Classification

Daniel Oberski

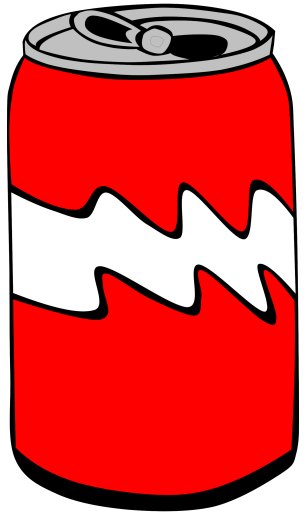
d.l.oberski@uu.nl

Overview of classification, with a view to fraud detection

Styrofoam cup

Potato peels

Egg box thingy



Soda can

How to classify trash

1. **Rule-based**: You have enough experience with trash to know in which bin it's supposed to go…

   … or do you??

2. **Learning-based**:
   a) "**No supervision**": You simply think about all the objects you might encounter and invent some kind of taxonomy (hard!);
   b) "**Supervision**": You stand next to bins and observe objects being thrown away, until you can do it yourself.

Common to all solutions

- You could be wrong;
- The consequences of being wrong are not the same;
- Each can be done in an “infinite” number of ways;
- It is not obvious what is “best”.

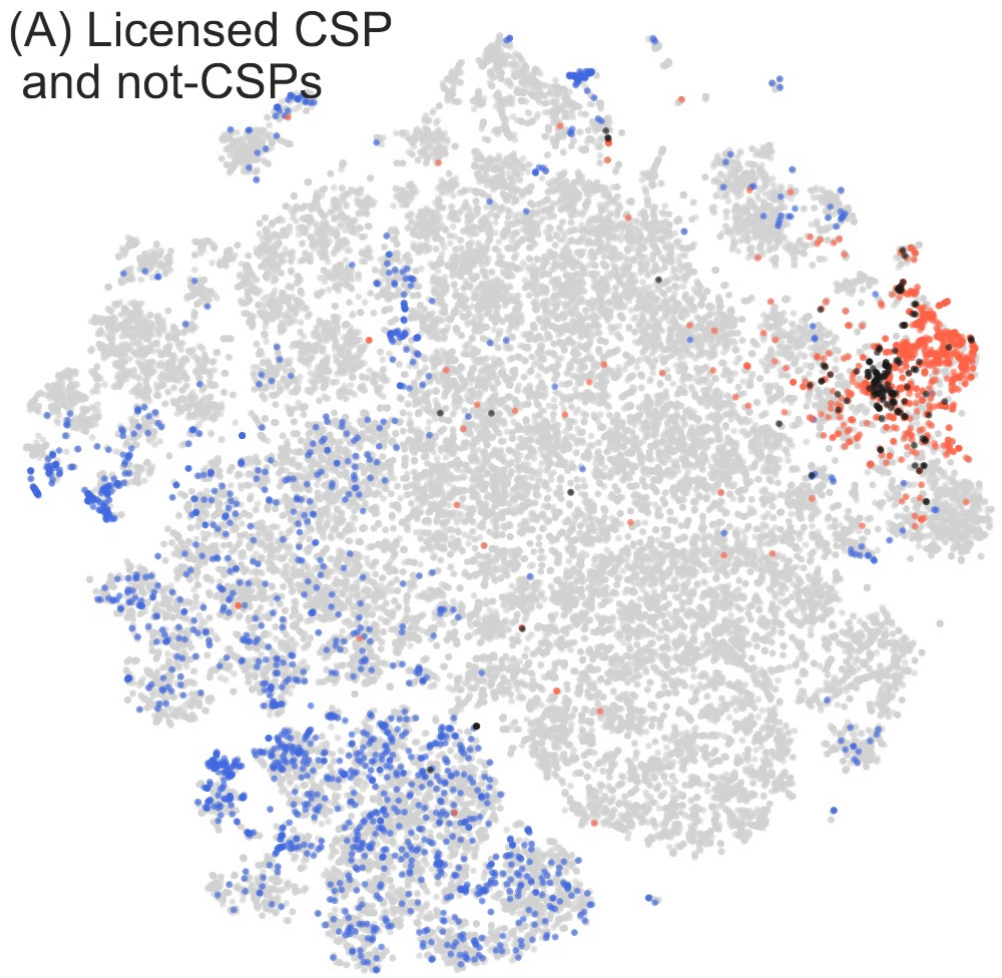
Classification in fraud detection

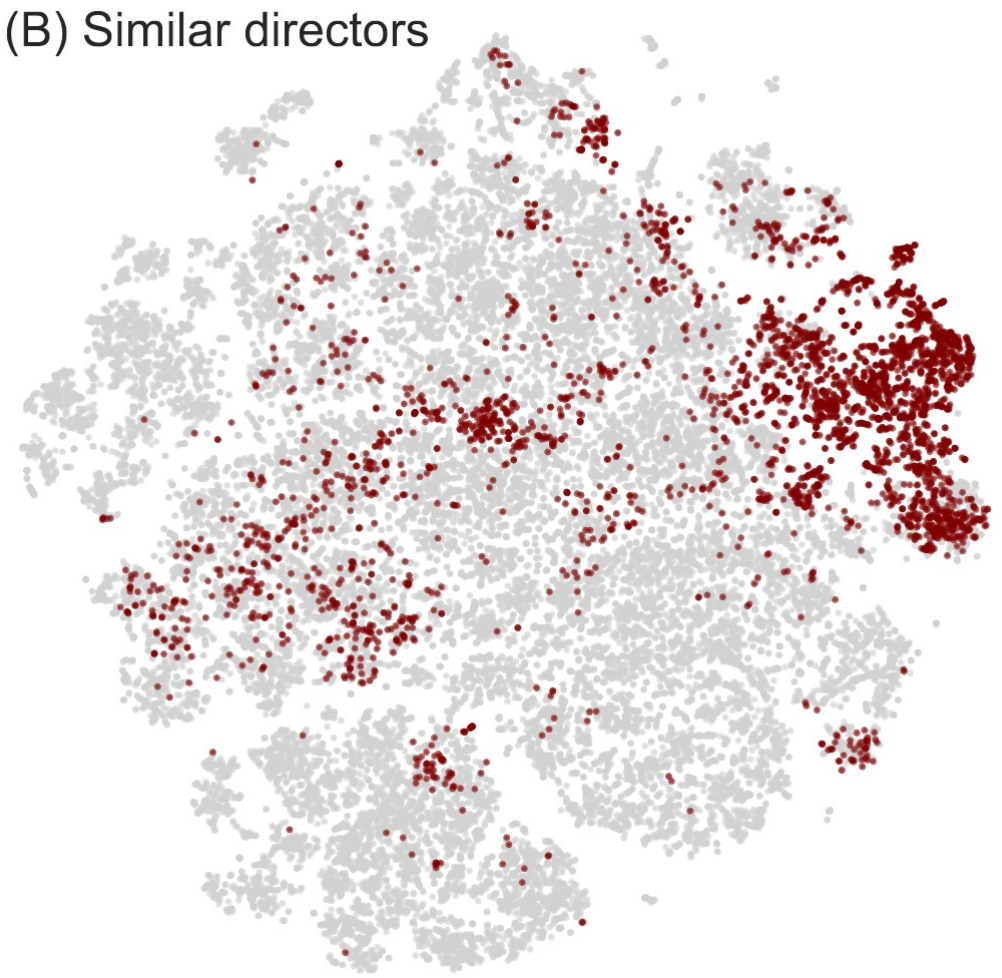
- **Expert rule-based**
  - *Example*: `IF risky_area AND large_amount -> fraud`

- **Learning-based**:
  - ***Supervised***: learn from examples of fraud *and* non-fraud ←
  - ***Unsupervised***: learn what examples tend to look like in general then label ”outlying” cases
  - ***Semi-supervised***: learn what examples tend to look like in general; use this information to distinguish fraud from non-fraud

Unsupervised unlicensed CSP detection

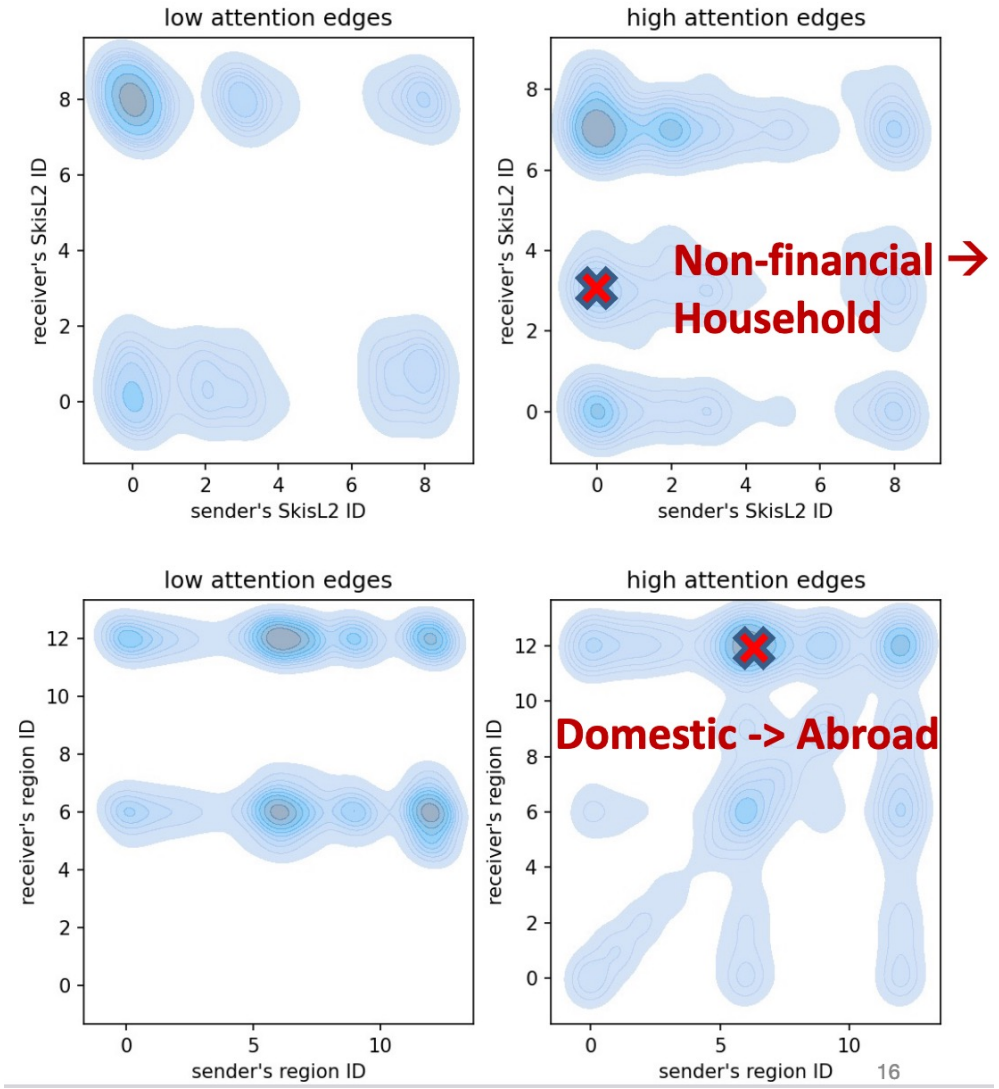
(Garcia-Bernardo et al. 2021) https://arxiv.org/pdf/2112.05019.pdf





Supervised fraud transactions

(Jiaxuan You, 2021)



Note that we are still learning “rules”, only the rules are not predetermined

Classification, the basic idea

Types of machine learning

We often distinguish **3 types** of machine learning:

- **Supervised Learning**: learn a model from labeled training data, then make predictions
- **Unsupervised Learning**: explore the structure of the data to extract meaningful information
- **Reinforcement Learning**: develop an agent that improves its performance based on interactions with the environment

SUPERVISED LEARNING: training phase

Data-Generating Process
"DGP"

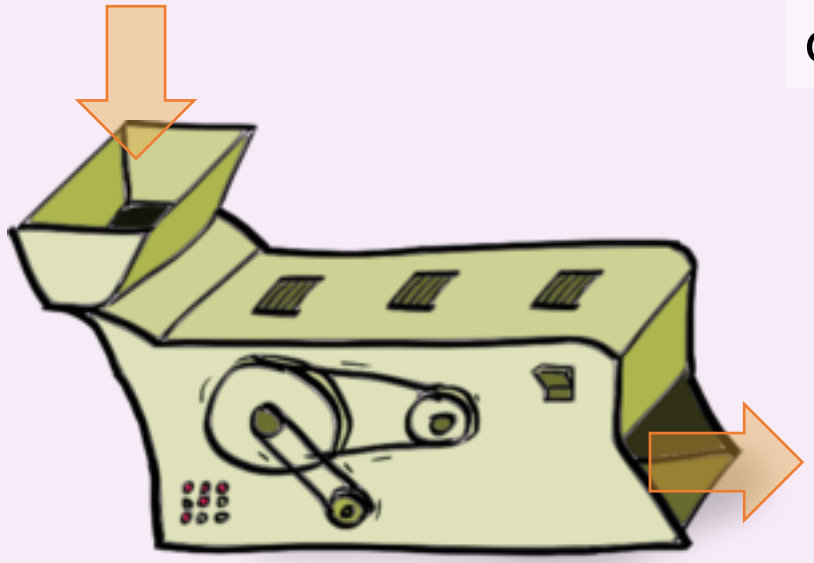
Training data

Features:
Transaction characteristics

Labels:
Fraud/not

Learning algorithm
(aka: "estimator", "model", "learner")

Input:
Features of *any* datapoint



Model:
A rule that depends on the training data

Output:
Guess "fraud" or "not"

SUPERVISED LEARNING: testing phase

Data-Generating Process
"DGP"

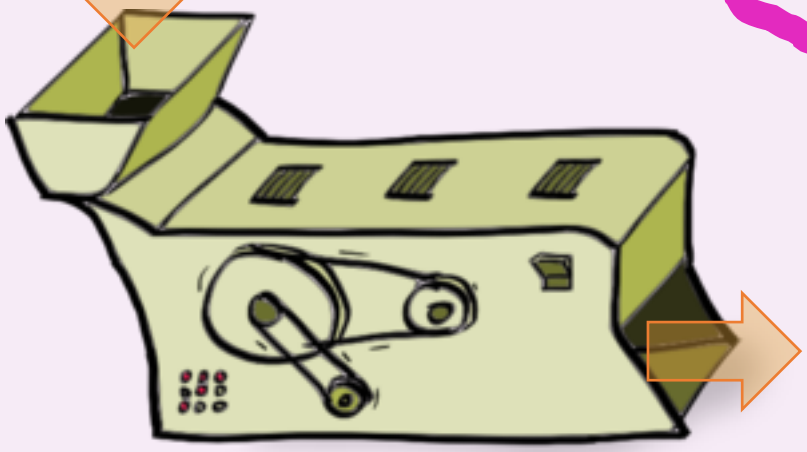
NEW!!

Test data

Features:
Transaction characteristics

Labels:
Fraud/not

Input:
Features of *any* datapoint



Model:
A rule that depends on the training data

SAME??

Output:
Guess "fraud" or "not"

Popular approaches to supervised classification

- Logistic regression (Peter van der Heijden's lecture)
- Random forests
- Boosting
- Support vector machines (SVM)
- Neural networks ("deep learning"), for example:
  - Graph neural networks (GNNs)
  - Recurrent neural networks (RNNs)
  - Convolution neural networks (CNNs)
  - … etc.

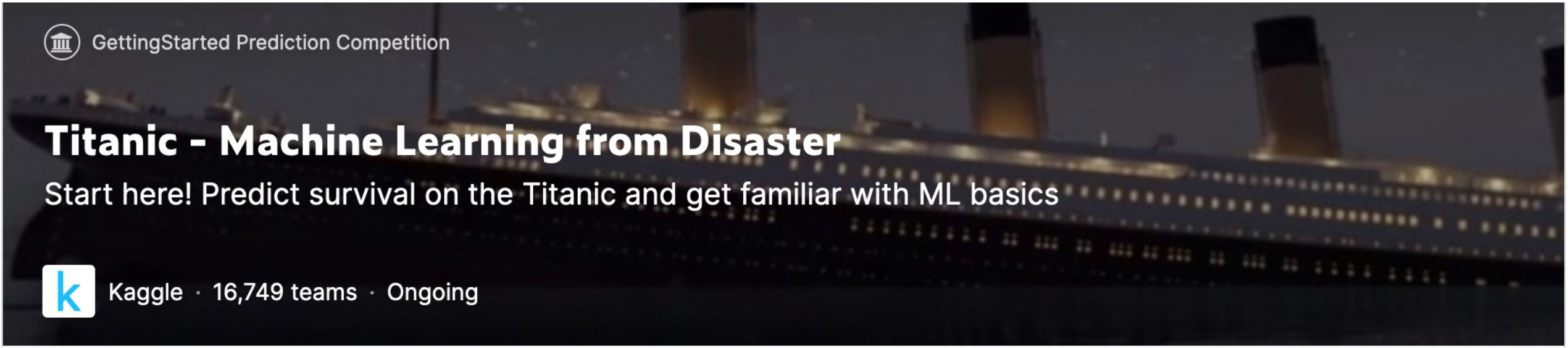
The plan

1. We will look a bit at the **training** part;

2. We will look a bit at the **testing**/evaluation part;

3. We may discuss other key issues, according to your interest:
   - Fairness;
   - Measurement error in labels;
   - “Runaway feedback loops”;
   - (Lack of) industry-standard auditing frameworks.

Classification trees



Overview

👋🚢 Ahoy, welcome to Kaggle! You're in the right place.

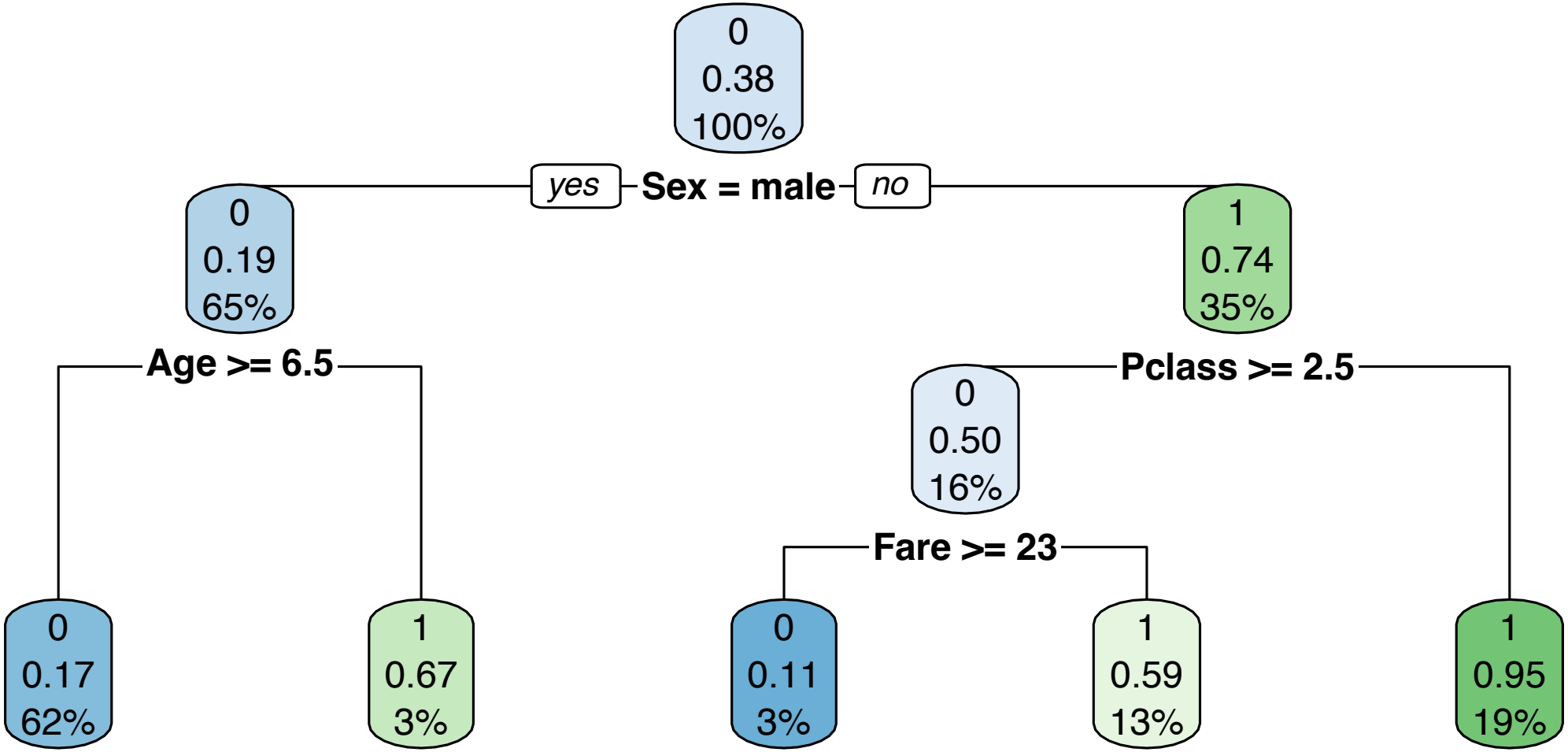
This is the legendary Titanic ML competition – the best, first challenge for you to dive into ML competitions and familiarize yourself with how the Kaggle platform works.

