Data Wrangling and Data Analysis
Classification

Daniel Oberski

Department of Methodology & Statistics

Utrecht University

VERSION: 2023-10-10

Supervised machine learning

1. Regression (predicting continuous outcomes)
2. Model evaluation
3. **Classification (predicting discrete outcomes)**
4. Deep learning

Reminder

- **Goal**: given a new, ***unseen***, vector data point $\mathbf{x}_i$ (with potentially many dimensions), predict a scalar (single value) y_i ;
- To achieve this task, we will learn a function $\hat{f}(\boldsymbol{x})$ (model) using an algorithm (estimator, learner) from labeled training data points (**x**, y);
- When asked to perform the task on new, unseen, data, we will output the prediction function learned earlier, $\hat{y} = \hat{f}(\boldsymbol{x})$.
- We will now look at some of these functions (models) and how they can be learned (estimated)

Some commonly used “learners”

- kNN
- trees
- random forests
- Boosting (e.g. xgboost)
- SVM
- neural nets

Classification

The thing you’re trying to predict is **discrete**:

- *Titanic*: Survival/Nonsurvival
- Banking data: Default on/payment of debt
- GPS/Accelerometer data:
- Work/Home/Friend/Parking/Other
- Imagenet: gazelle/tank/pirate/sea lion/tandem bicycle/: : :
- Etc.

Example from the book

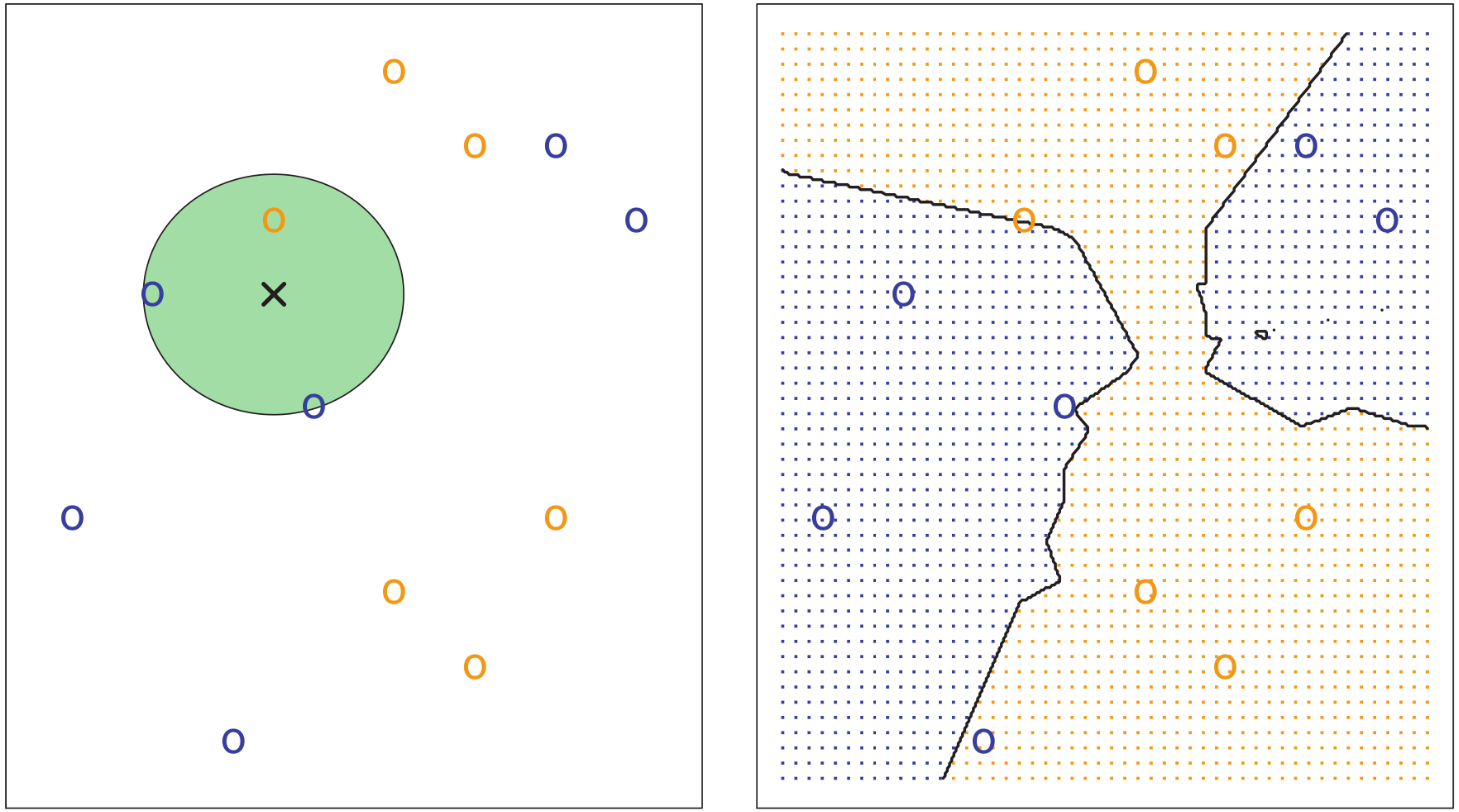
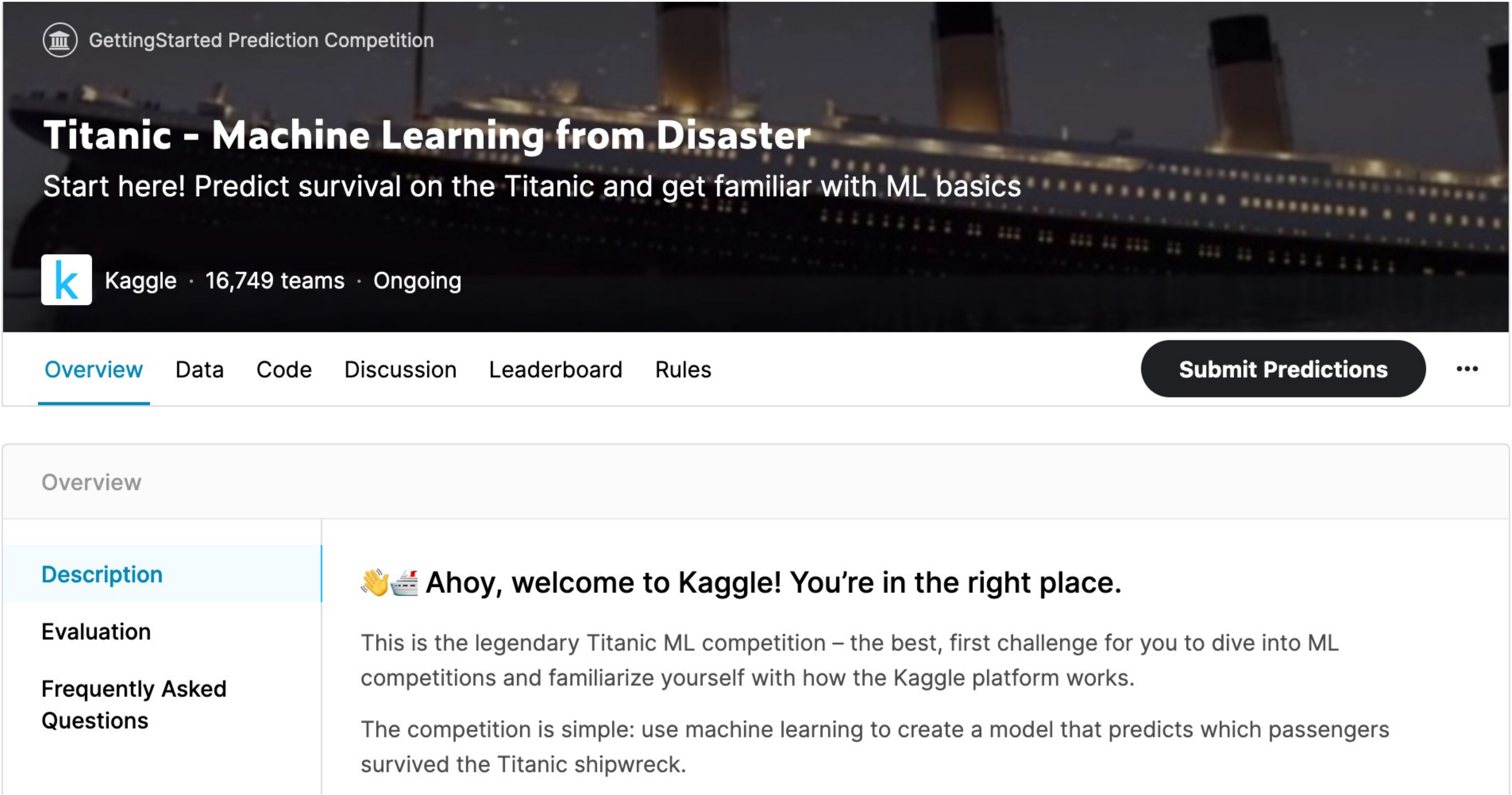


FIGURE 2.14. *The KNN approach, using $K = 3$, is illustrated in a s*

Classification trees



Titanic - Machine Learning from Disaster

Start here! Predict survival on the Titanic and get familiar with ML basics

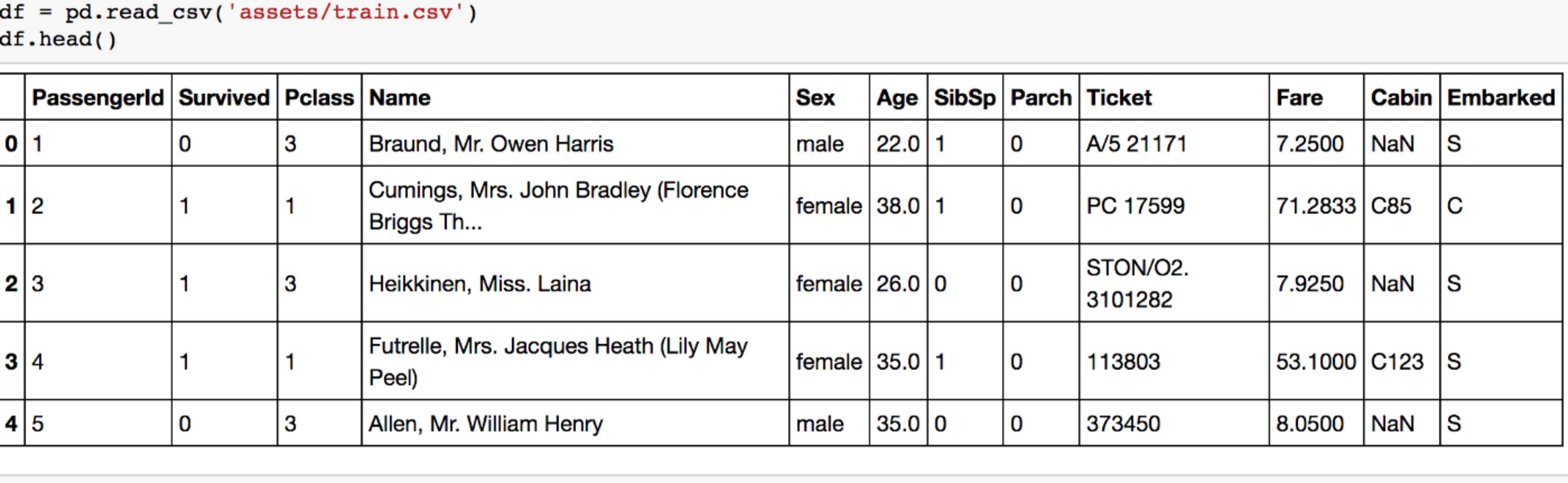
Kaggle · 16,749 teams · Ongoing

👋🛳️ 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.

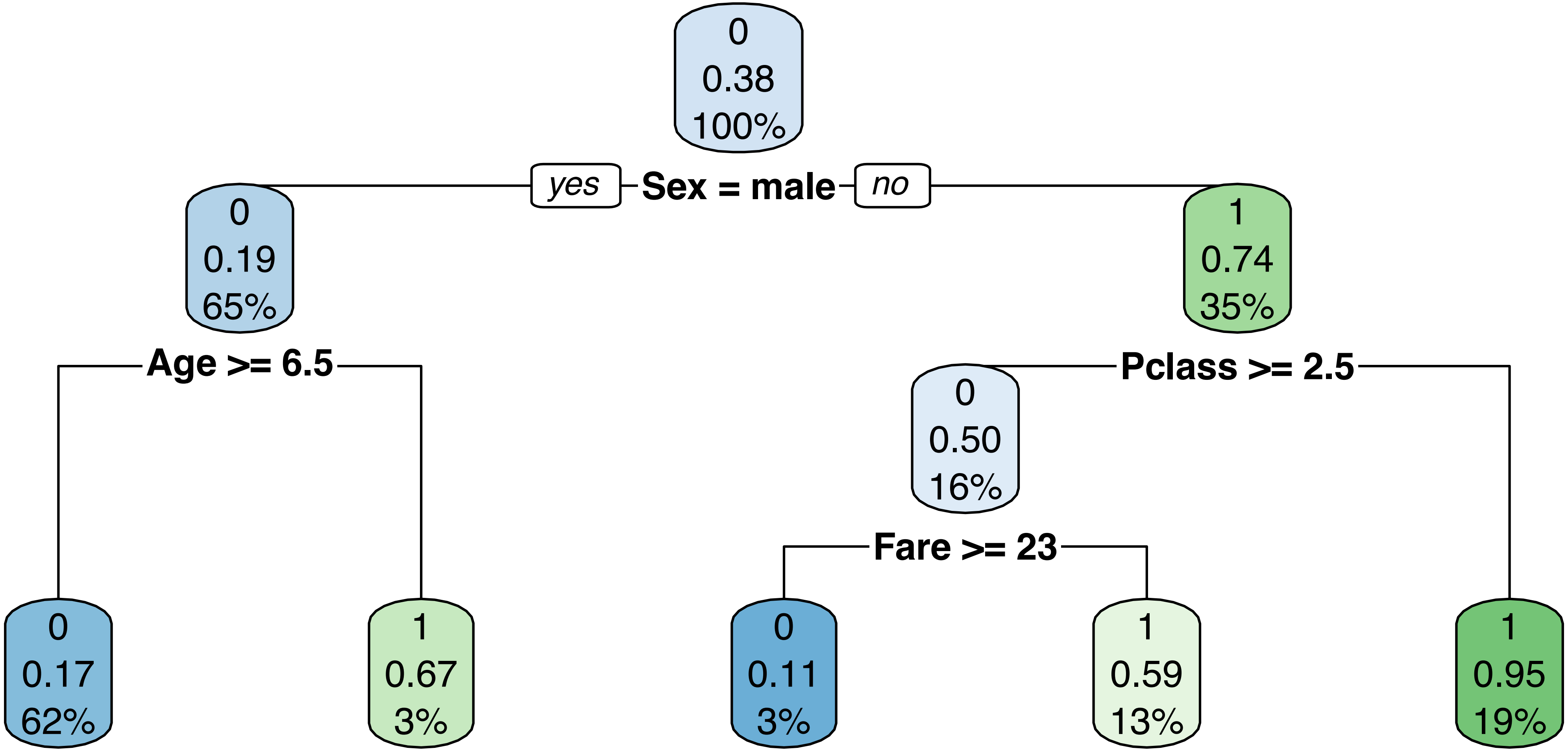
The competition is simple: use machine learning to create a model that predicts which passengers survived the Titanic shipwreck.

