Section 5

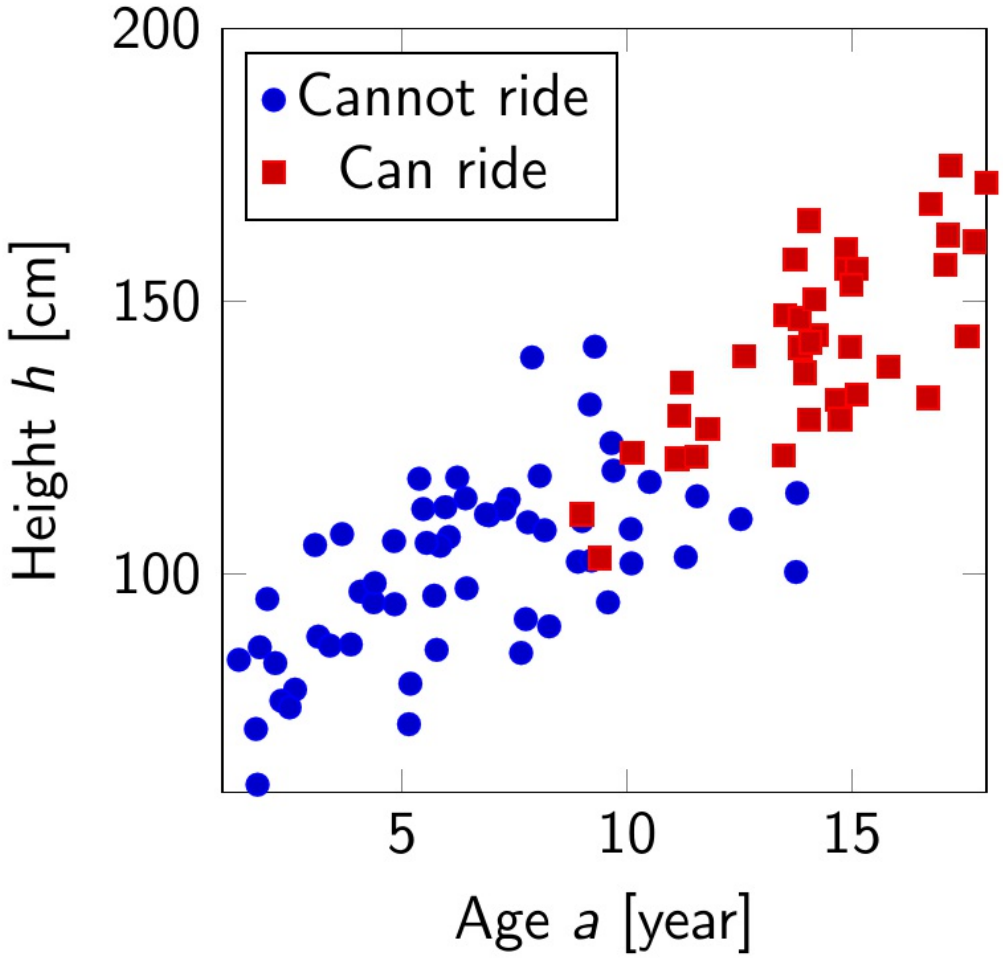
Tree-based methods

The carousel robot attendant

Problem: replace the carousel attendant with a robot which automatically decides who can ride the carousel.

Carousel: data

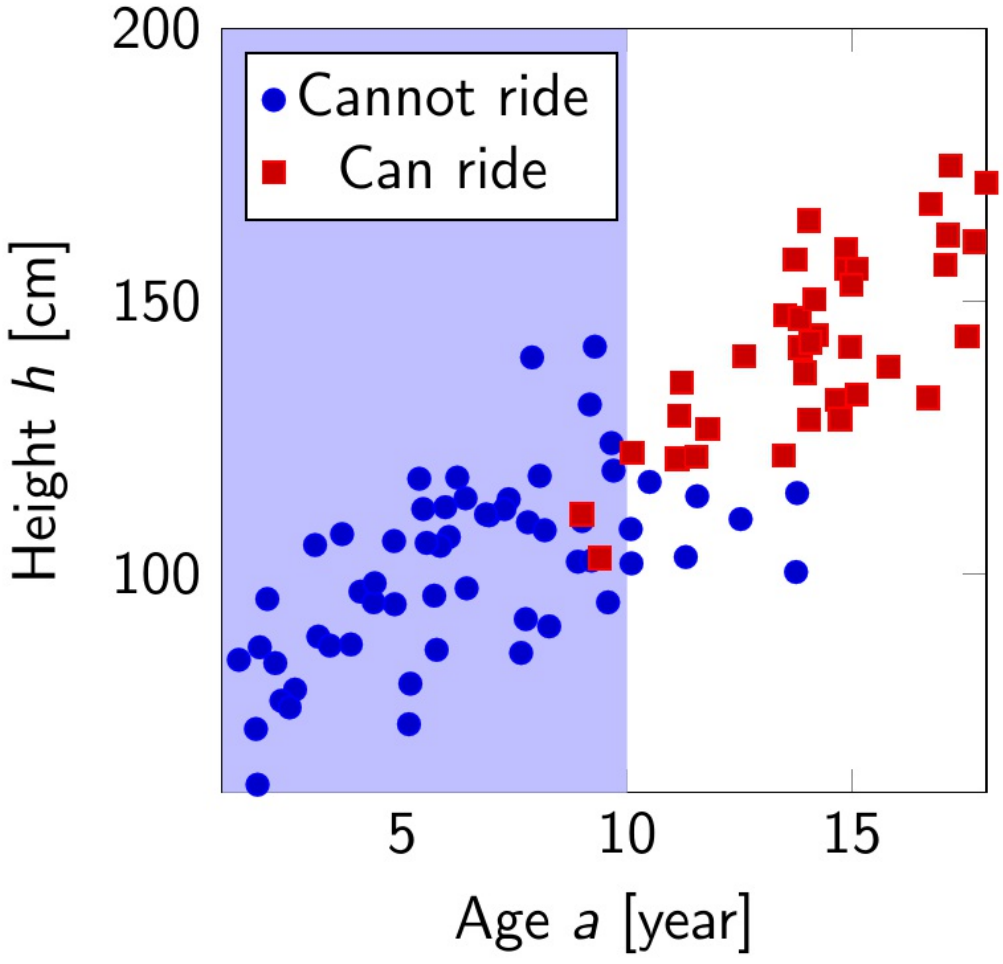
Observed human attendant's decisions.



