Lecture 10 – Classification and Regression 2

Bulat Ibragimov

bulat@di.ku.dk

Department of Computer Science
University of Copenhagen

Objectives

Decision tree

Splitting rule: information gain

Splitting rule: Gini coefficient

Decision tree generation

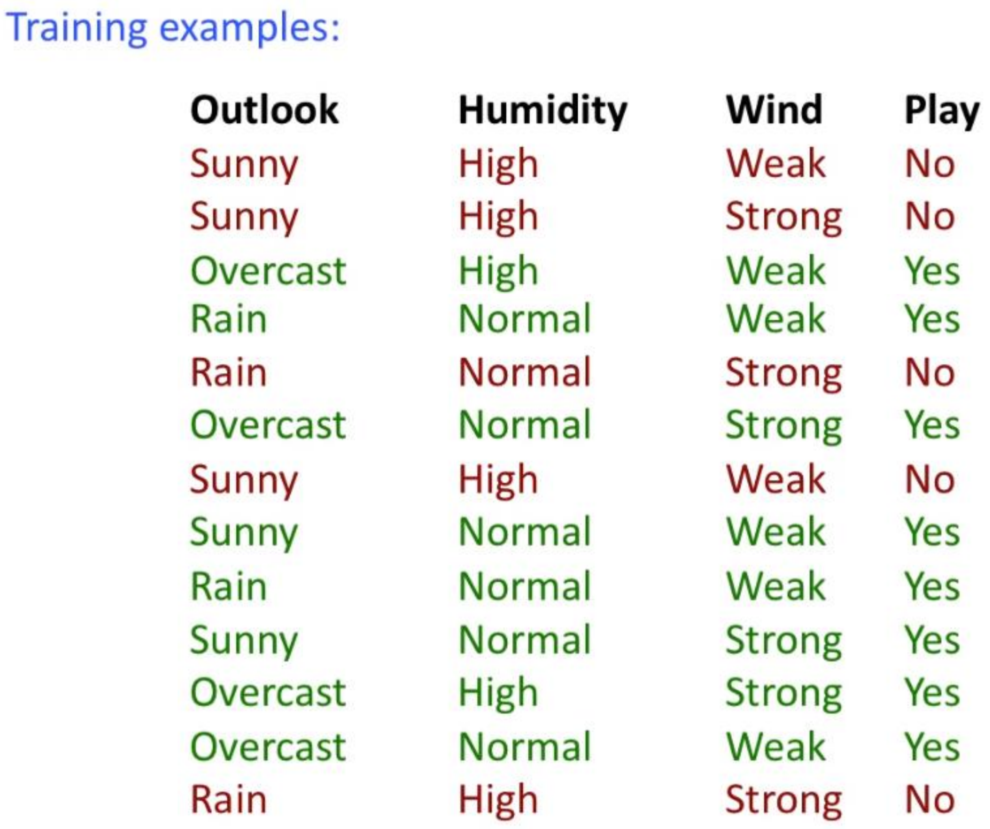
Random forest

Image analysis example

Decision tree

We want to predict if John plays tennis in a given day

- We collected history of 13 days with some factors we believe may be important
- We want make a prediction for new days:
  - D14 – [Rain, High, Weak]
- Divide and conquer:
  - Split training samples according to some attribute
  - If the result is pure – stop, overwise continue splitting