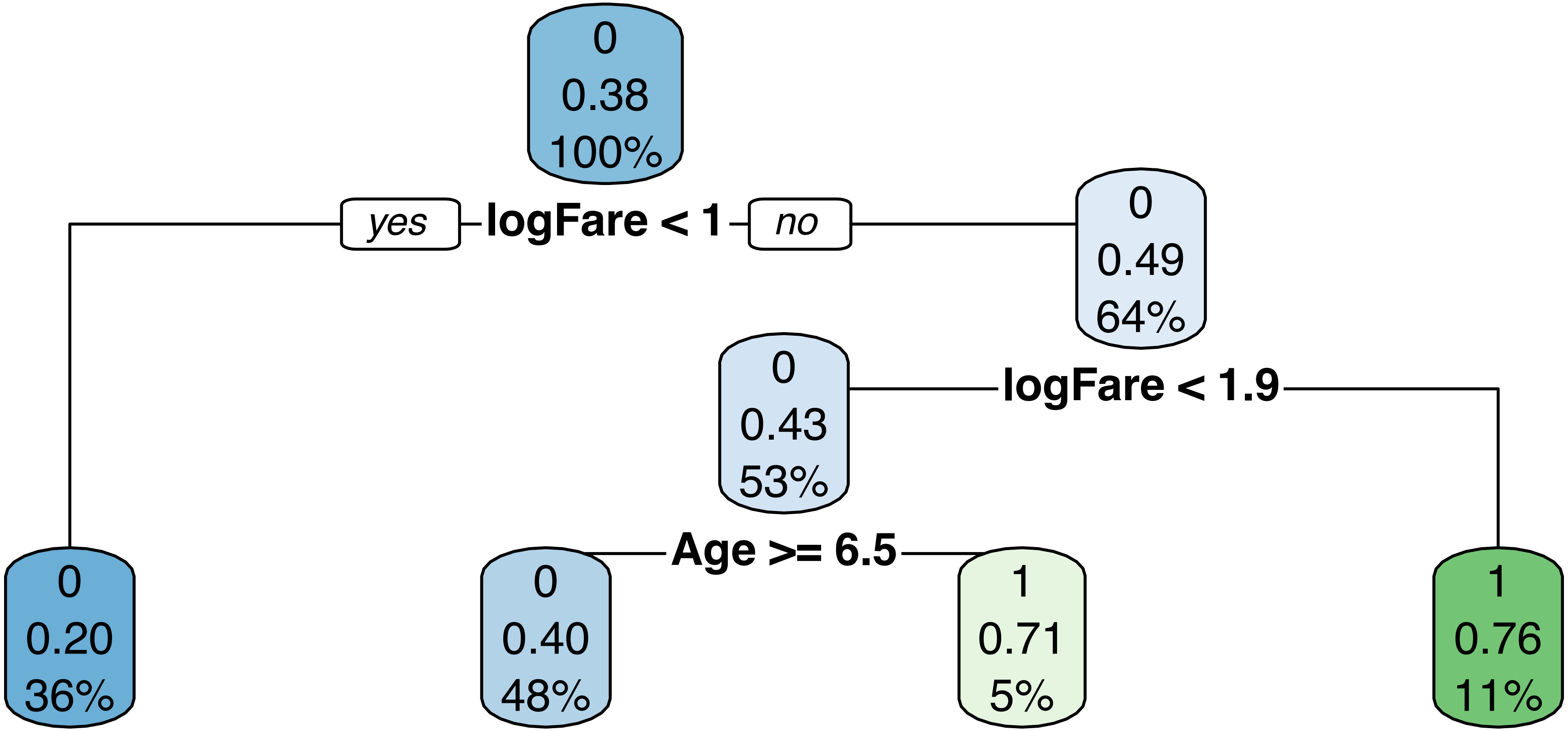
Titanic data



```
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
0.50
16%
Fare >= 23
0
0.17
62%
1
0.67
3%
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
Age >= 6.5
0
0.20
36%
0
0.40
48%
1
0.71
5%
1
0.76
11%

Learning algorithm

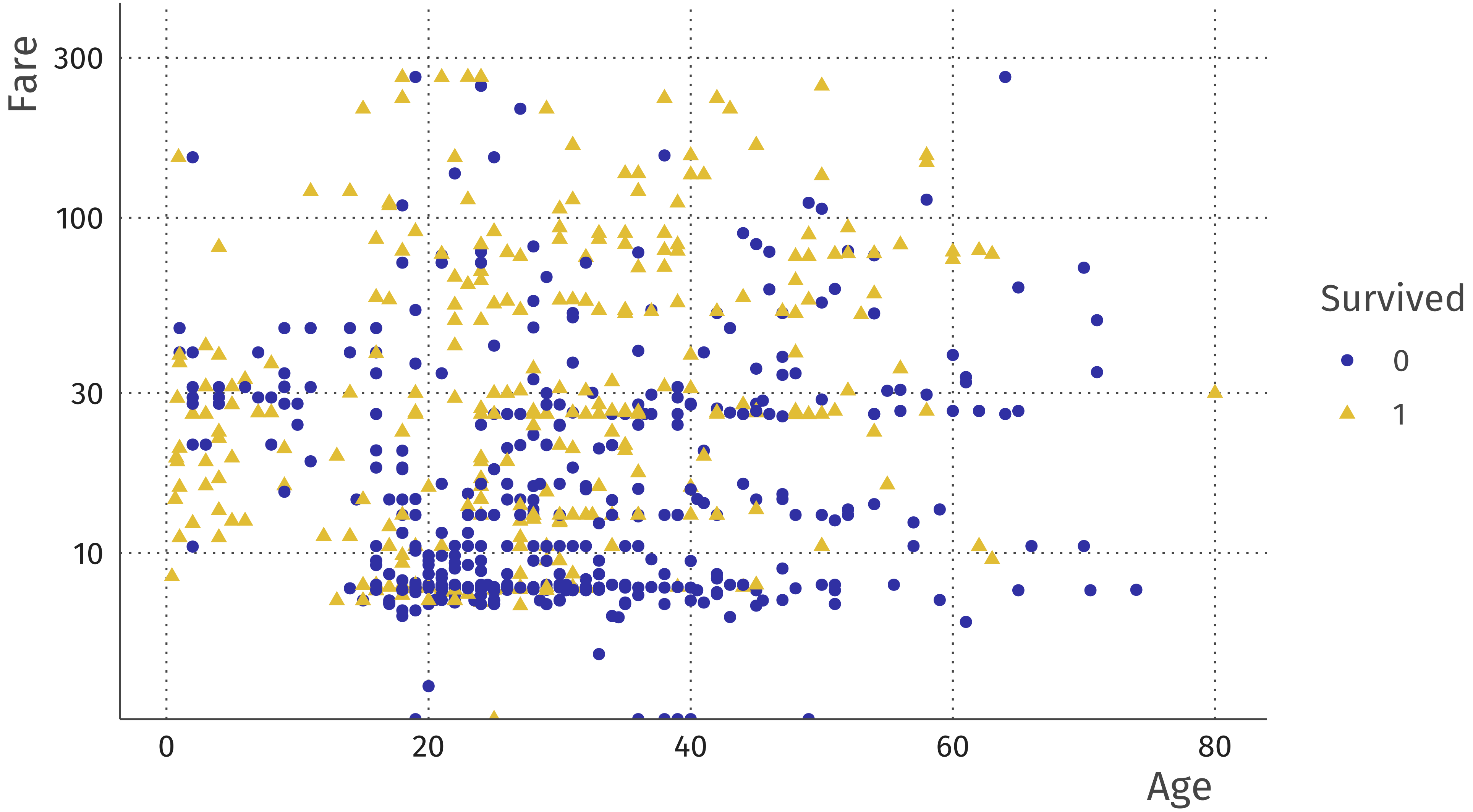
Recursive partitioning

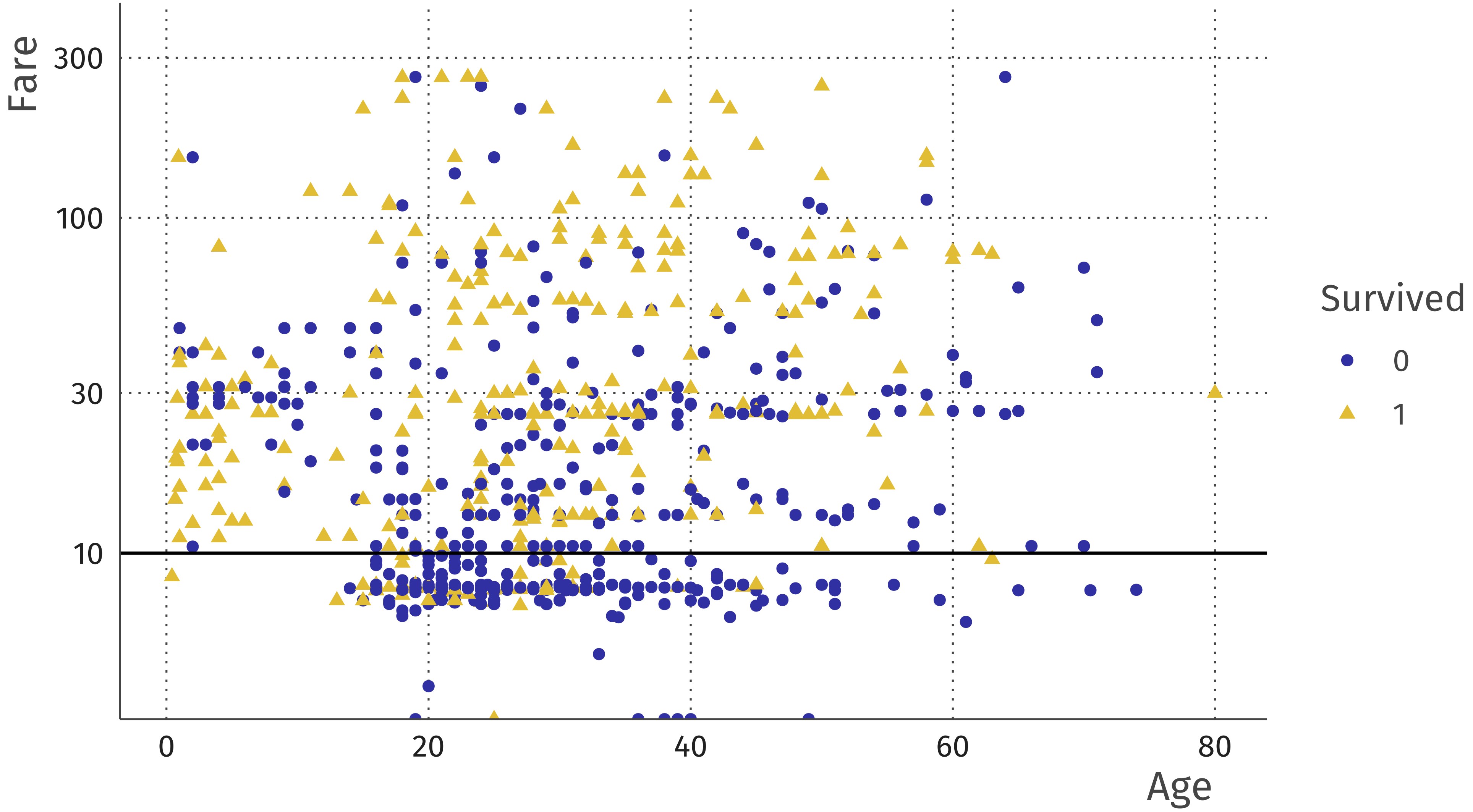
1. Find the split that makes observations as similar as possible on the outcome within that split;
2. Within each resulting group, do (1).

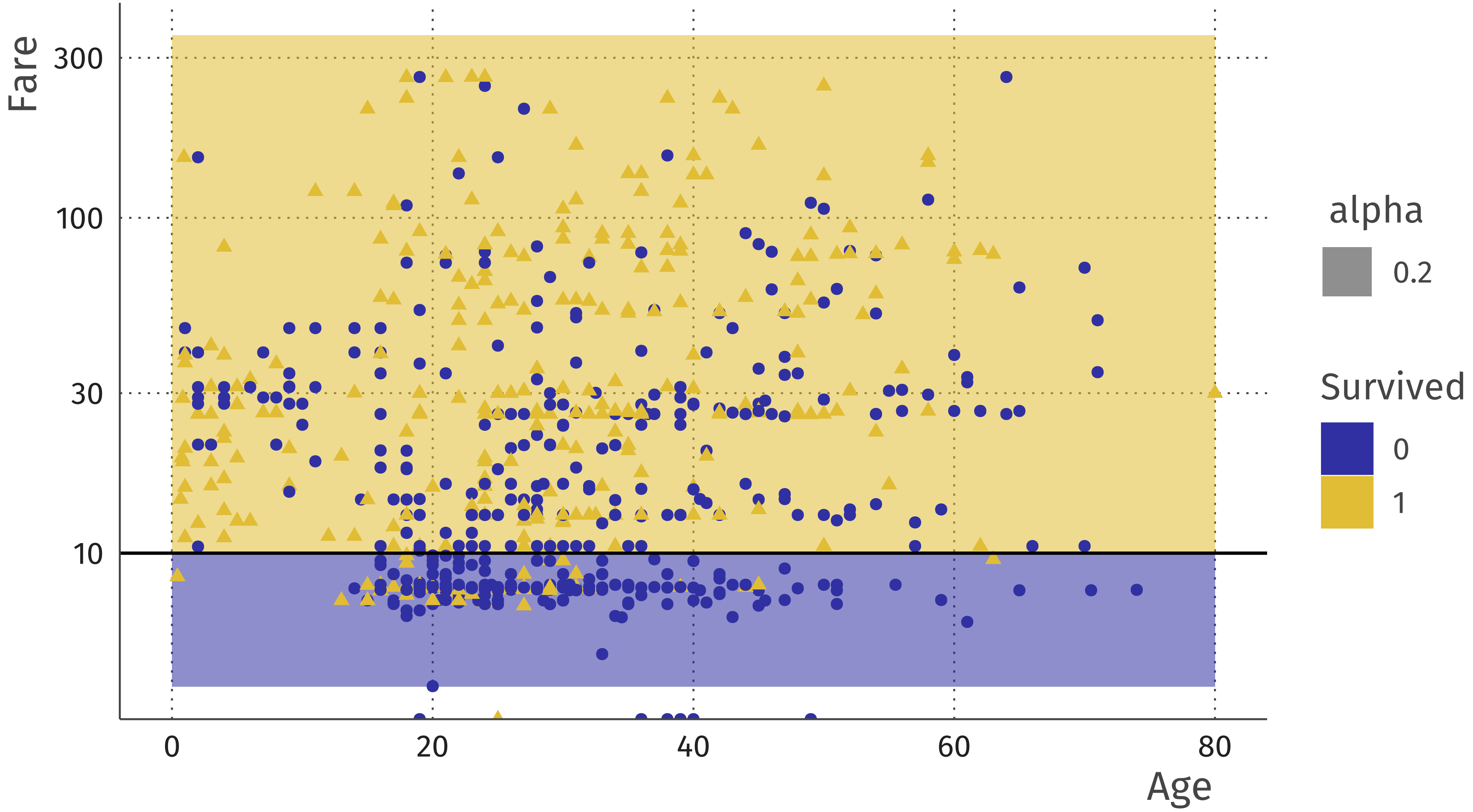
Recursive partitioning

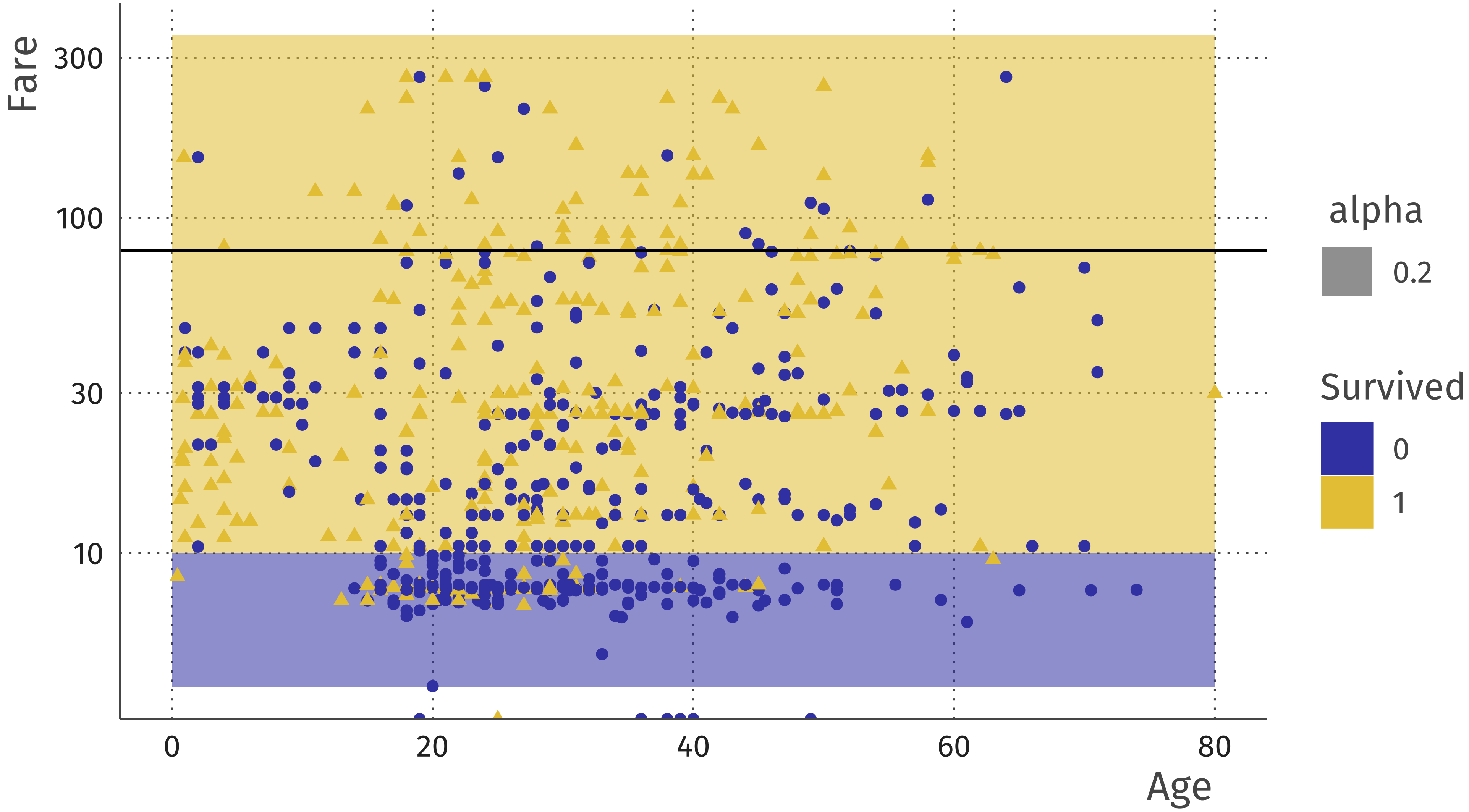
1. Find the split that makes observations as similar as possible on the outcome within that split;
2. Within each resulting group, do (1).

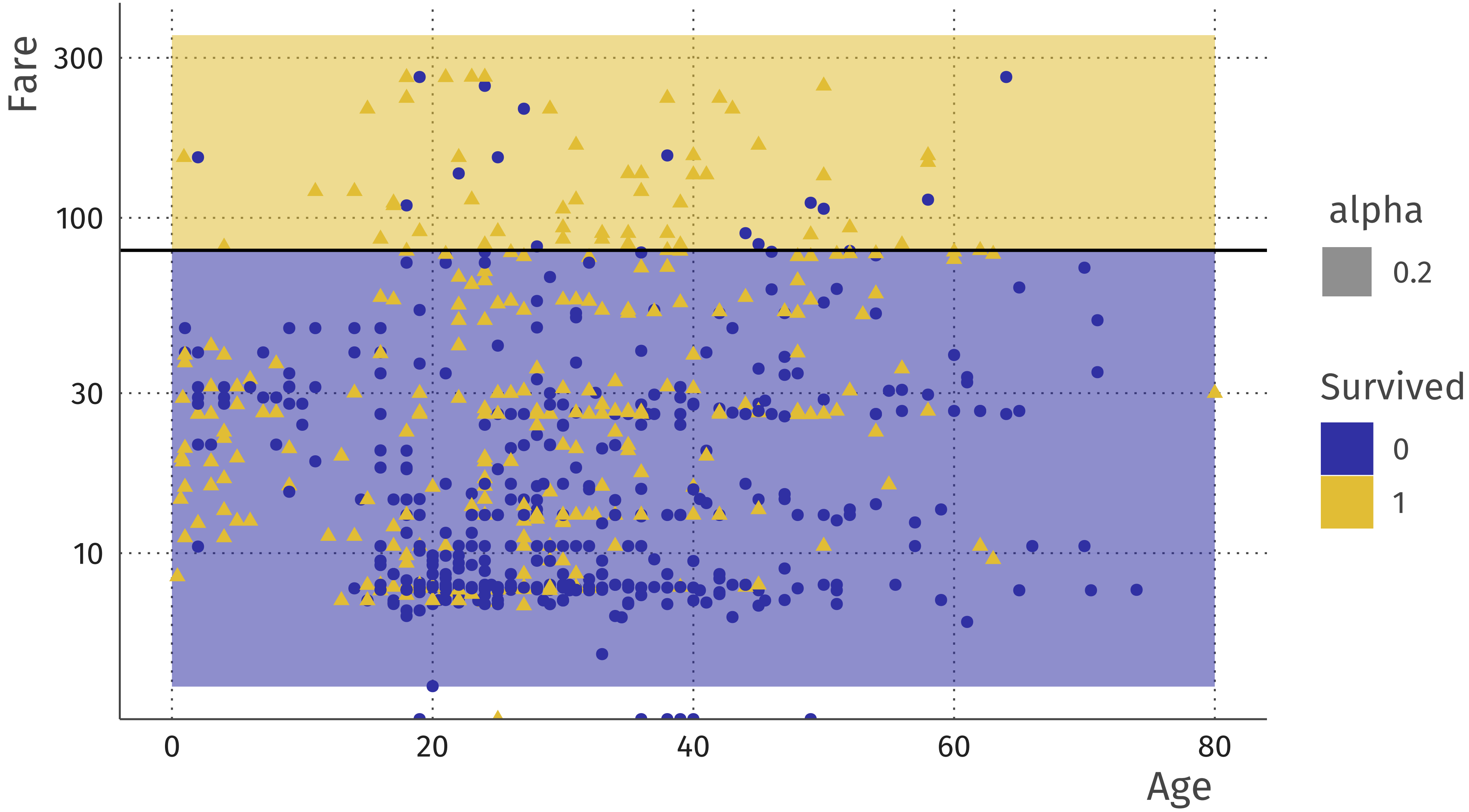
- **Criteria** for “as similar as possible”: Purity, MSE reduction, ...
- **Early stopping**: add after (2):
  - “unless fewer than `nmin` observations in the group” (typically 10);