How can the robot take the decision?

Carousel: data

Observed human attendant's decisions.

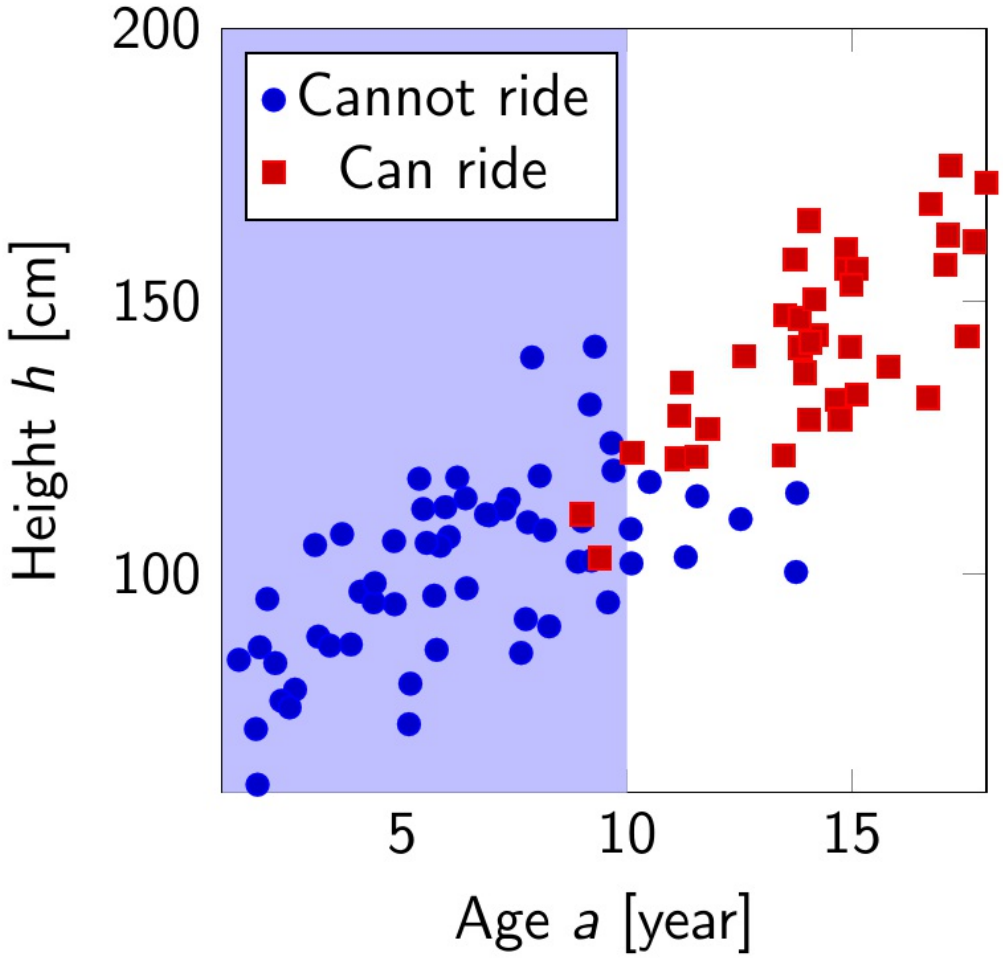


How can the robot take the decision?

- if younger than 10 $\rightarrow$ can't!

Carousel: data

Observed human attendant's decisions.

