LECTURE 18:
TREE-BASED METHODS PT. 1

November 13, 2017

SDS 293: Machine Learning

Announcement

- Minor change to Final Project Poster Session
- They will now take place **during class time** on Dec. 13th
  - **Location**: McConnell Foyer
  - Part of the larger Course Based Research Poster Symposium (put it on your resume!)
  - Better accommodates students that work, have to travel, etc.

Outline

- Final project activity: important questions
- Basic mechanics of tree-based methods
  - Classification example
  - Choosing good splits
  - Pruning
- How to avoid over-fitting
  - Bootstrap aggregation (“bagging”)
  - Random forests
  - Boosting
- Lab

Final Project Deliverables

✓Nov. 8th - FP1: Data Appendix

- **Nov. 27th – FP2: Initial Model**
- Dec. 4th – FP3: Revised Model
- Dec 13th – Final Project Reception (posters due!)
- Dec. 22nd - FP5: Final Write-Up

NEWS

Activity: big questions

- What **question** is the project is trying to help answer?
- How have people answered it / gotten around it **before**?
- What **new idea** does this project offer that improves on the old way of doing things?
- What are the (major) **building blocks** the project will need to be successful?
- Which ones are **in place already**, and which ones are **still in progress**?
- Are there any **potential roadblocks**?

2015 Example: supreme court decisions

2015 Example: supreme court decisions

What question is the project is trying to answer?

We are trying to help people better understand patterns in how the US Supreme Court votes.

For example:

- *How often do S.C. justices actually vote in 'blocks’?*
- *How does justice X vote on specific issues?*
- *How does justice X vote compared to justice Y?*

2015 Example: supreme court decisions

How have people answered it / gotten around it before?

Reading opinions written by justices generally helps experts understand how they vote. People haven't done as much research on aggregated data in this area, although some textbooks use graphs generated by:

supremecourtdatabase.org

There is not much information designed to help average citizens understand how the supreme court votes.

2015 Example: supreme court decisions

What new idea does this project offer that improves on the old way of doing things?

This project will provide a simple way to explore the data. Ideally, it will give the user enough flexibility to explore what they want, while not being too overwhelming (as some other databases are).

It will also be interactive; others are not.

2015 Example: supreme court decisions

What are the (major) building blocks the project will need to be successful?

The major building blocks we will need are

a) access to data on issues, votes, etc. and

b) access to written opinions

The project will also need to be intuitive, so we will need help to choose the right way to communicate this data.

2015 Example: supreme court decisions

Which ones are in place already, and which ones are still in progress?

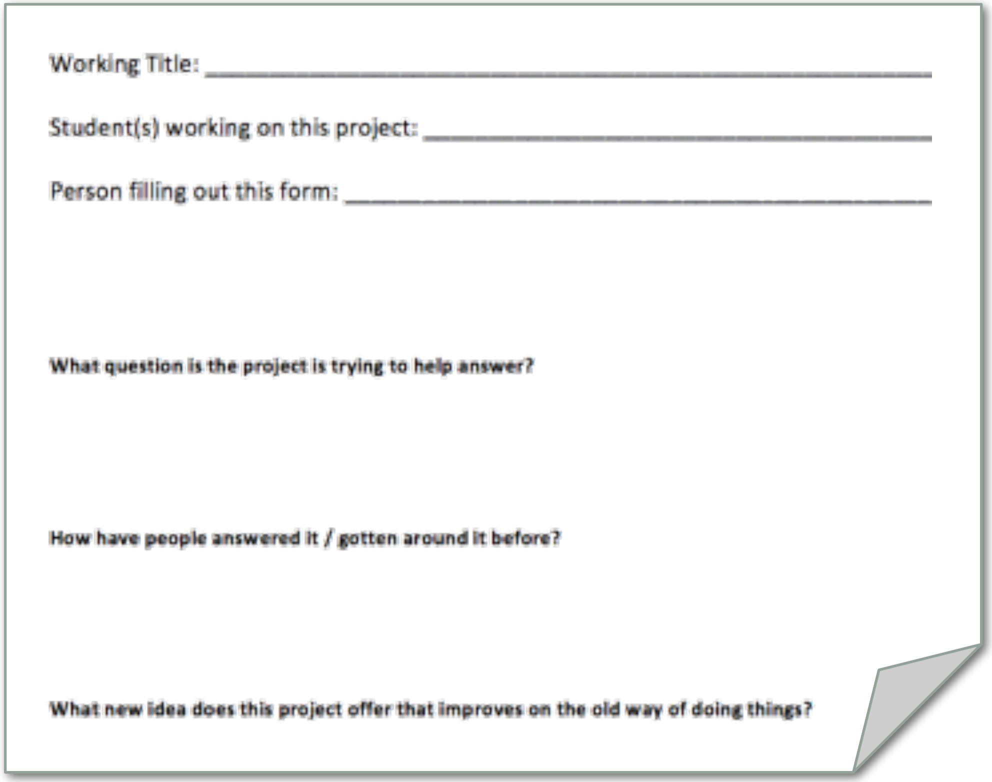
We have already gotten most of the data on issues and votes from online sources. We still need to figure out how to deal with the text of opinions; we are considering looking at word frequency, but are concerned that this won't capture enough context. We have not decided how to model the data yet.

2015 Example: supreme court decisions

Are there any potential roadblocks?

We haven’t learned how to work with text data yet, so that might be more difficult than we expect.

Activity 1: big questions



Working Title: ______________________________

Student(s) working on this project: ______________________________

Person filling out this form: ______________________________

What question is the project is trying to help answer?

How have people answered it / gotten around it before?

What new idea does this project offer that improves on the old way of doing things?

Discussion

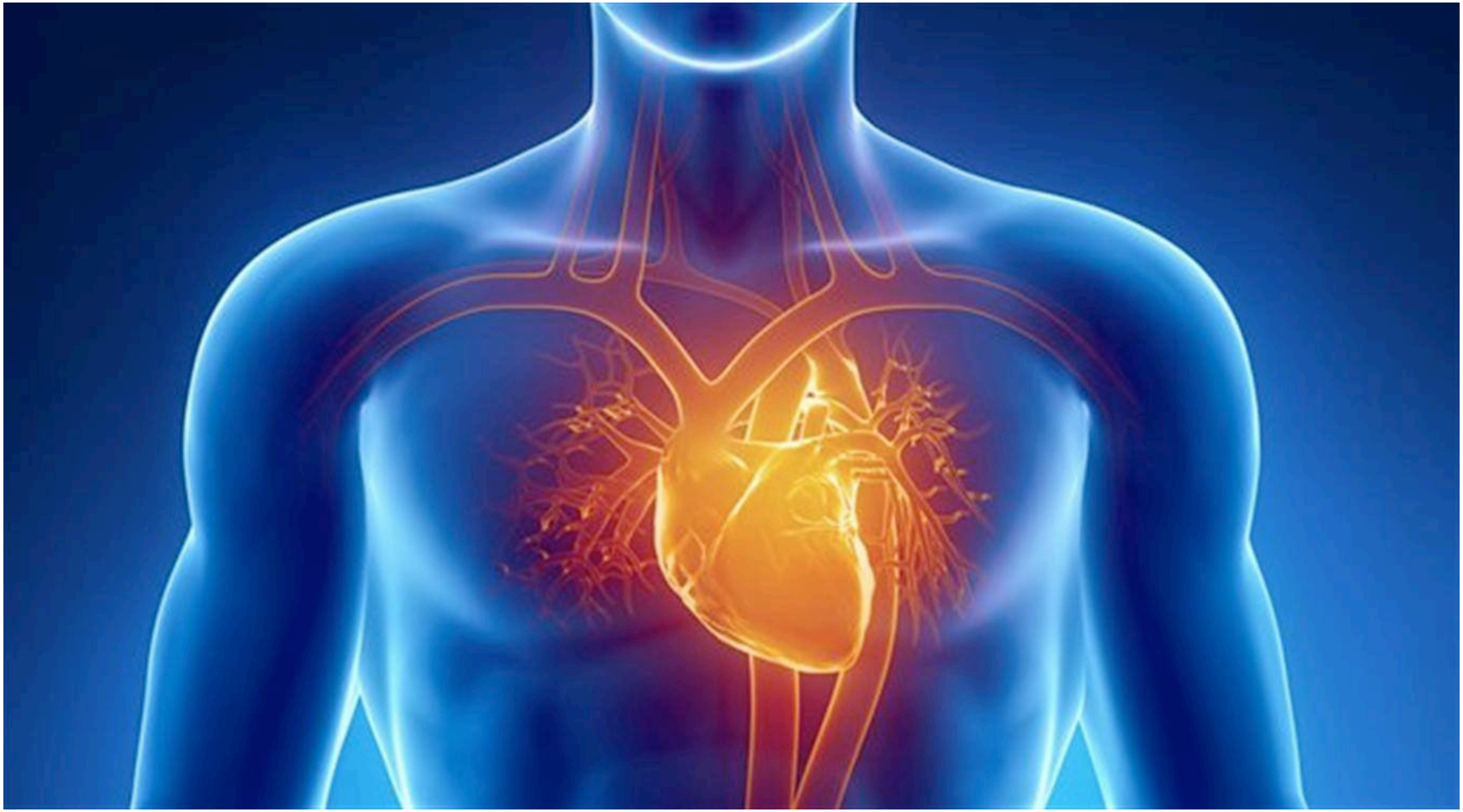
What potential **roadblocks** did you discover?