  - “unless improvement less than `cp`” (typically 0.05);

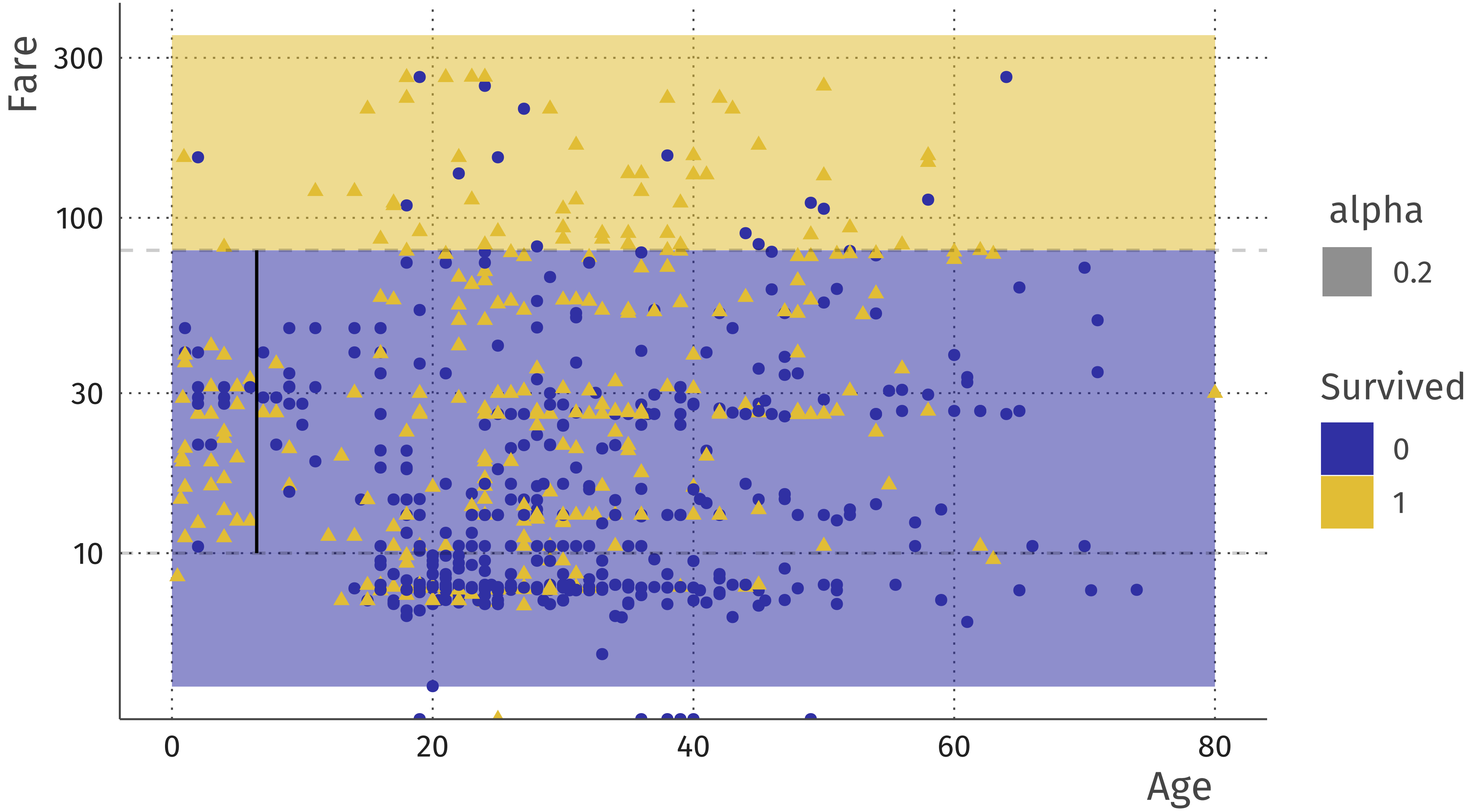

Fare
300
100
30
10
0
20
40
60
80
Age
Survived
● 0
▲ 1

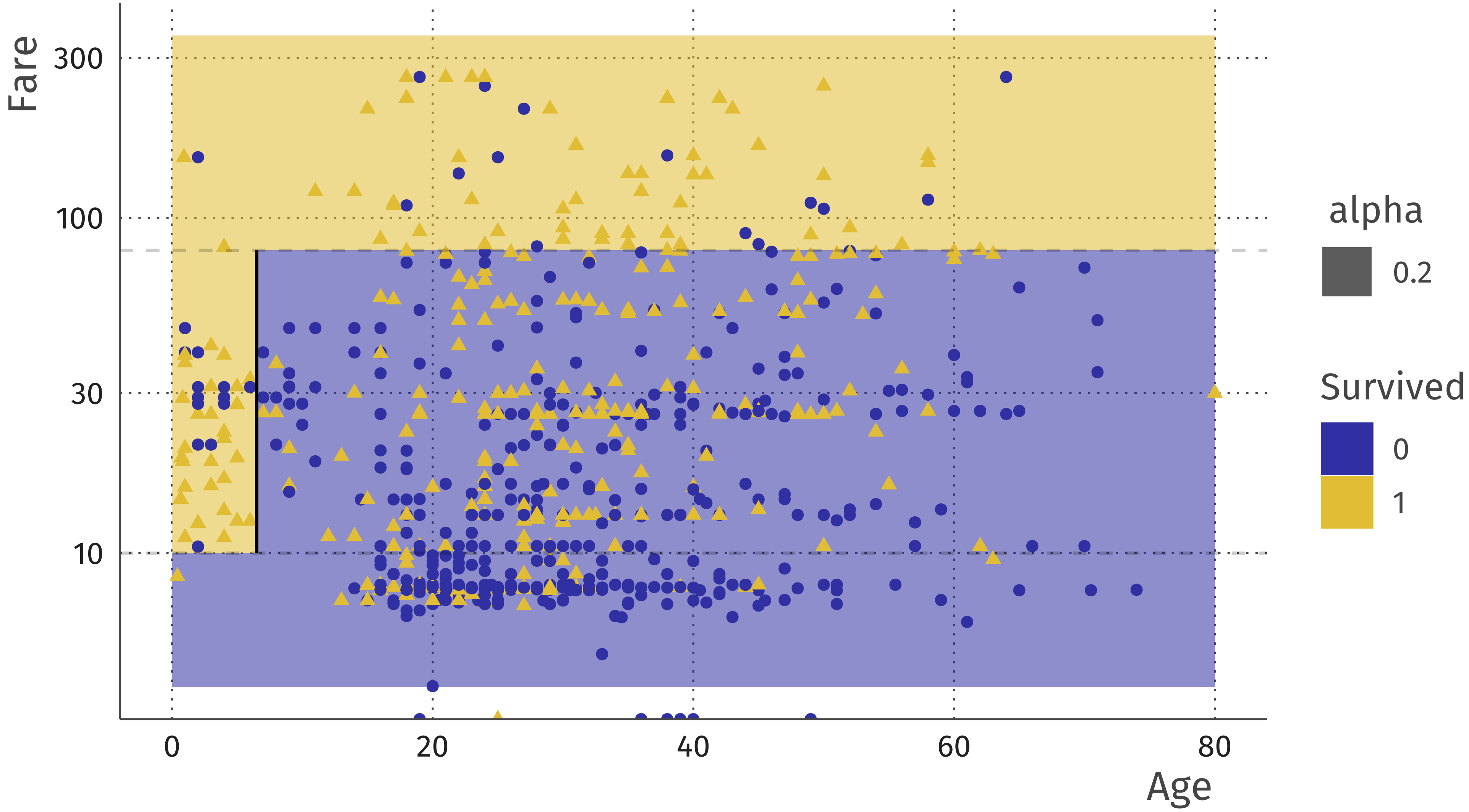

Fare
300
100
30
10
0
20
40
60
80
Age
Survived
● 0
▲ 1


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

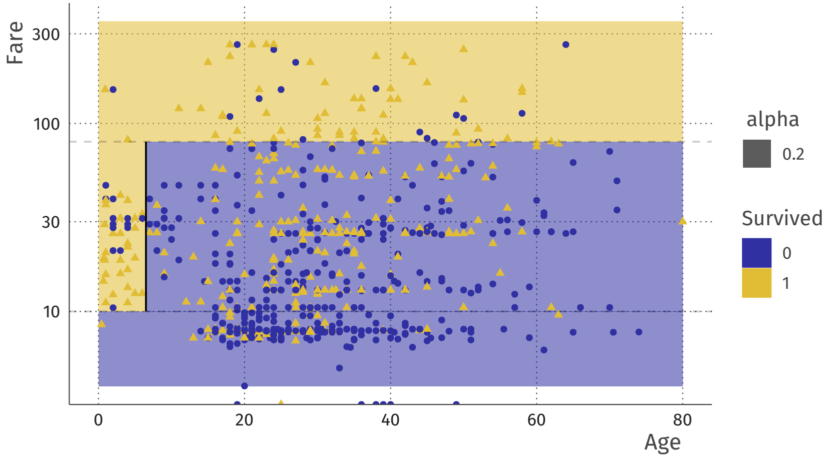

Fare
300
100
30
10
0
20
40
60
80
Age
alpha
0.2
Survived
0
1

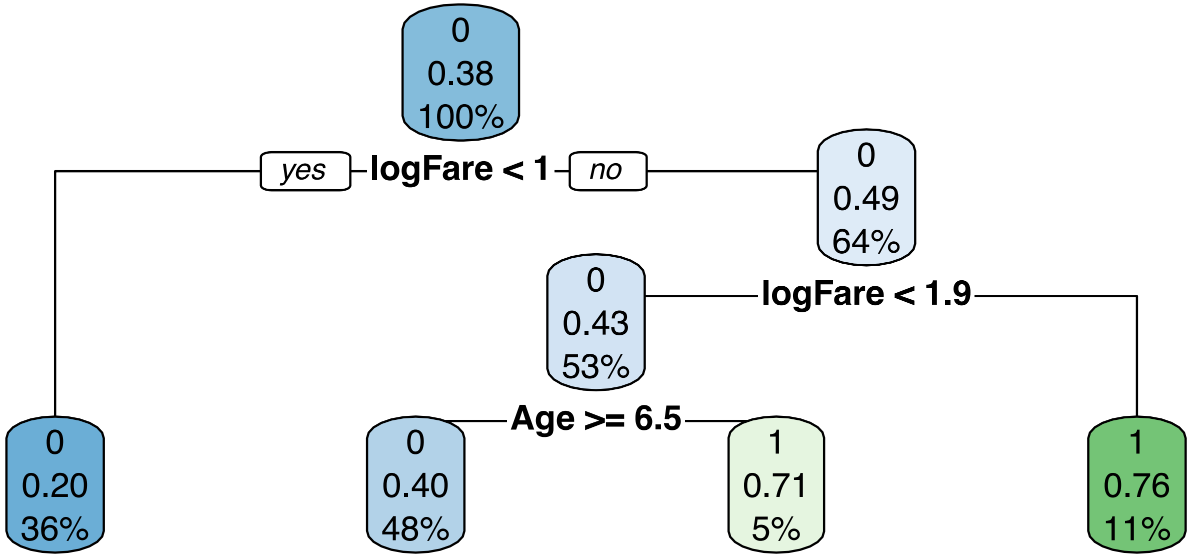

Fare
300
100
30
10
0
20
40
60
80
Age
alpha
0.2
Survived
0
1


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

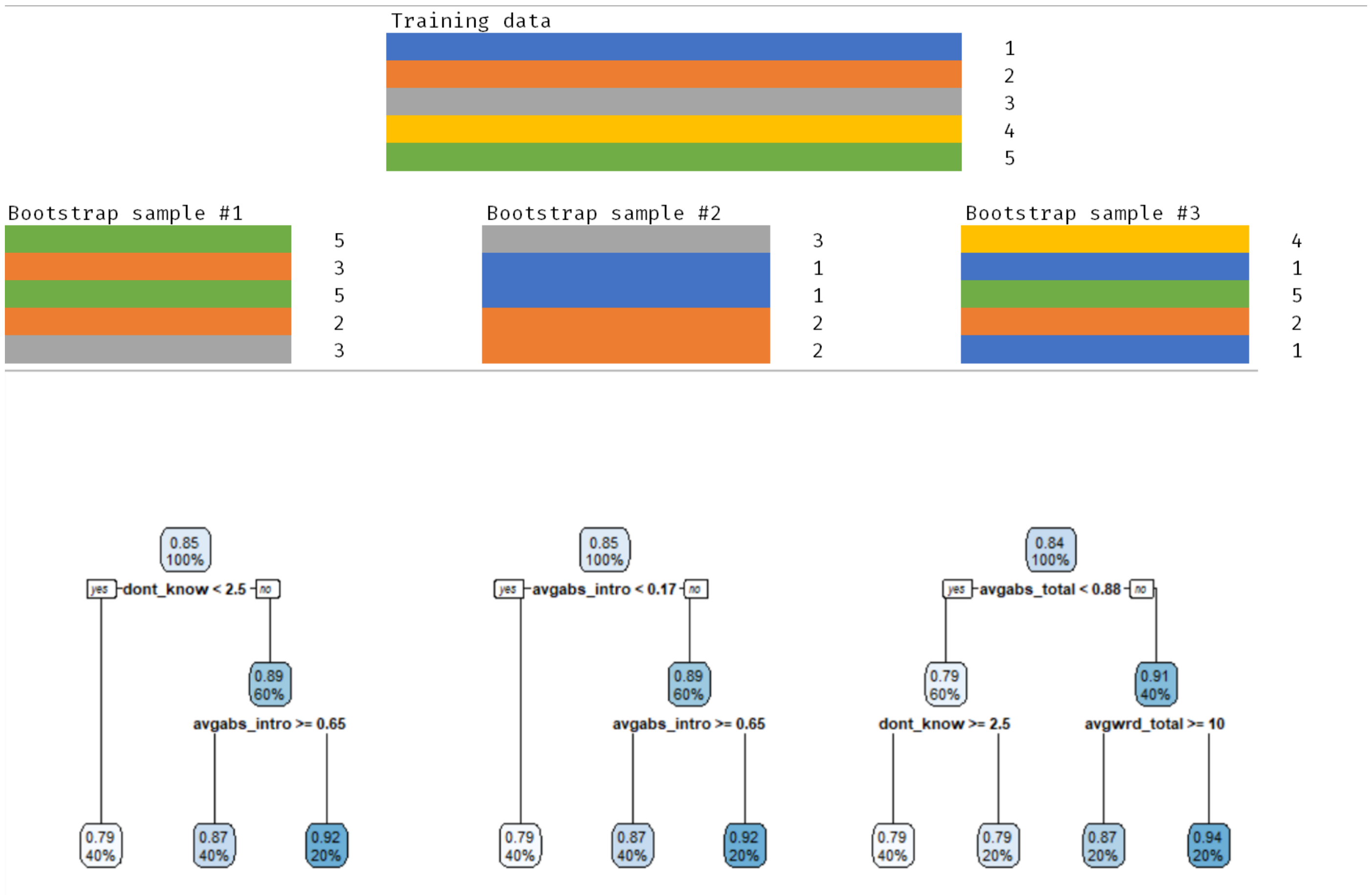
The problem with trees

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

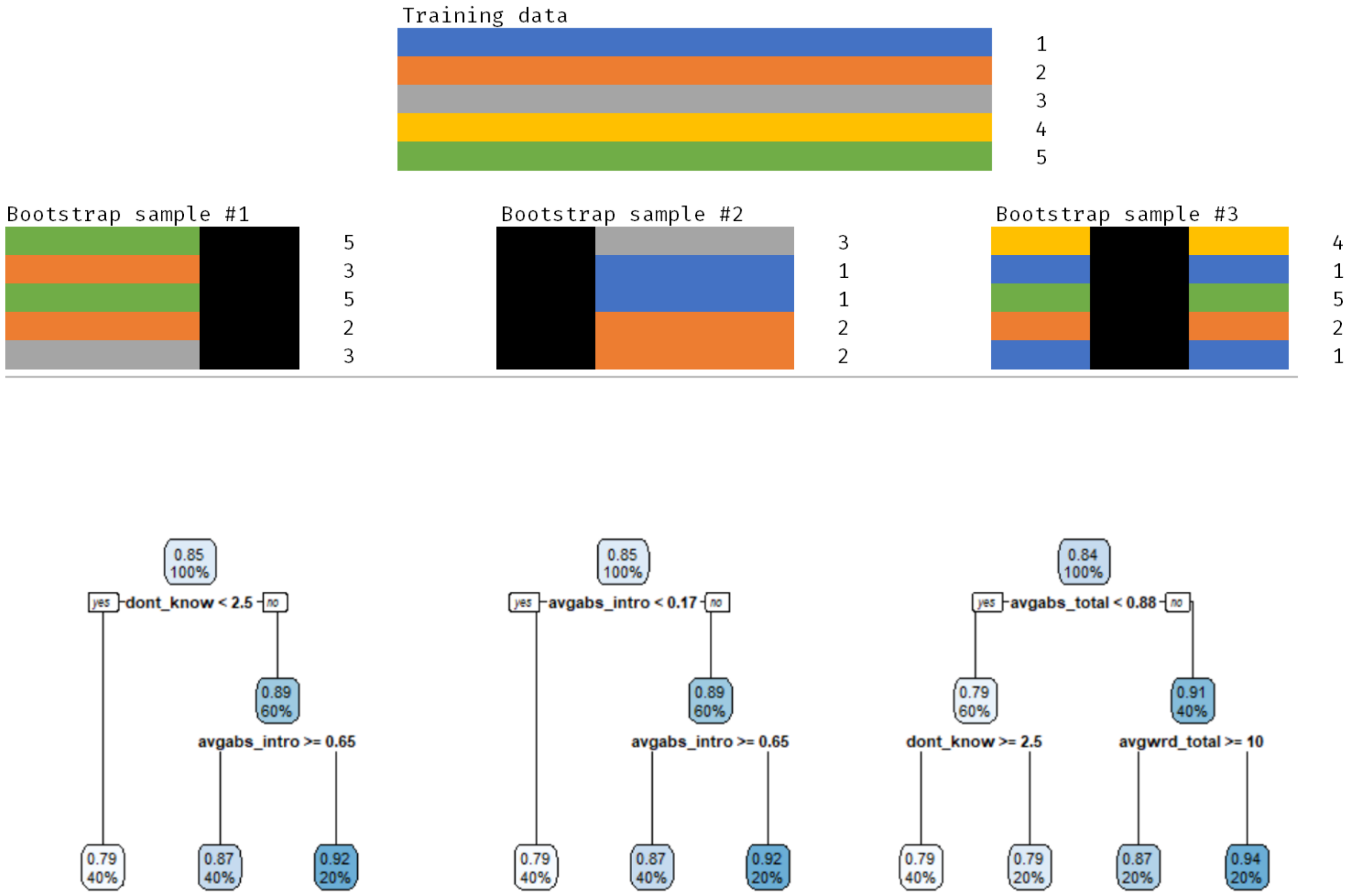
- **Random forests**: “bagged trees with feature sampling”
    - Make trees that are too complex (low bias, high variance);
    - Average over bootstrapped samples to cancel out the overfitting parts.