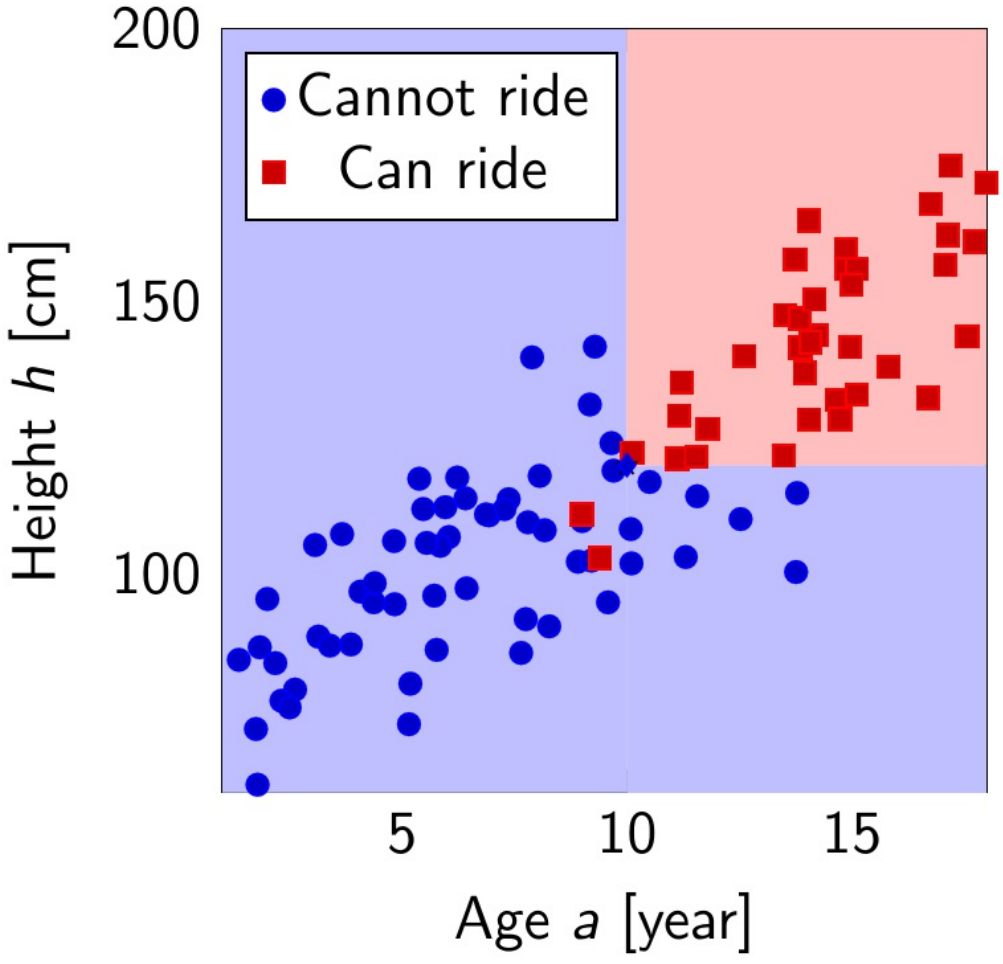


How can the robot take the decision?

- if younger than 10 $\rightarrow$ can't!
- otherwise:

Carousel: data

Observed human attendant's decisions.

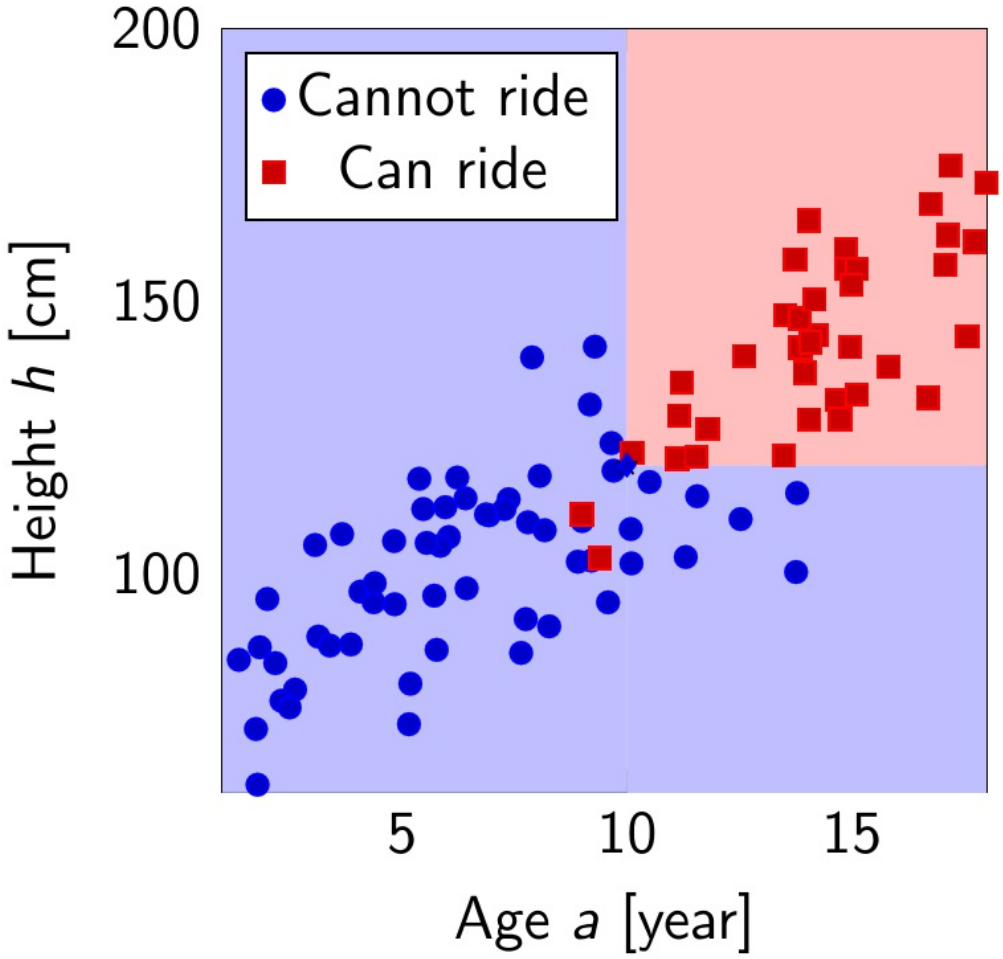


How can the robot take the decision?

- if younger than 10 $\rightarrow$ can't!
- otherwise:
  - if shorter than 120 $\rightarrow$ can't!
  - otherwise $\rightarrow$ can!

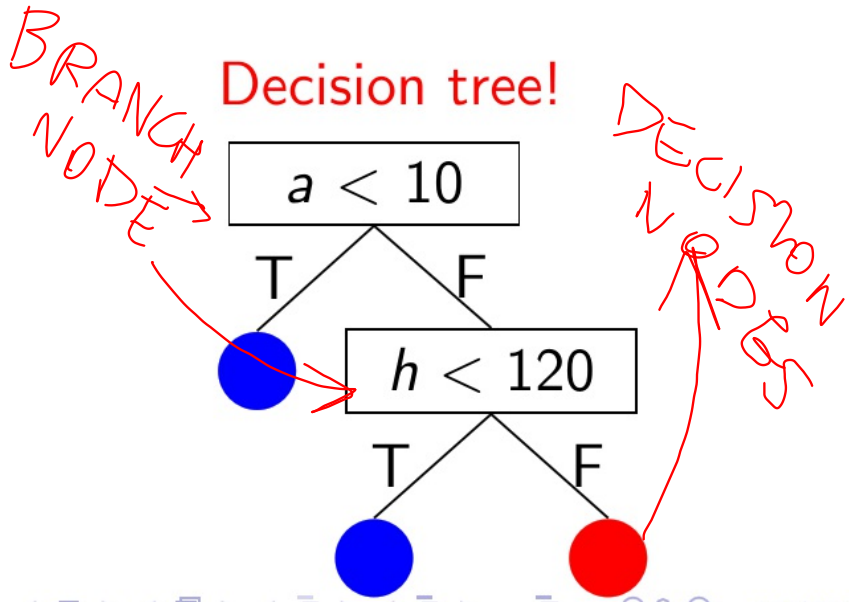
Carousel: data

Observed human attendant's decisions.