Outline

✓ Final project activity: important questions

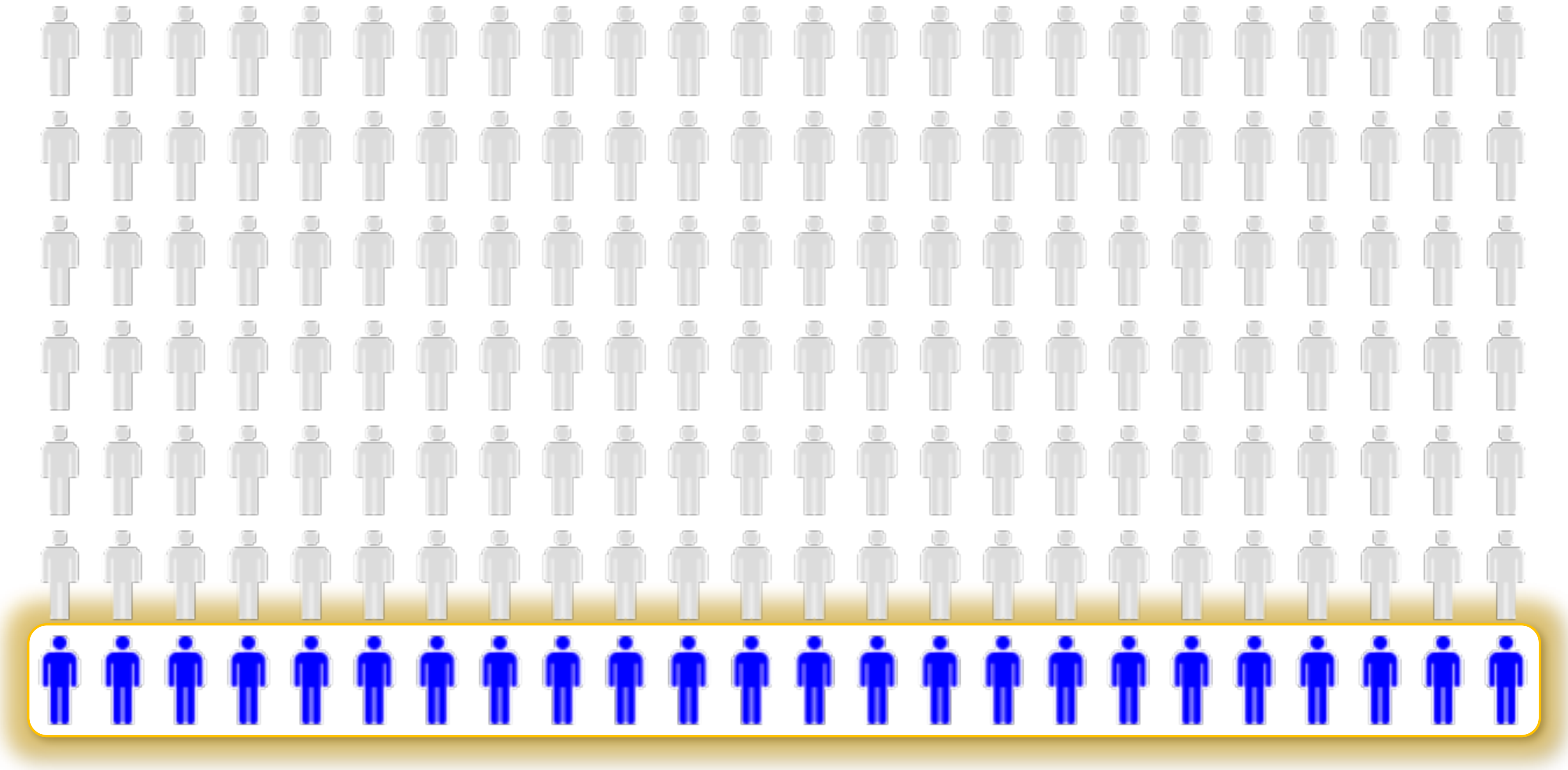
- Basic mechanics of tree-based methods
  - Classification example
  - Choosing good splits
  - Pruning
- How to avoid over-fitting
  - Bootstrap aggregation (“bagging”)
  - Random forests
  - Boosting
- Lab

Example: surviving cardiac arrest



Example: surviving cardiac arrest

Full dataset: 168 patients

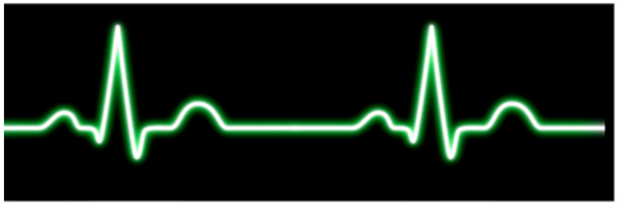


24 of 168 patients survived

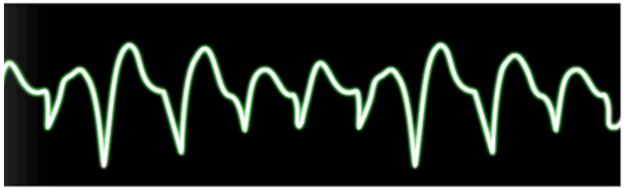
144 of 168 patients could not be revived

Crystal ball: best predictor

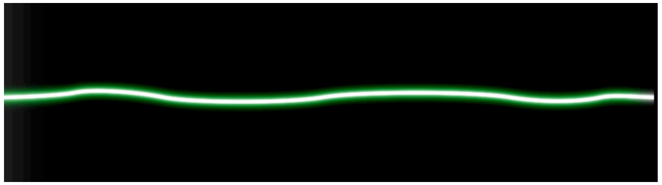
Different types of arrhythmia



Normal



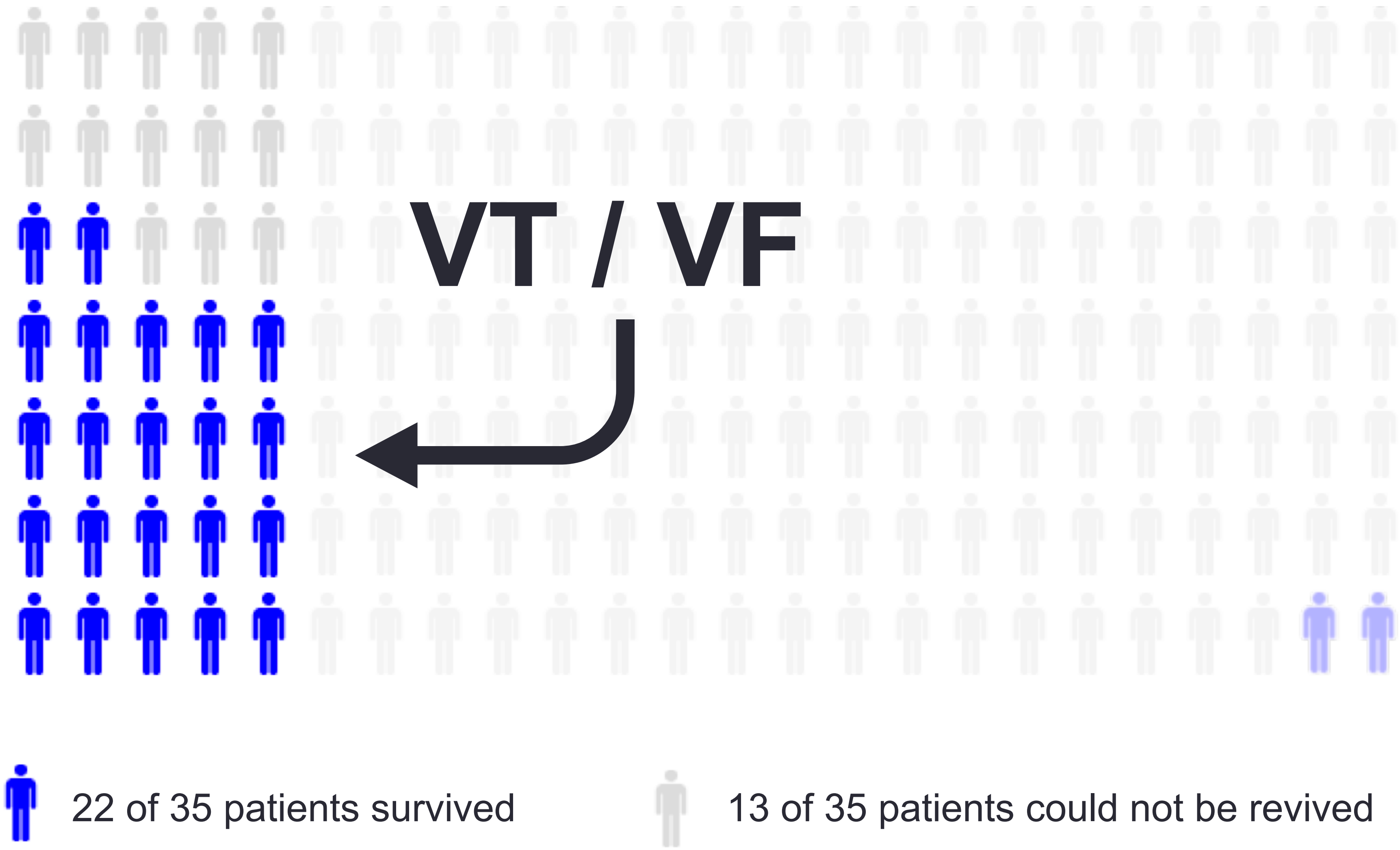
Ventricular Tachycardia (VT) /
Ventricular Fibrillation (VF)