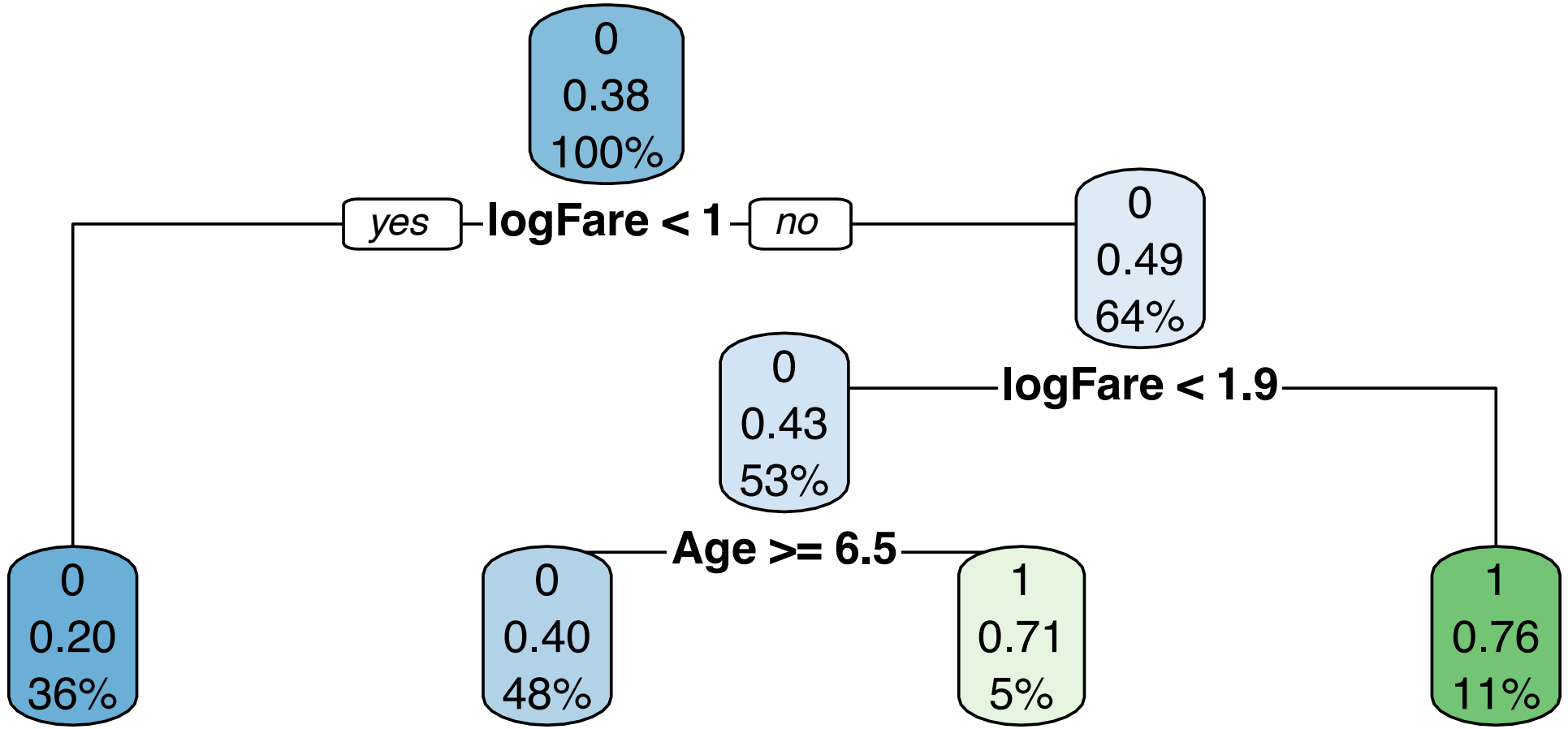
The competition is simple: use machine learning to create a model that predicts which passengers survived the Titanic shipwreck.