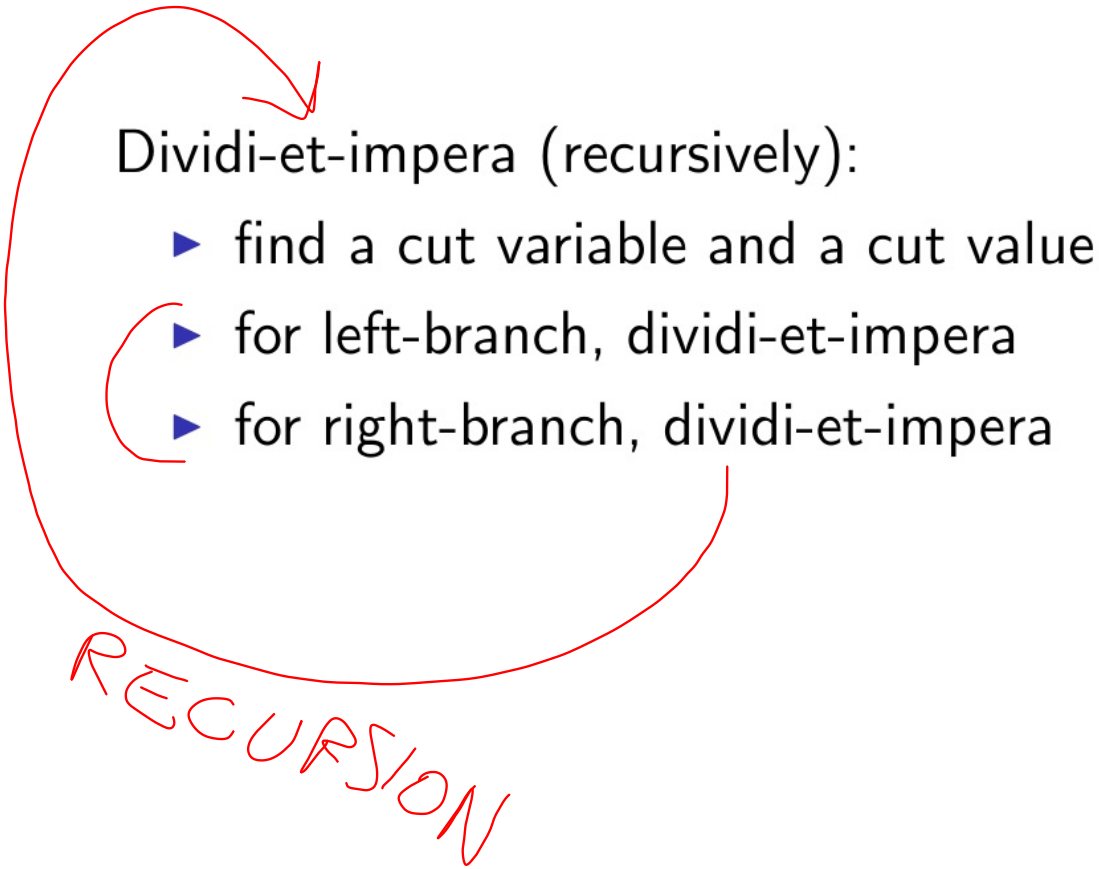


How can the robot take the decision?

- if younger than 10 → can't!
- otherwise:
  - if shorter than 120 → can't!
  - otherwise → can!



How to build a decision tree



How to build a decision tree: detail

Recursive binary splitting

function BUILDDECISIONTREE($\mathbf{X}, \mathbf{y}$)
  if SHOULDSTOP($\mathbf{y}$) **then**
    $\hat{y} \leftarrow$ most common class in $\mathbf{y}$
    return new terminal node with $\hat{y}$
  else
    $(i, t) \leftarrow$ BESTBRANCH($\mathbf{X}, \mathbf{y}$)
    $n \leftarrow$ new branch node with (i, t)
    append child BUILDDECISIONTREE($\mathbf{X}|_{\mathbf{x}_i < t}, \mathbf{y}|_{\mathbf{x}_i < t}$) to n
    append child BUILDDECISIONTREE($\mathbf{X}|_{\mathbf{x}_i \geq t}, \mathbf{y}|_{\mathbf{x}_i \geq t}$) to n
    return n
  end if
end function

- Recursive binary splitting
- Top down (start from the "big" problem)

Best branch

function BESTBRANCH($\mathbf{X}, \mathbf{y}$)
  $(i^\star, t^\star) \leftarrow \arg\min_{i,t} E(\mathbf{y}|_{\mathbf{x}_i \geq t}) + E(\mathbf{y}|_{\mathbf{x}_i < t})$
  return $(i^\star, t^\star)$
end function

Classification error on subset:

$$E(\mathbf{y}) = \frac{|\{y \in \mathbf{y} : y \neq \hat{y}\}|}{|\mathbf{y}|}$$

$\hat{y}$ = the most common class in $\mathbf{y}$

- Greedy (choose split to minimize error now, not in later steps)

Best branch

$$(i^\star, t^\star) \leftarrow \arg\min_{i,t} E(\mathbf{y}|_{\mathbf{x}_i \geq t}) + E(\mathbf{y}|_{\mathbf{x}_i < t})$$

The formula say what is done, not how is done!

Q: different "how" can differ? how?

Stopping criterion

```
function ShouldStop(y)
    if y contains only one class then
        return true
    else if |y| < k_min then
        return true
    else
        return false
    end if
end function
```

USER-PROVIDED PARAM OF THE LEARNING ALG.