EMD /
Asystole / Other

Images courtesy of Wikipedia

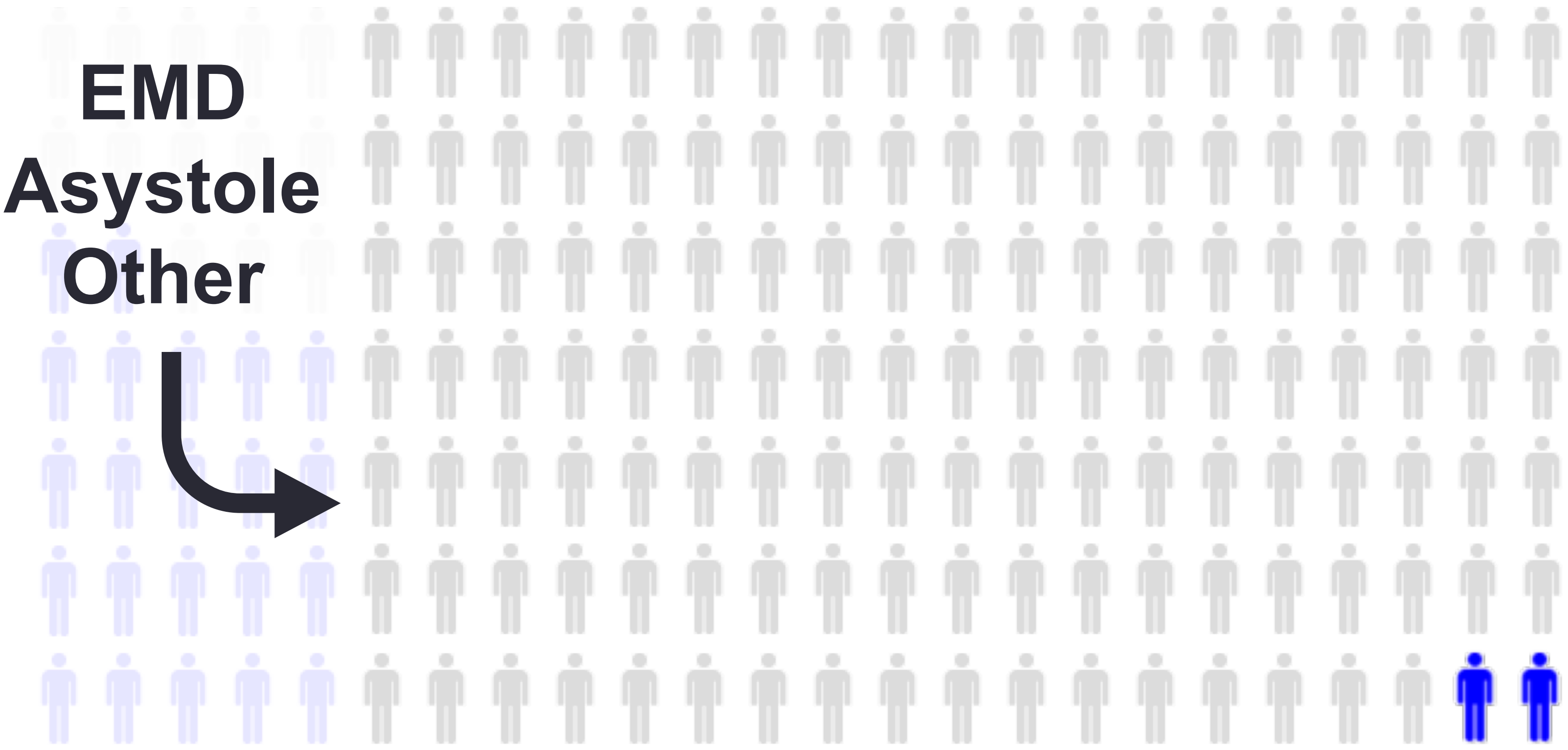
First Split: Initial Heart Rhythm



22 of 35 patients survived

13 of 35 patients could not be revived

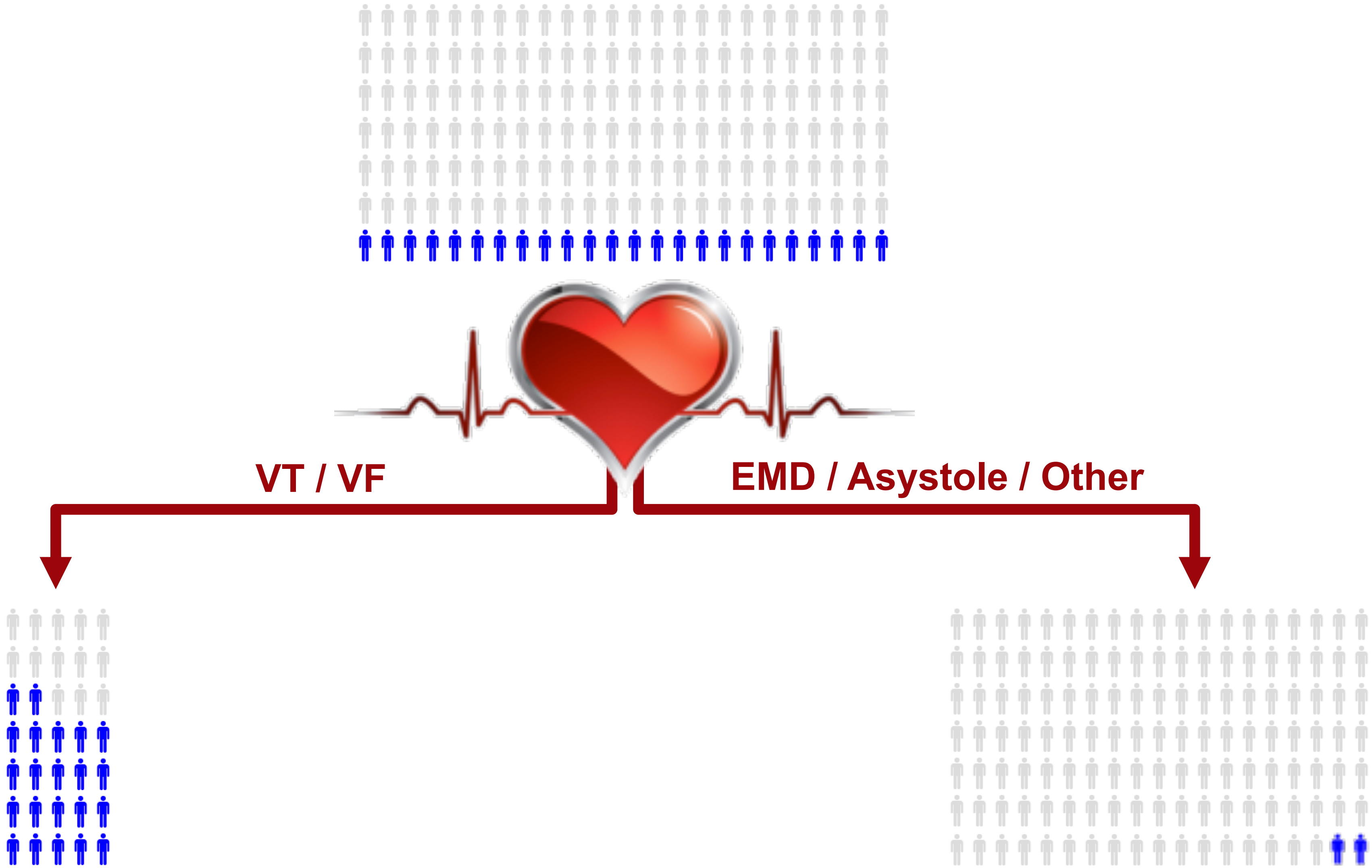
First Split: Initial Heart Rhythm



● 2 of 133 patients survived