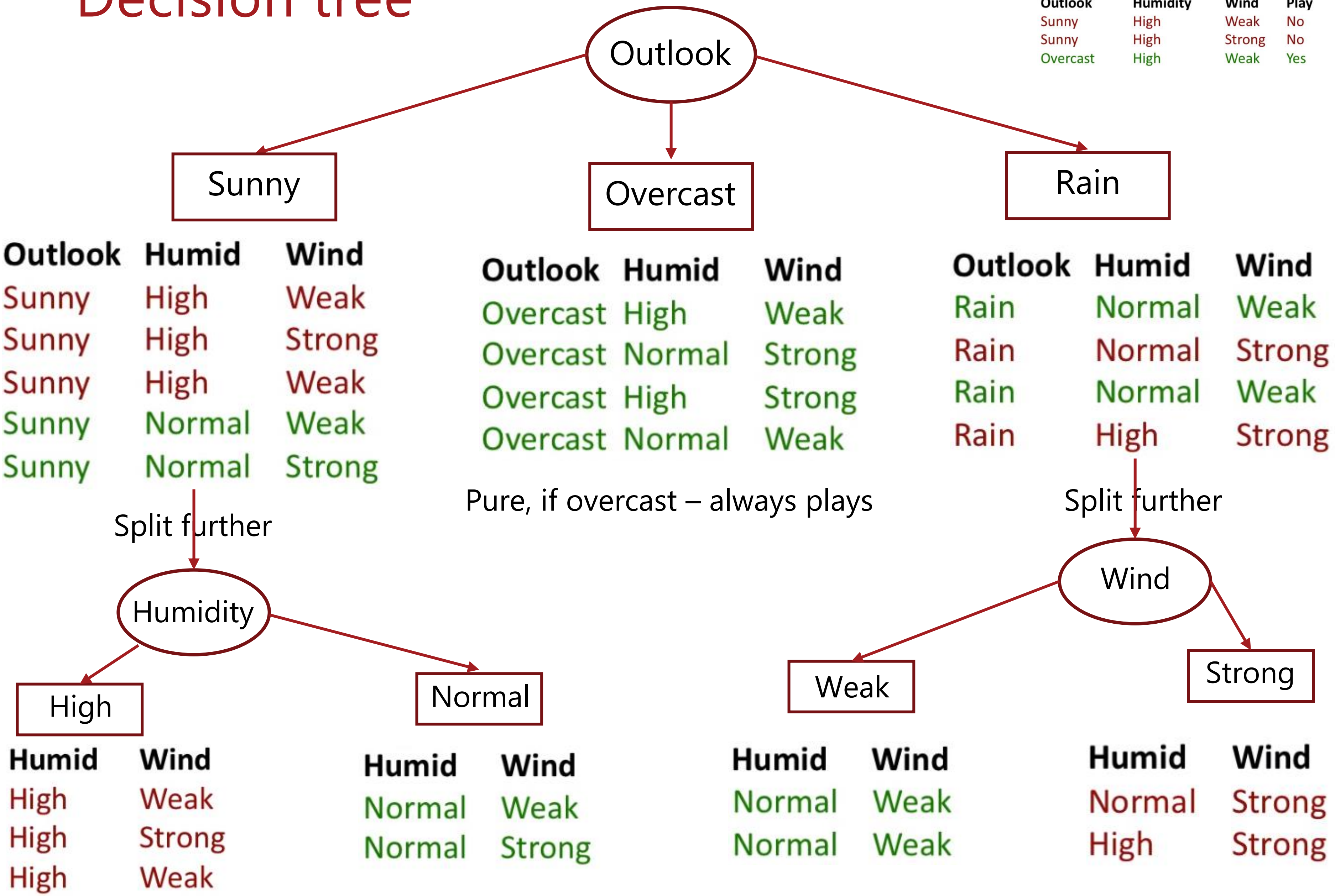
Training examples:

Outlook	Humidity	Wind	Play
Sunny	High	Weak	No
Sunny	High	Strong	No
Overcast	High	Weak	Yes
Rain	Normal	Weak	Yes
Rain	Normal	Strong	No
Overcast	Normal	Strong	Yes
Sunny	High	Weak	No
Sunny	Normal	Weak	Yes
Rain	Normal	Weak	Yes
Sunny	Normal	Strong	Yes
Overcast	High	Strong	Yes
Overcast	Normal	Weak	Yes
Rain	High	Strong	No

Decision tree

Training examples:

Outlook	Humidity	Wind	Play
Sunny	High	Weak	No
Sunny	High	Strong	No
Overcast	High	Weak	Yes



Outlook

Sunny

Outlook	Humid	Wind
Sunny	High	Weak
Sunny	High	Strong
Sunny	High	Weak
Sunny	Normal	Weak
Sunny	Normal	Strong

Split further

Humidity

High

Humid	Wind
High	Weak
High	Strong
High	Weak

Normal

Humid	Wind
Normal	Weak
Normal	Strong

Overcast

Outlook	Humid	Wind
Overcast	High	Weak
Overcast	Normal	Strong
Overcast	High	Strong
Overcast	Normal	Weak

Pure, if overcast – always plays

Rain

Outlook	Humid	Wind
Rain	Normal	Weak
Rain	Normal	Strong
Rain	Normal	Weak
Rain	High	Strong