Other possible criterion:

- tree depth larger than $d_{\max}$

Best branch criteria

Classification error $E()$ works, but has been shown to be "not sufficiently sensitive for tree-growing".

$$E(\mathbf{y}) = \frac{|\{y \in \mathbf{y} : y \neq \hat{y}\}|}{|\mathbf{y}|} = 1 - \max_c \frac{|\{y \in \mathbf{y} : y = c\}|}{|\mathbf{y}|} = 1 - \max_c p_{\mathbf{y},c}$$

Other two option:

- Gini index

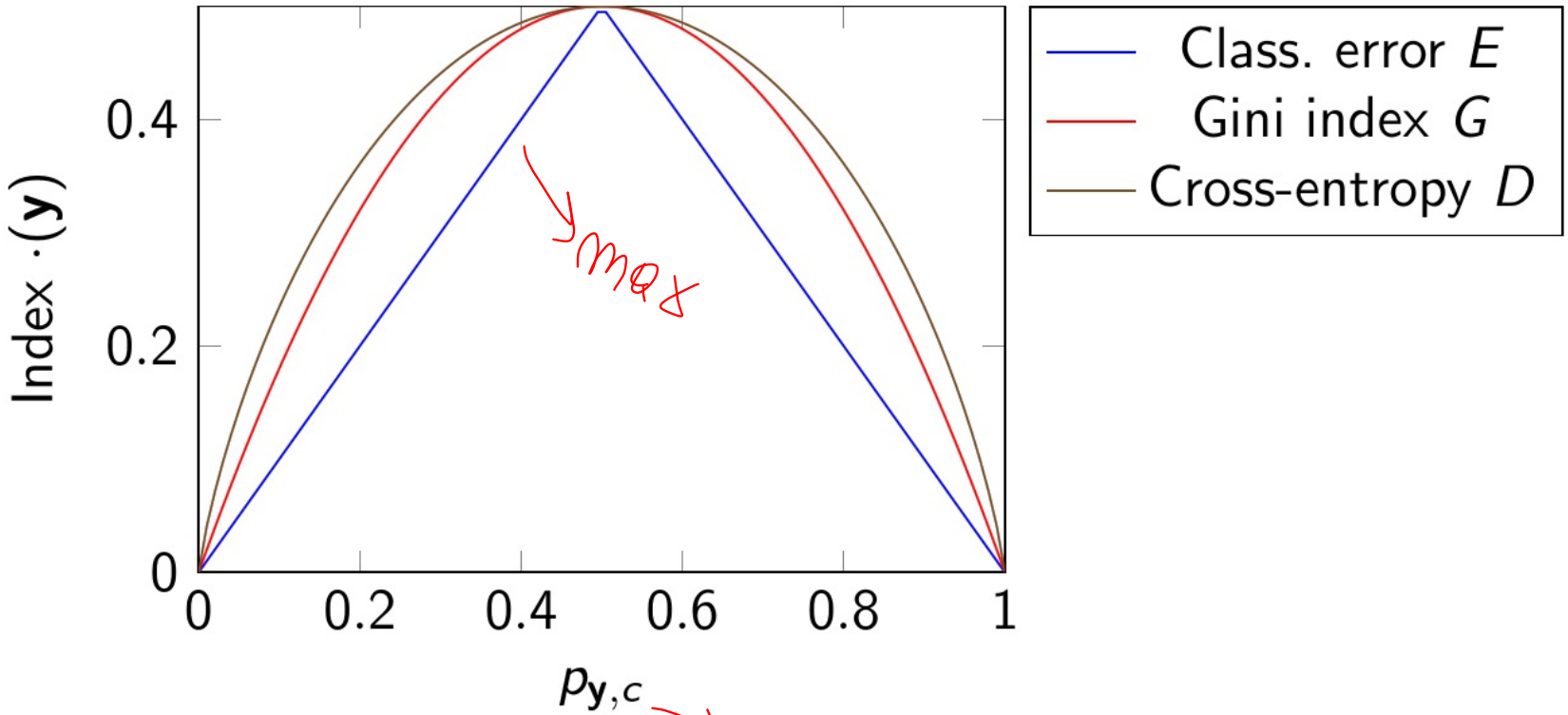
$$G(\mathbf{y}) = \sum_c p_{\mathbf{y},c}(1 - p_{\mathbf{y},c})$$

- Cross-entropy

$$D(\mathbf{y}) = -\sum_c p_{\mathbf{y},c} \log p_{\mathbf{y},c}$$

For all indexes, the lower the better (**node impurity**).

Best branch criteria: binary classification



Cross-entropy is rescaled.

Q: what happens with multiclass problems?

