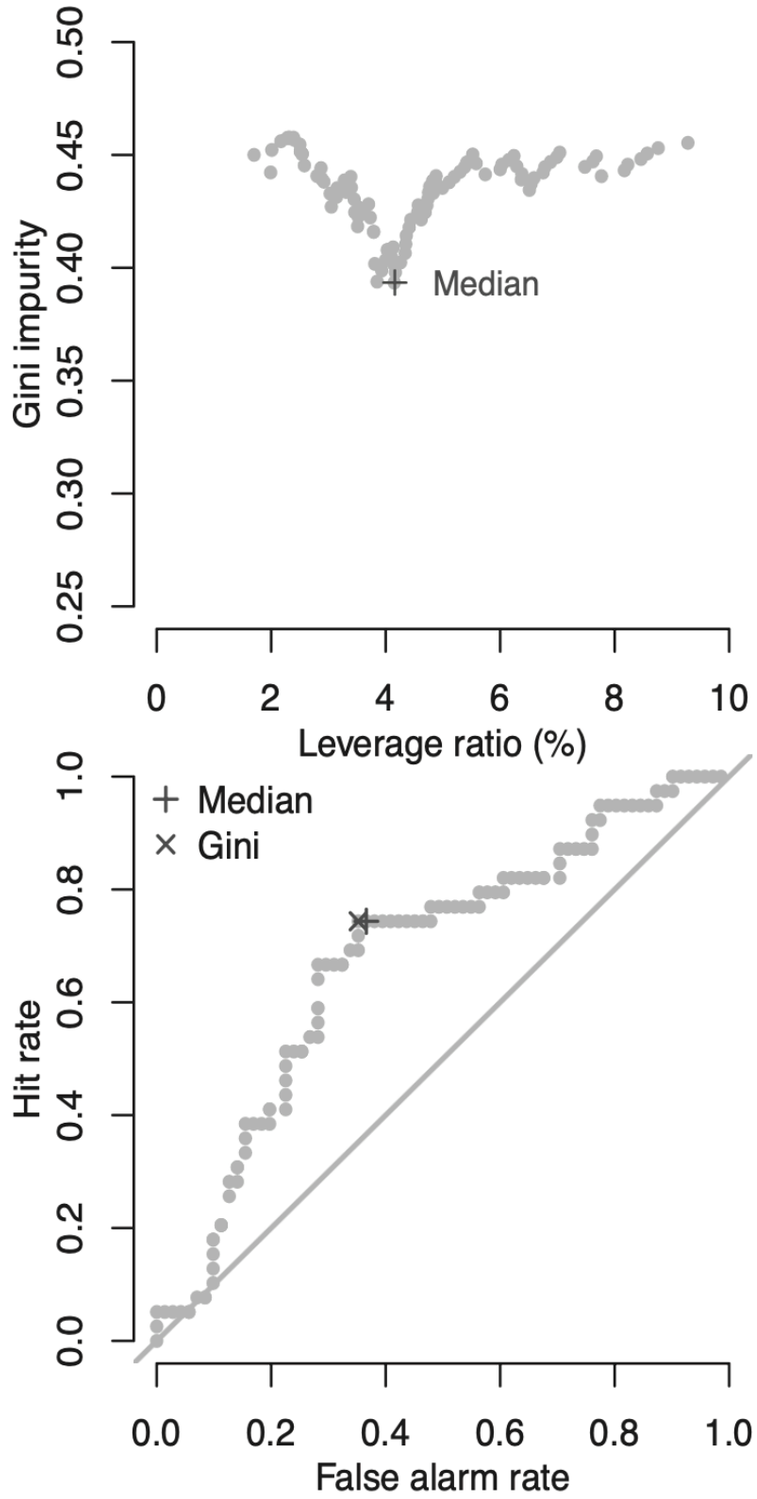
Cross Entropy Parameters

starts = number of models to train (best fit will be selected)
learning_rate = learning rate of the cross-entropy optimisation
maximum_time = allocated compute budget in seconds
iterations = number of cross-entropy iterations
early_stopping = if training loss does not decrease after this many iterations, stop training
thresholds = the maximum number of values on each variable considered for splits. For example, when you have a variable with 1000 distinct values, you may not want to consider all of these as possible splits. Instead you pick 100 (default value) equidistant split points between the minimum and maximum value. On a scale of 0 to 1 that would be 0.01,0.02, 0.99)
split_percentiles = with outliers, may want to pick the split points by using percentiles instead of equidistant values between the min and max.
samples and **elite_samples:** standard parameters of cross-entropy optimisation
threads determines on how many cores to run in parallel.