Titanic data

```
df = pd.read_csv('assets/train.csv')
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S


0
0.38
100%
yes
Sex = male
no
0
0.19
65%
1
0.74
35%
Age >= 6.5
Pclass >= 2.5
0
0.17
62%
1
0.67
3%
0
0.50
16%
Fare >= 23
0
0.11
3%
1
0.59
13%
1
0.95
19%


0
0.38
100%
yes
logFare < 1
no
0
0.49
64%
0
0.43
53%
logFare < 1.9
0
0.20
36%
0
0.40
48%
Age >= 6.5
1
0.71
5%
1
0.76
11%

Learning algorithm

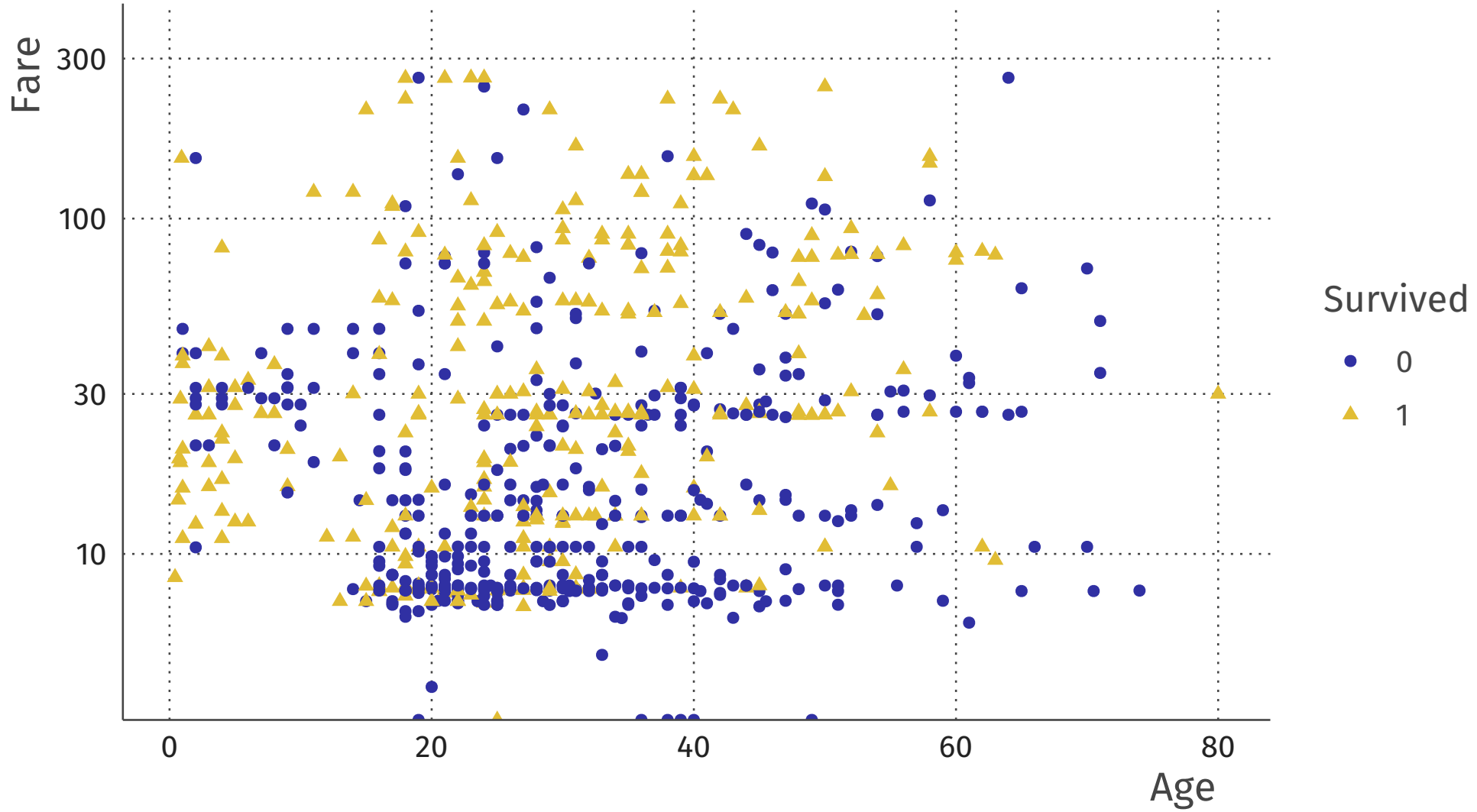
Recursive partitioning

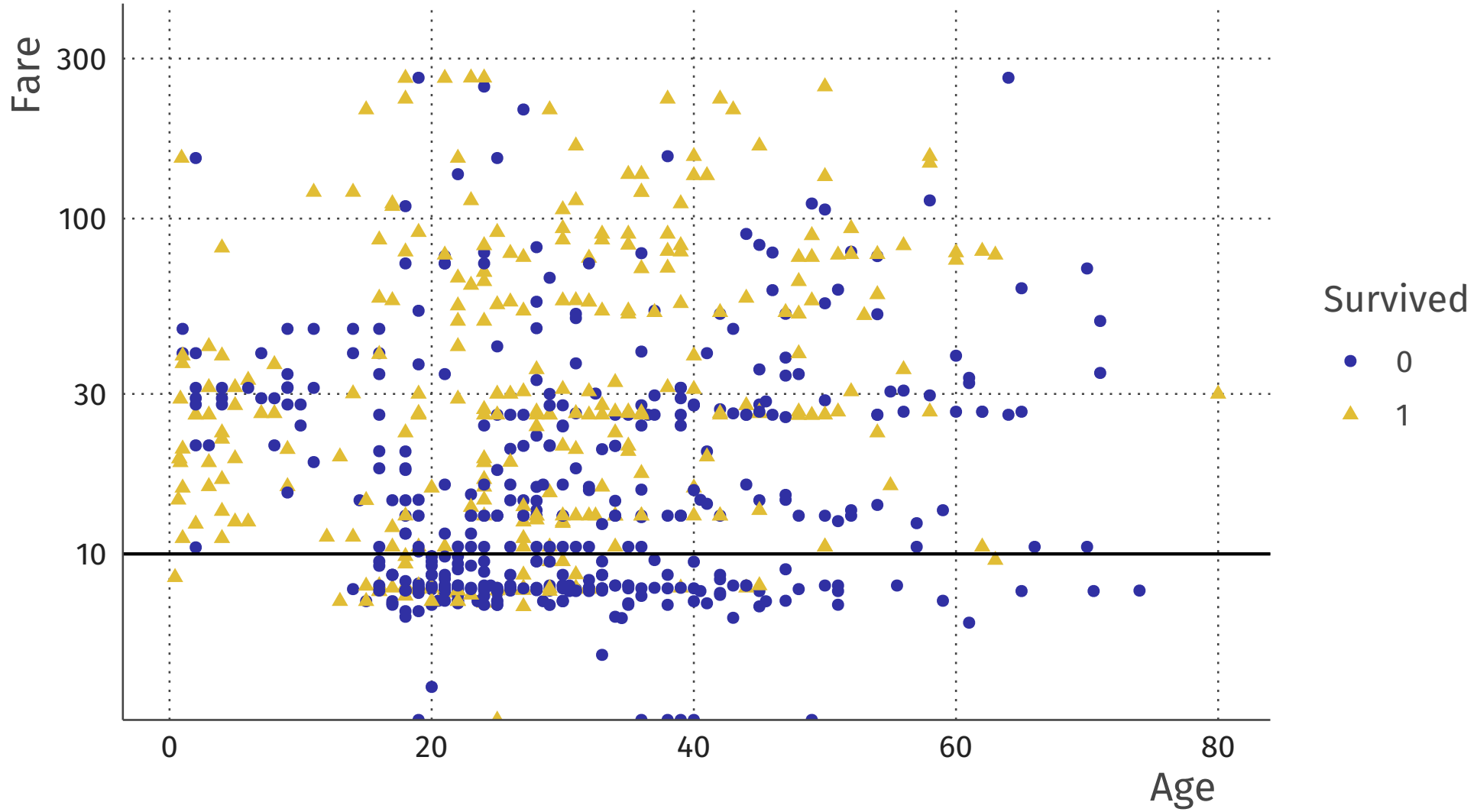
1. Find the split that makes observations as similar as possible on the outcome within that split;
2. Within each resulting group, do (1).

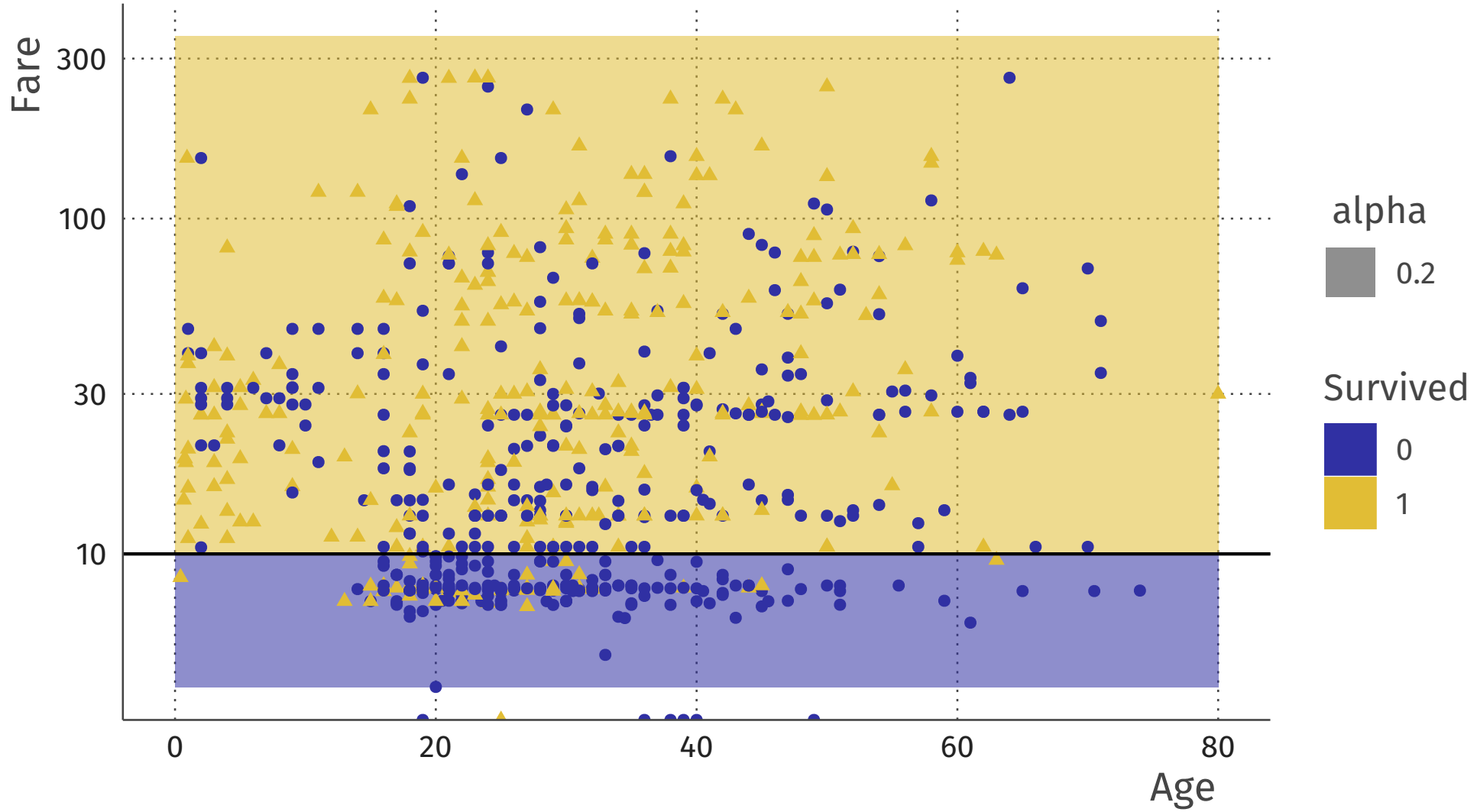
Recursive partitioning

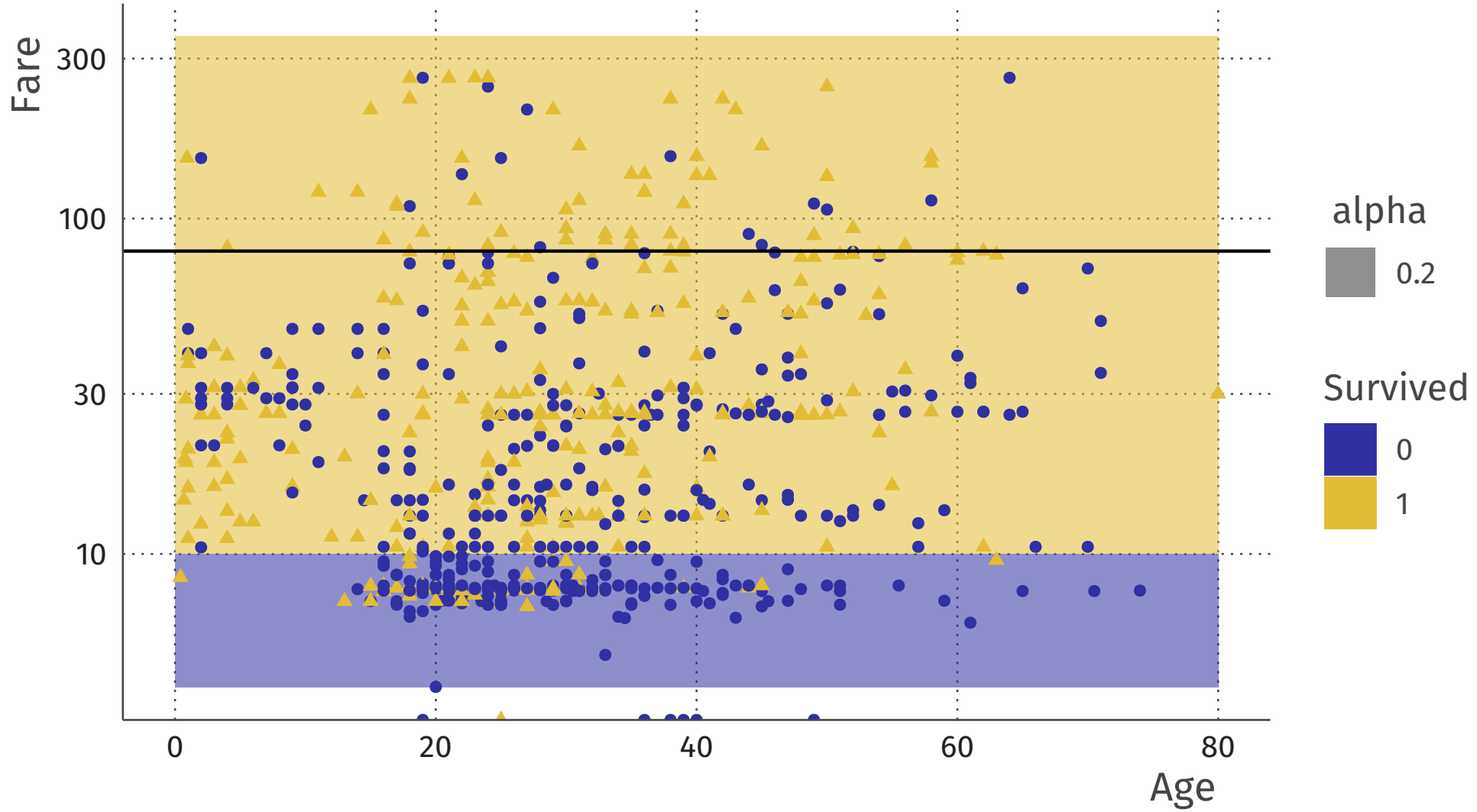
1. Find the split that makes observations as similar as possible on the outcome within that split;
2. Within each resulting group, do (1).

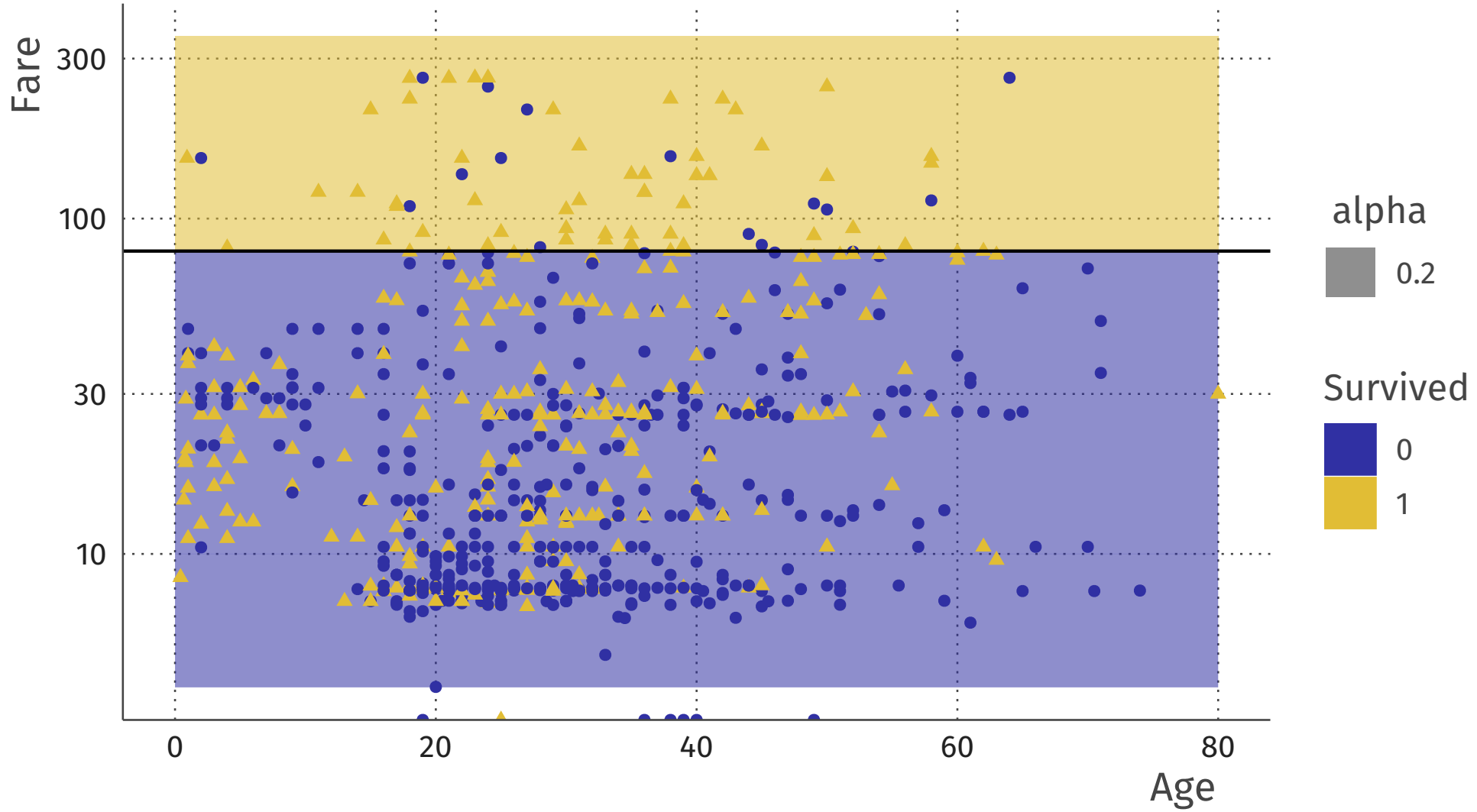
- **Criteria** for “as similar as possible”: Purity, MSE reduction, ...
- **Early stopping**: add after (2):
  - “unless fewer than `nmin` observations in the group” (typically 10);
  - “unless improvement less than `cp`” (typically 0.05);

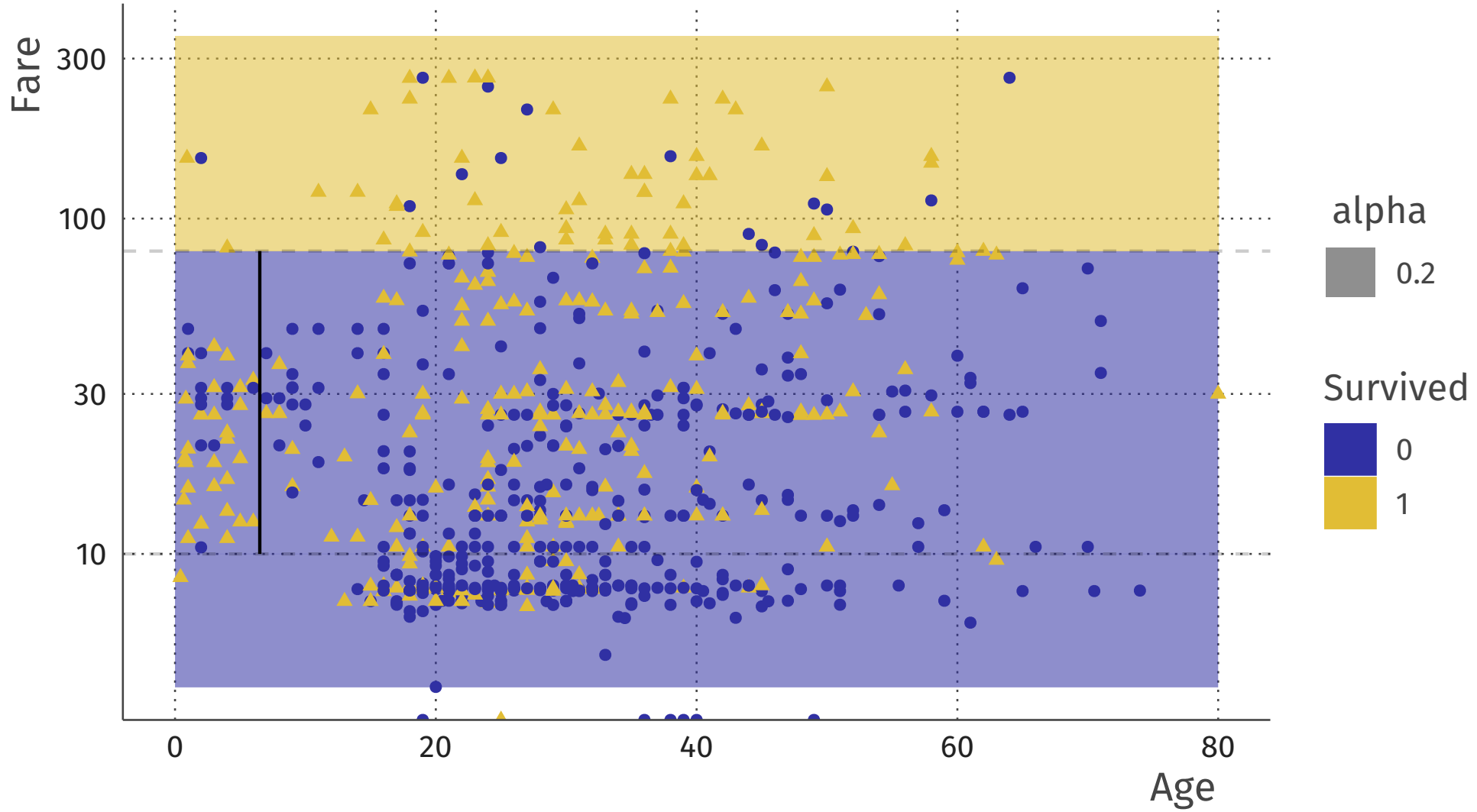

Fare
300
100
30
10
0
20
40
60
80
Age
Survived
0
1

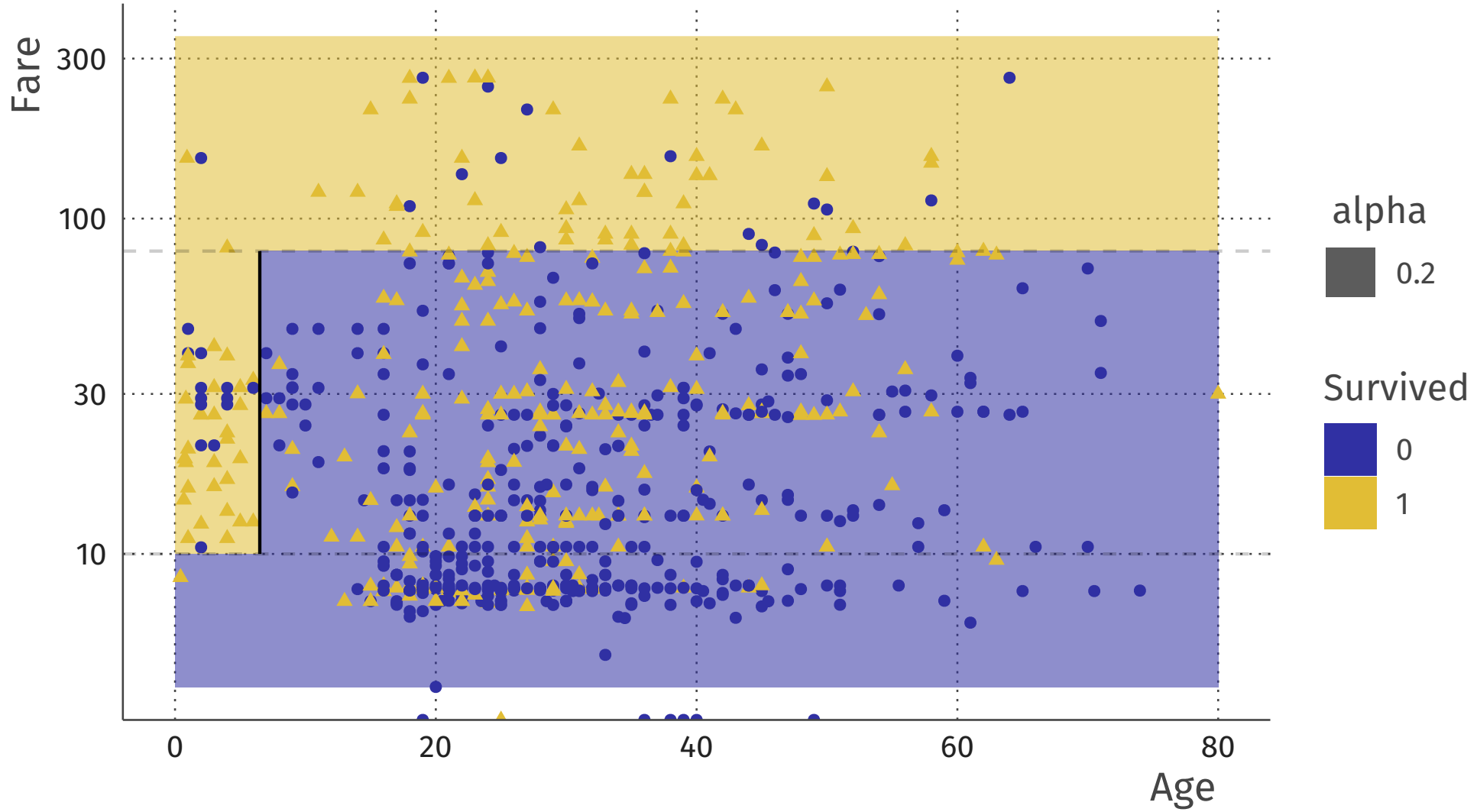

Fare
300
100
30
10
0
20
40
60
80
Age
Survived
0
1


Fare
300
100
30
10
0
20
40
60
80
Age
alpha
0.2
Survived
0
1

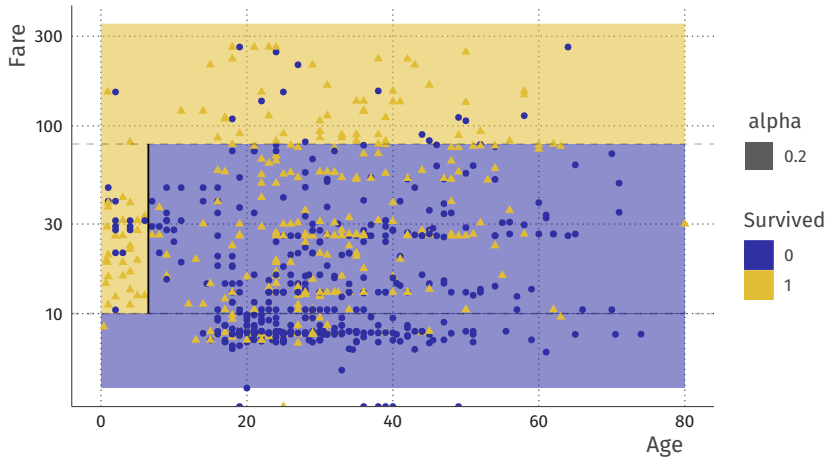

Fare
300
100
30
10
0
20
40
60
80
Age
alpha
0.2
Survived
0
1

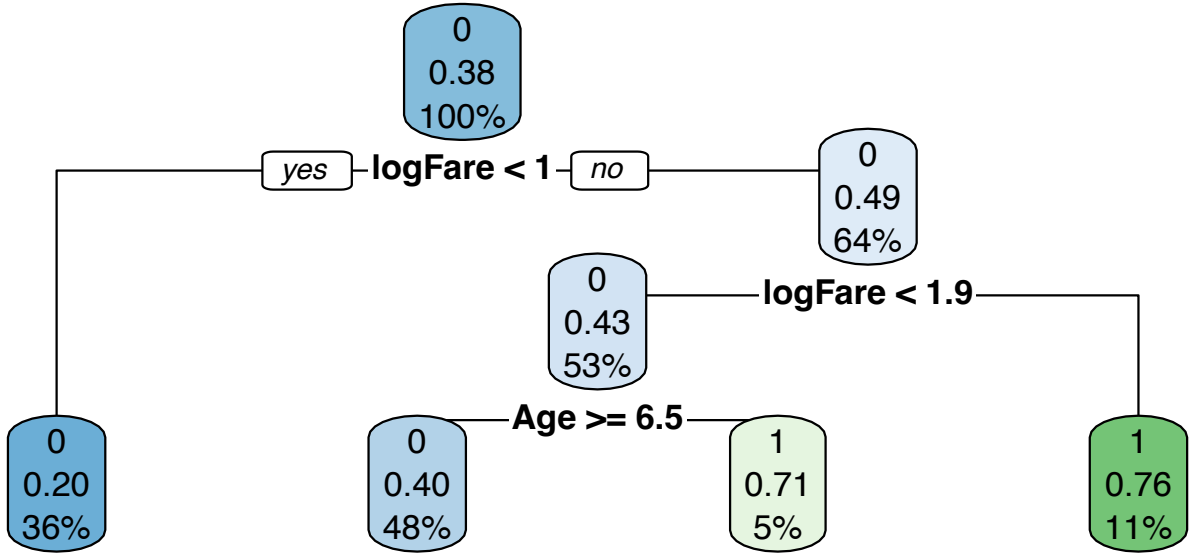

Fare
300
100
30
10
0
20
40
60
80
Age
alpha
0.2
Survived
0
1


Fare
300
100
30
10
0
20
40
60
80
Age
alpha
0.2
Survived
0
1


Fare
300
100
30
10
0
20
40
60
80
Age
alpha
0.2
Survived
0
1

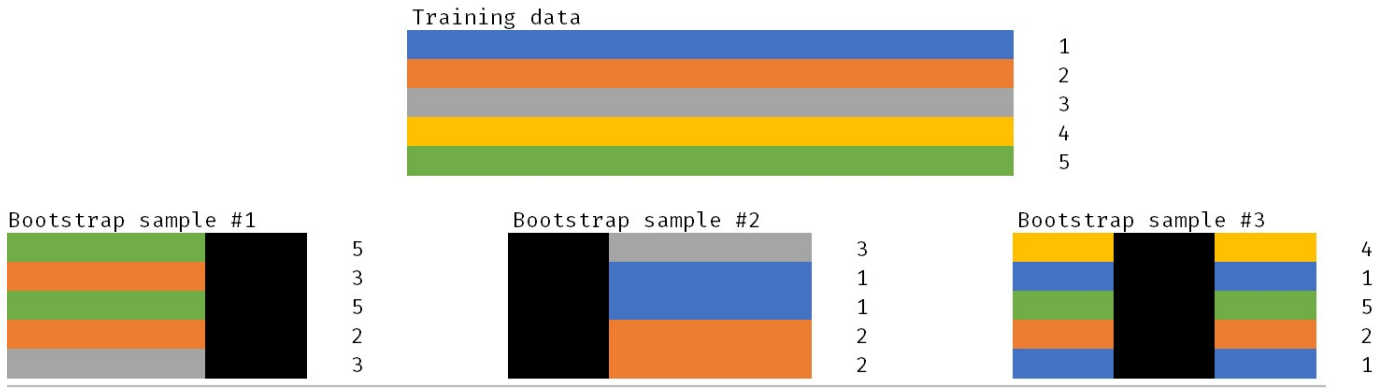
These are the same!

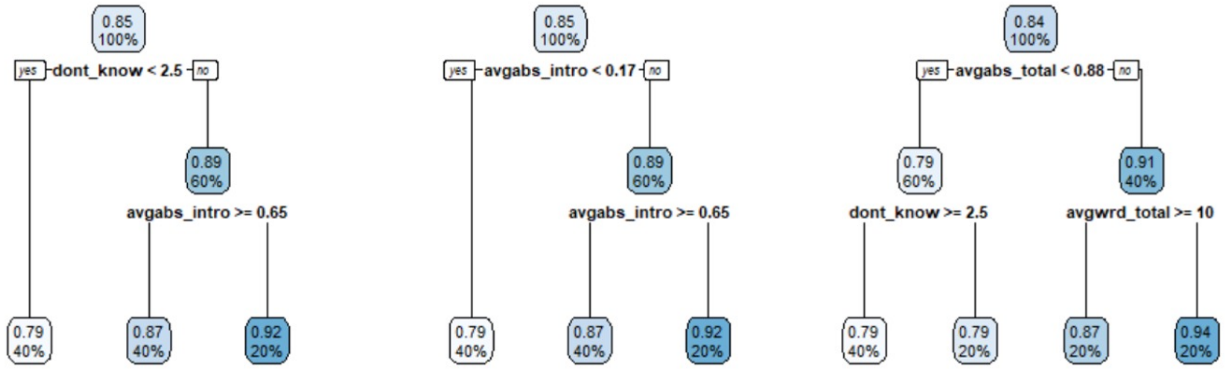




Ensemble learners based on trees

Random forests reminder





The problem with trees

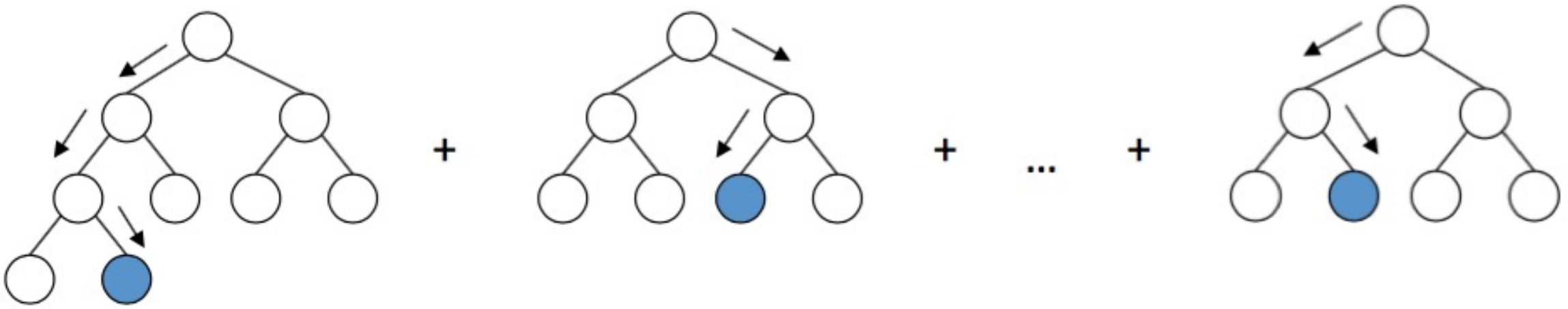
- Trees are not a bad idea, but in practice they tend to overfit
- Use them as **basic building block** for **ensembles**

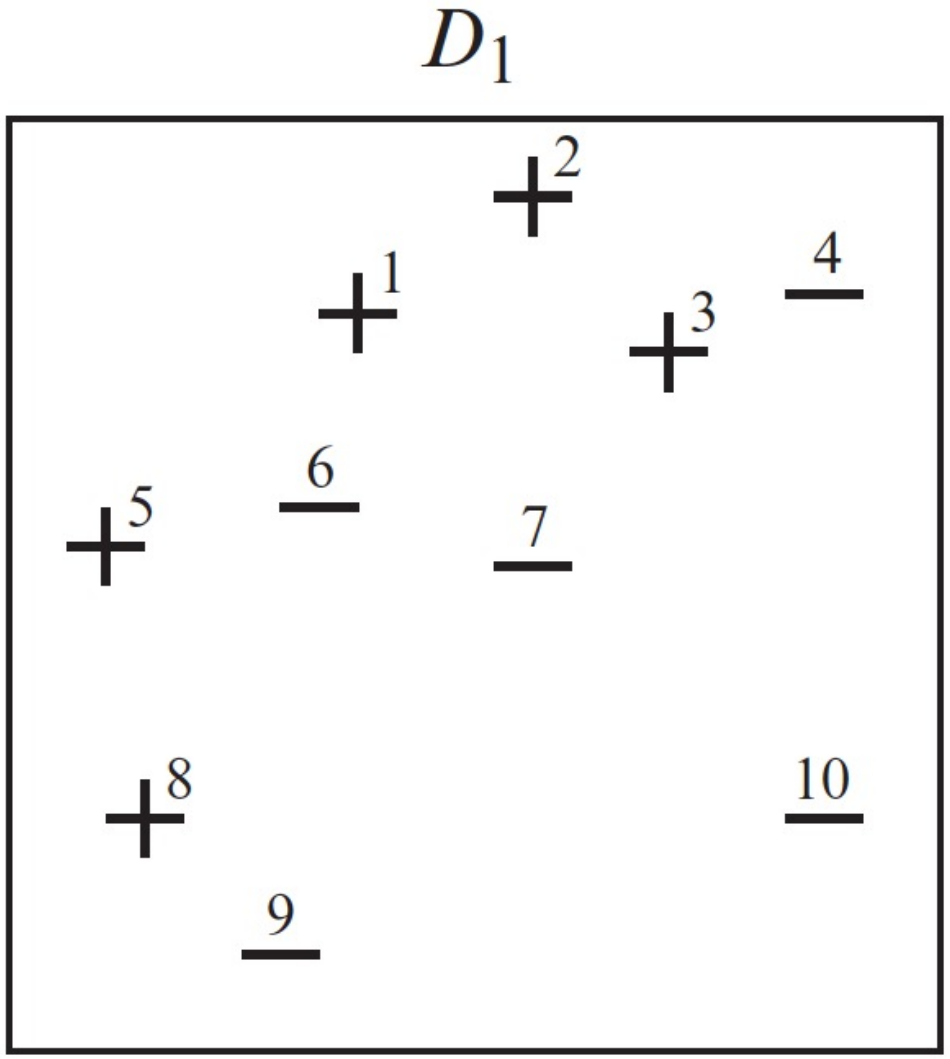
- **Random forests**: “bagged trees with feature sampling”
  - Make trees that are too complex (low bias, high variance);
  - Average over bootstrapped samples to cancel out the overfitting parts.

- **Boosting**: “ensemble of weak learners”
  - Make trees that are too simple (high bias, low variance);
  - Make more of them for observations with big residuals;
  - Average them.

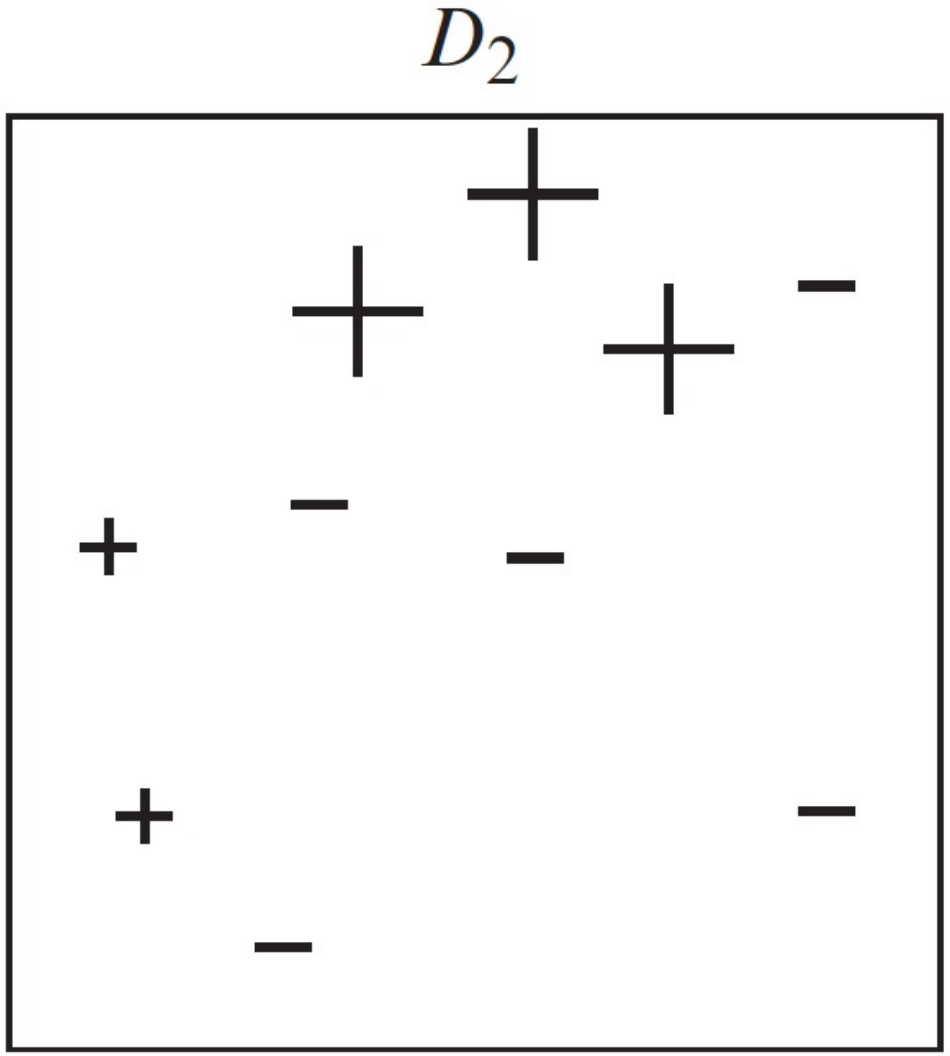
Boosting

- By combining many “weak learners”, a good model is created
- “Wisdom of the crowds”

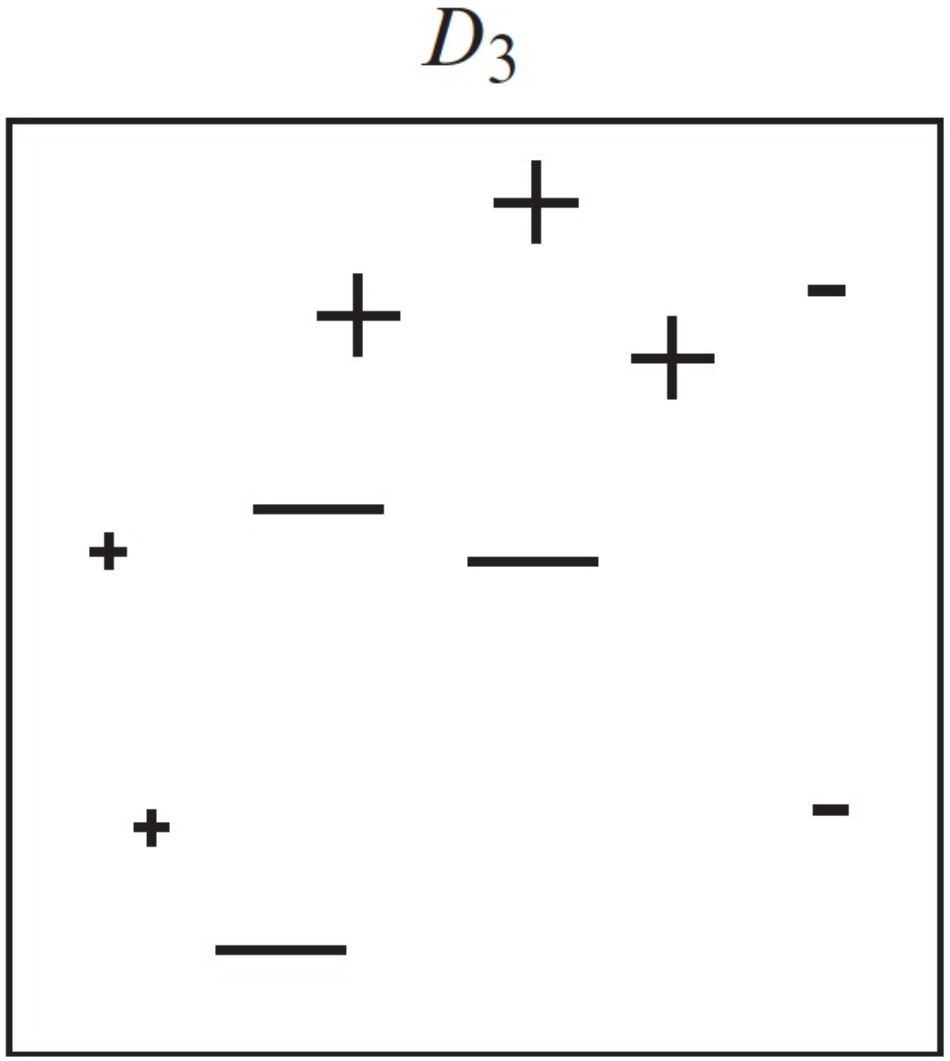




Source: Schapire & Freund (2012). *Boosting: Foundations and Algorithms.*



Source: Schapire & Freund (2012). *Boosting: Foundations and Algorithms.*



Source: Schapire & Freund (2012). *Boosting: Foundations and Algorithms.*

The final prediction model (classifier)

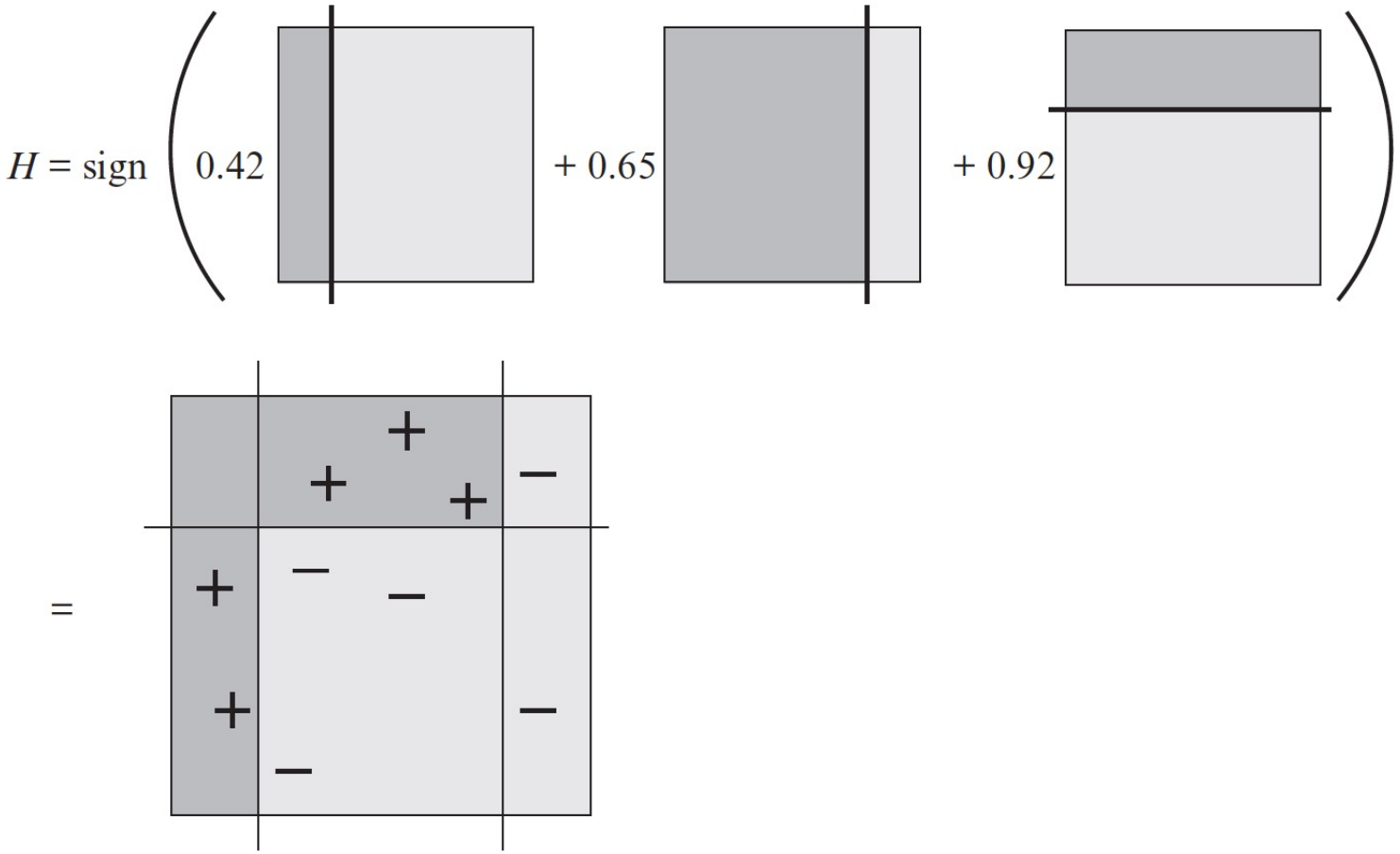
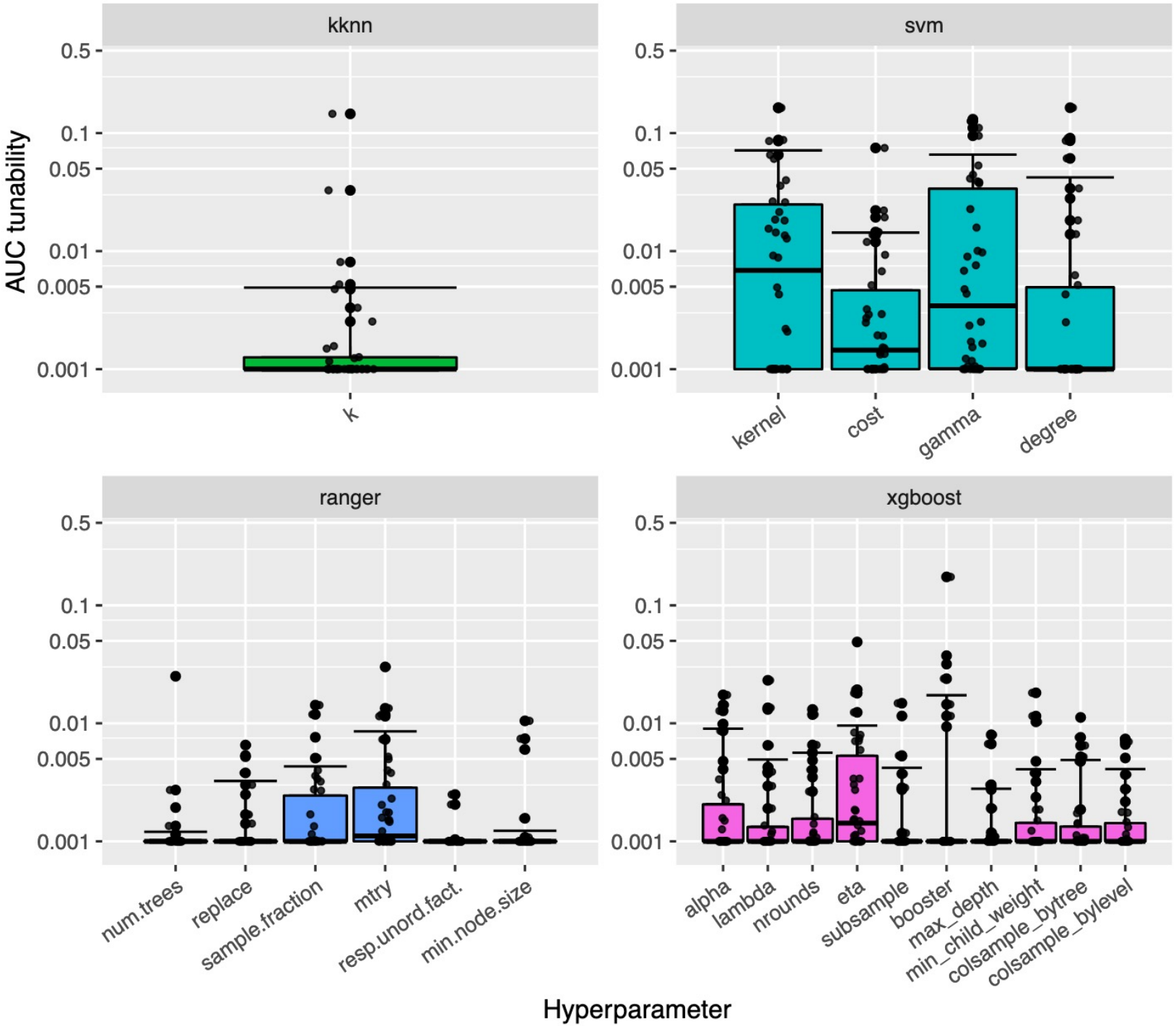


Figure 1.2