Categorical independent variables

VARS WITH DISCRETE DOMAIN W/O ORDER

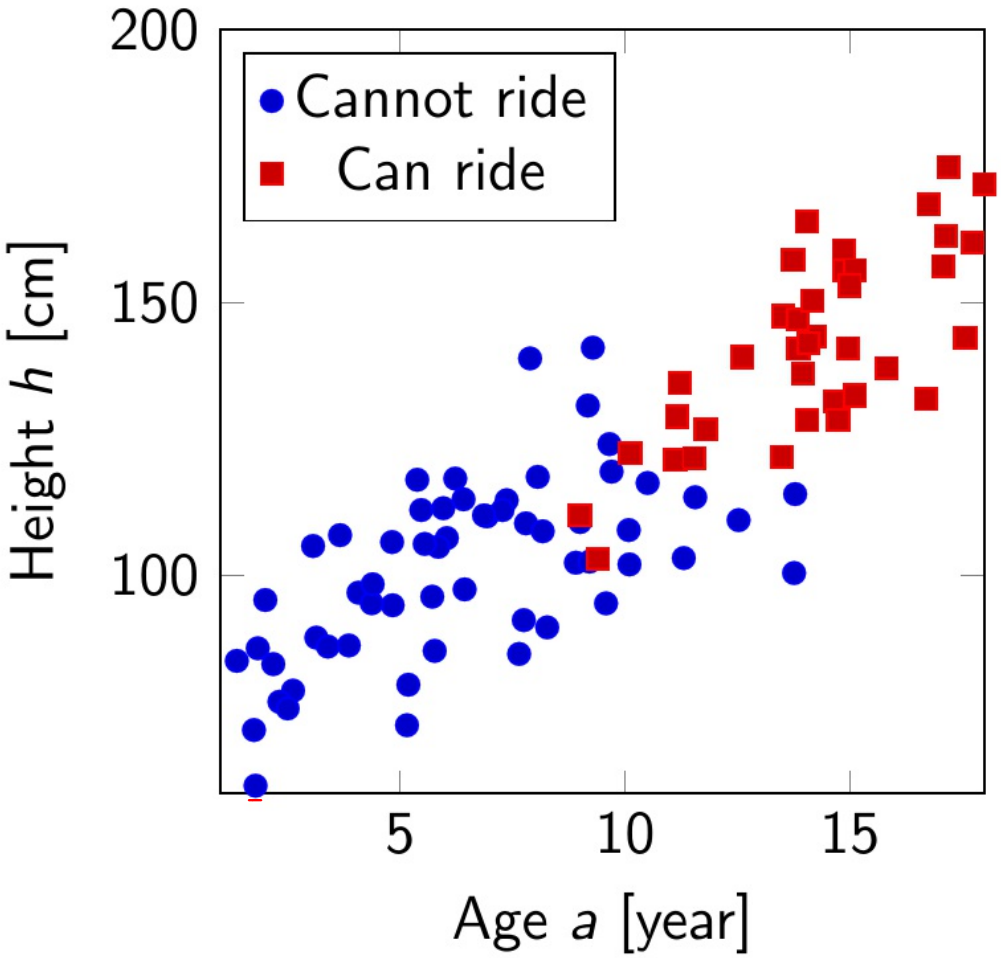
- Trees can work with categorical variables
- Branch node is $x_i = c$ or $x_i \in C' \subset C$ (c is a class)
- Can mix categorical and numeric variables

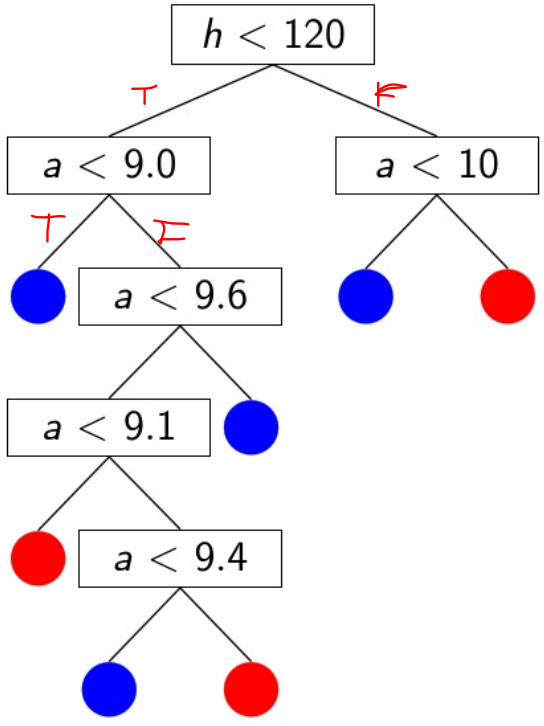
NUM $x_i < t$

CAT

Stopping criterion: role of $k_{\min}$

Suppose $k_{\min} = 1$ (never stop for **y** size)





Q: what's wrong? (recall: "a model of what?") OR ATTENDANT

Tree complexity

When the tree is "too complex"

- less readable/understandable/explicable
- maybe there was noise into the data

ALG IS OK, OUTPUT NOT OK ⟹ INPUT NOT OK

Q: what's noise in carousel data?

NOISE IN DATA COLLECTION OR ATTEND. BEHAVIOR

Tree complexity is not related (only) with $k_{\min}$, but also with data

Tree complexity: other interpretation

- maybe there was noise into the data

The tree *fits* the learning data too much:

- it overfits (**overfitting**) BAD THING
- does not generalize (high **variance**: model varies if learning data varies)

High variance

"model varies if learning data varies": what? why data varies?

- learning data is about the system/phenomenon/nature S
  - a collection of *observations* of S
  - a point of view on S

High variance

"model varies if learning data varies": what? why data varies?

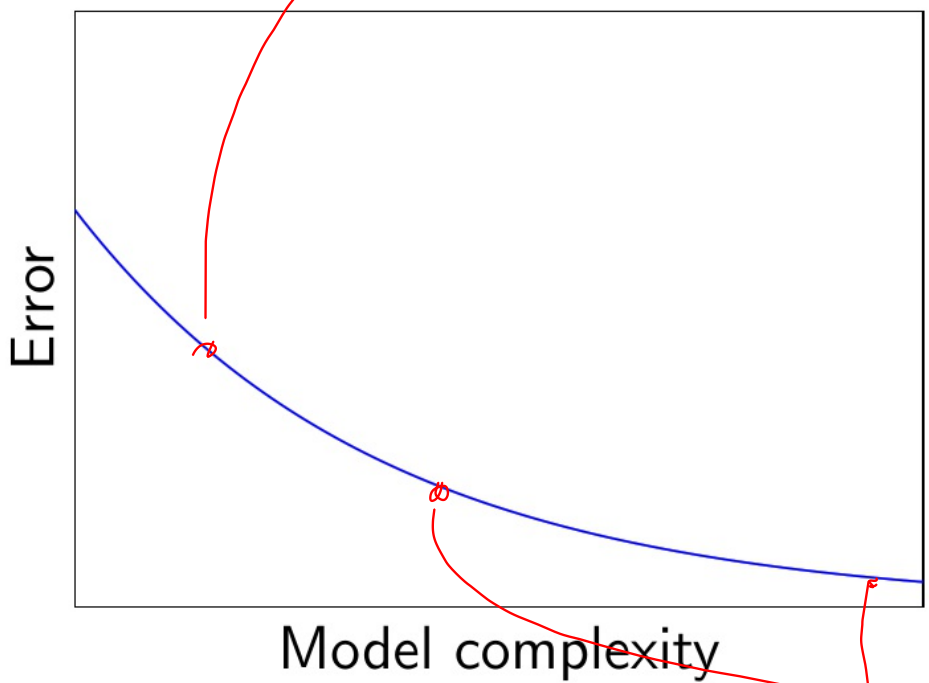
- learning data is about the system/phenomenon/nature S
  - a collection of *observations* of S
  - a point of view on S
- learning is about understanding/knowing/explaining S

High variance

"model varies if learning data varies": what? why data varies?