- **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.

Bagged trees



Random forests:
Bagged trees with feature sampling



Training data

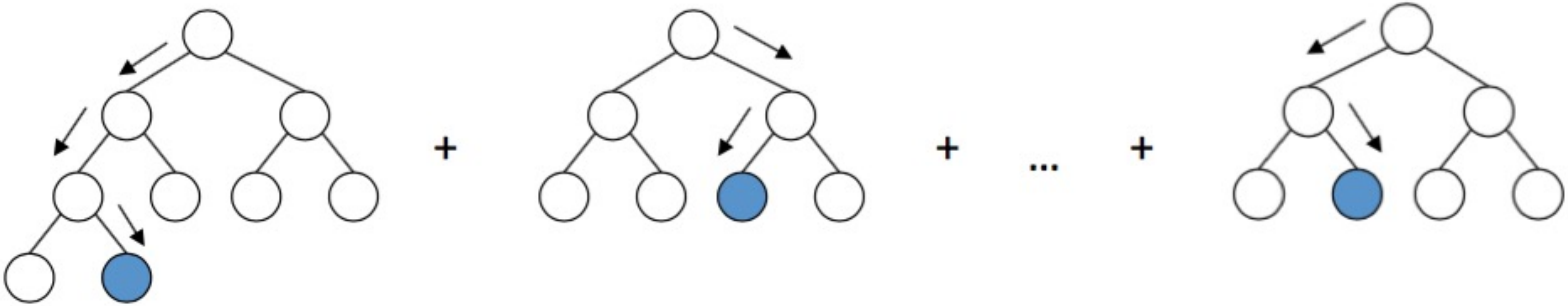
Bootstrap sample #1

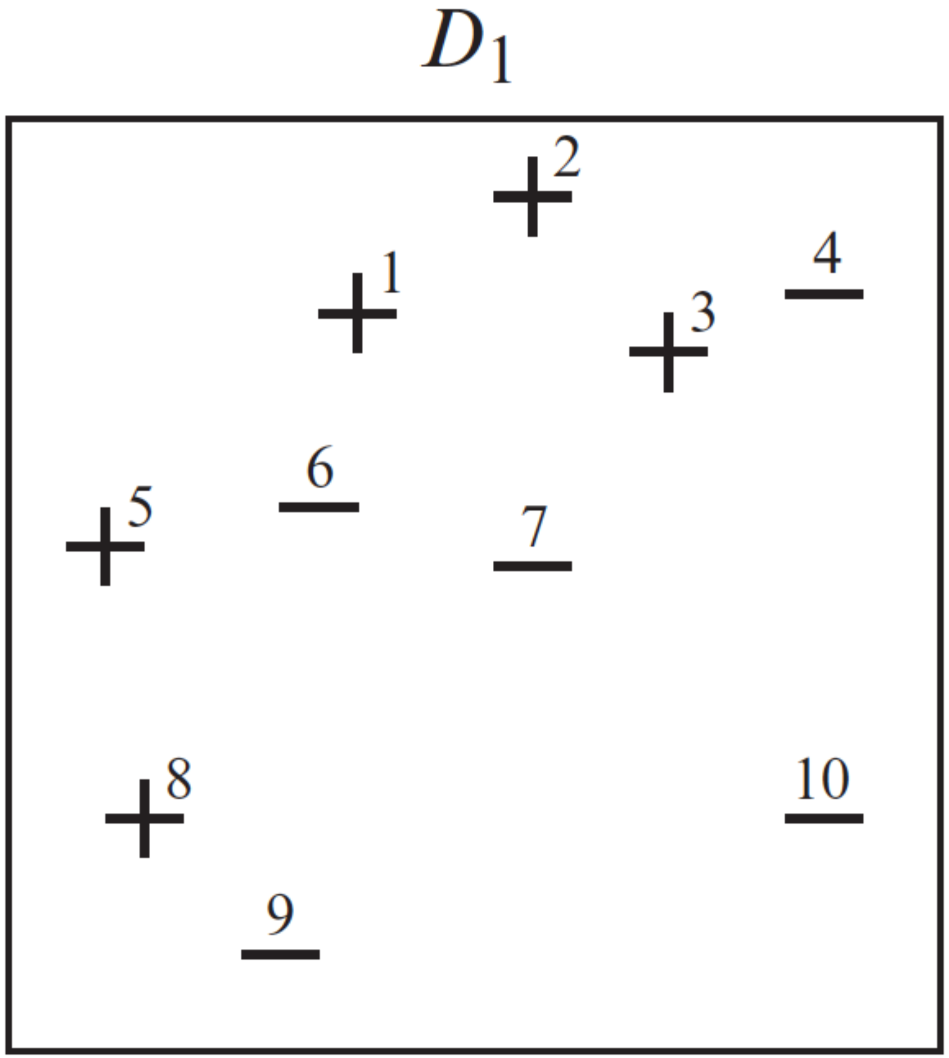
Bootstrap sample #2

Bootstrap sample #3

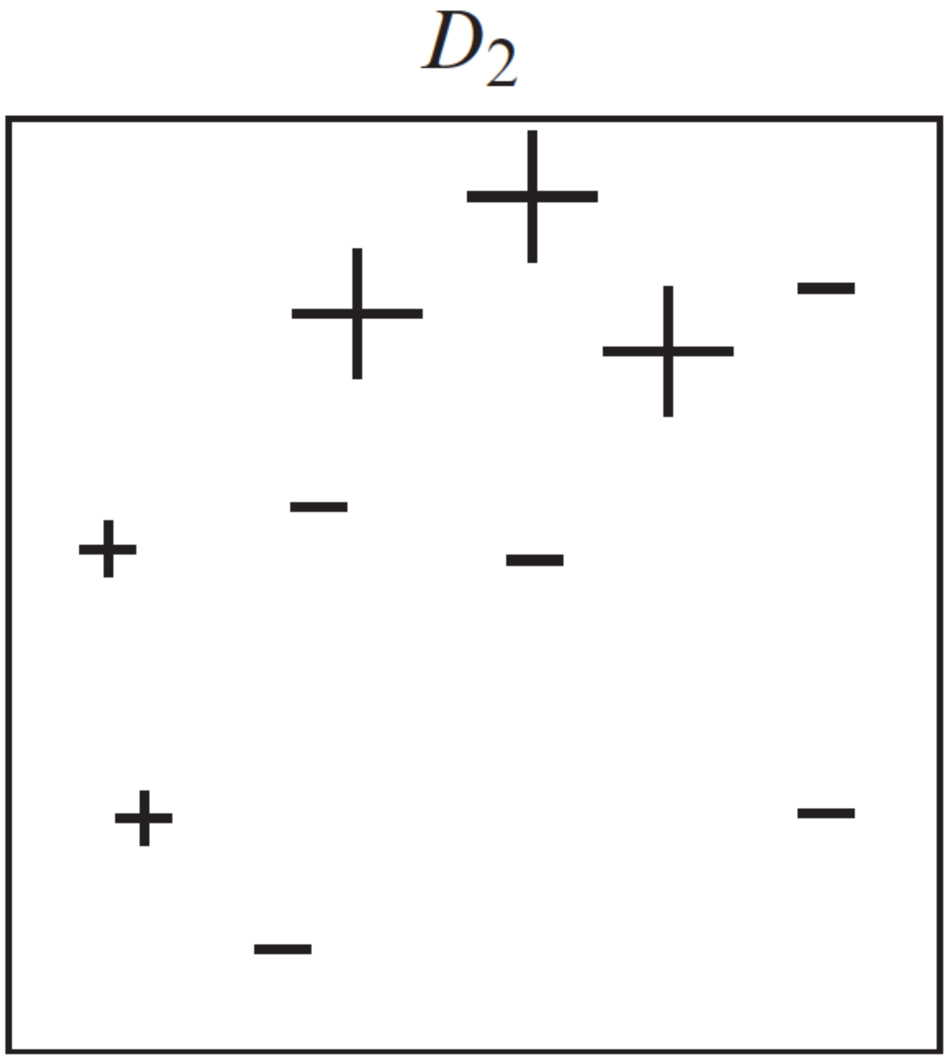
Boosting

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

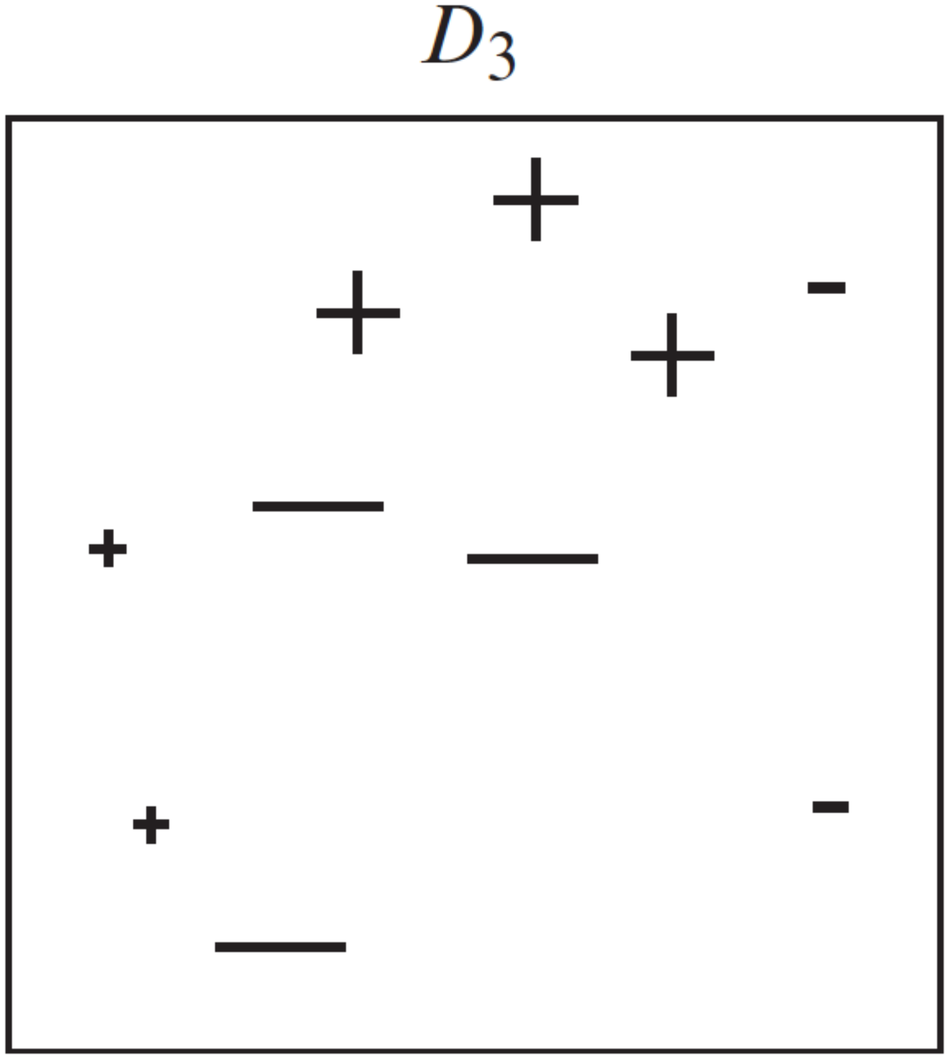




Source: Schapire & Freund (2012). *Boosting: Foundations and Algorithms.*



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



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

The final prediction model (classifier)

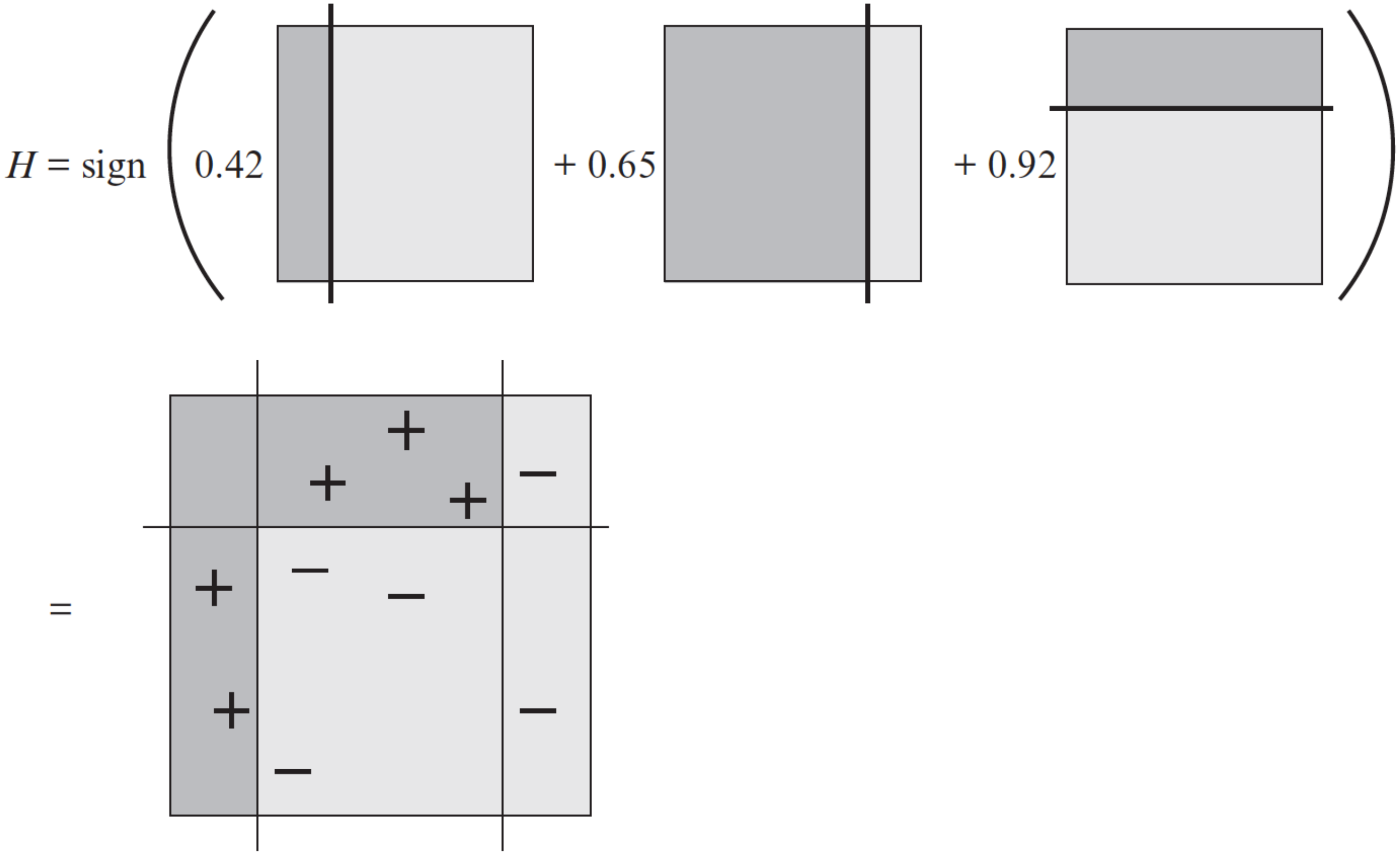


Figure 1.2
The combined classifier for the toy example of figure 1.1 is computed as the sign of the weighted sum of the three weak hypotheses, $\alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3$, as shown at the top. This is equivalent to the classifier shown at the bottom. (As in figure 1.1, the regions that a classifier predicts positive are indicated using darker shading.)