The combined classifier for the toy example of figure 1.1 is computed as the sign of the weighted sum of the three weak hypotheses, $\alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3$, as shown at the top. This is equivalent to the classifier shown at

Boosting

- Current go-to implementation is `xgboost`
- Very powerful idea and easy to apply to other things than classification trees
- Regression boosting, Survival boosting, etc. etc.
- Often SotA in tabular data challenges, …
- … *after* extensive hyperparameter tuning



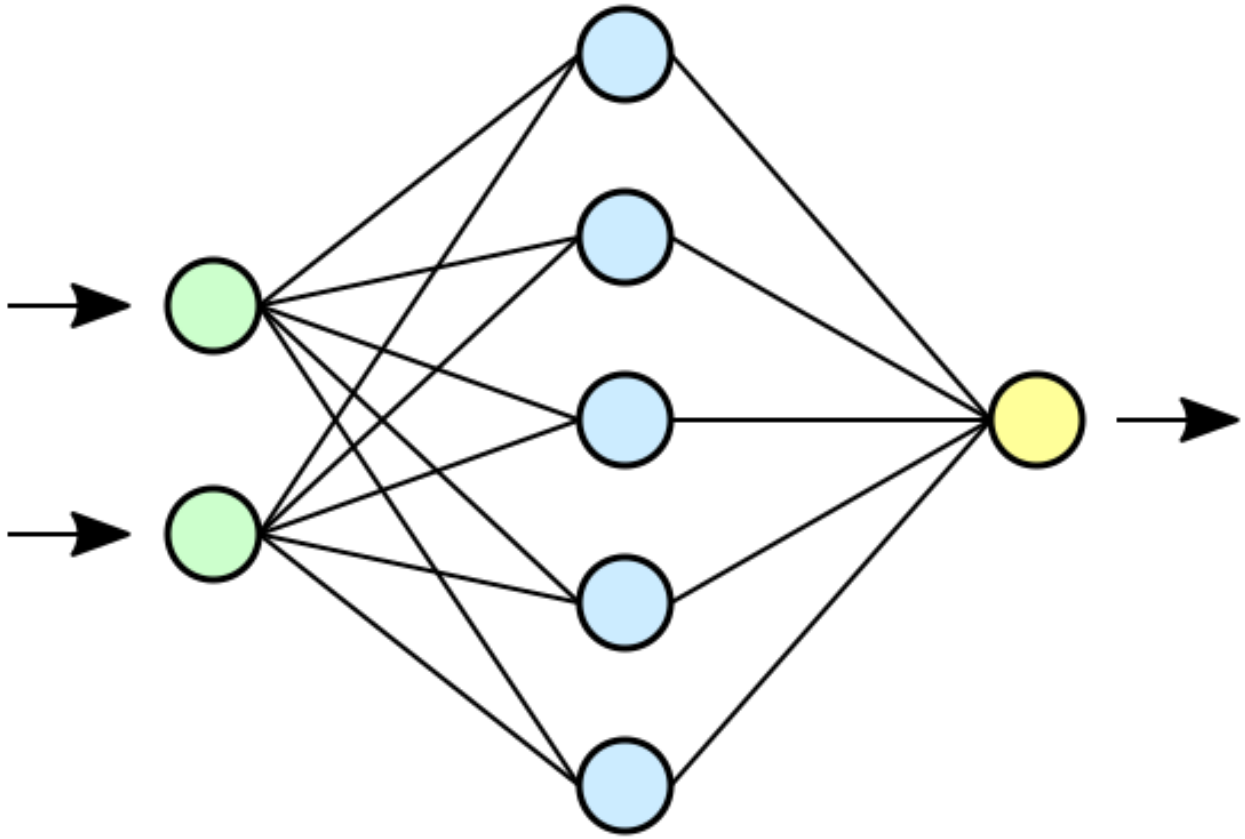
Probst et al. (2019). JMLR

Neural networks

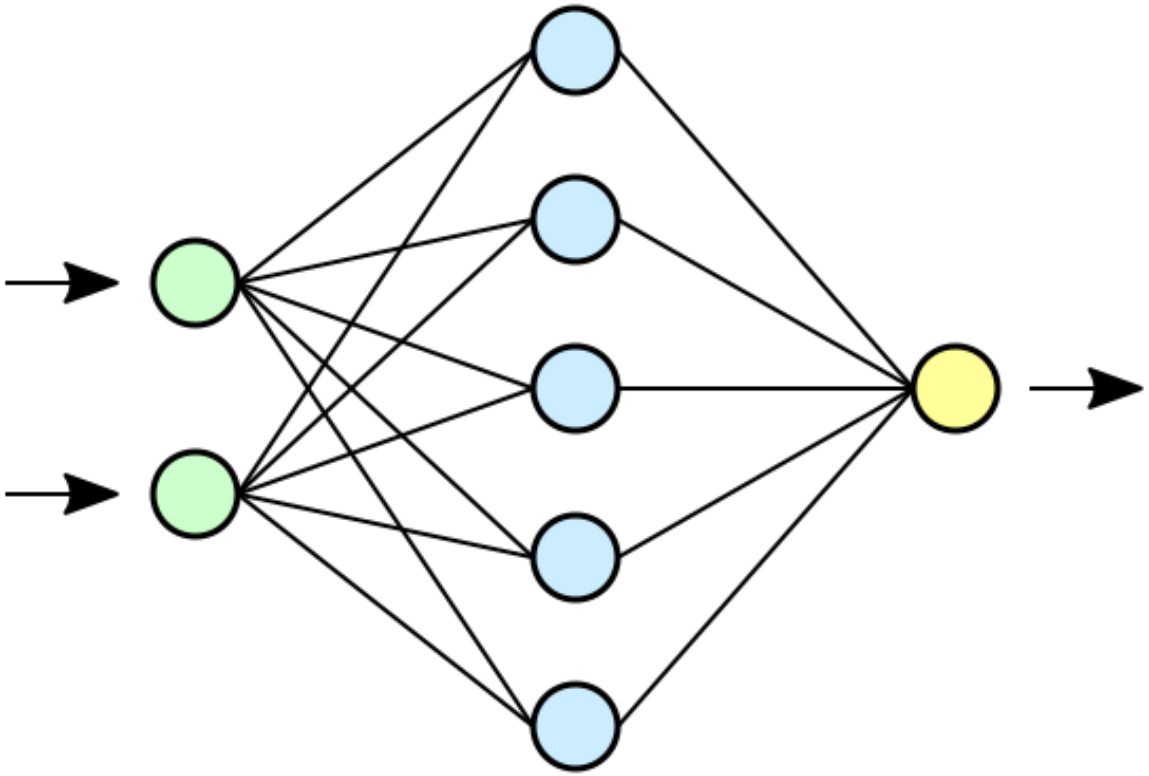
Neural networks, “deep learning”

- Compositional approach to curve-fitting;
- “Biologically inspired”
  (but don’t take that too seriously);
- State-of-the-art in some domains (see https://paperswithcode.com/sota)
- Sound cool.

Neural network



Neural network



"Hidden" nodes:

Example:

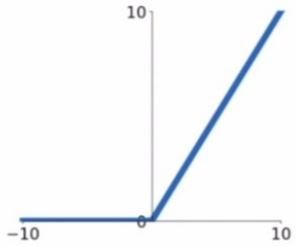
$$h_1 = f(w_{11}x_1 + w_{12}x_2)$$

Output:

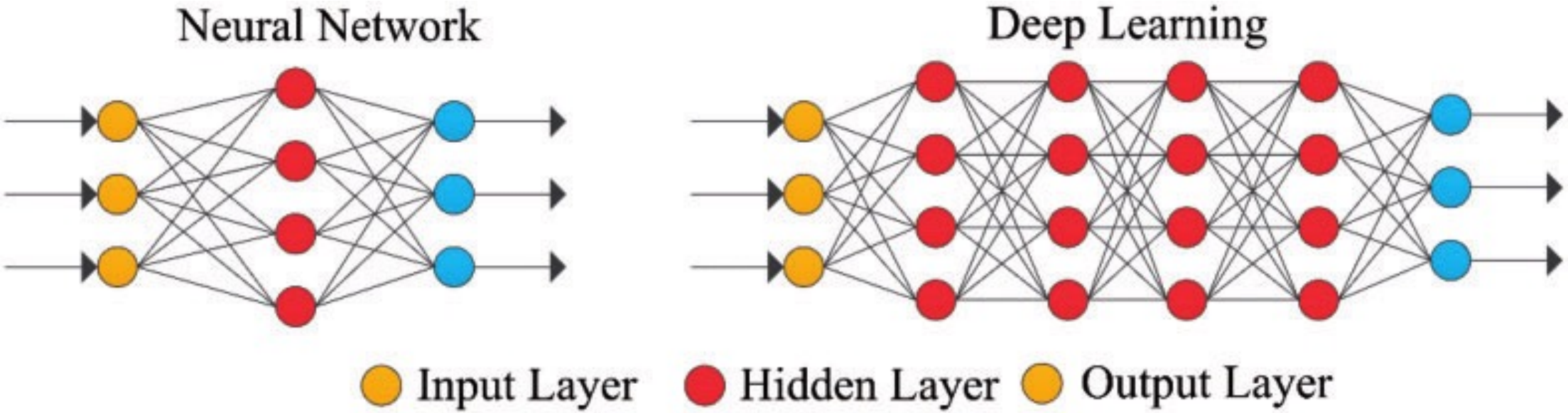
$$y = f(w_{21}h_1 + w_{22}h_2 + \cdots + w_{25}h_5)$$

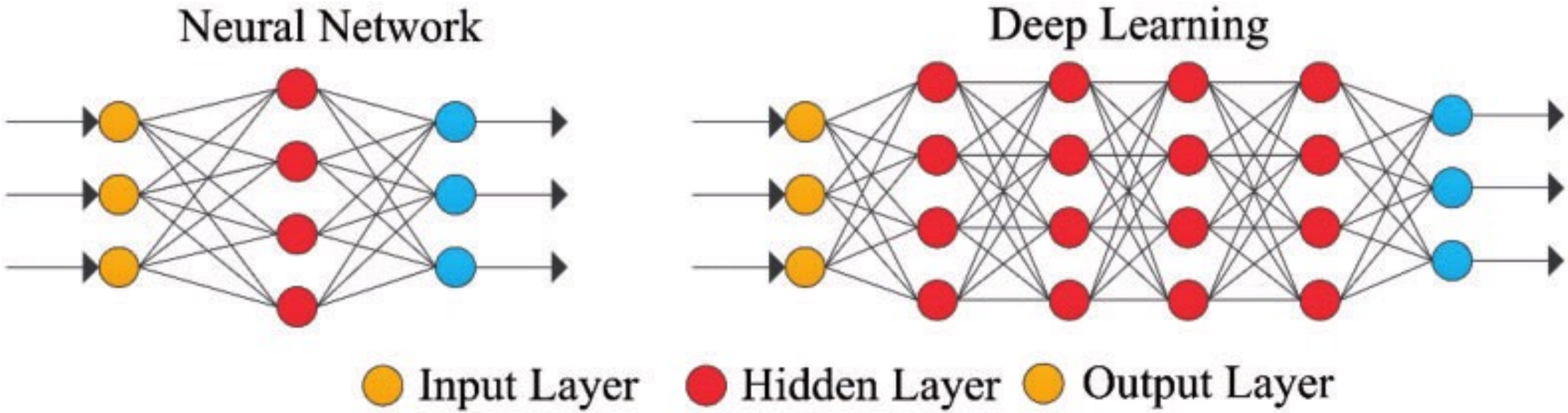
"Activation function":

- RElu $f(z) =$
- …



What makes a neural net “deep”?





Keep doing

$$z = g^{(n_h)}(g^{(\dots)}(g^{(2)}(g^{(1)}(\mathbf{x}))))$$

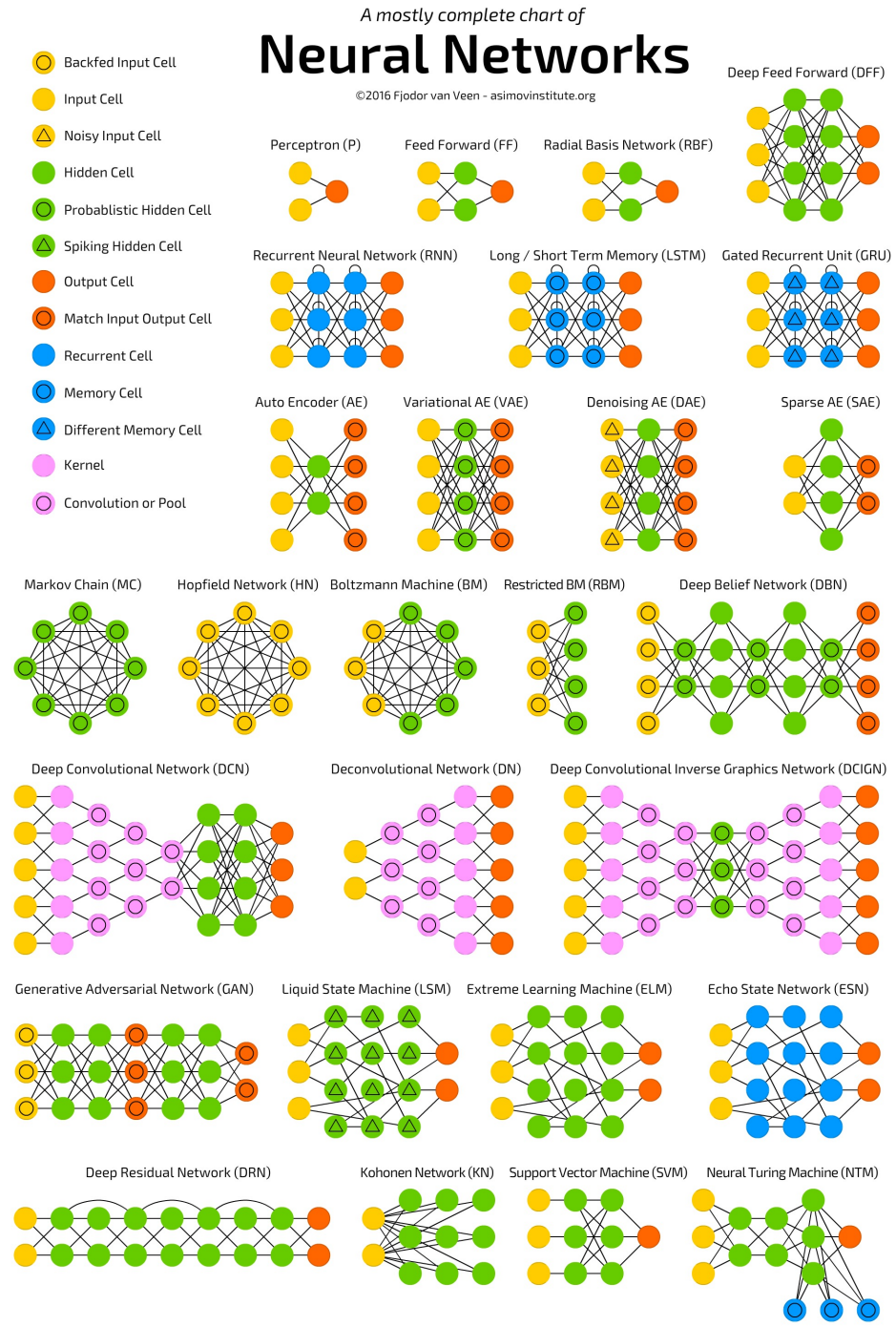
then $y \approx f(z)$.

Deep learning

- Output of each hidden layer is input to subsequent one
- Allow representation learning by building complex features out of simpler ones
- Go deep: exponential advantages, less overfitting
- Aggressive parameterization + aggressive regularization
- Compositional: efficient parametrization
- Learn relevant features: “End-to-end”

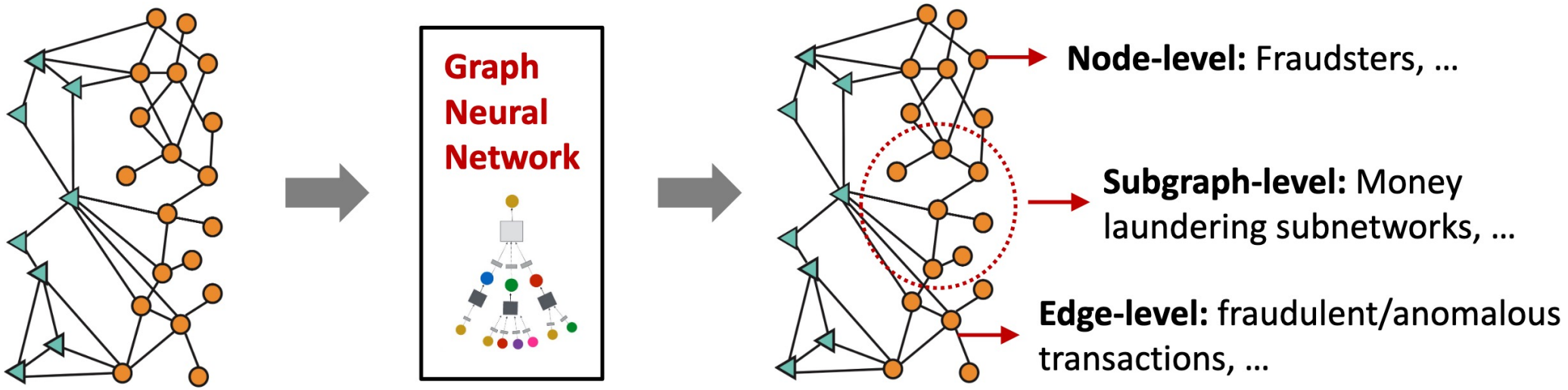
Different architectures

- By adjusting the arrows, layers, and activation functions, you can create models that are tailored to specific data, e.g.
- Convolutional (CNN): images, text, sound
- Recurrent (RNN): time series, text
- Graph (GNN): networks
- …



Example with transaction data

(Jiaxuan You, 2021)



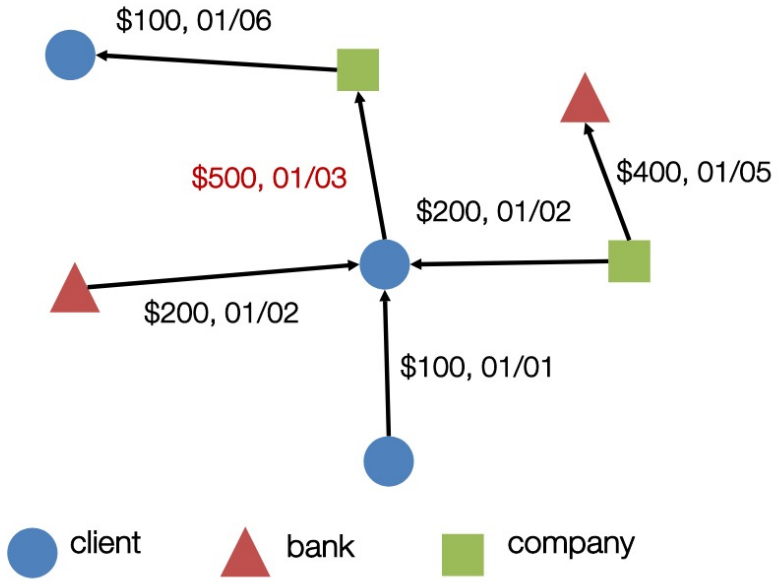
Input: Financial networks

Output: Predictions

Example with transaction data

(Jiaxuan You, 2021)

- Transaction-based approach
  - "On 01/03, Client A sends Company B $500"
  - Build models based on transaction attributes
  - **Issues:** ignore the context of a transaction
- **Graph-based approach**
  - Represent transactions as a **dynamic graph**
  - Predictions are made based on the entire graph
  - **Benefits:**
    - Represents a transaction with a broader context
    - Requires fewer feature-engineering



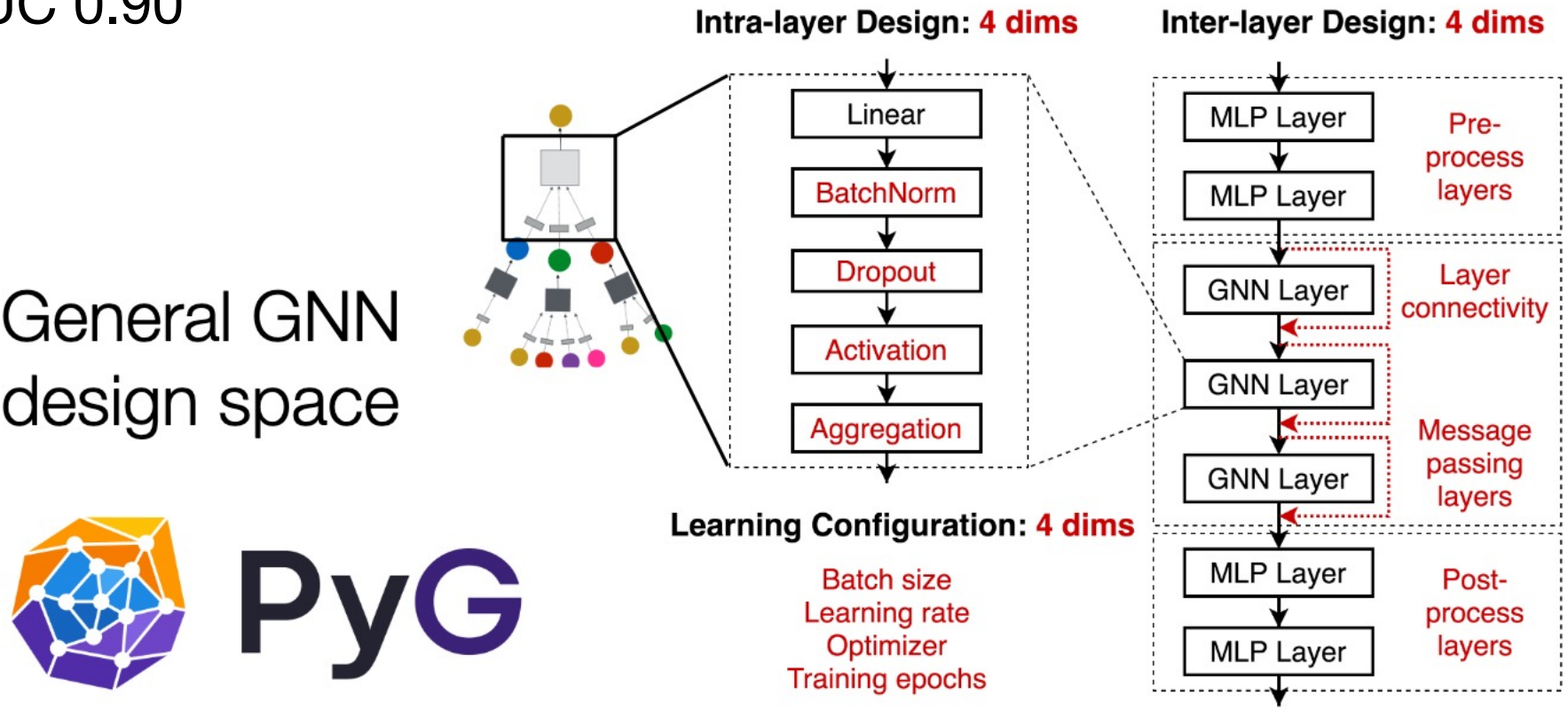
Example with transaction data

(Jiaxuan You, 2021)

Fraud transaction classification:

- Baseline: AUC 0.80
- SotA GNN: AUC 0.90

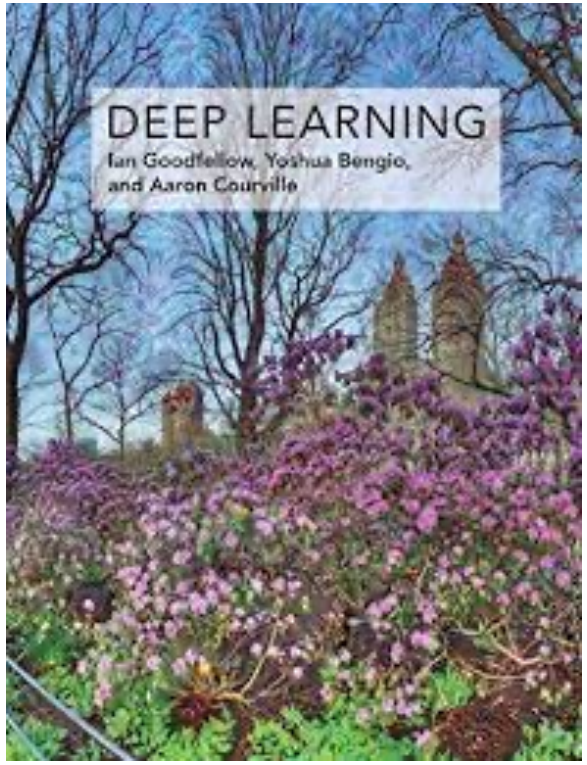
General GNN design space



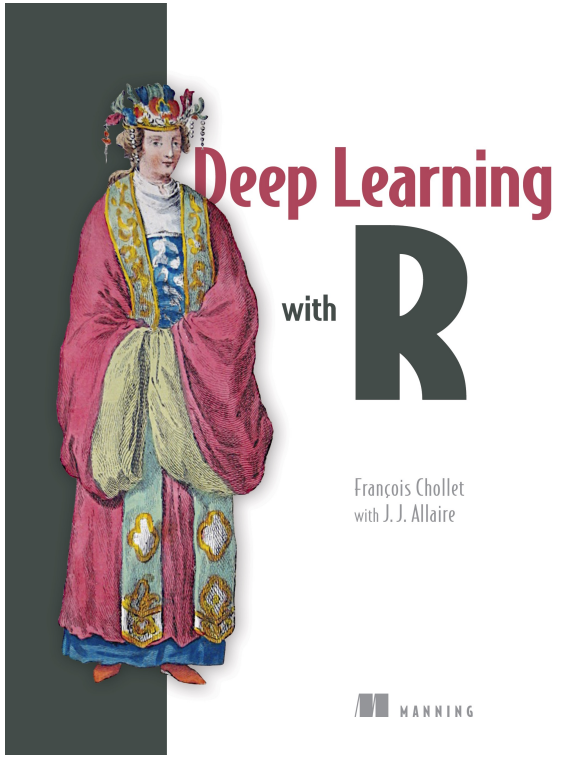
PyG

Deep learning in practice

- Good places to start:
  - https://keras.rstudio.com/
- ISLR Chapter 10



Goodfellow et al.



Chollet (R/Python version)

Evaluating classifiers

No free lunch

“Any two optimization algorithms are equivalent when their performance is averaged across all possible problems”

(Wolpert & MacReady)

Confusion matrix: Counts