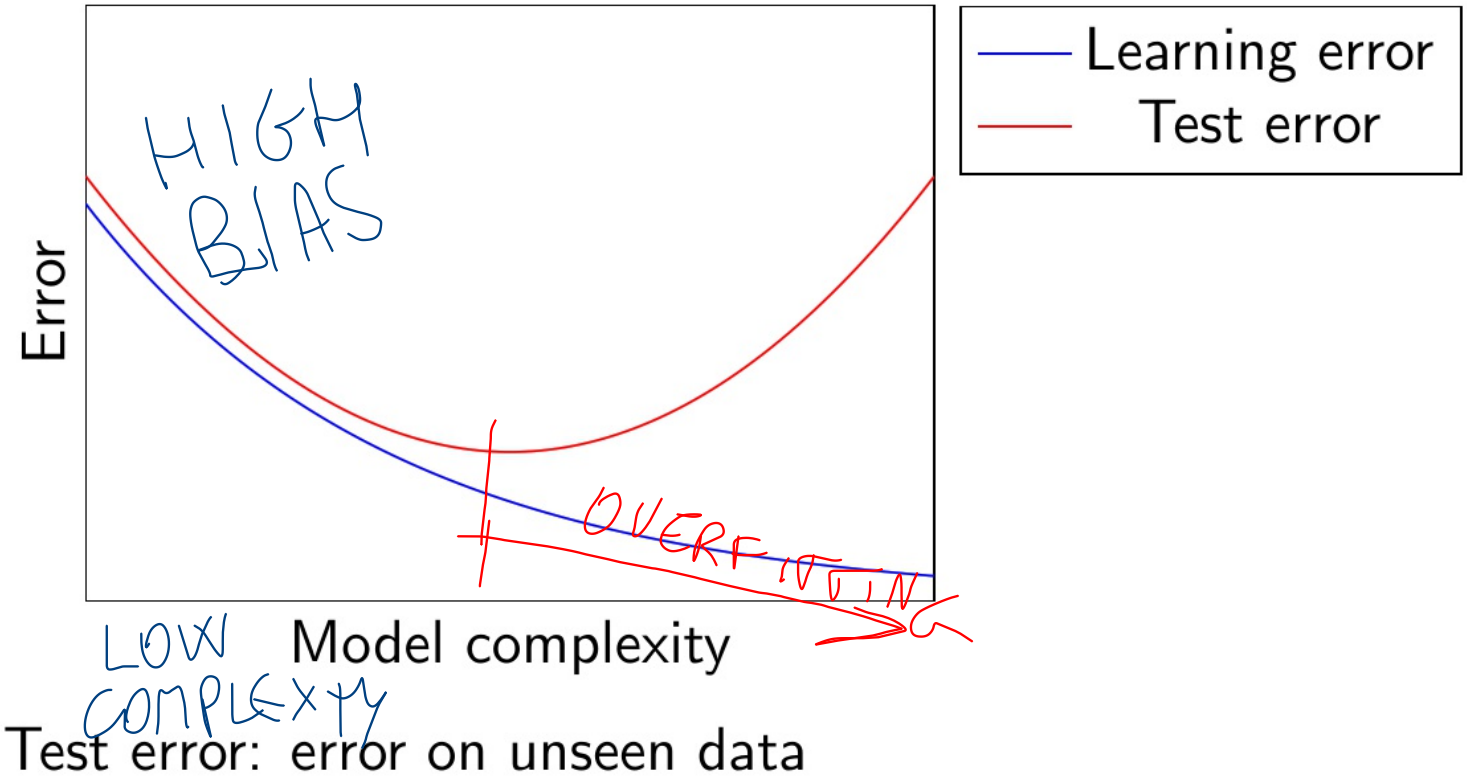
- learning data is about the system/phenomenon/nature S
  - a collection of *observations* of S
  - a point of view on S
- learning is about understanding/knowing/explaining S
  - if I change the point of view on S, my knowledge about S should remain the same!