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

Tuning

- Learners have “hyperparameters”
- Example: number of trees in RF
- General idea is to use cross-validation to select hyperparameters
- Some models more sensitive to good choice of hyperparamters
- Examples are boosting and neural nets

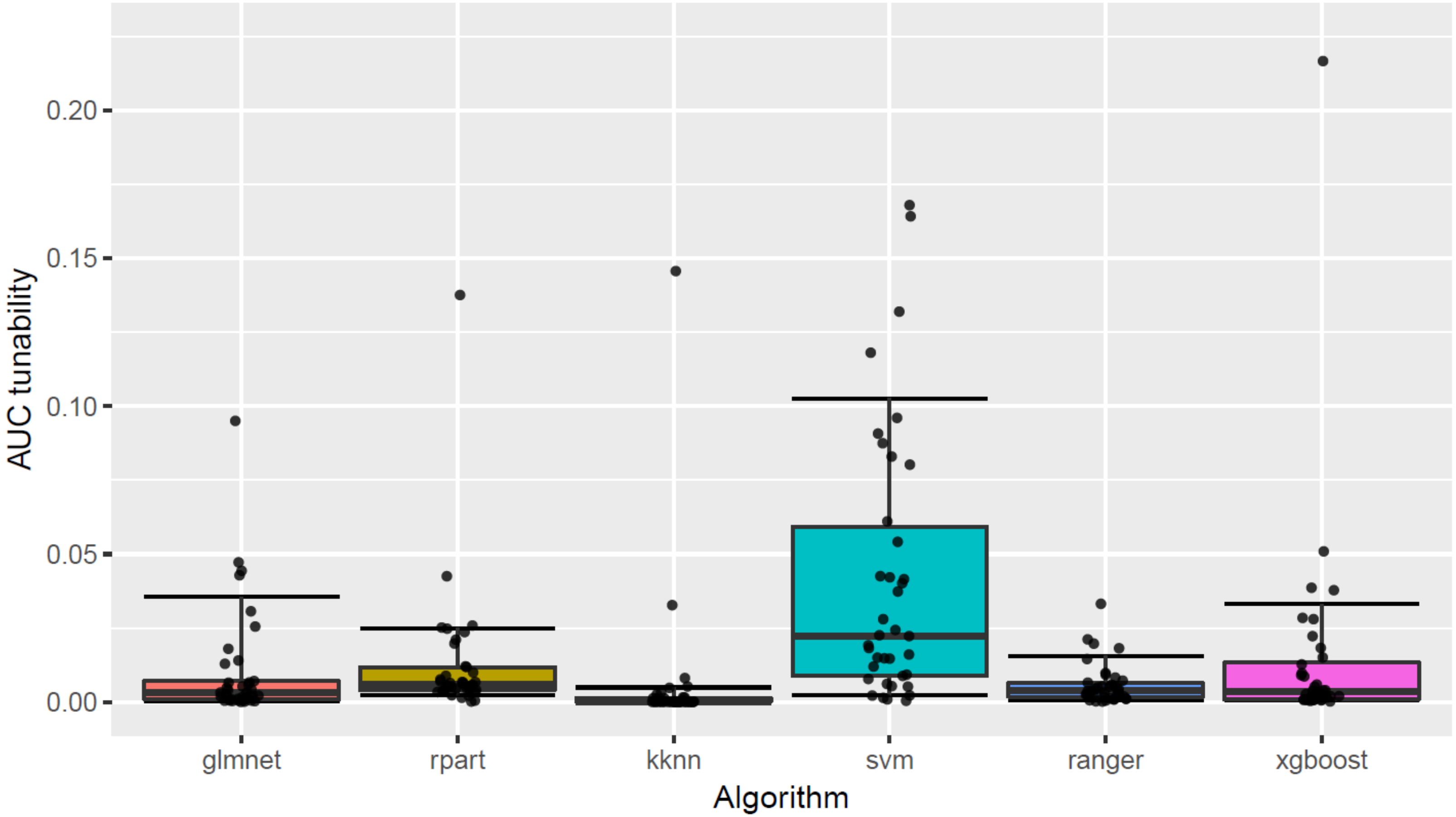
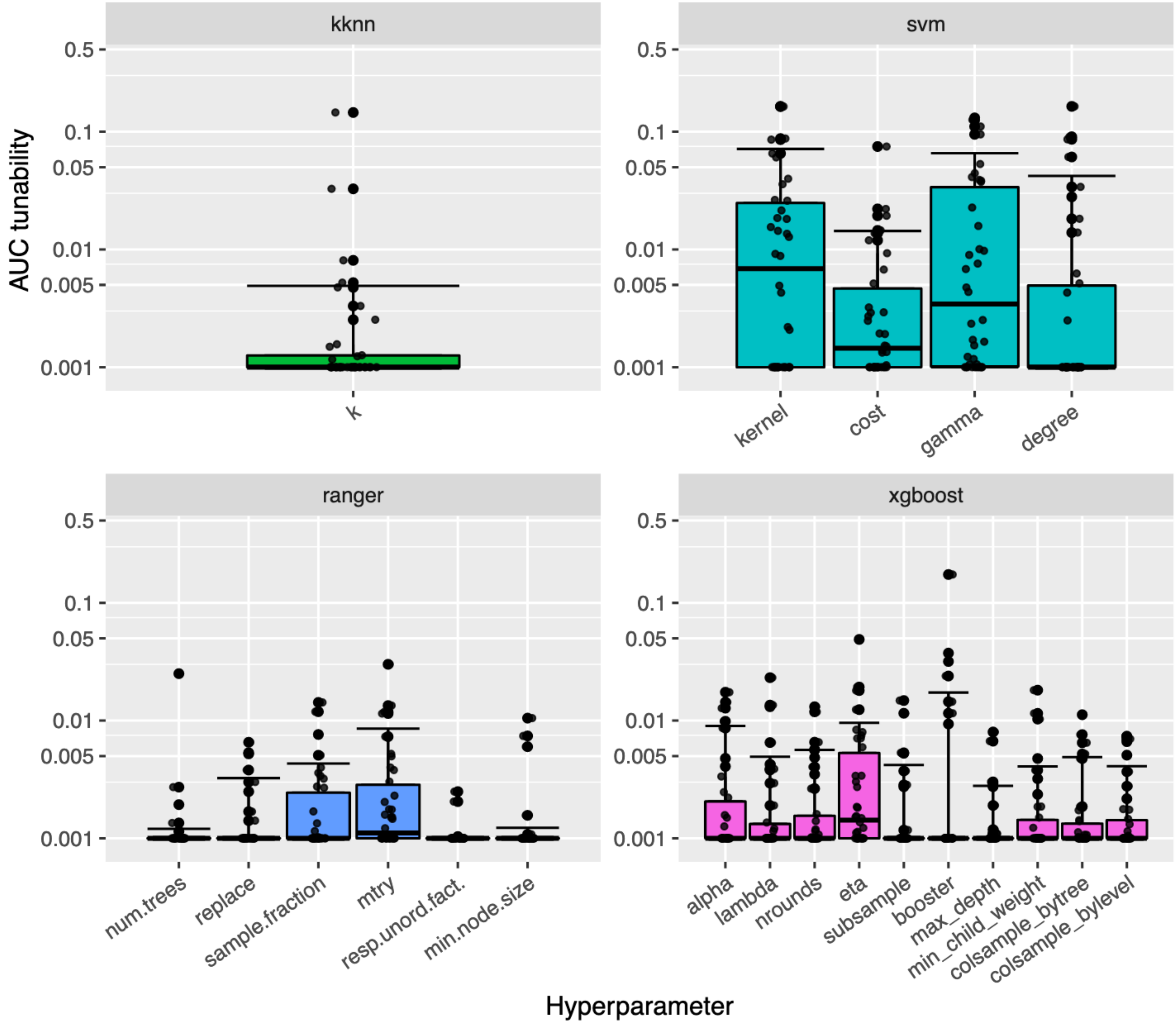


Figure 2: Boxplots of the tunabilities (AUC) of the different algorithms with respect to optimal defaults. The upper and lower whiskers (upper and lower line of the boxplot rectangle) are in our case defined as the 0.1 and 0.9 quantiles of the tunability scores. The 0.9 quantile indicates how much performance improvement can be expected on at least 10% of datasets. One outlier of `glmnet` (value 0.5) is not shown.

Probst et al. JMLR



Probst et al. (2019). JMLR