Split further

Wind

Weak

Humid	Wind
Normal	Weak
Normal	Weak

Strong

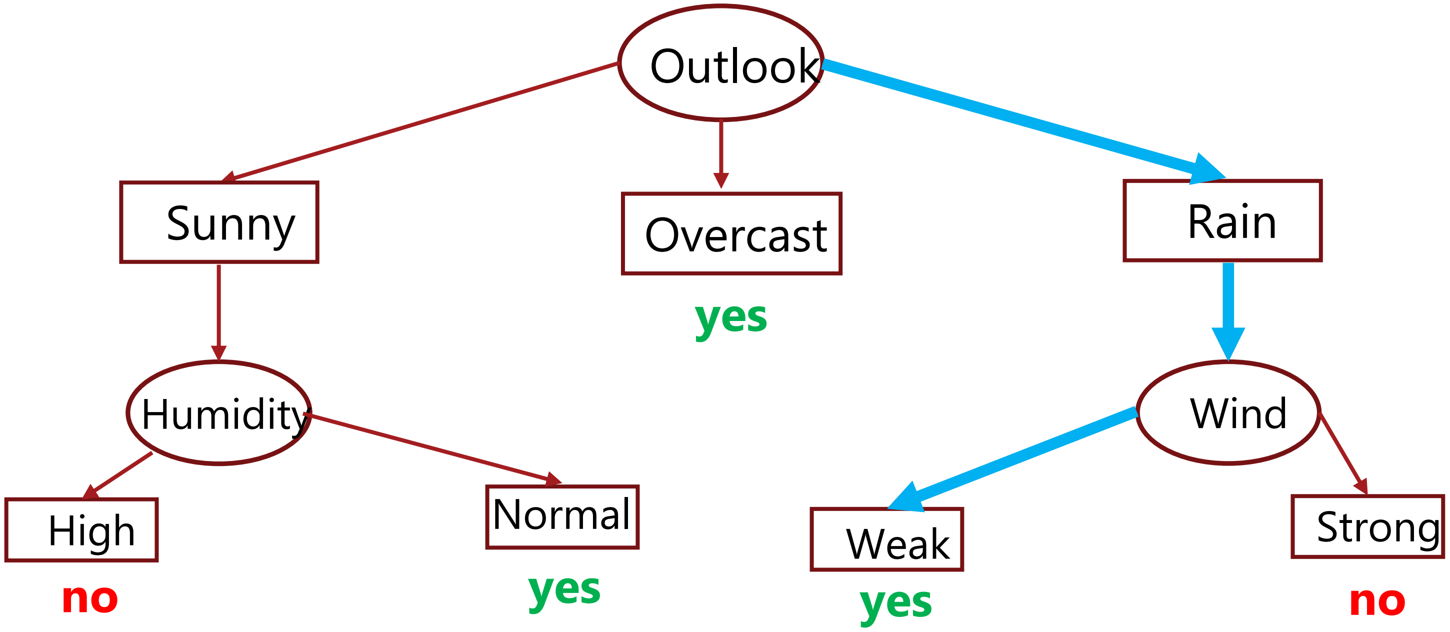
Humid	Wind
Normal	Strong
High	Strong

Decision tree

Training examples:

Outlook	Humidity	Wind	Play
Sunny	High	Weak	No
Sunny	High	Strong	No
Overcast	High	Weak	Yes

We generated a decision tree



For our test example [Rain, High, Weak], we predict John to play

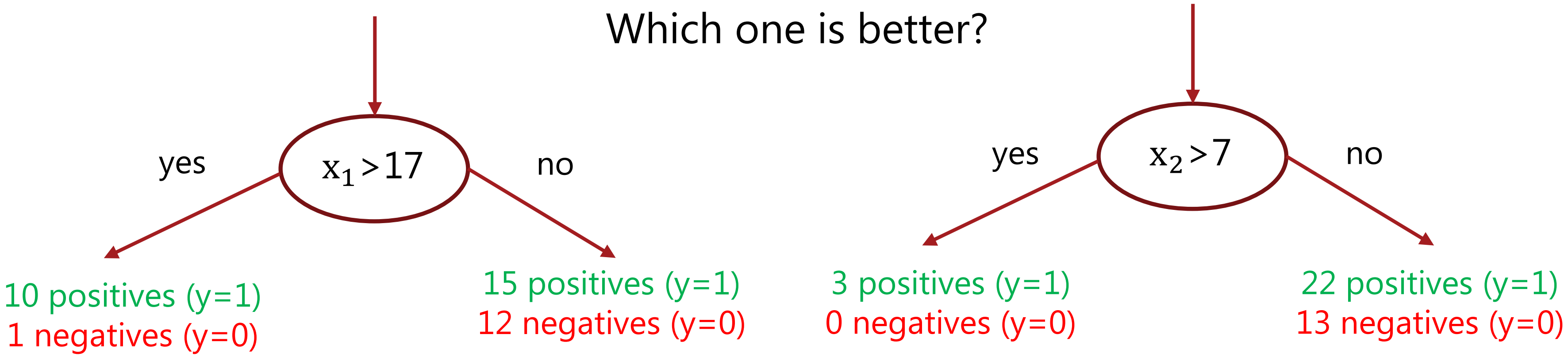
Binary decision tree: algorithm

- Training dataset $\mathbf{T} = \{(\mathbf{x}_1, \mathrm{y}_1), (\mathbf{x}_2, \mathrm{y}_2), \dots, (\mathbf{x}_n, \mathrm{y}_n)\}$; $y \in \{0,1\}$:

- Select feature j using the current subset of examples T.
- Find the optimal split of T using feature j:
  - The new subsets T_0 and T_1.
- Continue for each T_0 and T_1:
  - If stopping criteria is not reached for T_k, go to step 1

- How to select j? How to split T?

Splitting rule

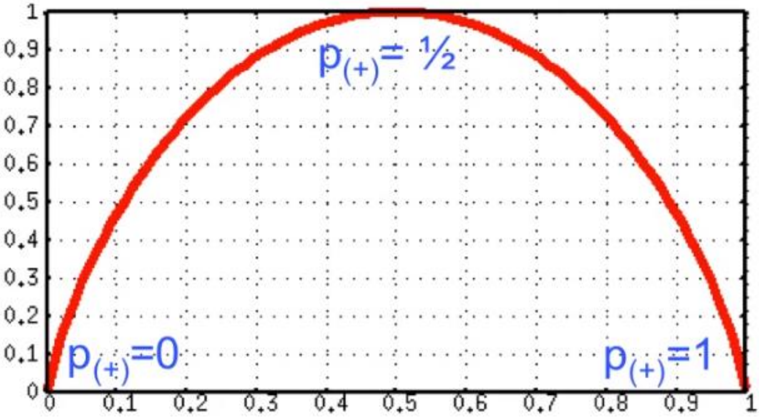


Which one is better?