```
> p_pred <- predict(titanic_tree, newdata = val_df)

> with(val_df, table(p_pred > 0.5, Survived))

       Survived
          0   1
  FALSE 134  40
  TRUE   19  75
```

Confusion matrix: counts

	Survived (observed)			
	No		Yes	
Survived (predicted)				
No	134	(TN)	40	(FN)
Yes	19	(FP)	75	(TP)

- False positives (FP): 19
- False negatives (FN): 40
- Total errors: FP + FN

Confusion matrix:
Sensitivity ("recall") and Specificity

```
> with(val_df, table(p_pred > 0.5, Survived)) %>% prop.table(2)
```

	Survived (observed)	
	No	Yes
Survived (predicted)		
No	0.876	0.348
Yes	0.124	0.652
TOTAL	1	1

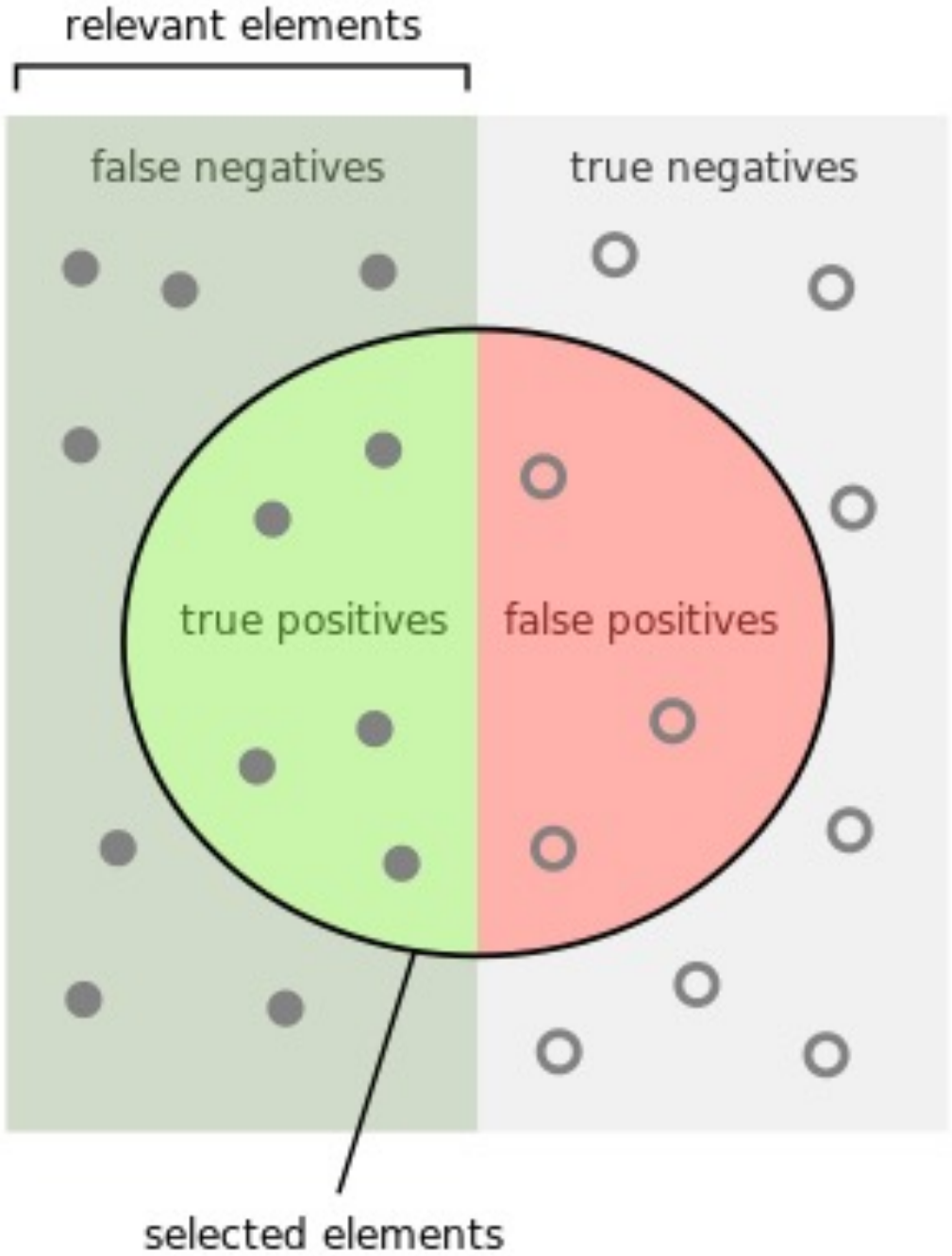
- Specificity: $\frac{\text{TN}}{\text{TN}+\text{FP}}$ = 134 / (134 + 19) ≈ 0.876
- Sensitivity ("recall"): $\frac{\text{TP}}{\text{TP}+\text{FN}}$ = 75 / (75 + 40) ≈ 0.652
- Accuracy (ACC): $\frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{TN}+\text{FN}} \approx 0.780$
- Error rate: $1 - \text{Accuracy} \approx 0.220$

Confusion matrix:
Negative & positive predictive value