Spotting overfitting



Test error: error on unseen data

Spotting overfitting



Test error: error on unseen data

k-fold cross-validation

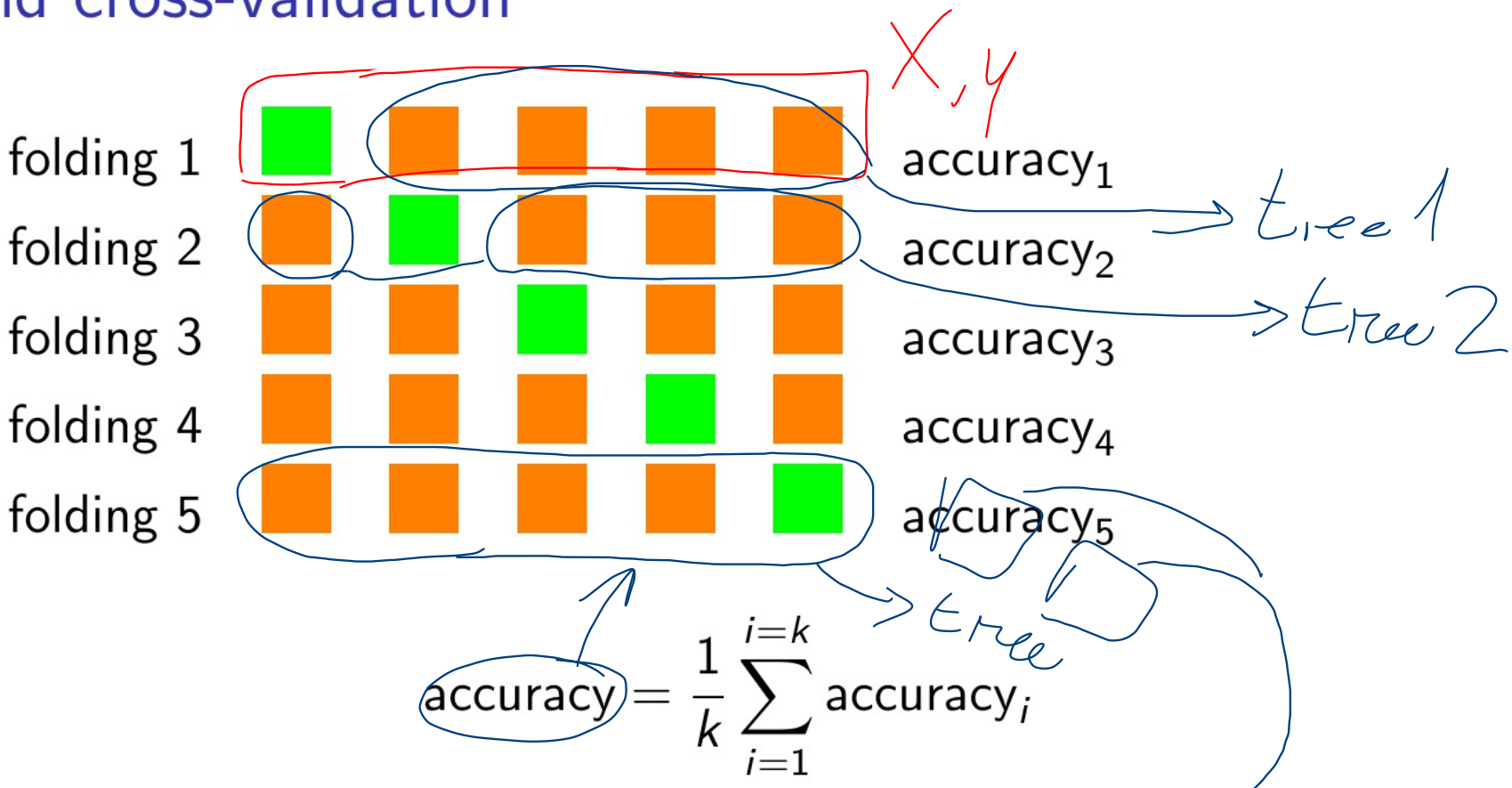
Where can I find "unseen data"? Pretend to have it!

1. split learning data ($\mathbf{X}$ and $\mathbf{y}$) in k equal slices (each of $\frac{n}{k}$ observations/elements)
2. for each split (i.e., each $i \in \{1, \ldots, k\}$)
   - 2.1 learn on all but k-th slice → LEARNED MODEL → E_{train}
   - 2.2 compute classification error on **unseen** k-th slice → E_{test}
3. average the k classification errors

In essence:

- can the learner generalize on available data?
- how the learned artifact will behave on unseen data?

k-fold cross-validation



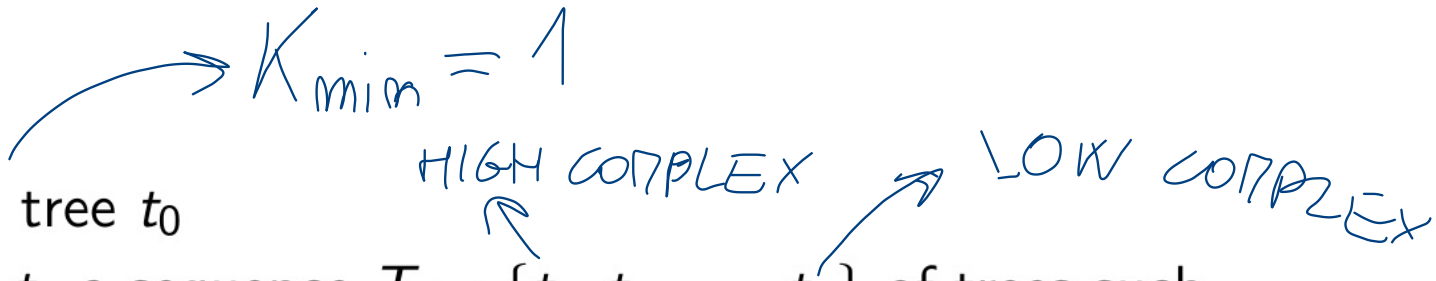
Or with classification error rate or any other meaningful (effectiveness) measure

Q: how should data be split?

Fighting overfitting with trees

- large k_{min} (large w.r.t. what?)
- when building, limit depth
- when building, don't split if low overall impurity decrease
- after building, *prune*

Pruning: high level idea



1. learn a full tree t_0
2. build from t_0 a sequence $T = \{t_0, t_1, \ldots, t_n\}$ of trees such that
   - t_i is a root-subtree of t_{i-1} ($t_i \subset t_{i-1}$)
   - t_i is always less complex than t_{i-1}
3. choose the $t \in T$ with minimum classification error with k-fold cross-validation

k-fold cross-validation: data splitting

Q: how should data be split?

Example: Android Malware detection

- Gerardo Canfora et al. "Effectiveness of opcode ngrams for detection of multi family android malware". In: *Availability, Reliability and Security (ARES), 2015 10th International Conference on*. IEEE. 2015, pp. 333–340
- Gerardo Canfora et al. "Detecting android malware using sequences of system calls". In: *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*. ACM. 2015, pp. 13–20

Android Malware detection