- Intuitively:
  - Purity (more certain separation) :
    - 9 pos / 1 neg => very certain separation (90%)
    - 5 pos / 4 neg => not certain (56%)
  - Power of separation:
    - Let's say we have 30 samples
    - Separation with 10 pos / 2 neg is better than
    - Separation with 5 pos / 1 neg
  - Symmetric:
    - 10 pos / 2 neg is as good as 2 pos / 10 neg

Splitting rule: entropy

- Entropy $H(T) = -p_+ \log_2 p_+ - p_- \log_2 p_-$:
  - H(9 pos, 1 neg):
    - $H(T) = -0.9 \log_2 0.9 - 0.1 \log_2 0.1 = 0.137 + 0.332 = \mathbf{0.469}$
  - H(5 pos, 4 neg):
    - $H(T) = -0.56 \log_2 0.56 - 0.44 \log_2 0.44 = 0.468 + 0.521 = \mathbf{0.989}$

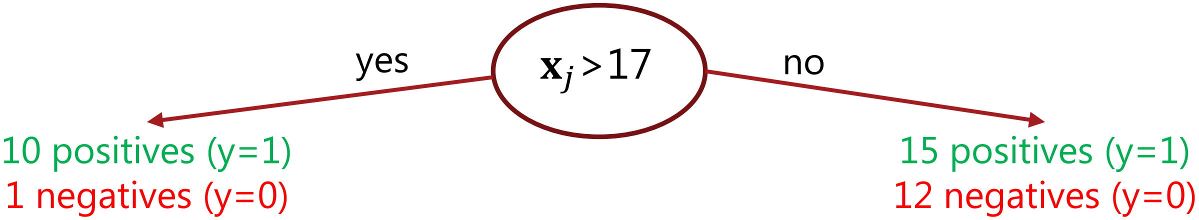


- H(8 class1, 7 class2, 5 class3):
  - $H(T) = -0.4 \log_2 0.4 - 0.35 \log_2 0.35 - 0.25 \log_2 0.25 = 0.53 + 0.53 + 0.5 = \mathbf{1.56}$
- H(11 class1, 9 class2, 0 class3):
  - $H(T) = -0.55 \log_2 0.55 - 0.45 \log_2 0.45 - 0\ \log_2 0 = 0.47 + 0.52 = \mathbf{0.99}$

Splitting rule: information gain

- Sets with good separation and large power:

$$Gain(T,j) = H(T) - \sum_{i=0,1} \frac{|T_i|}{|T|} H(T_i)$$

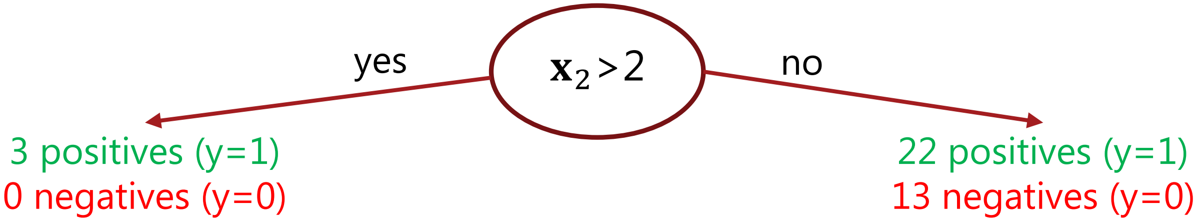


$$\begin{aligned} &Gain(T,j) \\ &\quad = H(25\text{pos}\,,13\text{pos}) - \frac{11}{38}H(10\text{pos},1\text{pos}) - \frac{27}{38}H(15\text{pos},12\text{pos}) \\ &\quad = 0.927 - \frac{11}{38}0.44 - \frac{27}{38}0.99 = 0.096 \end{aligned}$$