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

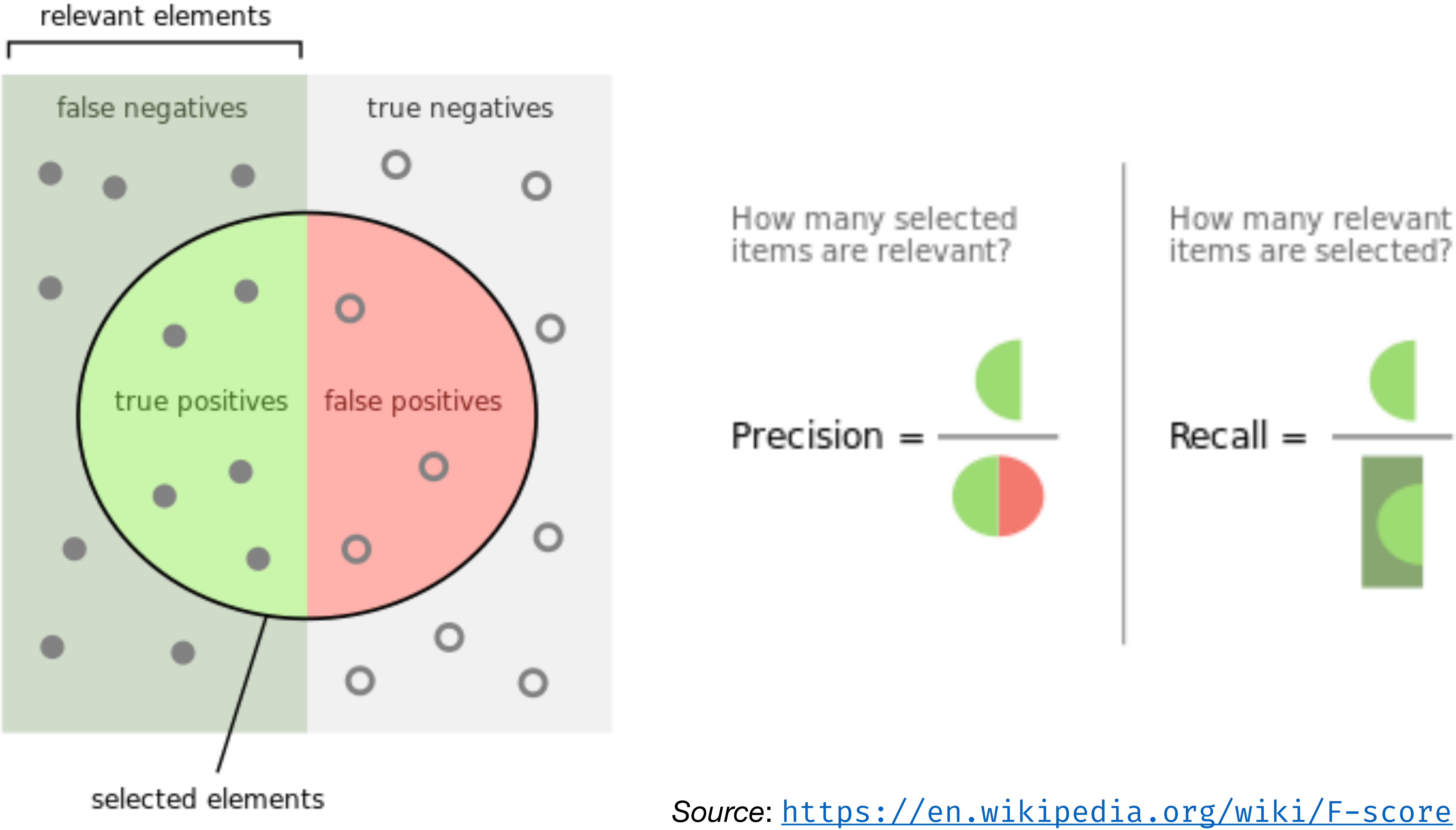
- Specificity: $\frac{\text{TN}}{\text{TN+FP}}$ = 134 / (134 + 19) ≈ 0.876
- Sensitivity (“recall”): $\frac{\text{TP}}{\text{TP+FN}}$ = 75 / (75 + 40) ≈ 0.652
- Accuracy (ACC): $\frac{\text{TP+TN}}{\text{TP+FP+TN+FN}} \approx 0.780$
- Error rate: $1 -$ Accuracy ≈ 0.220

Confusion matrix:
Negative & positive predictive value

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

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

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



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

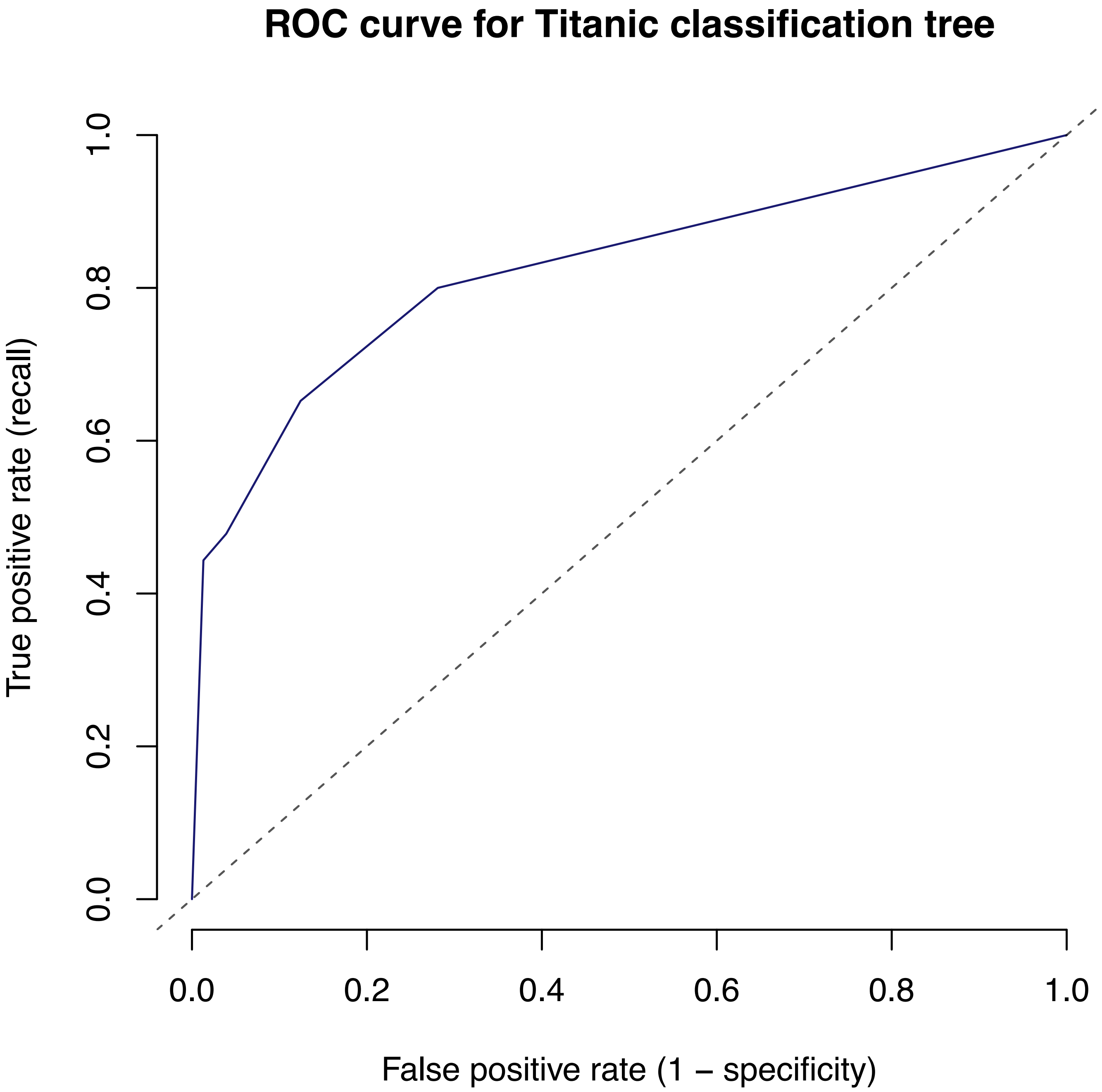
Overview: some classification metrics

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

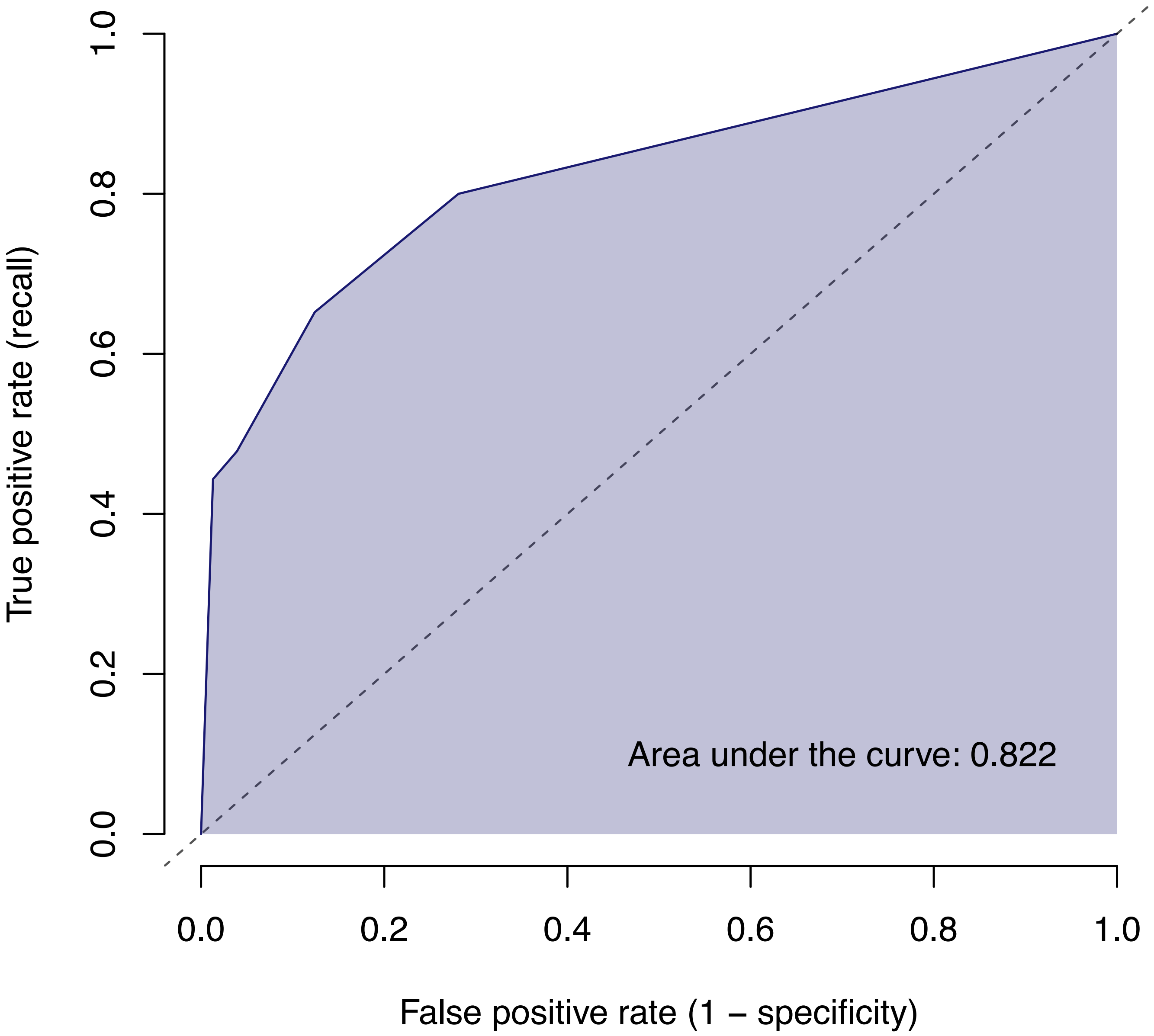
Different thresholds than 0.5

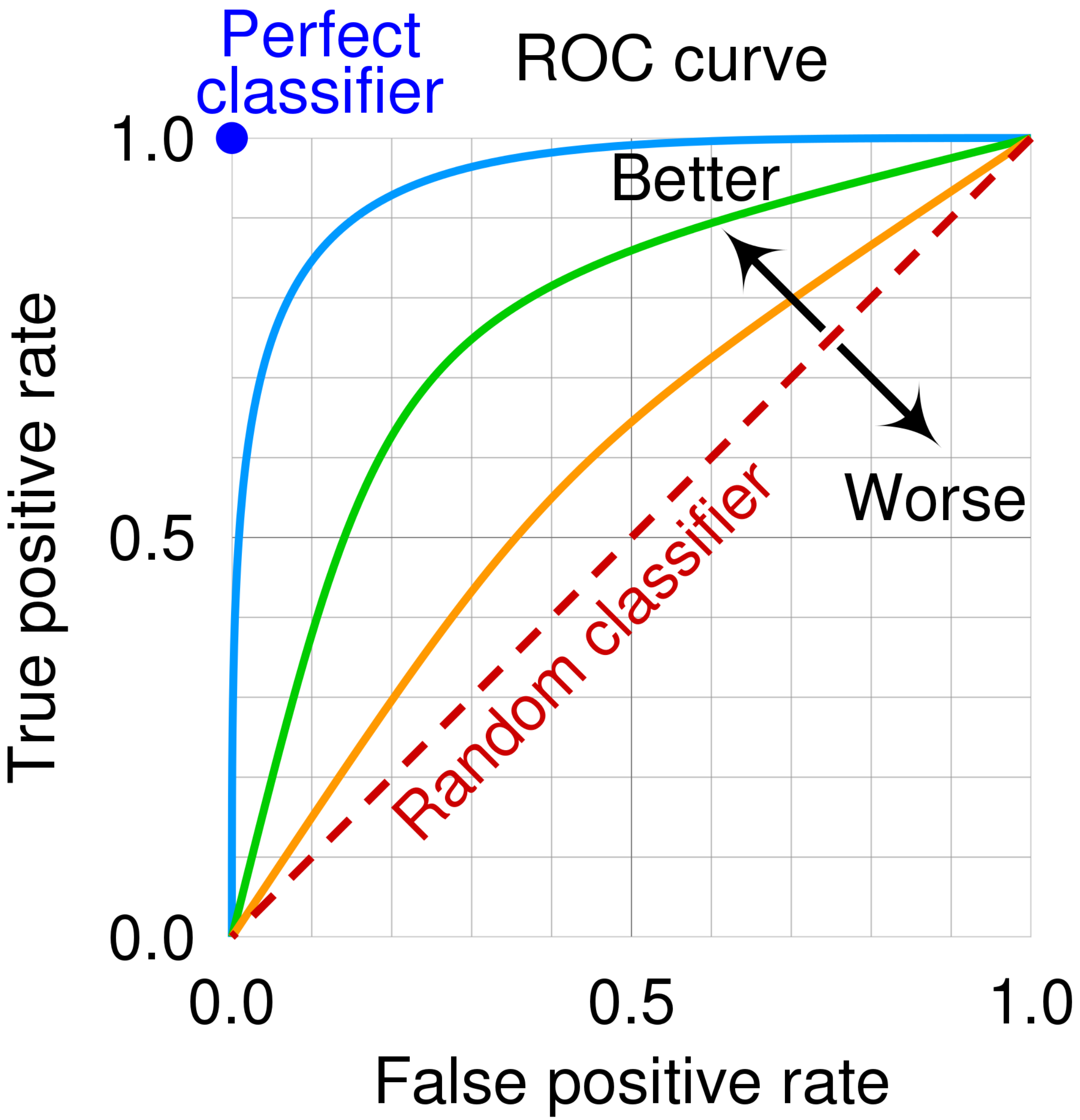
Moving around the threshold affects sensitivity and specificity!