● 131 of 133 patients could not be revived

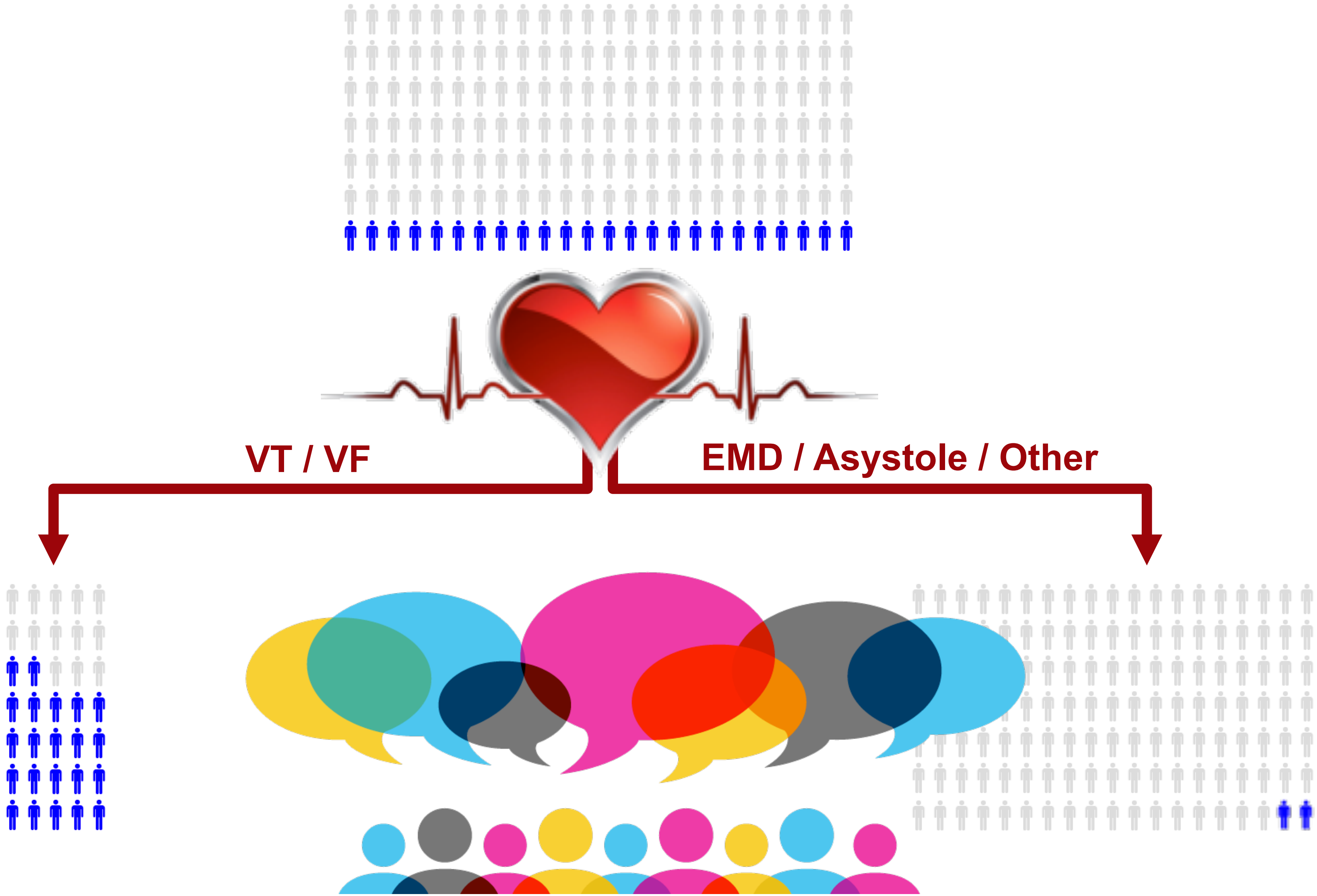
Another view: partitioning



VT / VF

EMD / Asystole / Other

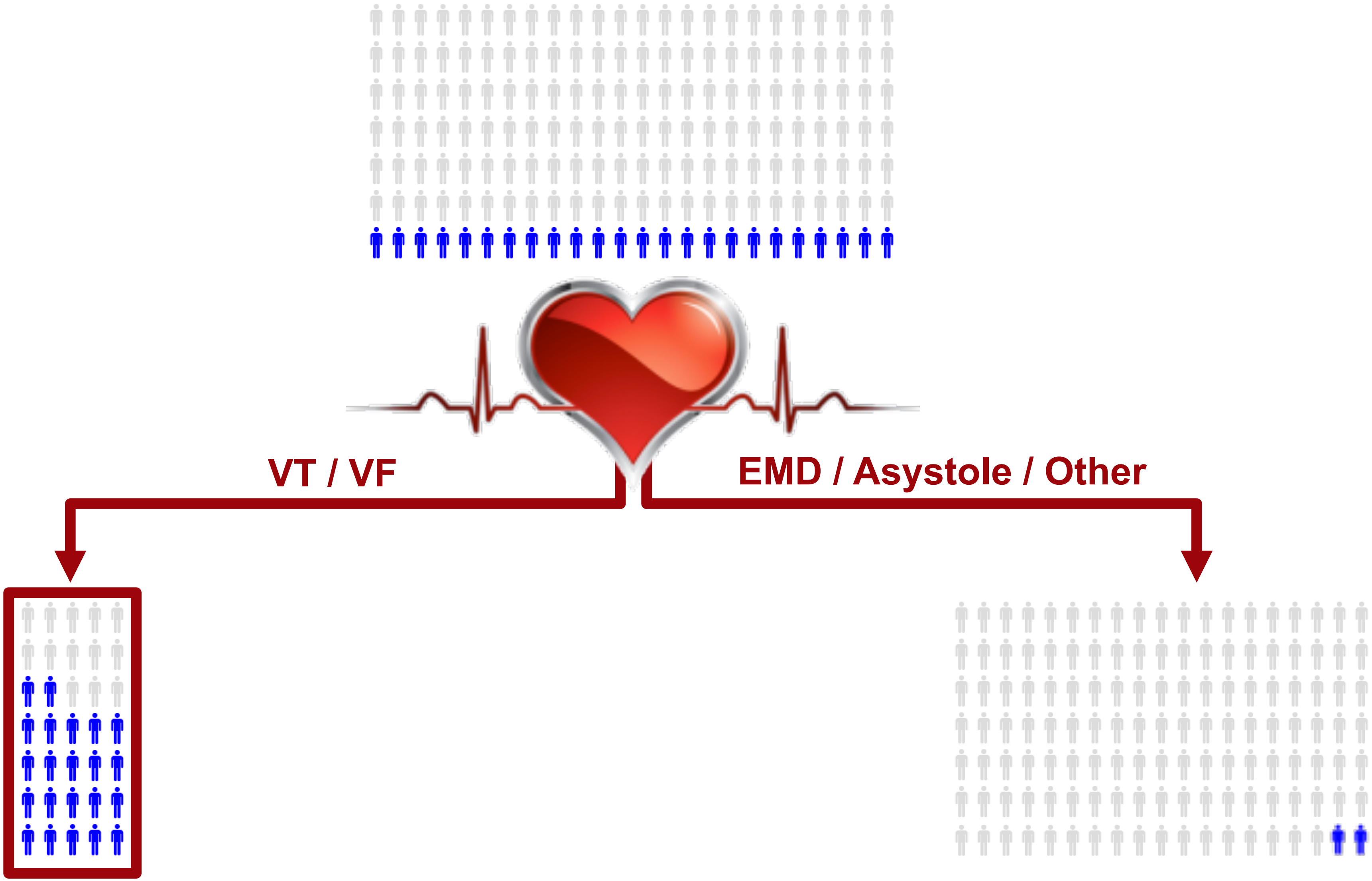
Now what do we do?



VT / VF

EMD / Asystole / Other

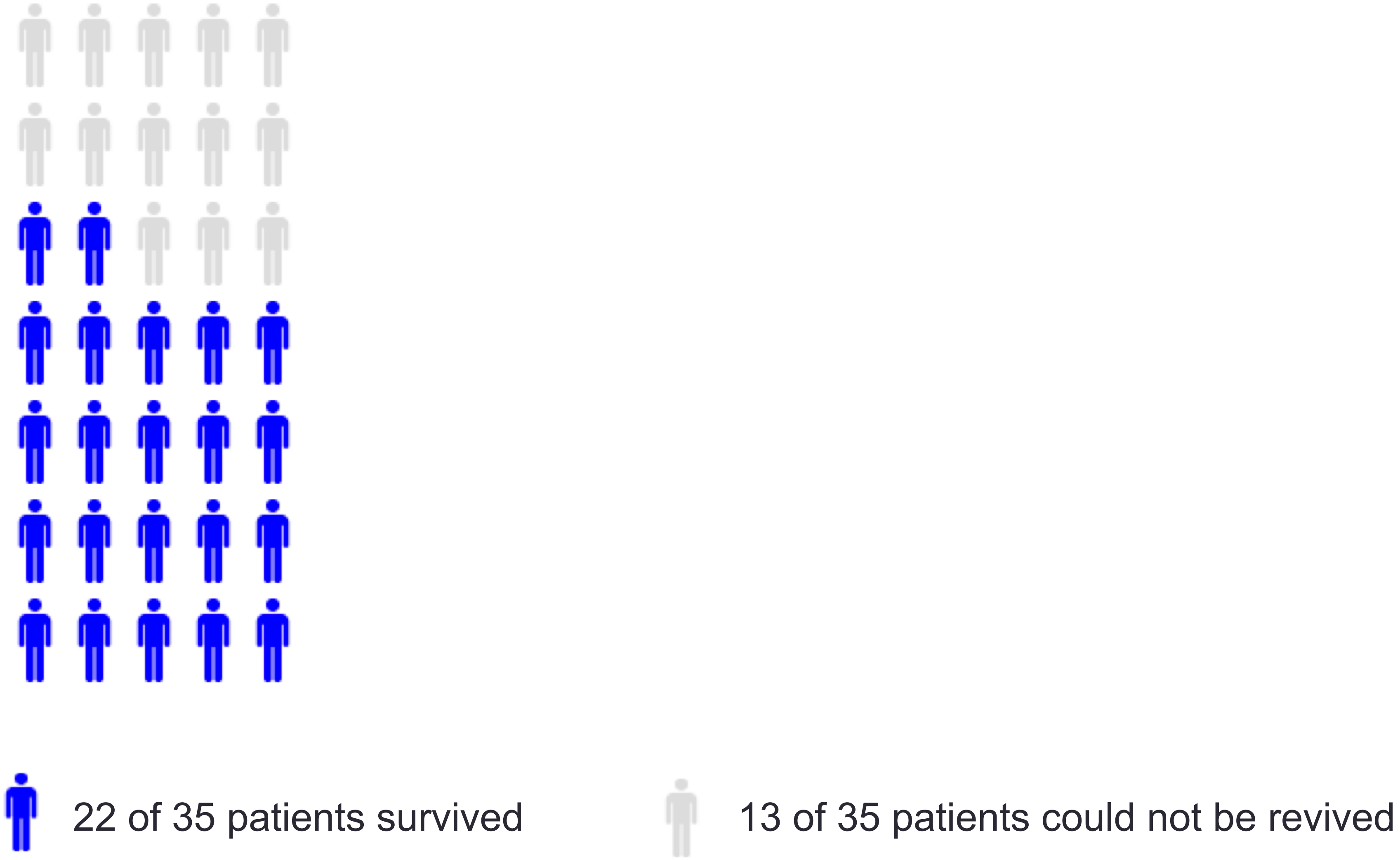
Recursion!