Splitting rule: information gain

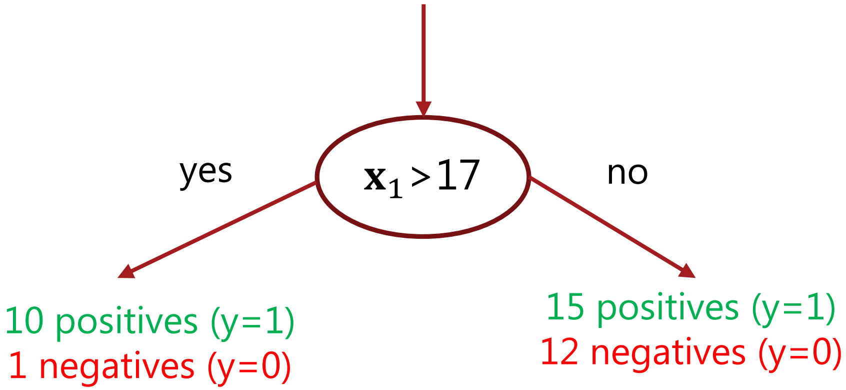
- Sets with good separation and large power:

$$Gain(T,j) = H(T) - \sum_{i=0,1} \frac{|T_i|}{|T|} H(T_i)$$



$$Gain(T,j)$$
$$= H(\text{25pos}\,,\text{13pos}) - \frac{3}{38} H(\text{3pos}, \text{0pos}) - \frac{35}{38} H(\text{22pos}, \text{13pos})$$
$$= 0.927 - \frac{3}{38} 0 - \frac{35}{38} 0.95 = 0.05$$

Splitting rule: information gain



$\mathbf{x}_1 > 17$

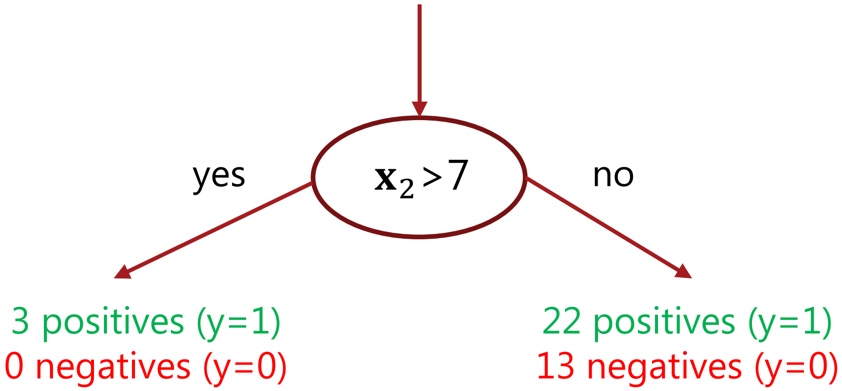
yes

10 positives (y=1)
1 negatives (y=0)

no

15 positives (y=1)
12 negatives (y=0)

Information gain is 0.096



$\mathbf{x}_2 > 7$

yes

3 positives (y=1)
0 negatives (y=0)

no

22 positives (y=1)
13 negatives (y=0)

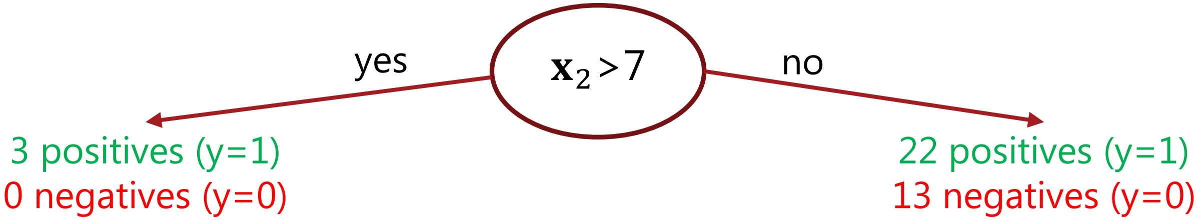
Information gain is 0.05

- We want to maximize the gain:
  - It is better to split using $\mathbf{x}_1 > 17$
  - $Gain(T, j) = H(T) - \sum_{i=0,1} \frac{|T_i|}{|T|} H(T_i)$

Splitting rule: Gini index

- Alternative splitting rule:

$$gini(T) = 1 - \sum_i \left(\frac{|T_i|}{|T|}\right)^2:$$



yes

$\mathbf{x}_2 > 7$

no

3 positives (y=1)
0 negatives (y=0)

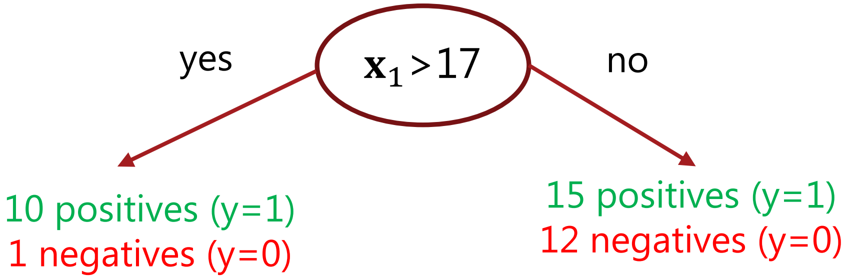
22 positives (y=1)
13 negatives (y=0)

$$Gini(T,j) = gini(T) - (\frac{3}{38} gini(T_0) + \frac{35}{38} gini(T_1))$$

$$= (1 - 0.433 - 0.117) - \left(\frac{3}{38}(1 - 1 - 0) + \frac{35}{38}(1 - 0.395 - 0.138)\right)$$