```
> 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

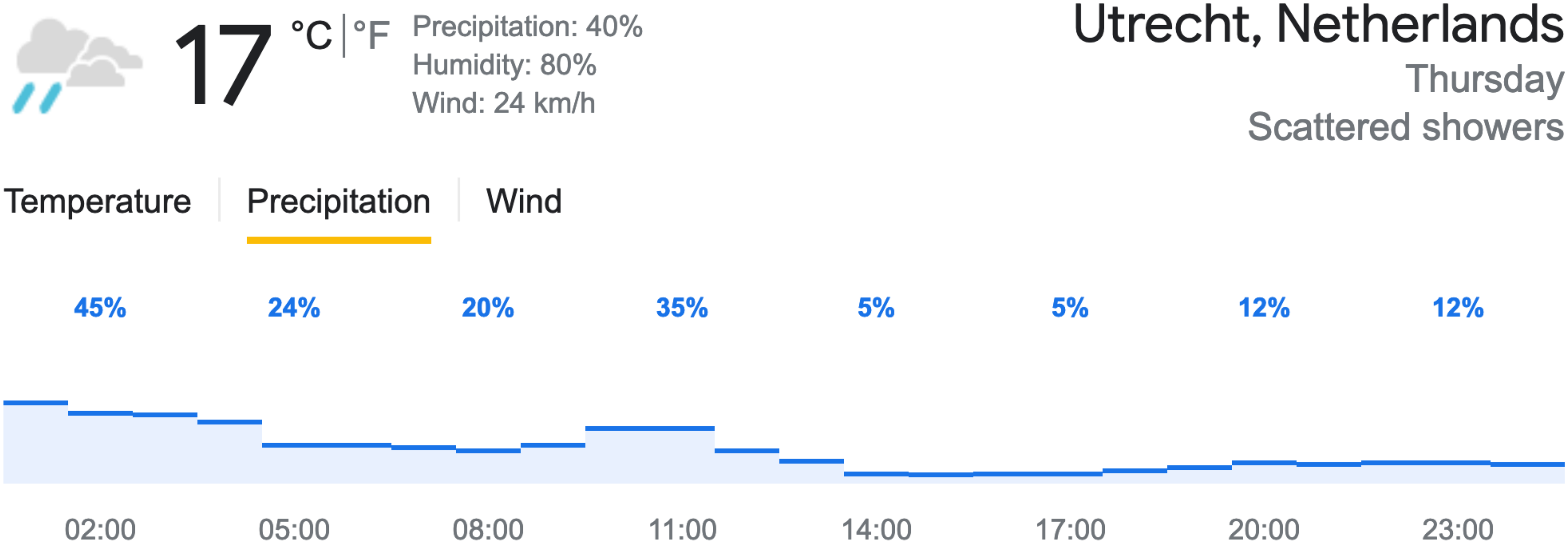
ROC curve for Titanic classification tree




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

- 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?