```
> with(val_df, table(p_pred > 0.5, Survived)) %>% prop.table(1)
```

	Survived (observed) No	Yes	TOTAL
Survived (predicted)			
No	0.770	0.230	1
Yes	0.202	0.798	1

- NPV: $\frac{\text{TN}}{\text{TN}+\text{FN}}$ = 134 / (134 + 40) ≈ 0.770
- PPV ("precision"): $\frac{\text{TP}}{\text{TP}+\text{FP}}$ = 75 / (75 + 19) ≈ 0.798



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

Source: https://en.wikipedia.org/wiki/F-score

F_1 score

The **F_1 score** is the harmonic mean of precision and recall:

$$F_1 = \sqrt{\text{precision} \cdot \text{recall}}$$

- **Like** accuracy, the F_1 quantifies overall amount of error
- **Unlike** accuracy, F_1 is not as affected by uneven class distributions

Titanic example: $F_1 = \sqrt{0.798 \times 0.652} \approx 0.52$

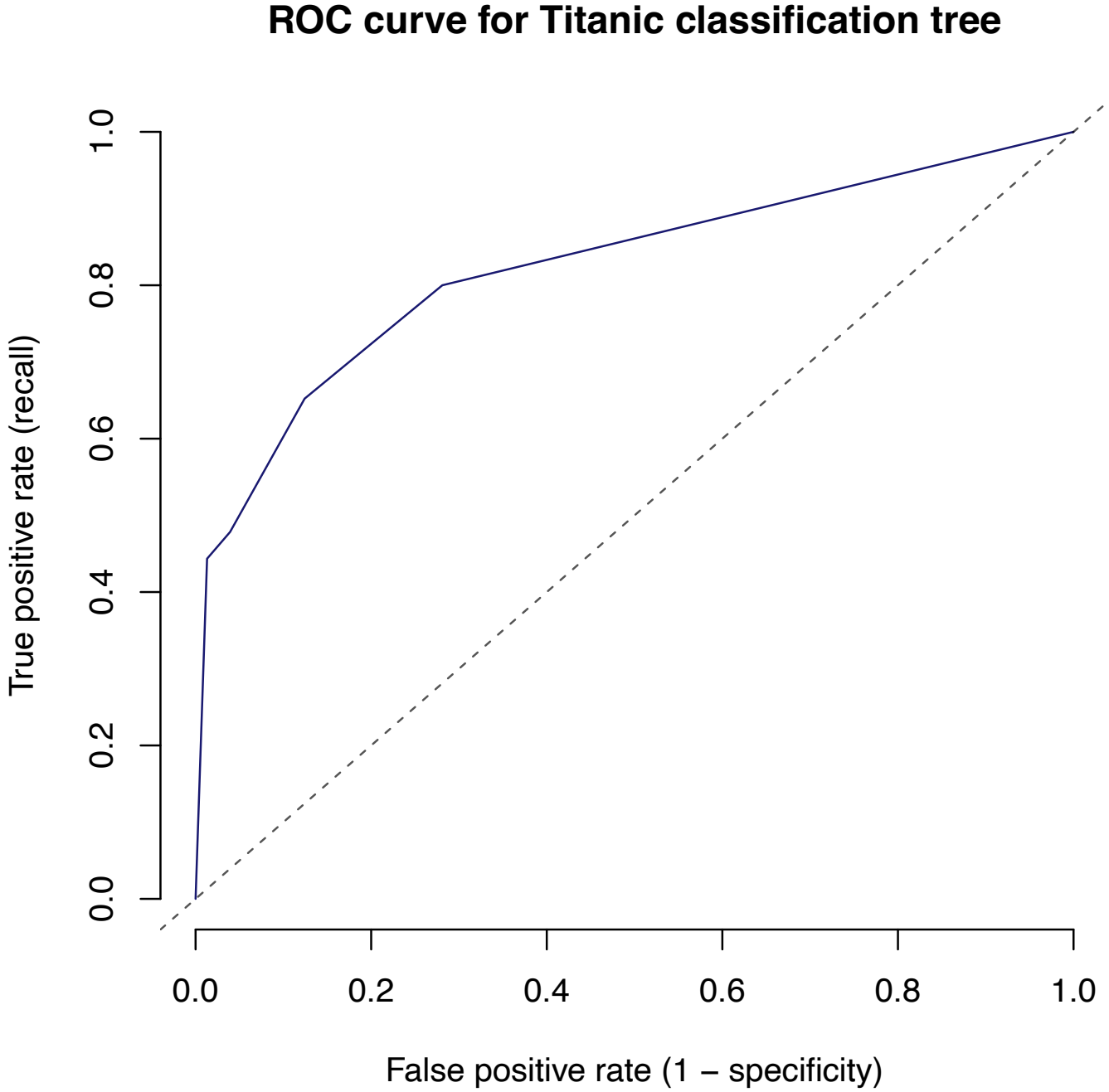
Overview: some classification metrics

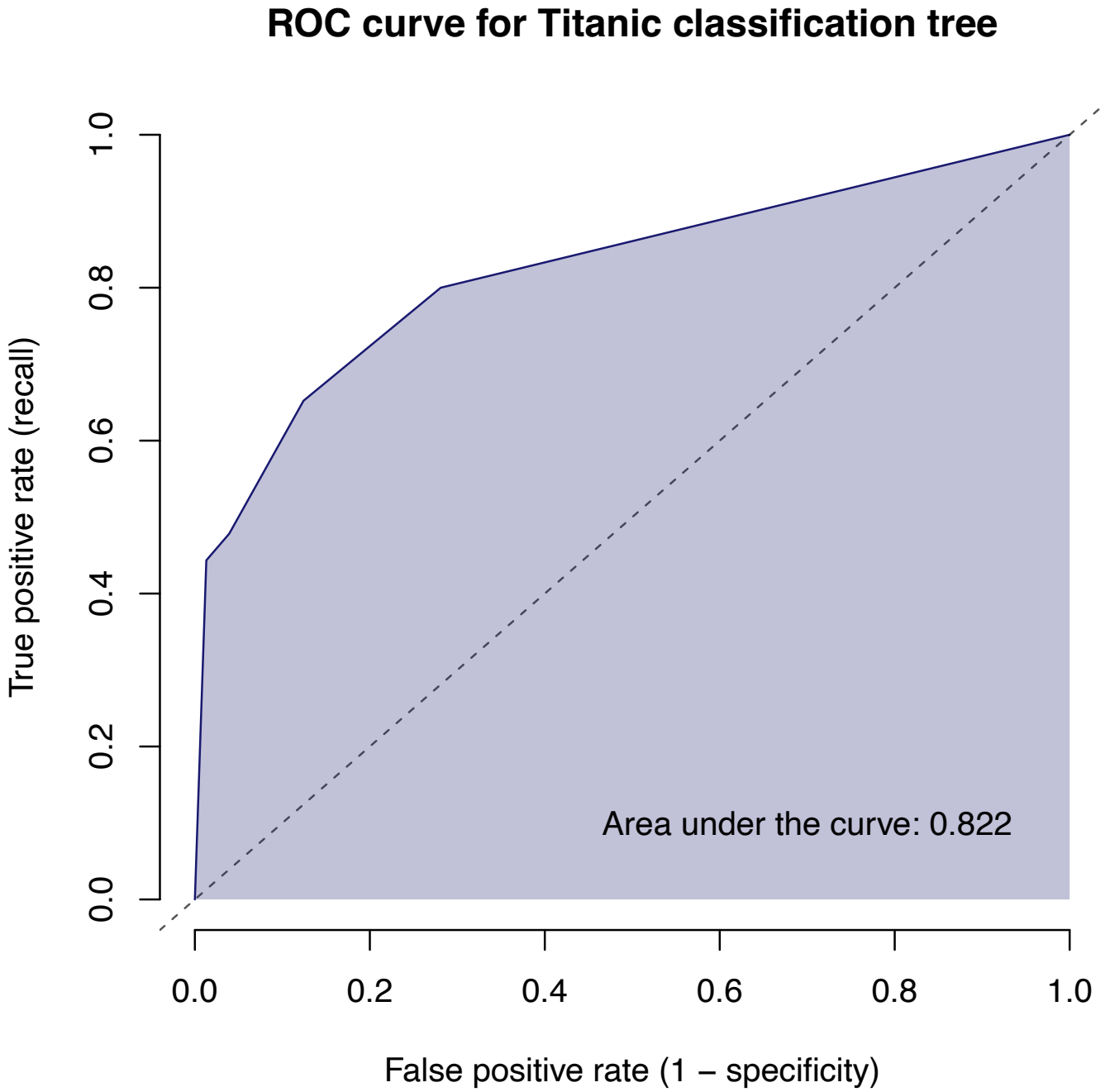
- Sensitivity (=Recall)
- Specificity
- Positive predictive value (=Precision)
- Negative predictive value
- Accuracy
- Many! more: https://en.wikipedia.org/wiki/Sensitivity_and_specificity

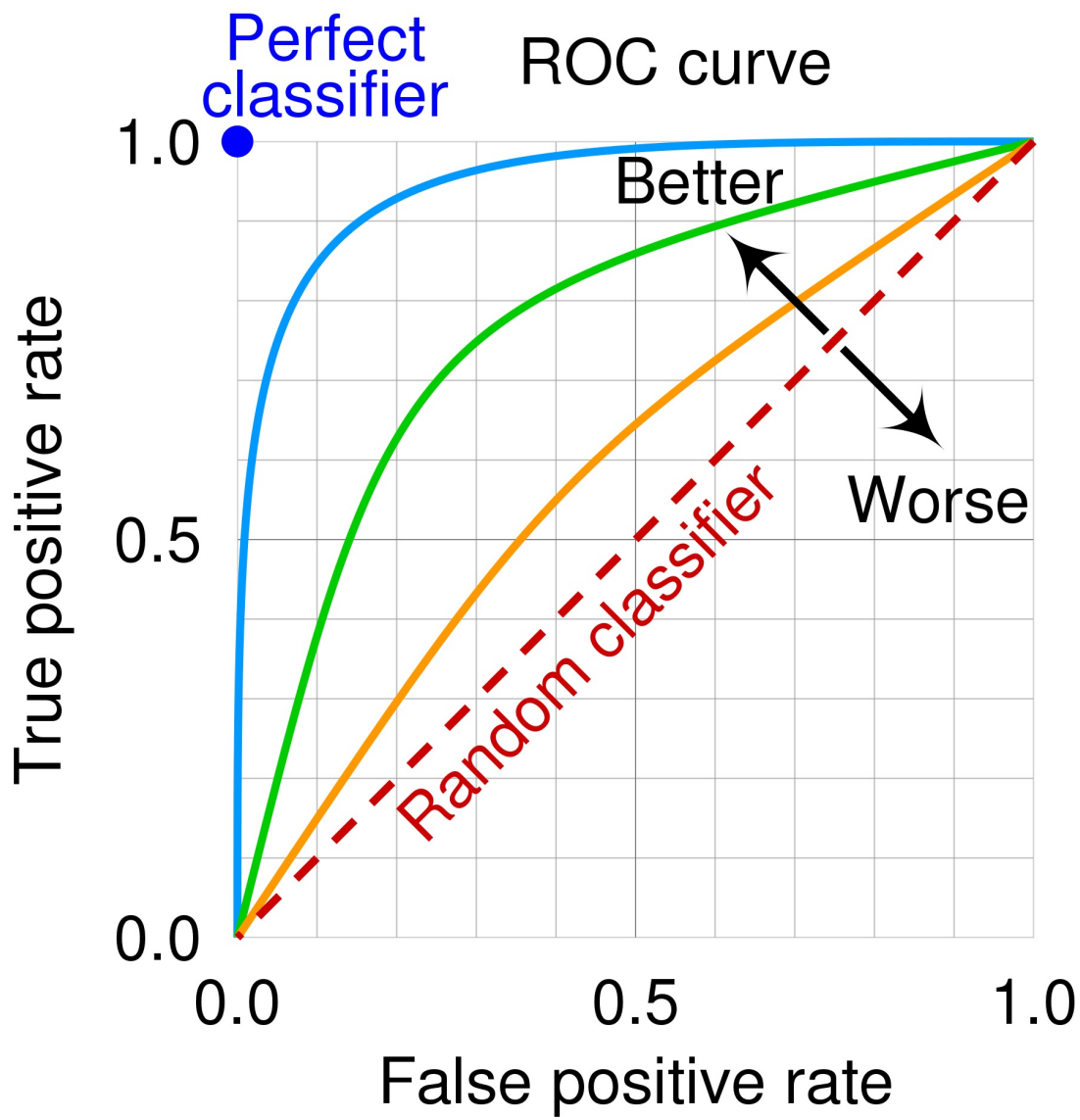
Different thresholds than 0.5

Moving around the threshold affects sensitivity and specificity!

```
> with(val_df, table(p_pred > 0.4, Survived)) %>% prop.table(2)

       Survived
            0     1
  FALSE 0.876 0.348
  TRUE  0.124 0.652

> with(val_df, table(p_pred > 0.6, Survived)) %>% prop.table(2)