$$= 0.45 - (0 + 0.43) = 0.02$$

Splitting rule: Information gain based on Gini



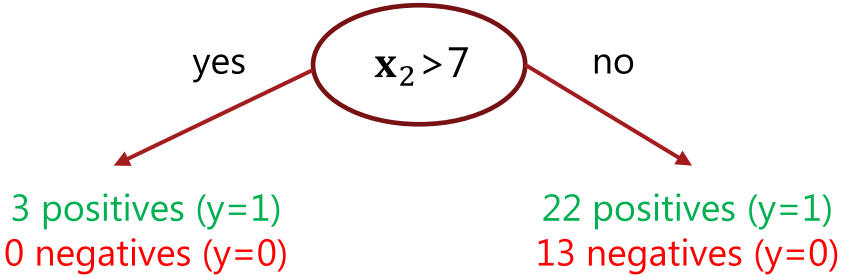
$\mathbf{x}_1 > 17$

yes

no

10 positives (y=1)
1 negatives (y=0)

15 positives (y=1)
12 negatives (y=0)



$\mathbf{x}_2 > 7$

yes

no

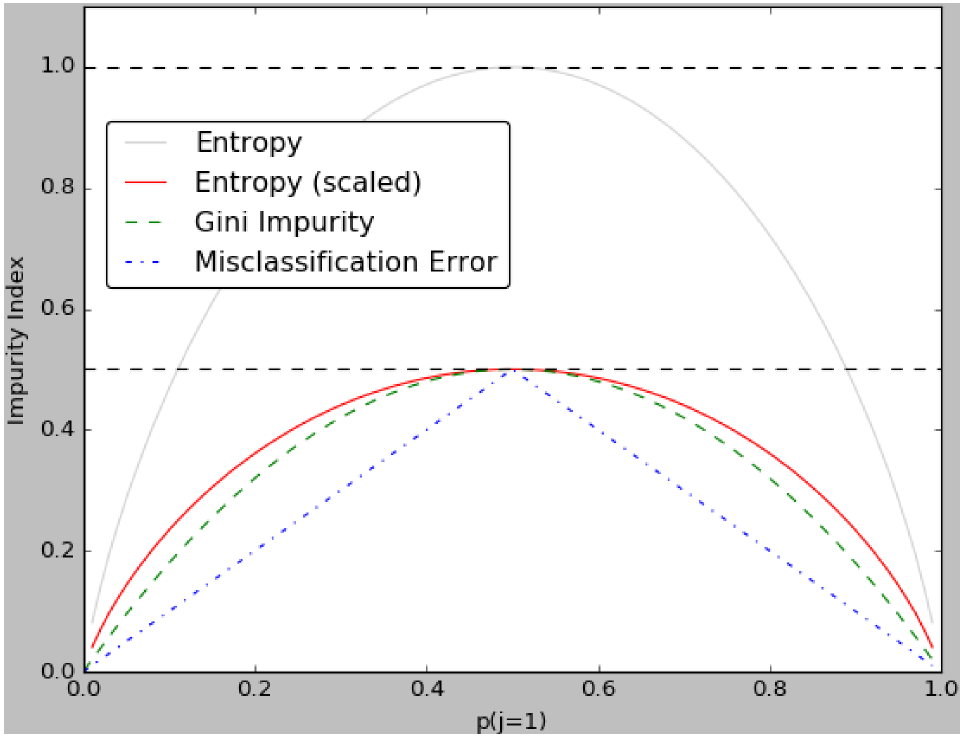
3 positives (y=1)
0 negatives (y=0)

22 positives (y=1)
13 negatives (y=0)