VT / VF

EMD / Asystole / Other

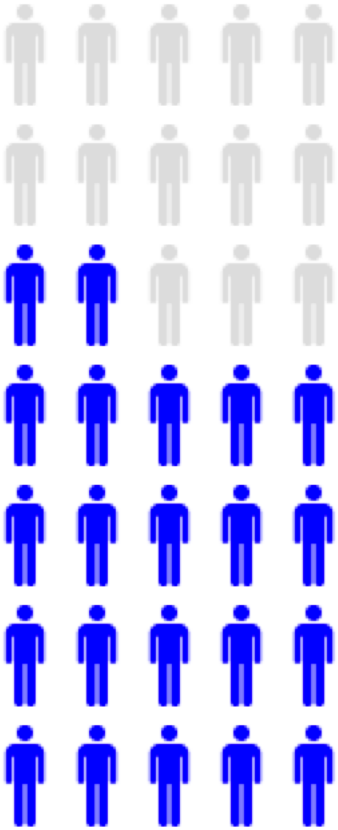
VT / VF group only

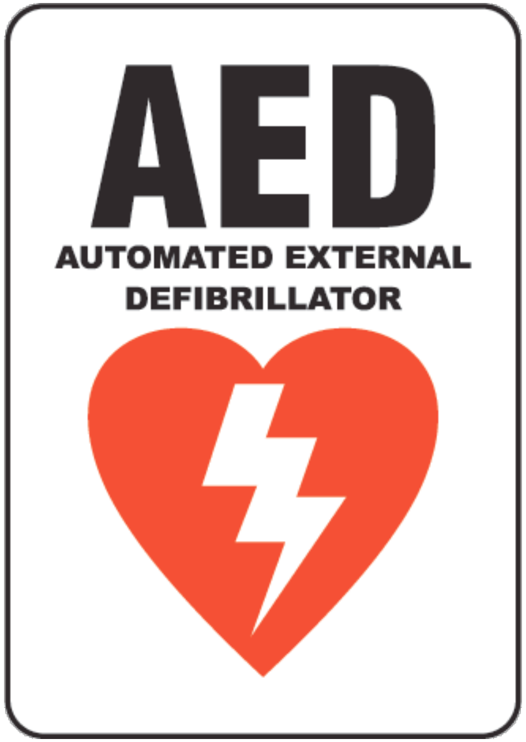


22 of 35 patients survived

13 of 35 patients could not be revived

Next split: response to defibrillation

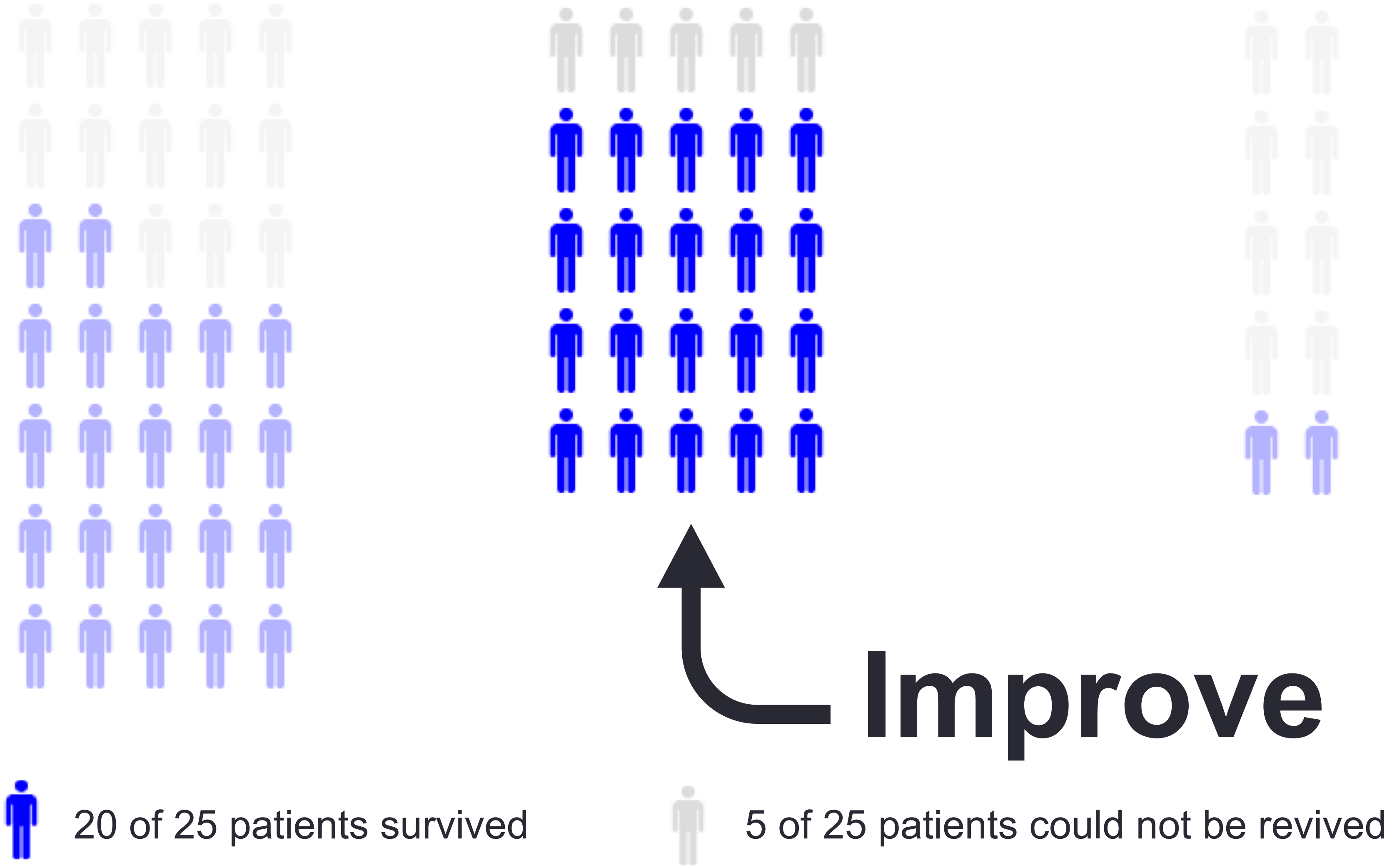




22 of 35 patients survived

13 of 35 patients could not be revived

Next split: response to defibrillation