17 °C | °F

Precipitation: 40%
Humidity: 80%
Wind: 24 km/h

Utrecht, Netherlands
Thursday
Scattered showers

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**”.

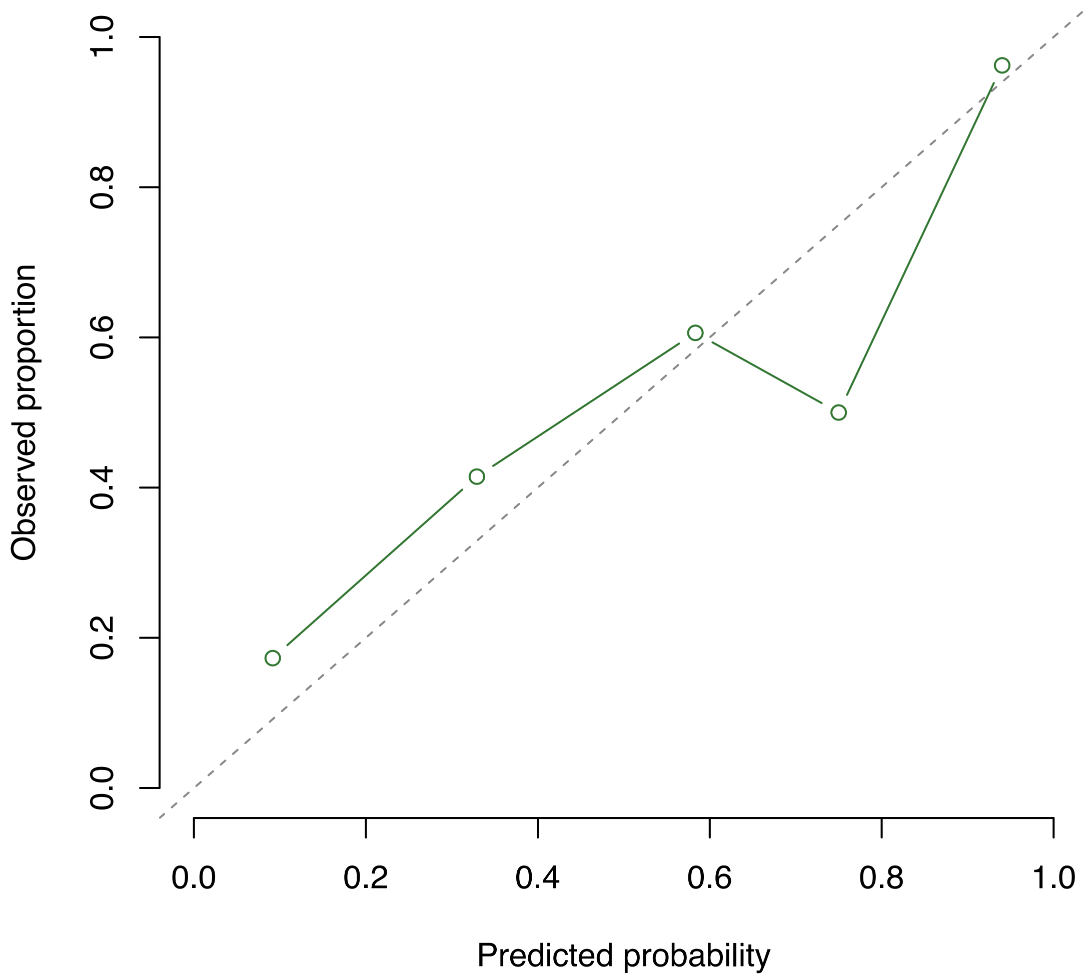
Calibration plot

Probability

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

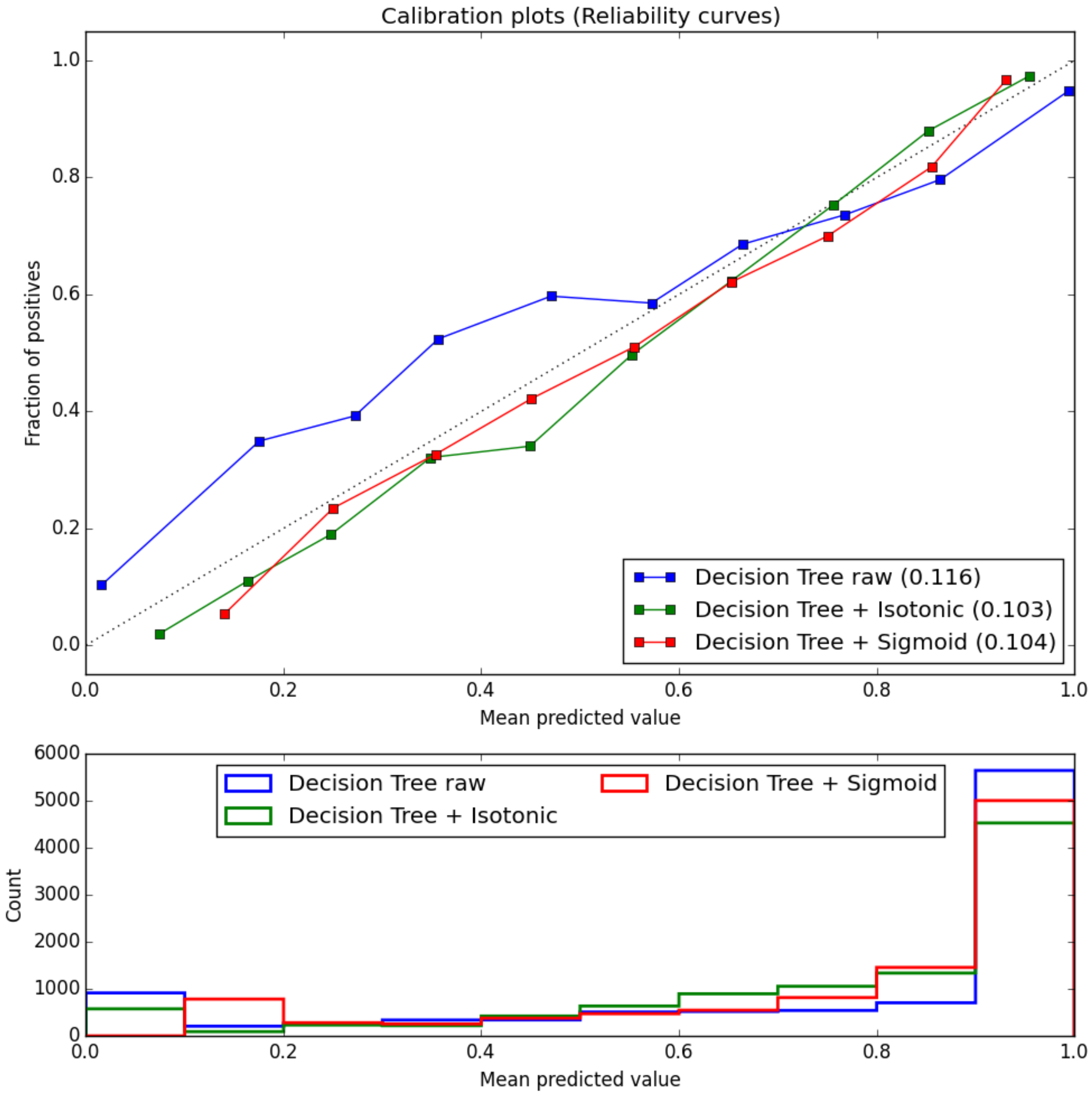
- A predicted probability is **calibrated** when it conforms to the definition above;
- Check this using a **calibration plot**

Calibration of Titanic classification



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
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted value
Decision Tree raw (0.116)
Decision Tree + Isotonic (0.103)
Decision Tree + Sigmoid (0.104)
Count
0
1000
2000
3000
4000
5000
6000
Decision Tree raw
Decision Tree + Isotonic
Decision Tree + Sigmoid
Mean predicted value

MSE (“Brier score”)

Setting $y = 1$ for Yes and $y = 0$ for No, then the “average” $\mathrm{E}(.)$ of y is the true proportion p, $\mathrm{E}(y) = p$.

We can again evaluate the Mean Square Error (MSE), now called “Brier score” of our guess, $\hat{p}$:

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

Turns out MSE can be reworked into two terms:

MSE = Calibration term + AUC term

OPTIONAL!! Not assessed.

MSE can be reworked into two terms

Suppose there are K bins of unique(ish) values for our prediction, $\hat{p}$.

Then the **calibration score** C is

$$C = n^{-1} \sum_{k=1}^{K} n_k \left(\hat{p}_k - p\right)^2$$

The calibration score is the squared deviation in the calibration plot showed earlier. Perfect ($C = 0$) when predicted probabilities equal observed proportion of events.

The **discrimination** (”refinement”) score is

$$D = n^{-1} \sum_{k=1}^{K} n_k \, p_k (1 - p_k)$$

Perfect ($D = 0$) when each prediction bin corresponds to a 1 or a 0. Equivalent to AUC. Same as “Gini node impurity” used in CART tree learning.

And now:

$$\mathrm{MSE} = C + D$$

[Blattenberger & Lad 1985]

Calibration in evaluation of ML models

- Important for downstream decision-making
- Sometimes overlooked in ML model evaluation
- Evaluate using more advanced methods in standard software

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**.

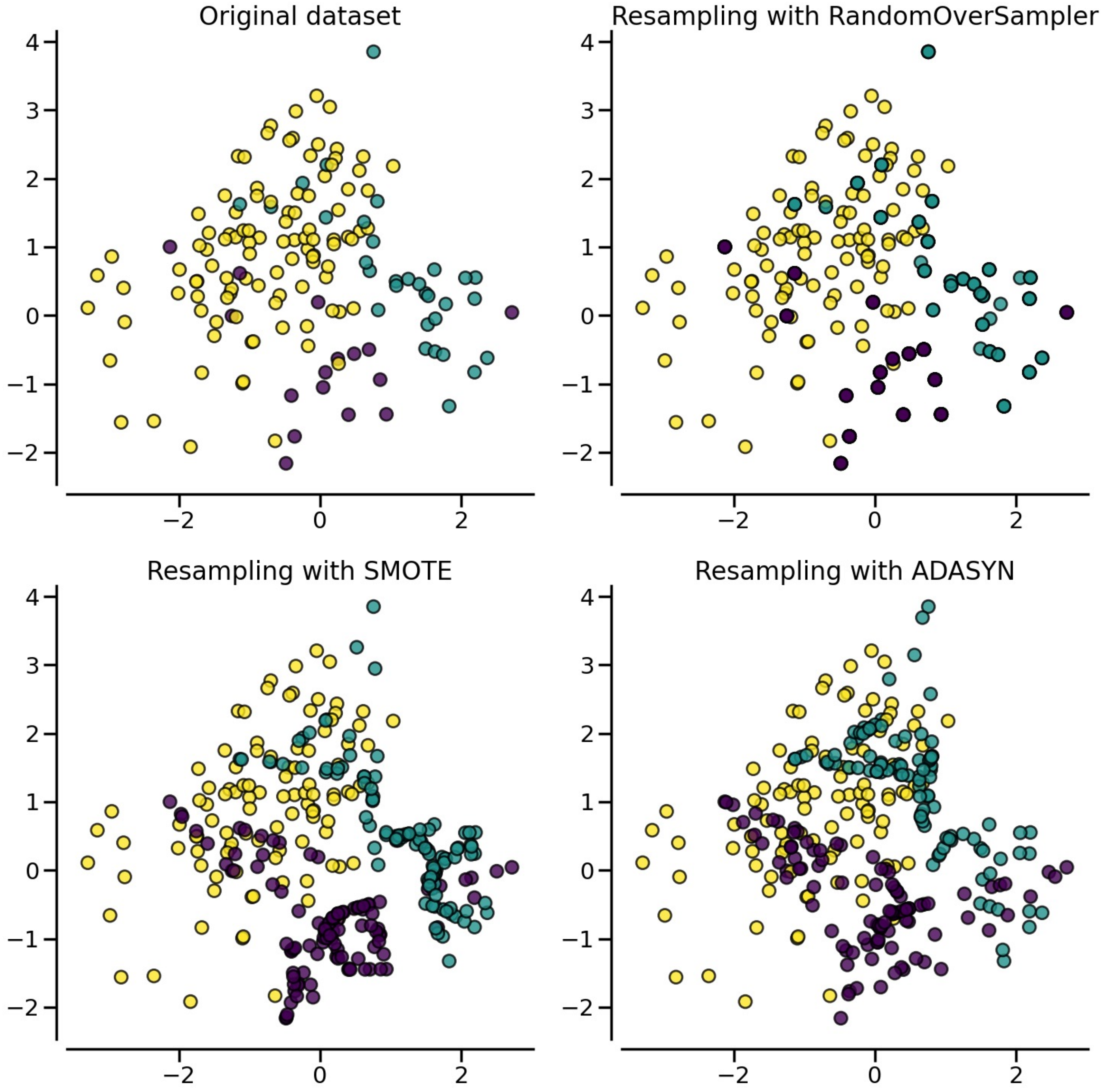
Class imbalance

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”



https://imbalanced-learn.org

Supervised learning 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
- Some of these lead to specific issues in classification (e.g. calibration, class imbalance) discussed today (& in book)
- **You should now be equipped to start using supervised learning in practice!**