Information gain is 0.051

Information gain is 0.02

Case $\mathbf{x}_1 > 17$ is again better

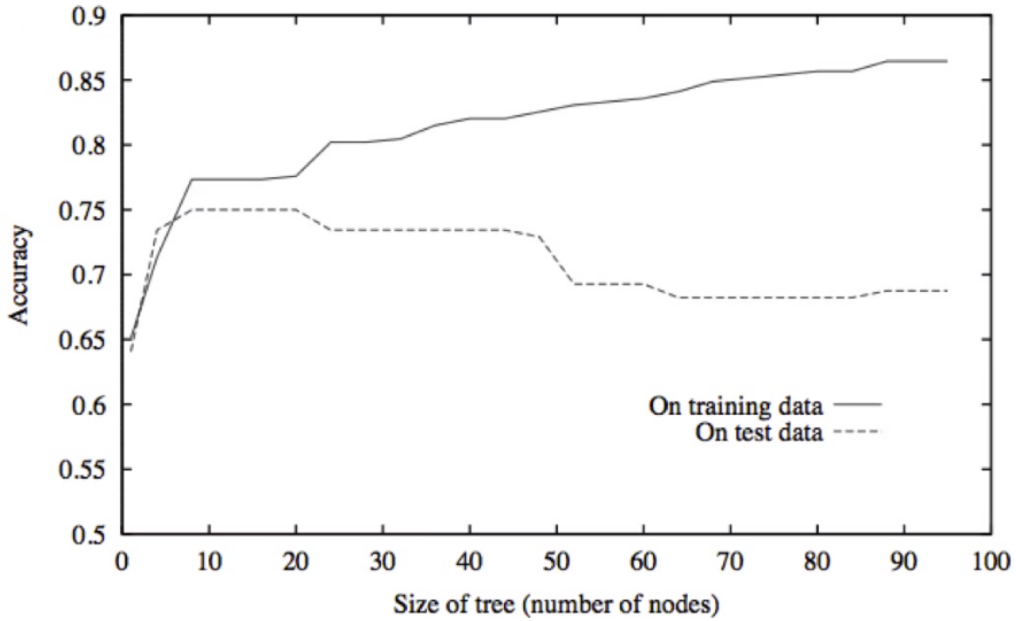


Stopping condition

Overfitting problem:

- If we allow the tree to grow until all terminal leaves are pure, the training classification will be perfect
- Leaves can contain only one element
- Testing accuracy may be unreliable

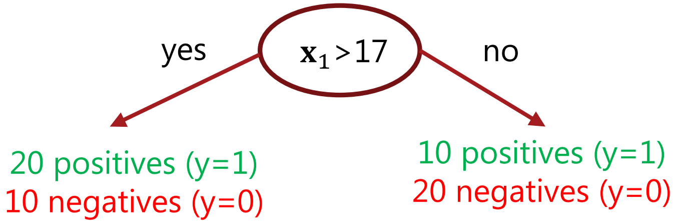
- How can we prevent it?
  - Validation data
  - Restrict tree depth
  - Restrict minimal leaf power
  - Pruning

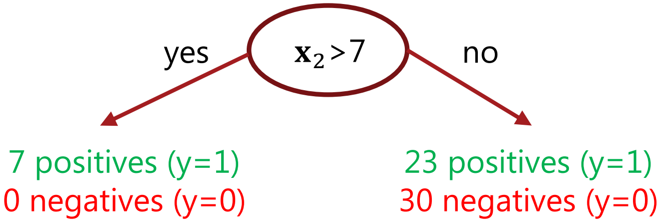


Stopping condition

- Restrict the tree complexity:
  - No more than N splits are allowed before a leaf
  - Split cannot have less that x% of the original data
  - Allow only statistically significant restrictions

- What can be the problem with such restrictions?
  - Imbalanced splits

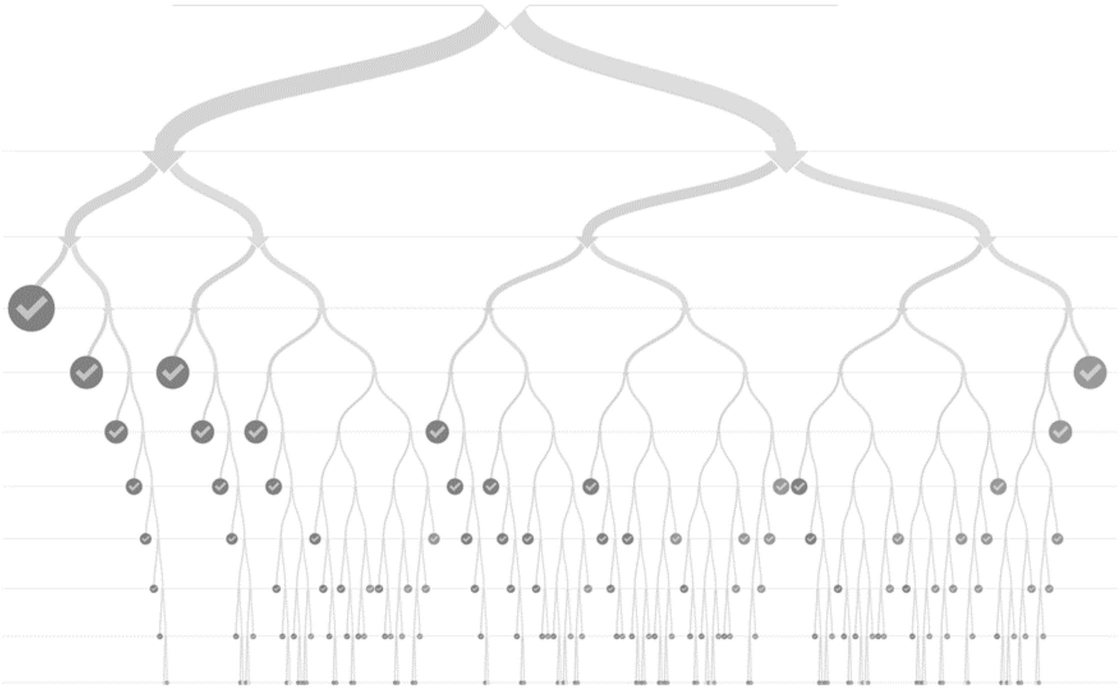




Information gain ($\mathbf{x}_1 > 17$) < Information gain ($\mathbf{x}_2 > 7$)