       Survived
            0     1
  FALSE 0.961 0.522
  TRUE  0.039 0.478
```


ROC curve for Titanic classification tree
True positive rate (recall)
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate (1 – specificity)
0.0
0.2
0.4
0.6
0.8
1.0

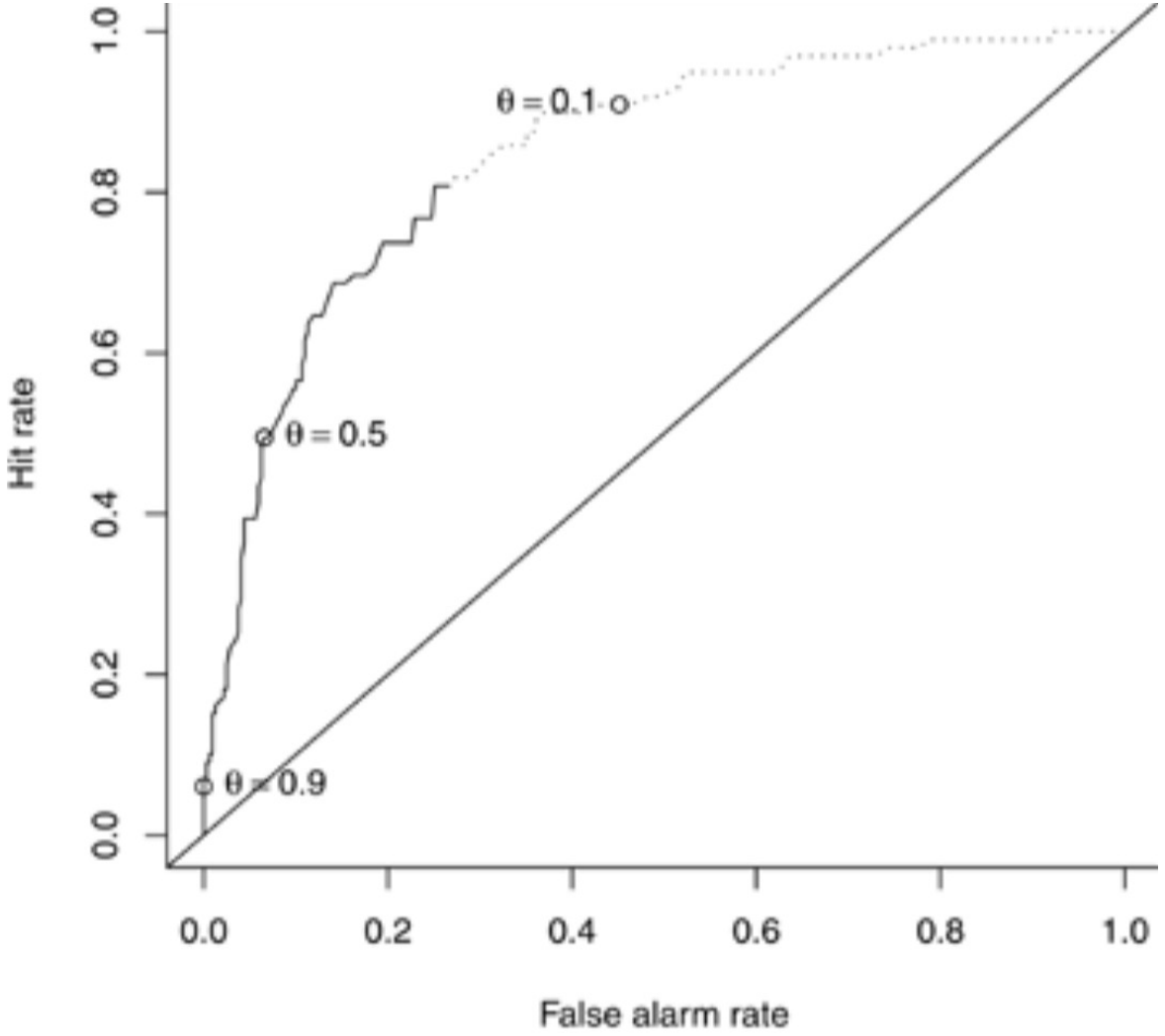

ROC curve for Titanic classification tree
True positive rate (recall)
False positive rate (1 – specificity)
0.0
0.2
0.4
0.6
0.8
1.0
Area under the curve: 0.822


Perfect classifier
ROC curve
1.0
Better
0.5
Worse
Random classifier
True positive rate
0.0
0.0
0.5
1.0
False positive rate

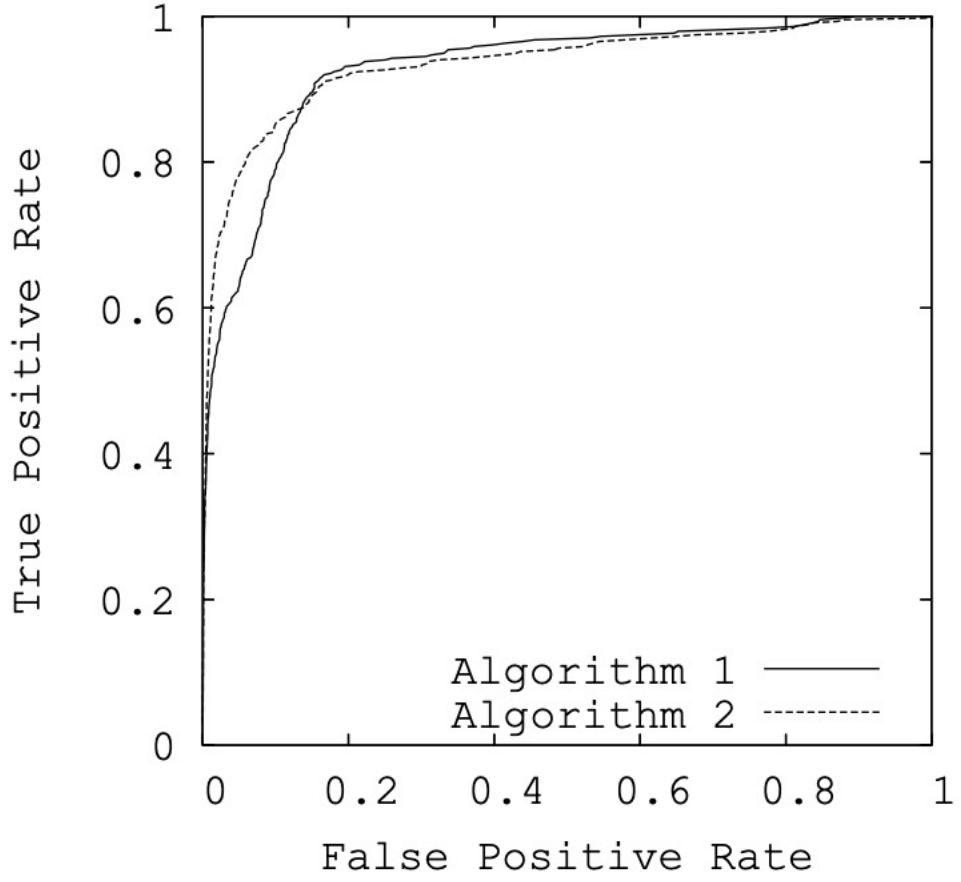
Problems with ROC curves

- **Imbalance**: Does not “care” about small classes
- **Hidden baseline**: Reference is not “random”
- **Black-and-white**: Only shows effect of binary decision (“discrimination”), not accuracy of the probability output (“calibration”)

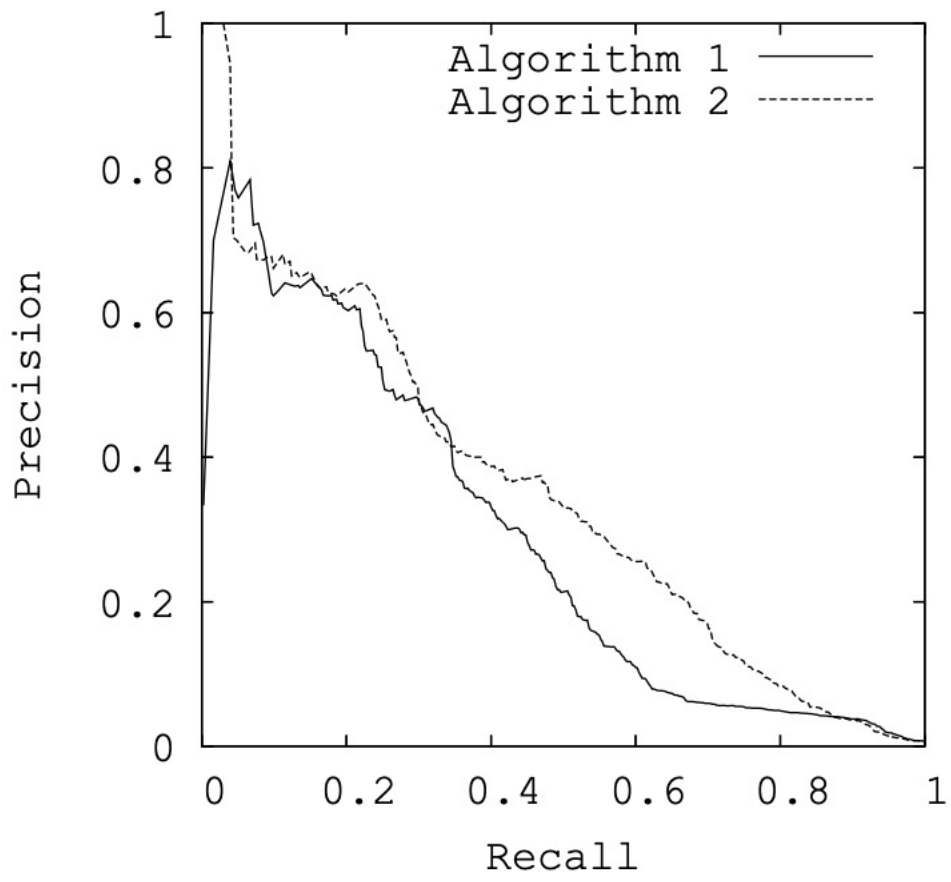
ROC with “skill” (Briggs & Zaretzki)



Precision-recall curves



(a) Comparison in ROC space



(b) Comparison in PR space

Actual cc fraud detection (ebay)

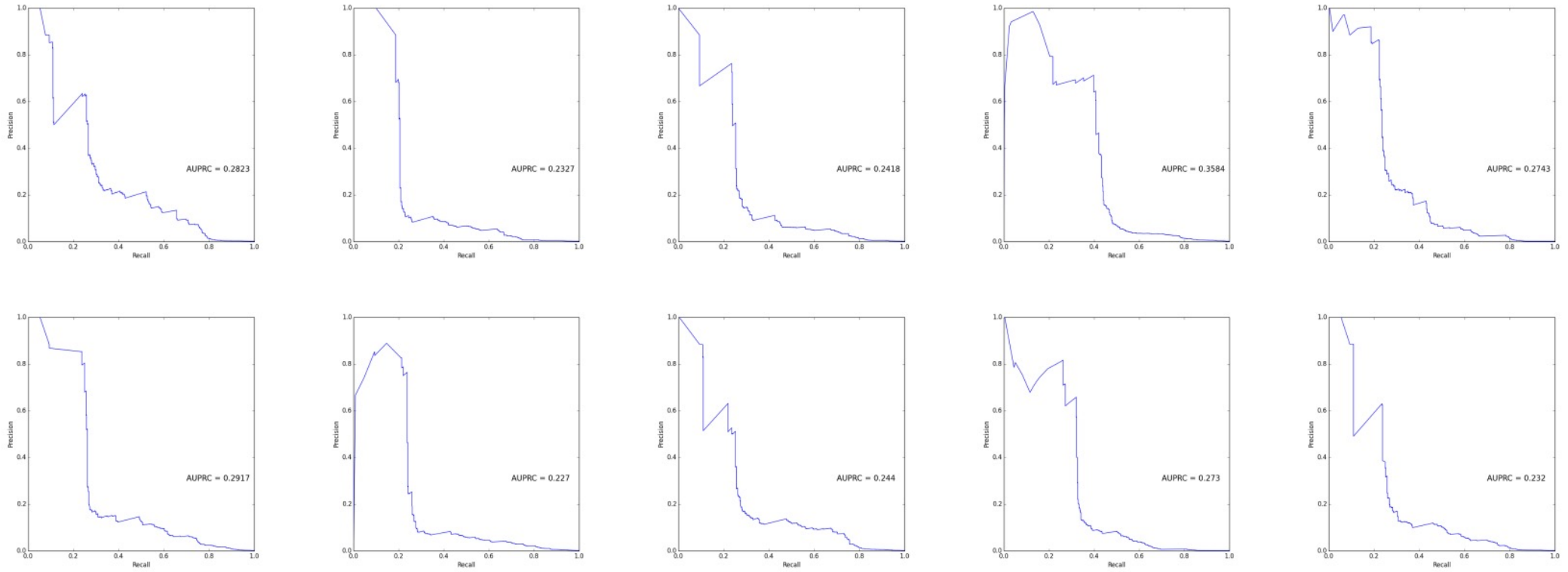


Figure 4: Precision Recall curves of different runs of the proposed algorithm on the Credit Card Fraud Detection Dataset

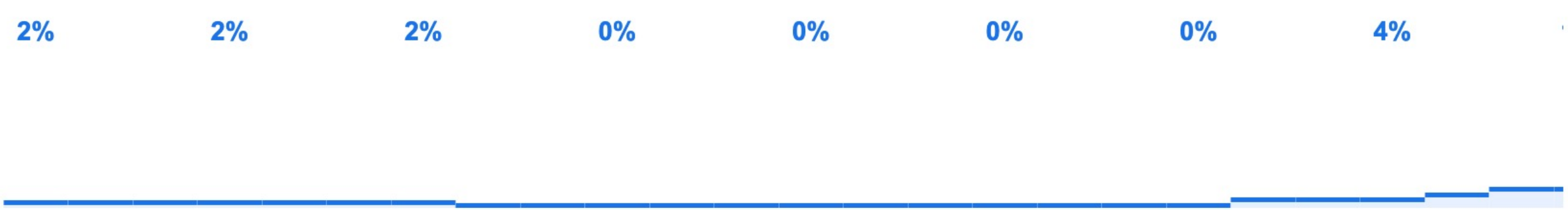
Calibration

- Besides the quality of a single-shot predicted class (“yes/no”, “survive/die”, ...),
- we could also be interested in the predicted probability.
- E.g.: “casemix adjustment” for hospital/school evaluation, risk scores in medicine, betting, ...

On days like today, how often does it rain?

Utrecht
Friday
Mostly sunny

Temperature | Precipitation | Wind



03:00	06:00	09:00	12:00	15:00	18:00	21:00	00:00
Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu
15° 3°	15° 8°	13° 7°	14° 4°	14° 4°	15° 5°	18° 6°	15° 3°

Probability

A *probability* is a number *p* such that the proportion of events given that number is about *p*.

- **Ideally**, the classification procedure (e.g. classification tree) outputs a predicted probability directly.
- **Unfortunately**,
  - Not all classifiers output a predicted probability (e.g. SVM);
  - Many classifiers that do give a number between 0 and 1 called a "predicted probability", the predicted probability does not give the correct proportion of events.

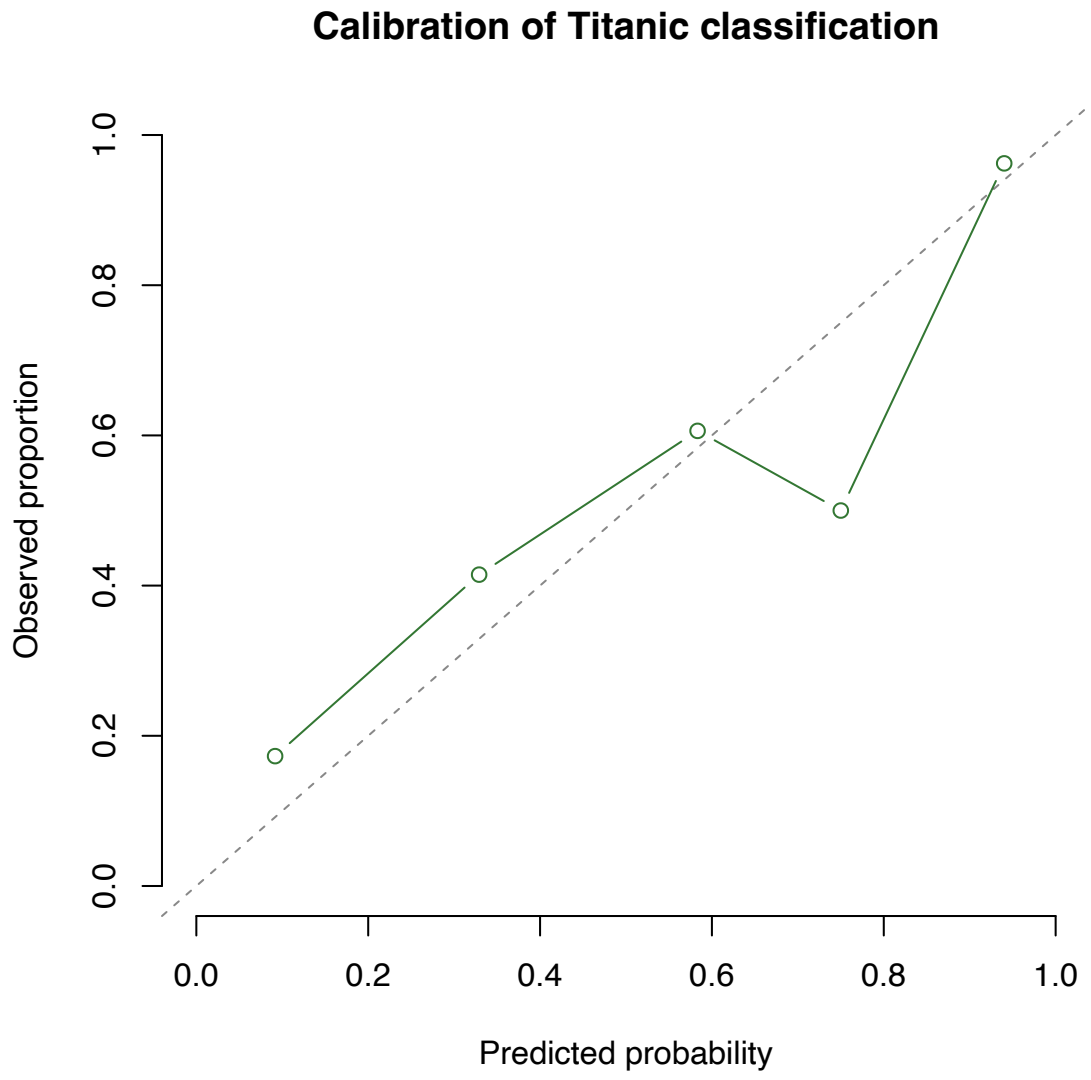
This is the "**calibration problem**".

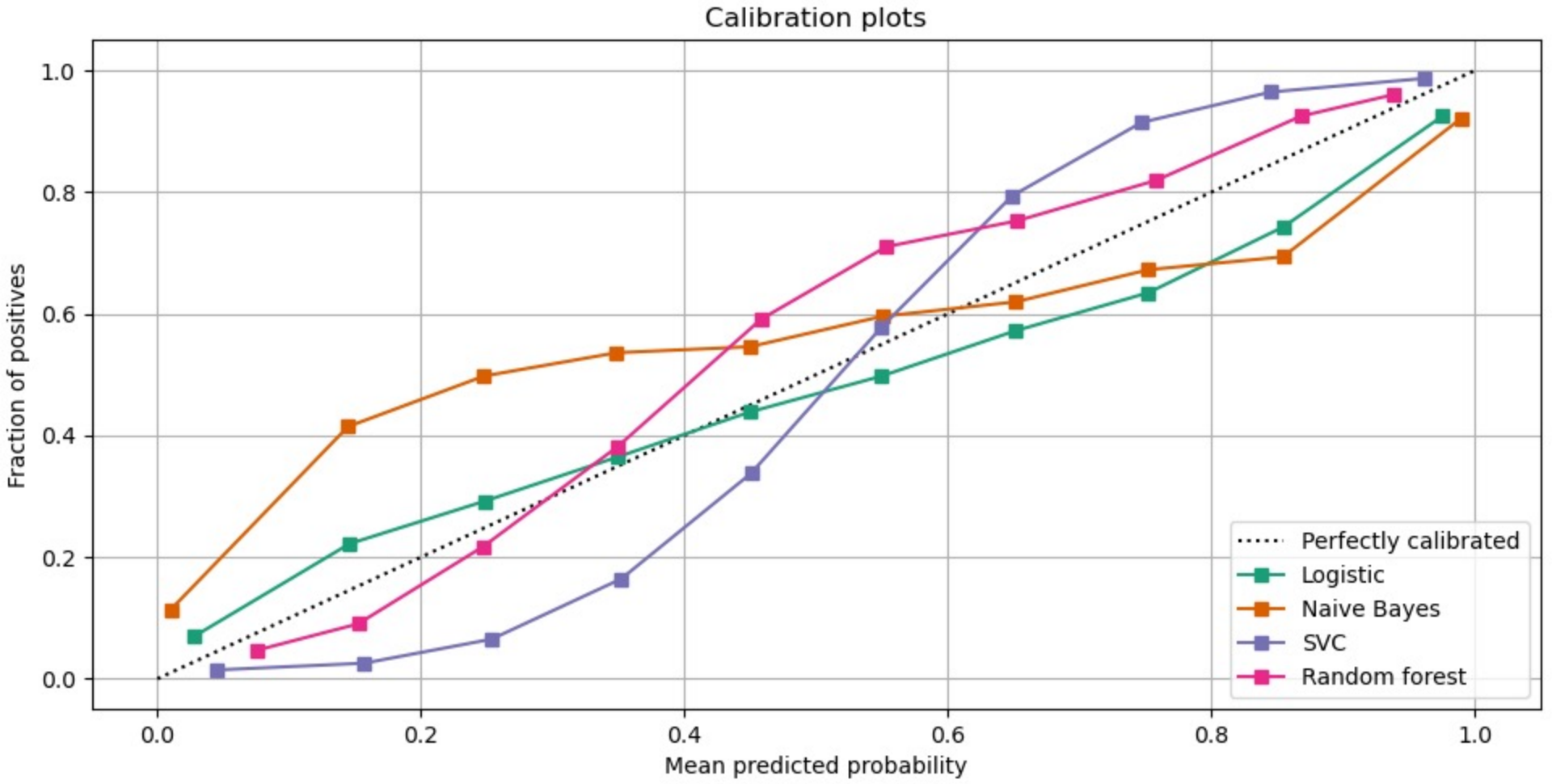
Calibration plot

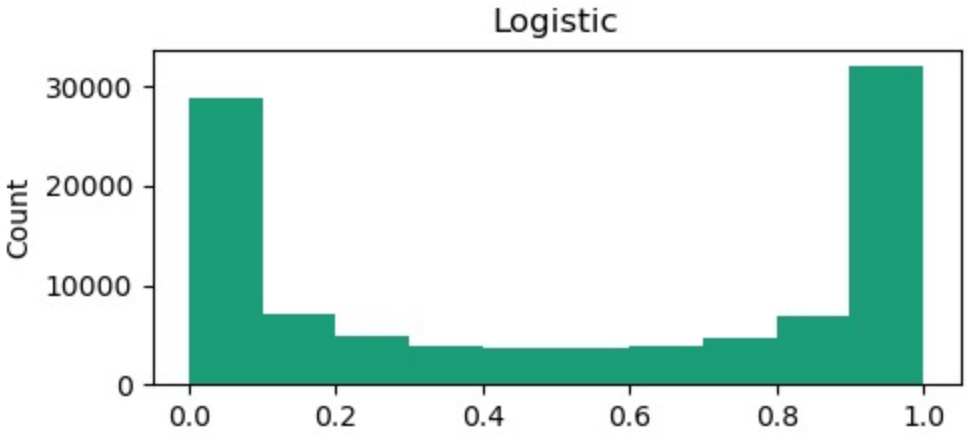
Probability

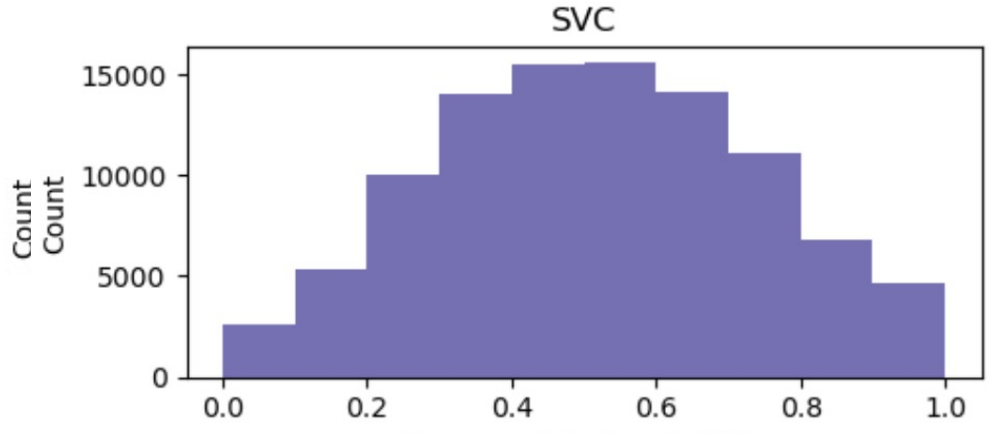
A *probability* is a number *p* such that the proportion of events given that number is about *p*.

- A predicted probability is **calibrated** when it conforms to the definition above;
- Check this using a **calibration plot**


Calibration of Titanic classification
Observed proportion
0.0
0.2
0.4
0.6
0.8
1.0
Predicted probability
0.0
0.2
0.4
0.6
0.8
1.0

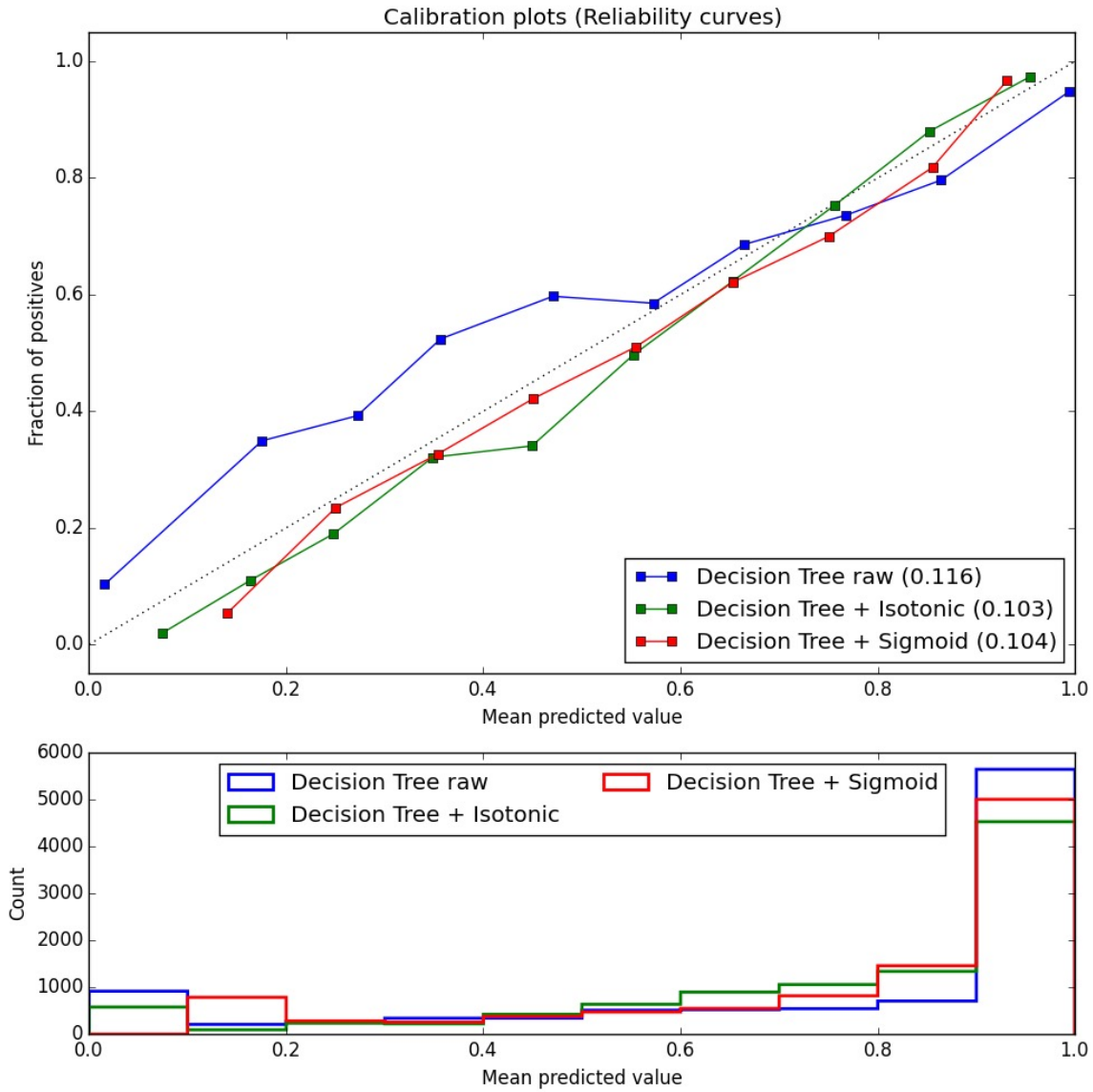

Calibration plots
Fraction of positives
Mean predicted probability
Perfectly calibrated
Logistic
Naive Bayes
SVC
Random forest


Logistic
Count


SVC
Count
Count

Post-hoc probability calibration

- Some libraries allow you to tweak the predicted probabilities so they fit on the curve. This is called “probability calibration”.
- There are many methods, but the most commonly used one takes a classification model we know is calibrated (“logistic regression”) and applies it to the uncalibrated scores outputted by the classifier;
- You may encounter this in your readings.


Calibration plots (Reliability curves)
Fraction of positives
Mean predicted value
Decision Tree raw (0.116)
Decision Tree + Isotonic (0.103)
Decision Tree + Sigmoid (0.104)
Count
Mean predicted value
Decision Tree raw
Decision Tree + Isotonic
Decision Tree + Sigmoid

MSE (“Brier score”)

Setting Yes = 1 and No = 0, we can again evaluate the Mean Square Error (MSE), now called “Brier score”:

$$\mathrm{MSE} = \mathrm{average}\big((\hat{p} - y)^2\big)$$

Turns out MSE can be reworked into two terms:

MSE = Calibration term + AUC term

This shows that the usual MSE metric combines “calibration” and “discrimination”.

Calibration is not always the target

(Friedman, 1997)

- “more accurate probability estimates do not necessarily lead to better classification performance and often can make it worse”.
- “these results suggest that **in situations where the goal is accurate classification**, focusing on improved probability estimation may be misguided”.

Class imbalance

- In the Titanic example, the outcome classes are pretty evenly **balanced**;
- That is *not* typical of many applications: debt default; illness; activity; buy/don't buy; tank/dog/selfie/..; solid/liquid/gas/plasma; ...
- When at least one class has very few observations, this is called **class imbalance**.

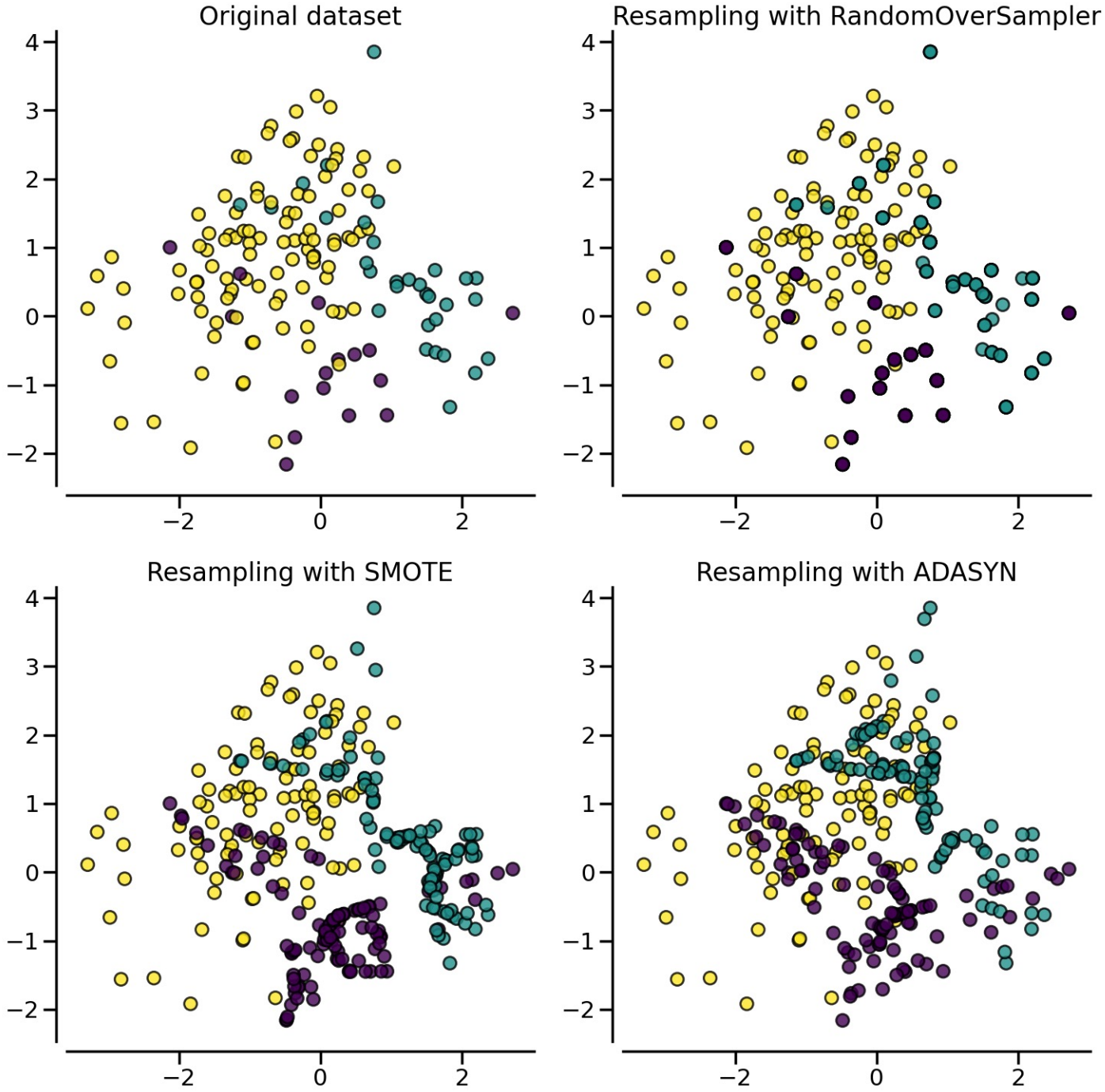
Class imbalance

Problem:

- Measures such as SEN/SPE/ACC/F1 emphasize larger classes;
- What if the *smaller* classes are the most/equally interesting?

Some **solutions**:

- Oversampling minority/undersampling majority
- Weighting
- “Embedded”: cost-sensitive learning



https://imbalanced-learn.org

Other key but tricky issues: fairness and feedback

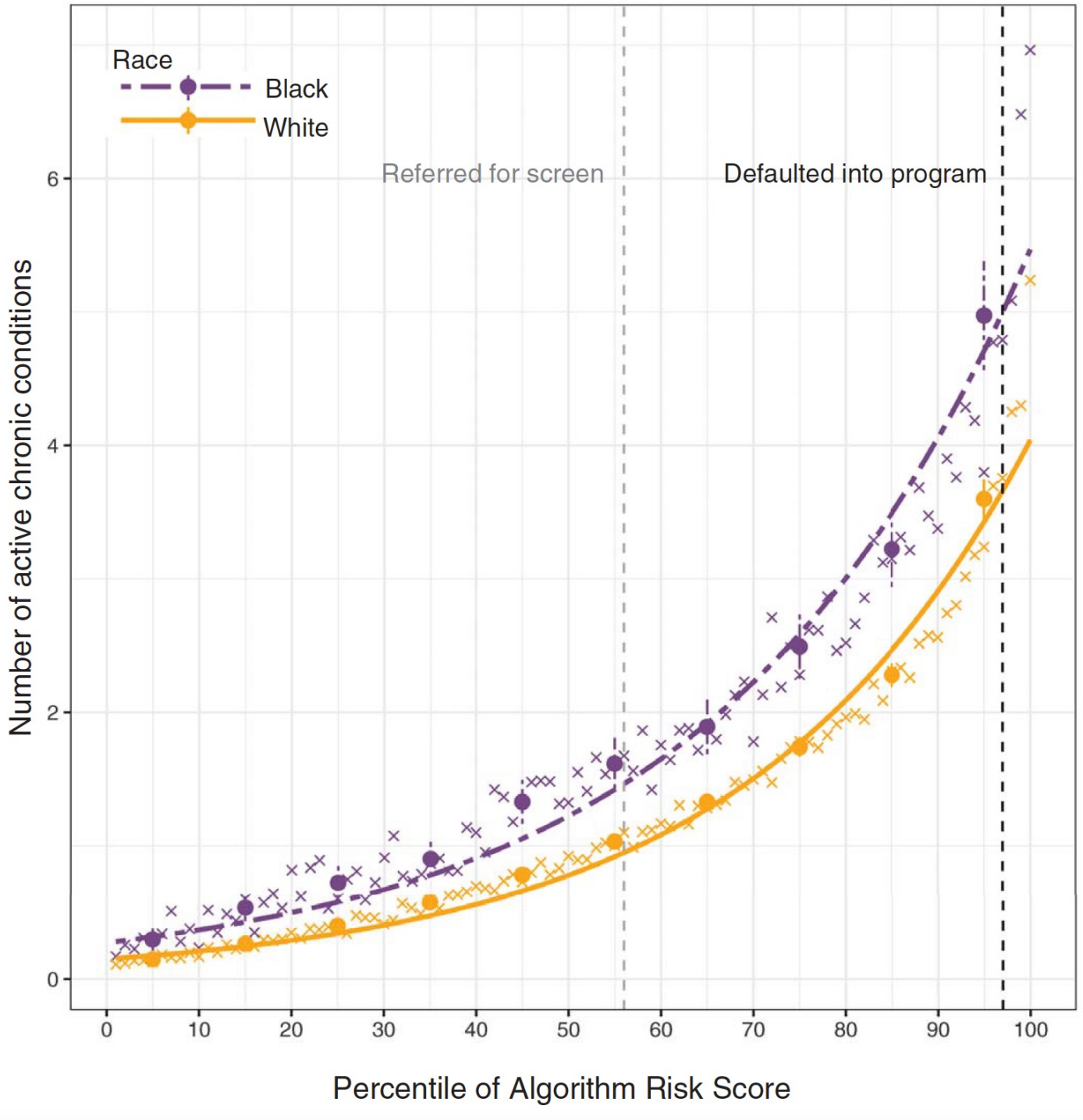
Imitating DGP is not always best

Random mistakes, for example:

- Random label noise, e.g. accidental paper cutlery in plastics
- Random feature noise

Systematic biases, for example:

- Styrofoam in plastics → learn to reproduce common mistake
- Belastingdienst finds “migration background” suspect → algorithm reproduces this labeling method (*toeslagenaffaire*)
- Immigrants live more in areas with high crime, but that does not mean that immigrants are doing the crime (ecological fallacy)

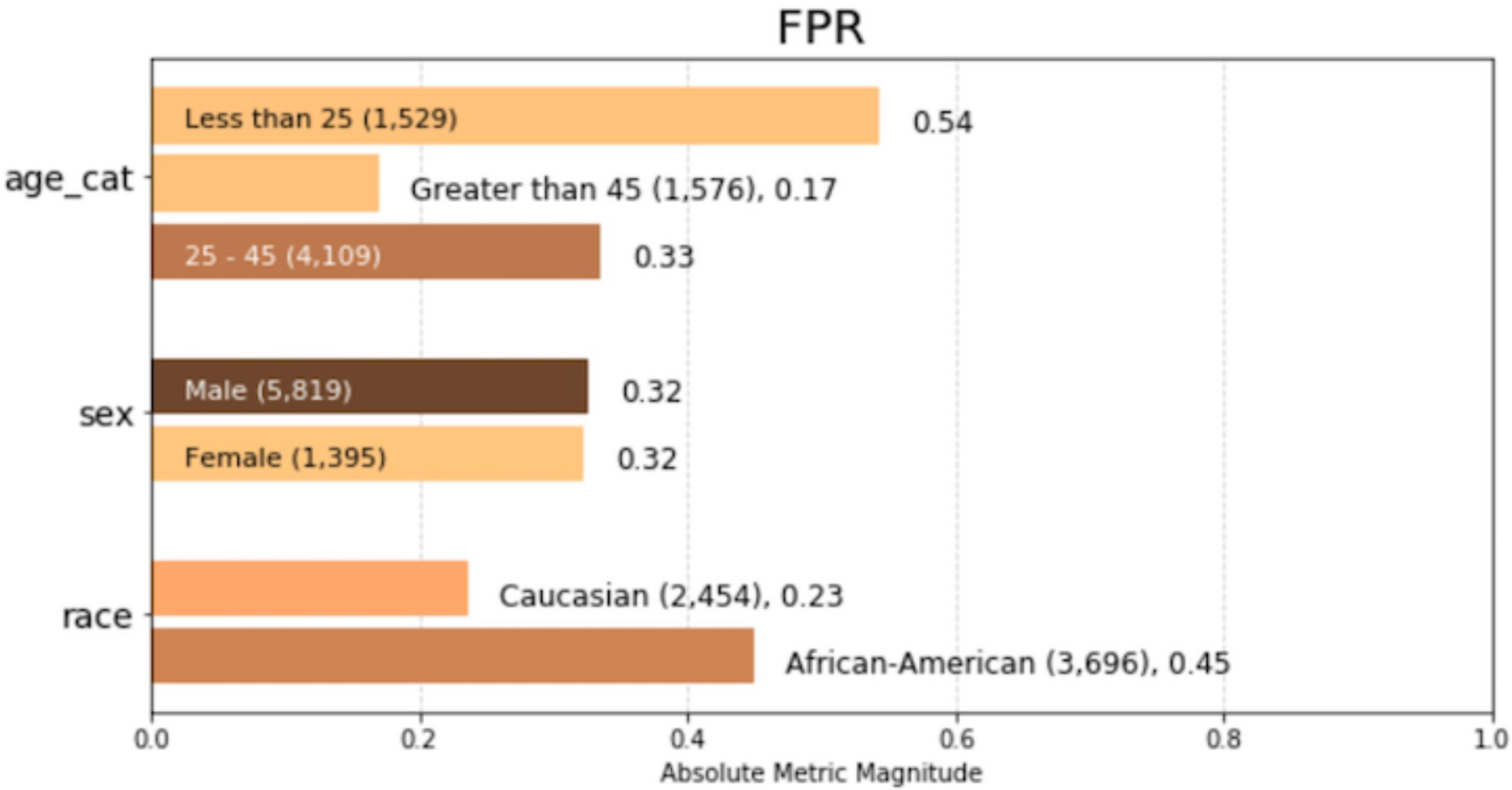


Obermeyer et al (2019)

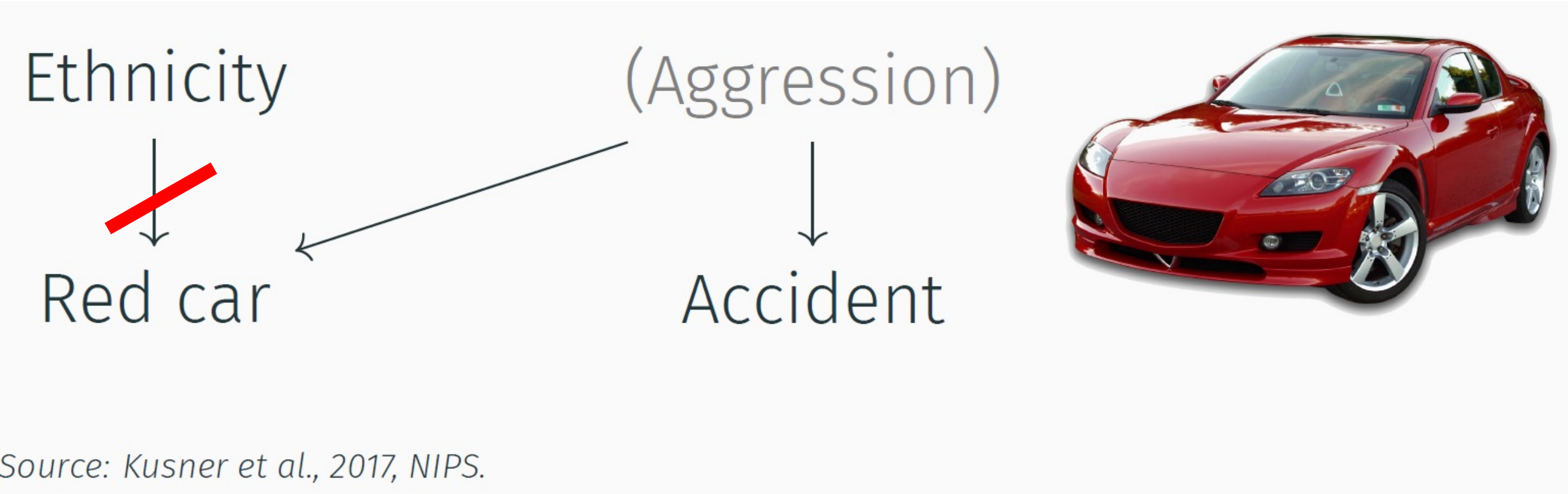
Notes about ML fairness

- **None** of the features in the Obermeyer algorithm were race,
- But a bias against black people still resulted.
- This demonstrates one strategy that does ***not*** work: "~~fairness through unawareness~~"
- Many definitions of "fairness" in ML literature
- Large number of proposed strategies to
  - Audit
  - Prevent
- E.g. https://dssg.github.io/aequitas/