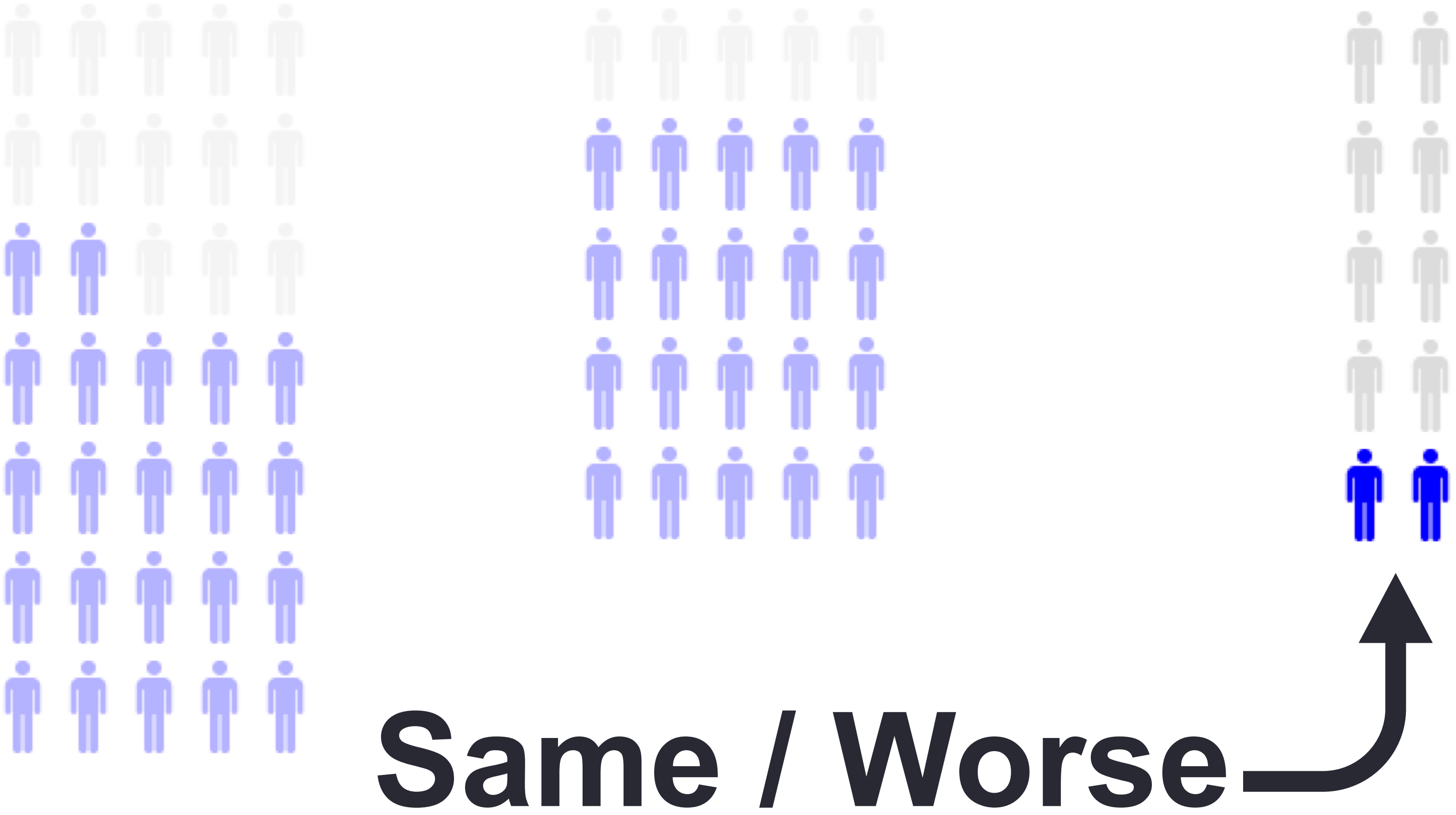


Improve

20 of 25 patients survived

5 of 25 patients could not be revived

Next split: response to defibrillation

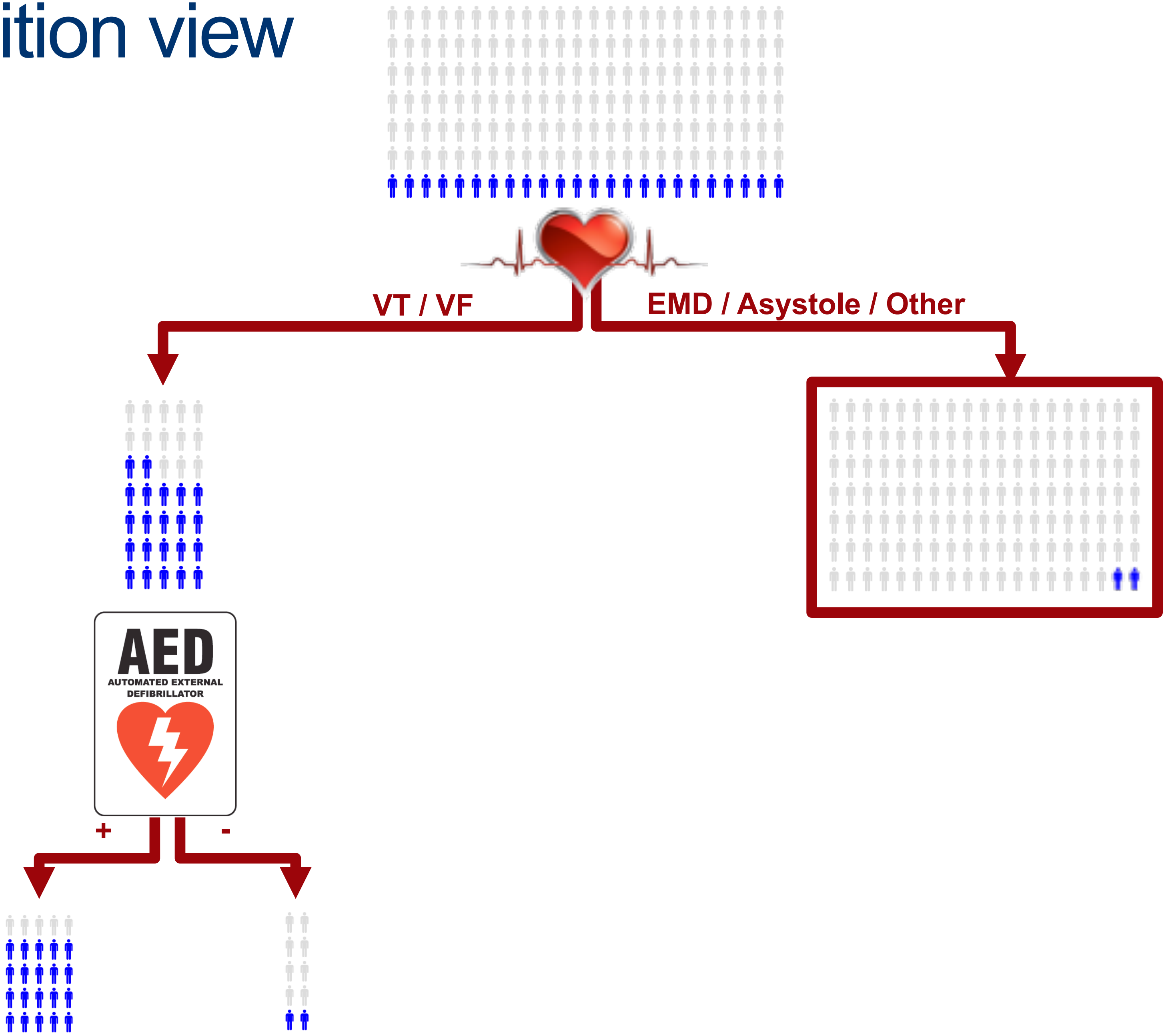


Same / Worse

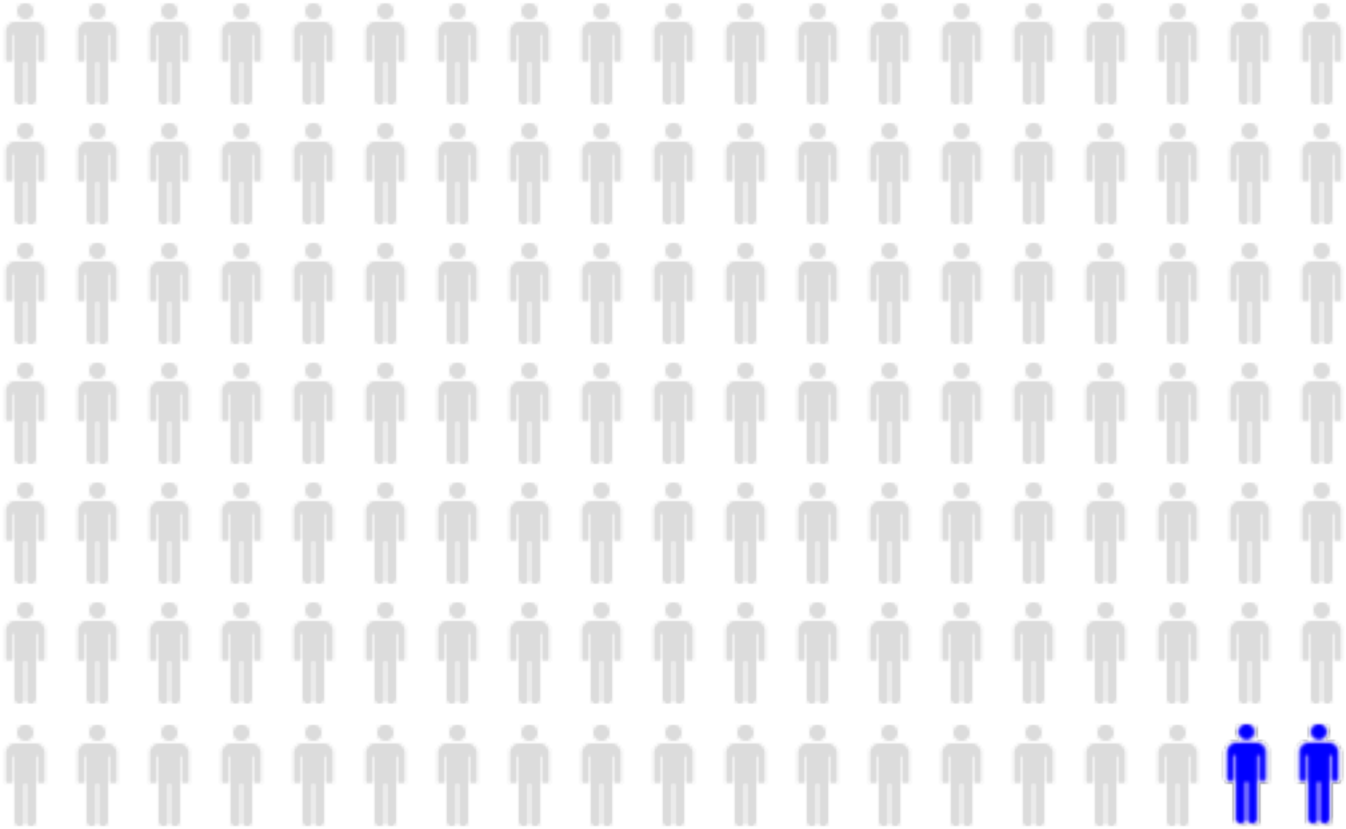
2 of 10 patients survived

8 of 10 patients could not be revived

Partition view



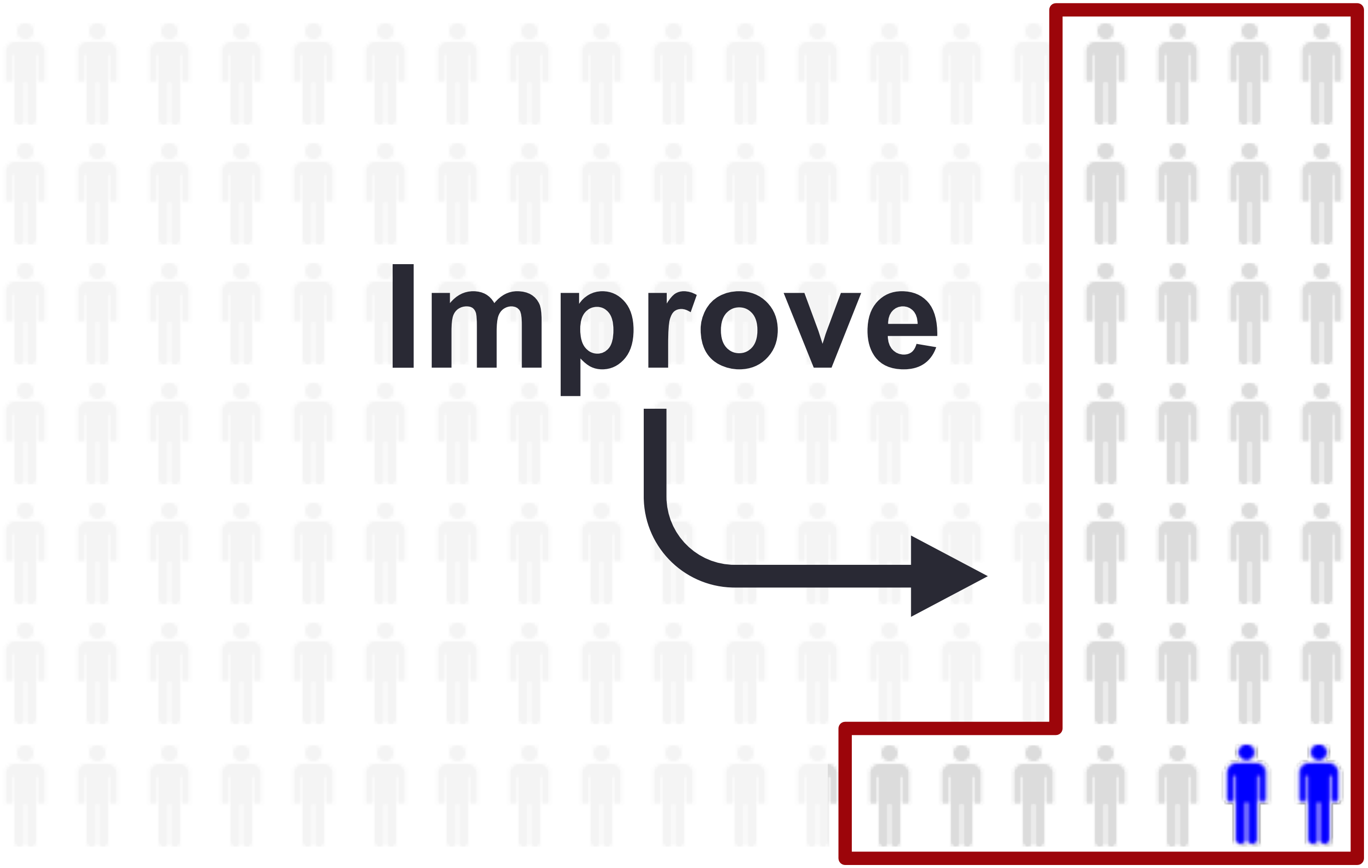
Next split: response to ~~defibrillation~~ medication