Measuring Purity




Frequency
0
2
4
6
8
10
12
0
2
4
6
8
10
Leverage ratio (%) of failing banks
Frequency
0
2
4
6
8
10
12
0
2
4
6
8
10
Leverage ratio (%) of healthy banks

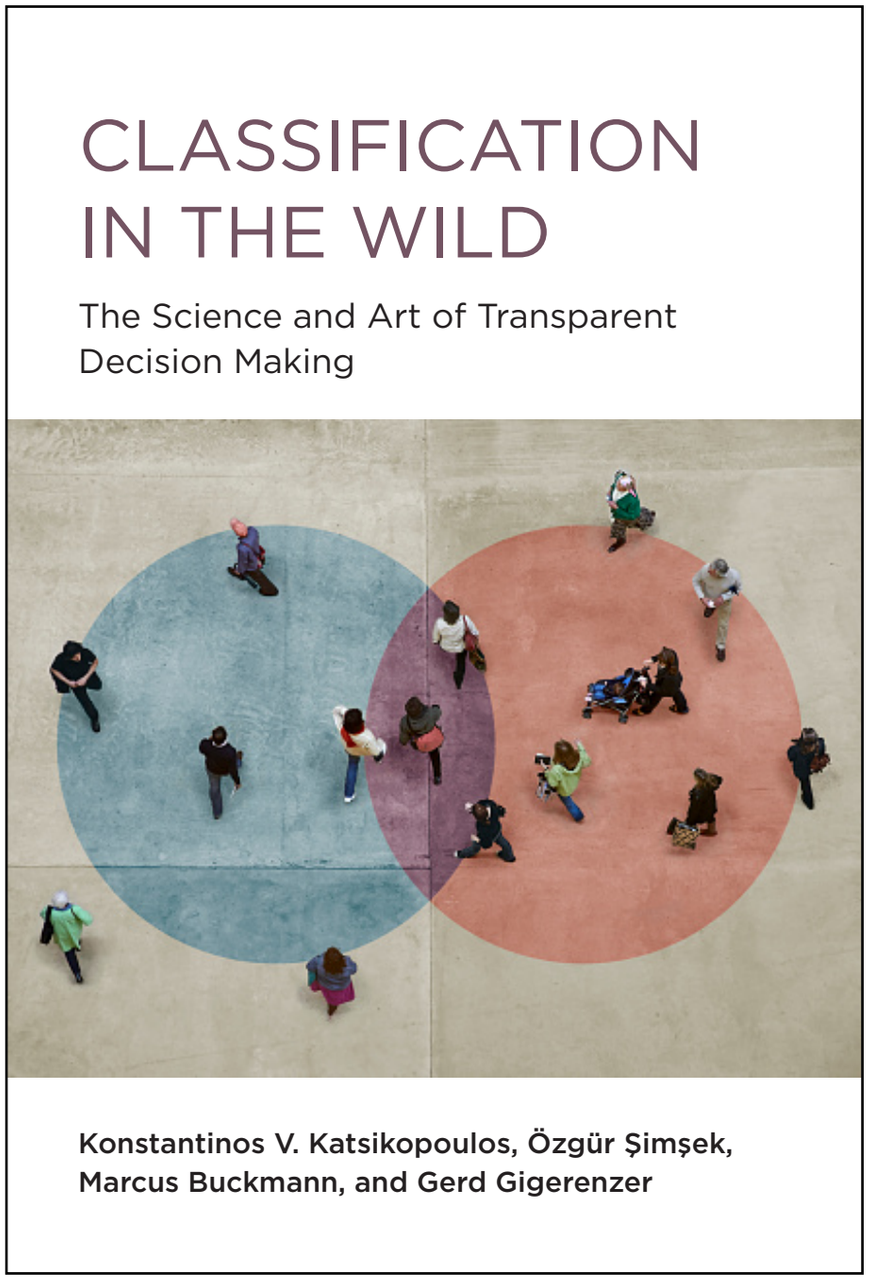

Gini impurity
0.25
0.30
0.35
0.40
0.45
0.50
Median
0
2
4
6
8
10
Leverage ratio (%)
Hit rate
0.0
0.2
0.4
0.6
0.8
1.0
Median
Gini
0.0
0.2
0.4
0.6
0.8
1.0
False alarm rate

Case studies

Building fast-and-frugal trees with domain experts when quantity or quality of the data not sufficient to learn from data

From 2004 to 2009, in 1,060 incidents in *NATO* checkpoints in Afghanistan, there were 7 “successful” suicide attacks. There were also 204 civilian casualties.

Keller and Katsikopoulos (2016),
European Journal of Operational Research



+

ffcr:
fast and frugal
classification in R

An R package for constructing fast-and-frugal trees and tallying models from data

https://github.com/marcusbuckmann/ffcr