```
from aequitas.plotting import Plot

aqp = Plot()
fpr_plot = aqp.plot_group_metric(xtab, 'fpr', min_group_size=0.05)
```

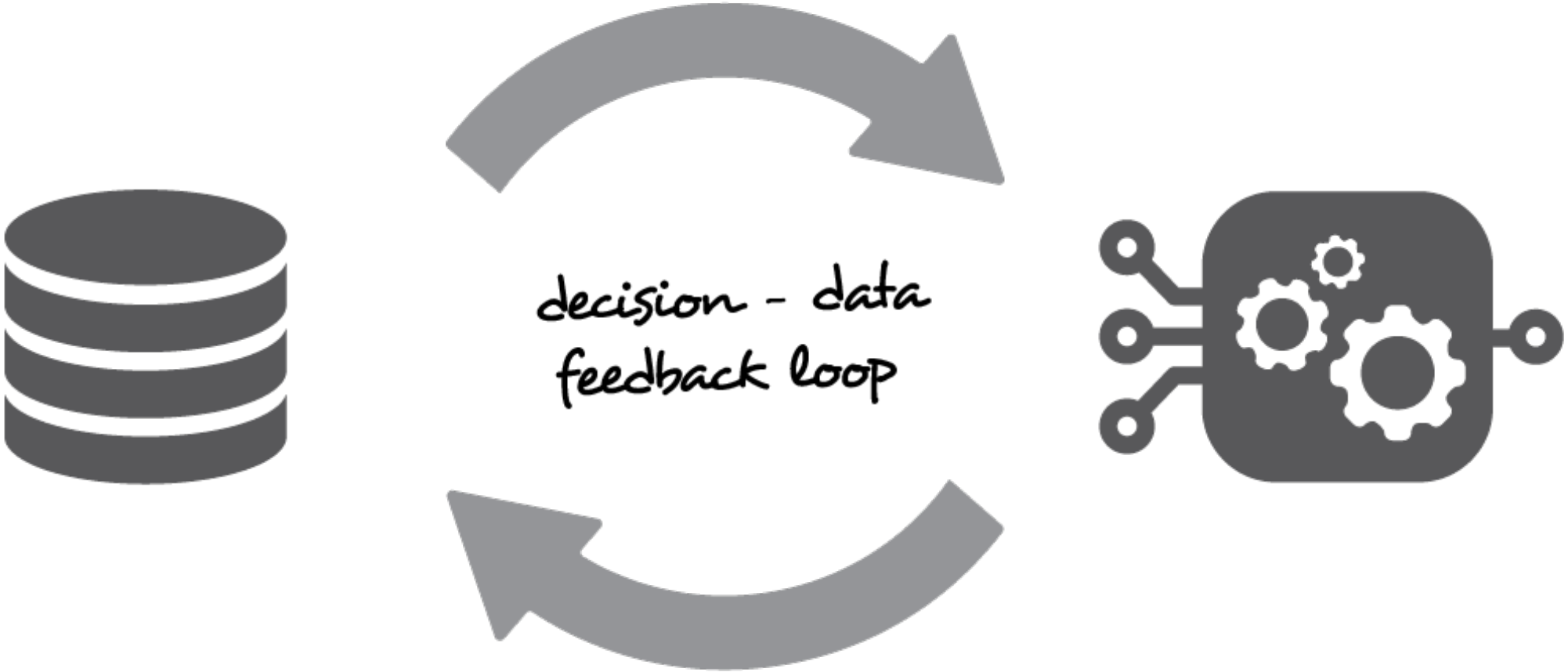


Counterfactual fairness



Source: Kusner et al., 2017, NIPS.

Data are used to train the model



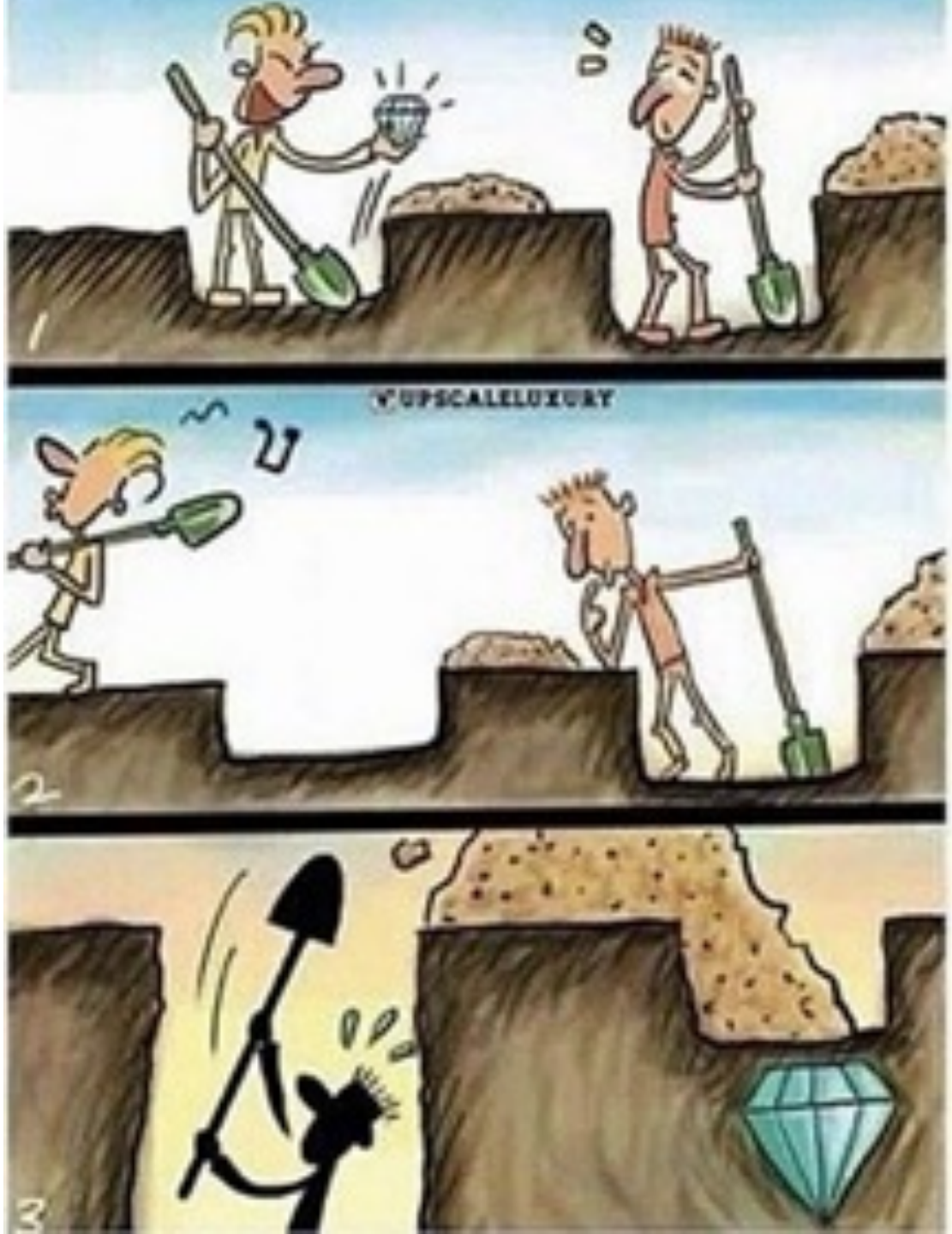
The use of model generates new data

Feedback loops in learning decisions

1. **Correlation**: *feature* “transfer amount” and *target* “fraud”;
2. → We observe correlation and so **learn** the following rule:
   `IF ”transfer amount” IS “high” THEN ”fraud”`
3. → Fraudsters who do big single transfers are caught and fraudsters who split transfers are not caught;
4. → No more correlation!

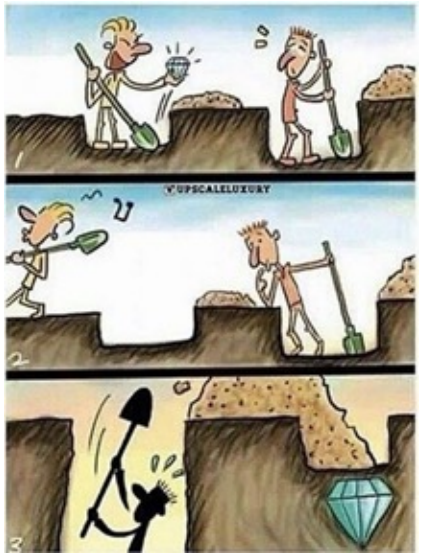
Should we:

- Forget the rule about high transfer amounts? (*bad*)
- Use the rule, which is now bad? (*also bad*)

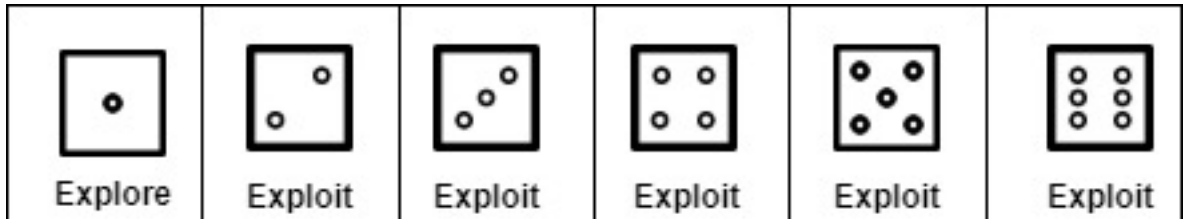


https://ai-ml-analytics.com/reinforcement-learning-exploration-vs-exploitation-tradeoff/

Exploration-exploitation tradeoff



- *Math*: Learning and Acting are in **conflict**
- If we *ALWAYS* send police to predicted “high-risk” area, ...
- …then we *NEVER* learn whether our prediction was any good or not!
- *Only* solution: change decision rule to a ***stochastic*** one:
  - *Sometimes* choose to act as model suggest, with probability *p*
  - *Sometimes* choose to act differently, prob. (1-*p*)
- Because we know the probability *p*, we can weight the evidence and **keep learning**.
- Glimpse into “bandit theory”/Reinforcement Learning



Classification is a large field

- We have emphasized
  - Pluralistic approach, **coupled with**
  - Honest model evaluation
- This includes thinking about the task, performance metric, and training and testing data
- It also includes task-specific issues, such as fairness, measurement error, and feedback
- **You should now be equipped to start using supervised learning in practice!**