2 of 133 patients survived

131 of 133 patients could not be revived

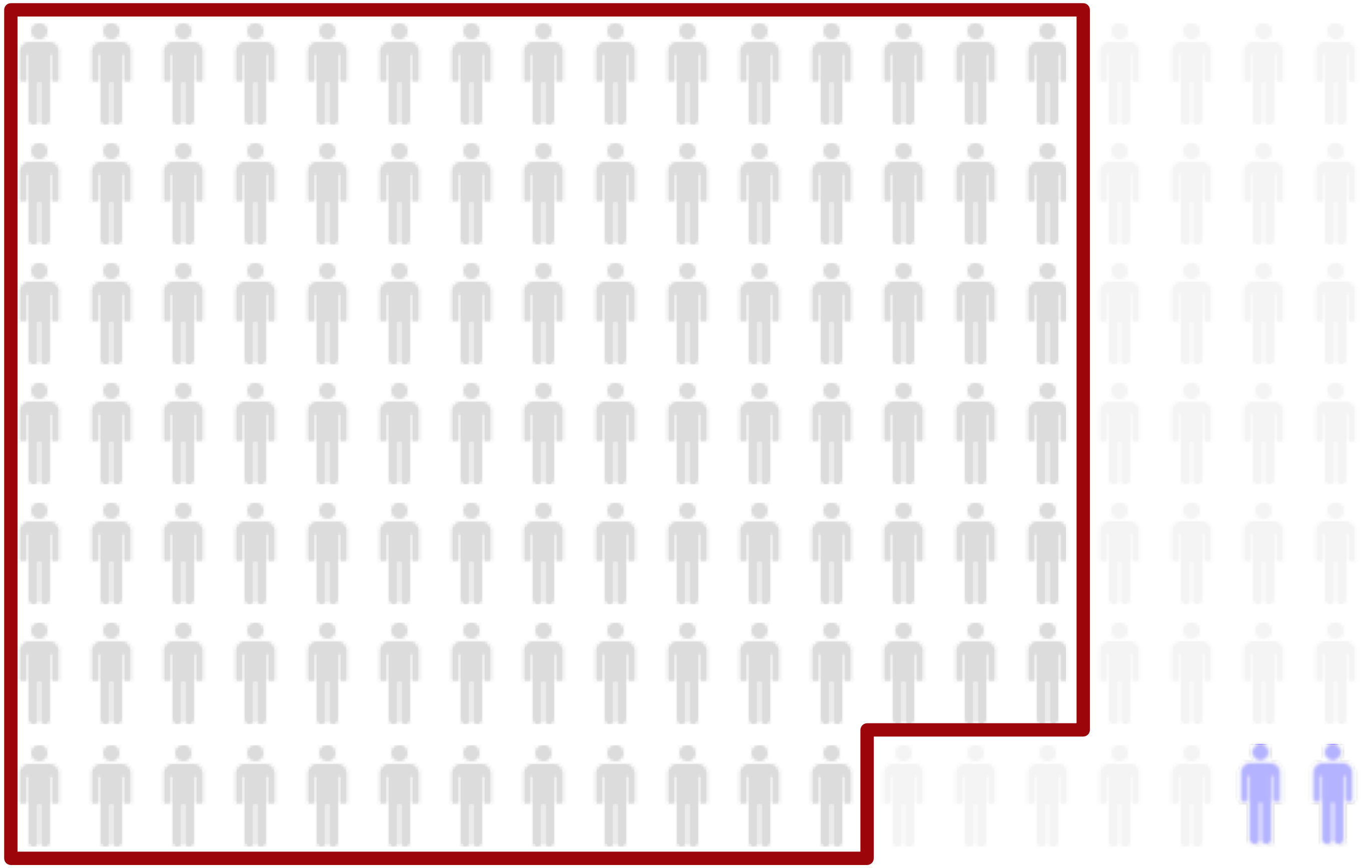
Next split: response to ~~defibrillation~~ medication



2 of 31 patients survived

29 of 31 patients could not be revived

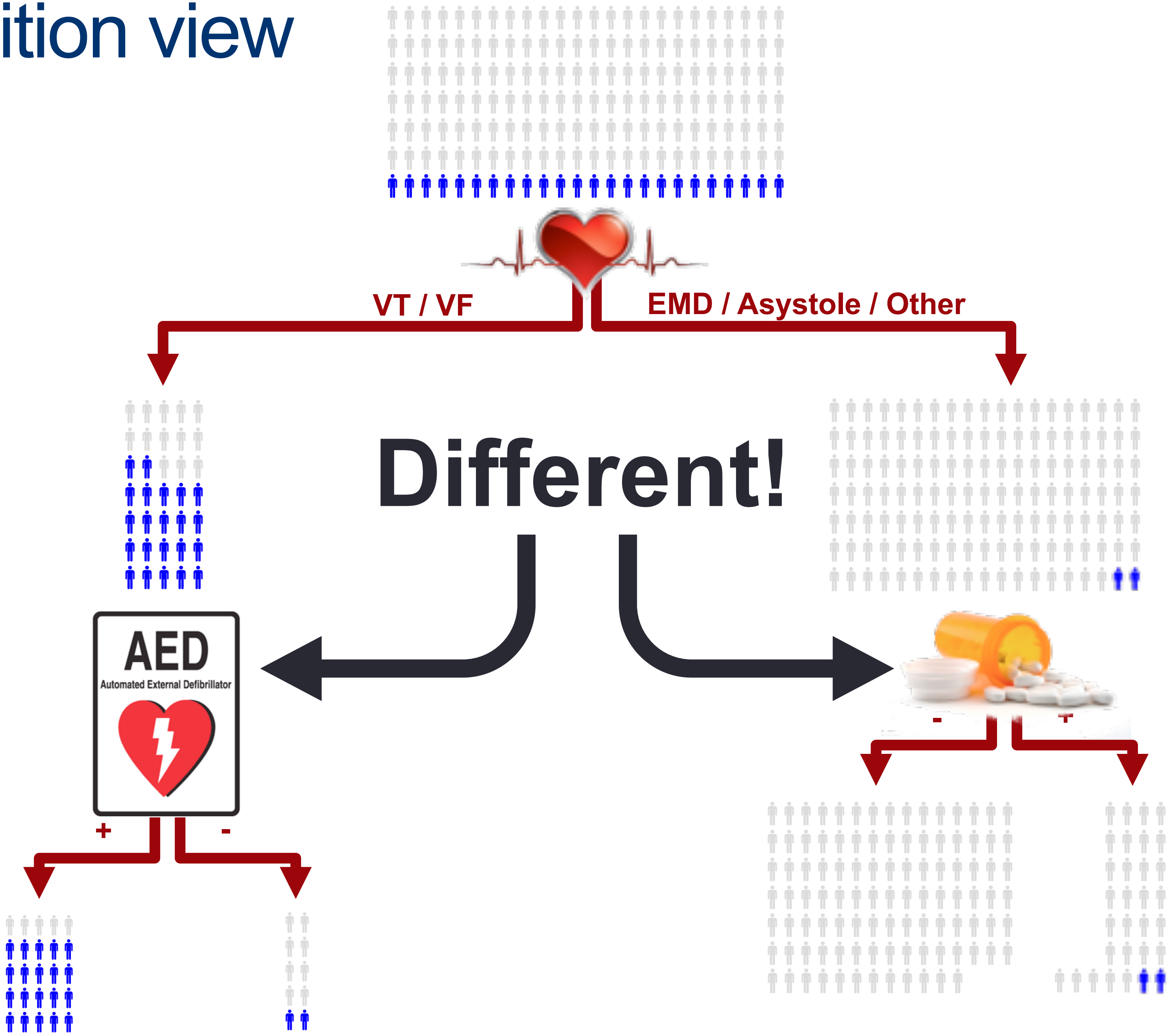
Next split: response to ~~defibrillation~~ medication



0 of 102 patients survived

102 of 102 patients could not be revived

Partition view



VT / VF

EMD / Asystole / Other

Different!

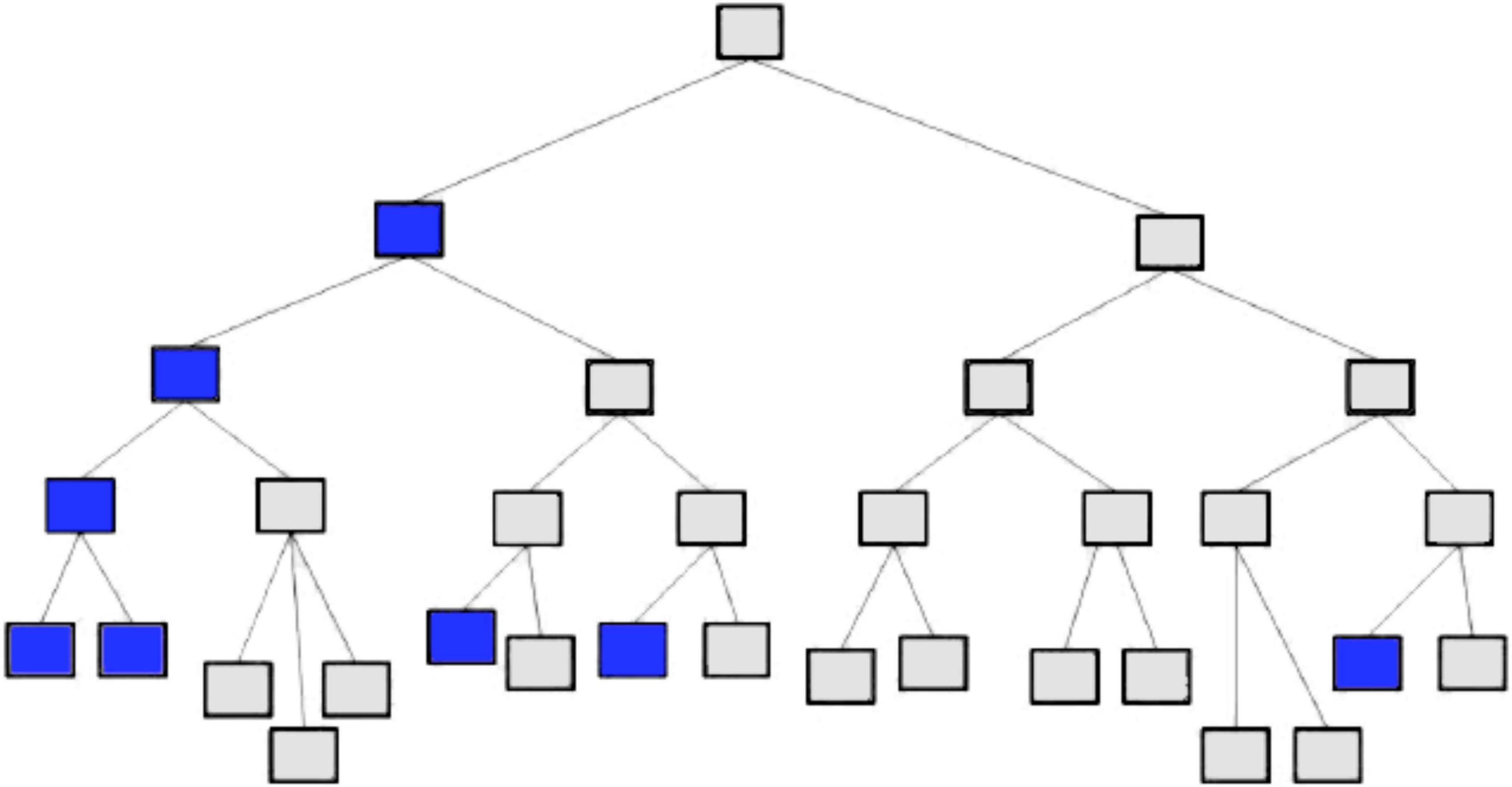
+

-

-

+

Continue ad nauseum…



Discussion

- **Question:** what could go wrong with this approach?
- **Answer:** the resulting tree might be too complex, leading to poor test set performance and difficulty interpreting the results

Growing (and pruning) trees

- **Big idea:** build a big tree, then cut off (“prune”) the branches that aren’t improving performance
- **Question:** why not just build a smaller tree to begin with?
- **Answer:** this is the same issue we had with our linear model selection methods: a branch that doesn’t seem useful early on may give rise to a better branch further down – if we stop too soon, we’d never know!

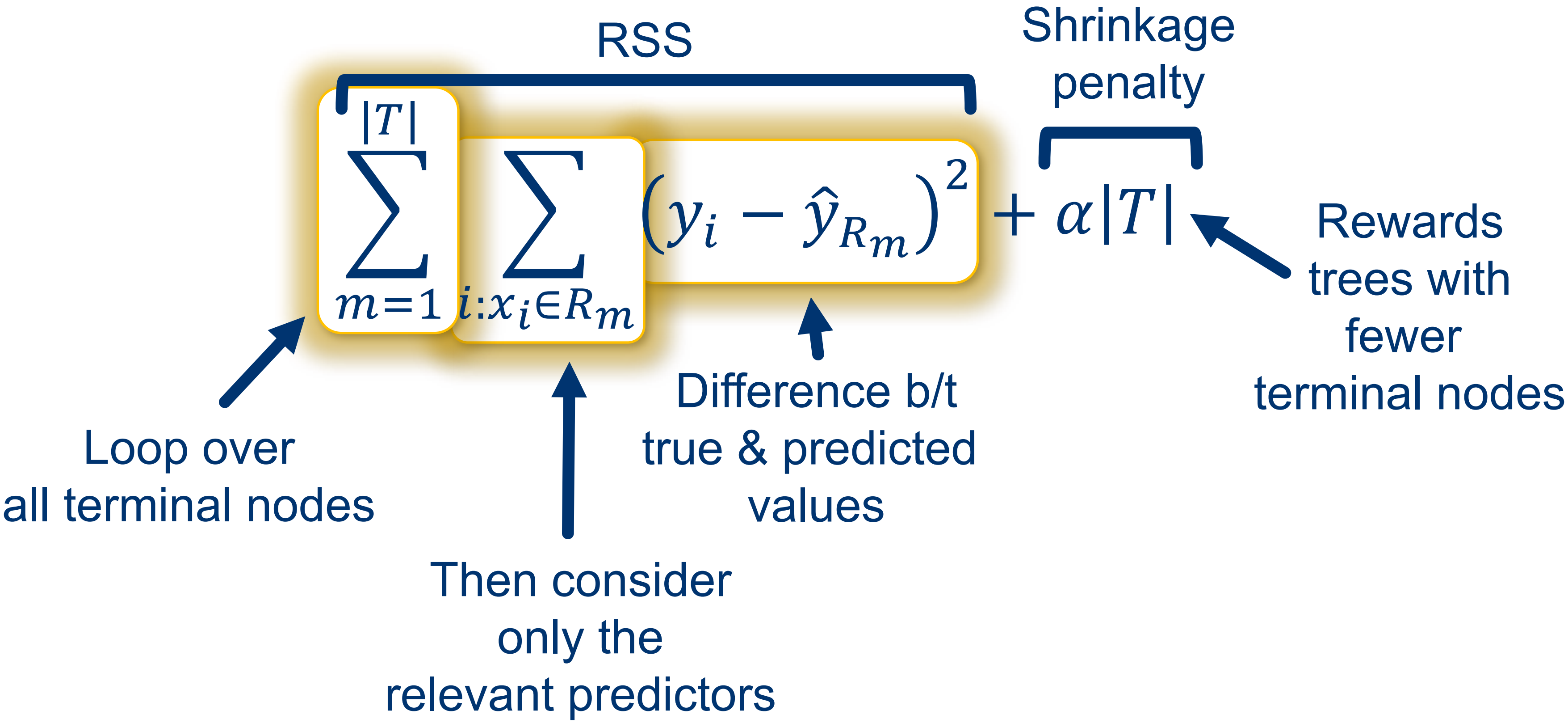
Flashback: the lasso

- **Big idea**: minimize RSS plus an additional penalty that rewards small (sum of) **coefficient values**

$$\overbrace{\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)}^{\text{RSS}}+\underset{\substack{\nearrow\\ \text{Tuning}\\ \text{parameter}}}{\lambda}\overbrace{\sum_{j=1}^{p}\beta_j^2}^{\substack{\text{Shrinkage}\\ \text{penalty}}}$$

Cost complexity pruning

- **Big idea**: minimize RSS plus an additional penalty that rewards small **trees**



$$\overbrace{\sum_{m=1}^{|T|}\sum_{i:x_i\in R_m}\left(y_i-\hat{y}_{R_m}\right)^2}^{\text{RSS}}+\overbrace{\alpha|T|}^{\text{Shrinkage penalty}}$$

Loop over all terminal nodes

Then consider only the relevant predictors

Difference b/t true & predicted values

Rewards trees with fewer terminal nodes

Cost complexity pruning

- **Big idea**: minimize RSS plus an additional penalty that rewards small **trees**

$$\overbrace{\sum_{m=1}^{|T|}\sum_{i:x_i\in R_m}\left(y_i-\hat{y}_{R_m}\right)^2}^{\text{RSS}}+\overbrace{\boxed{\alpha}|T|}^{\text{Shrinkage penalty}}$$

- **Fun fact:** as we increase α, branches get pruned in a nice, predictable (nested) fashion (why is this useful?)

Tree variation of backward selection

Start by growing some big tree on the training data

1. Use cost complexity pruning to get a sequence of "best subtrees" (as a function of α)
2. Select a single "best" α using cross-validated prediction error or something similar
3. Return the associated tree

Discussion

- The minimization we just saw would help us find the best **regression** tree, but our example was about **classification**

- **Question**: what needs to change?

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} \boxed{\left(y_i - \hat{y}_{R_m}\right)^2} + \alpha|T|$$

- **Answer**: just like in previous classification settings, we can't use RSS

Trouble in paradise…

- Usual approach (minimizing classification error) isn't sensitive enough to **build** good trees
- Alternative 1: *Gini index of each node*

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- Alternative 2: *cross-entropy of each node*

$$D = \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

- Both are measures of **purity***

*small values → node contains mostly observations from the same class

Discussion

- **Question**: what advantages / disadvantages might decision trees have when compared to other methods?
- **Answer:**
  - ✓Easy to explain and interpret
  - ✓Don’t need dummy variables to handle qualitative predictors
  - ✓Decision trees may more closely mirror human decision-making
  - x With what we’ve seen with so far*, trees aren’t going to be as accurate as other methods we’ve discussed

*foreshadowing Wednesday!

Coming up

- ✓ Final project activity: important questions
- ✓ Basic mechanics of tree-based methods
  - ✓ Classification example
  - ✓ Choosing good splits
  - ✓ Pruning
- How to avoid over-fitting
  - Bootstrap aggregation (“bagging”)
  - Random forests
  - Boosting
- Lab