Pruning

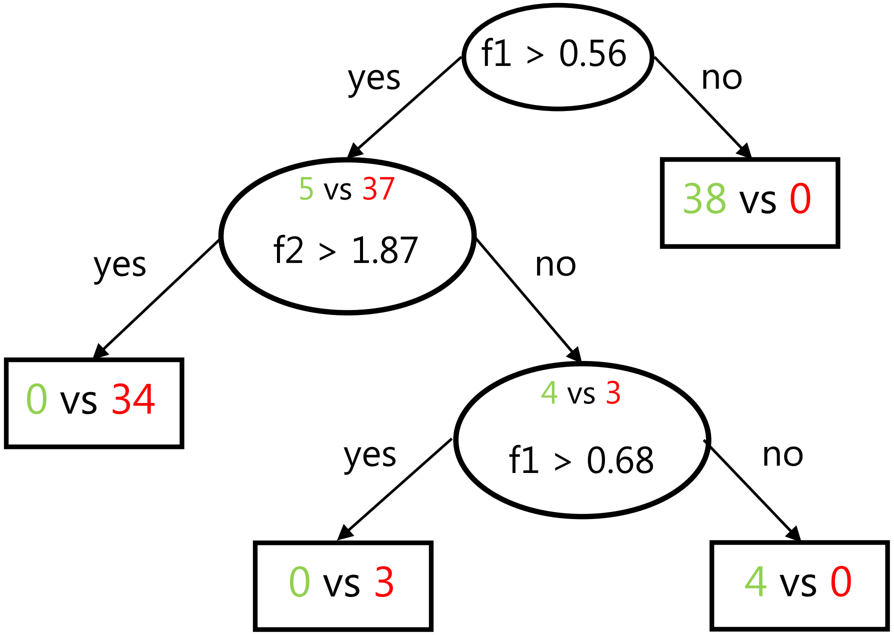
- Database is split into training and validation parts
- Go through every last level split:
  - Compare validation accuracy with the split and without (converted into a leaf)
  - If accuracy increases, replace the split with the leaf
- Go to next levels



Decision boundary

- Two classes in 2D space need to be separated

- Can decision tree generate such a visually-optimal border?



f1 > 0.56

yes

no

5 vs 37

f2 > 1.87

38 vs 0

yes

no

0 vs 34

4 vs 3

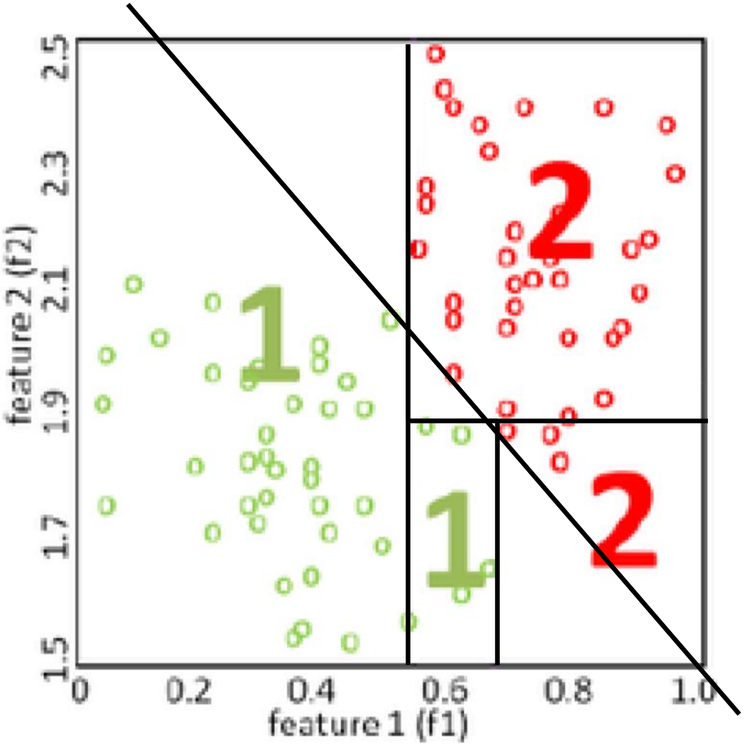
f1 > 0.68

yes

no

0 vs 3

4 vs 0

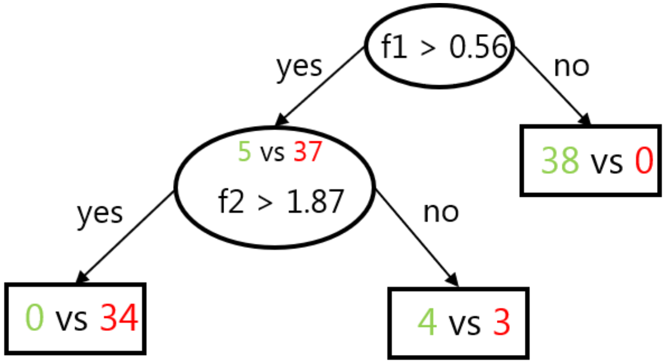


Decision boundary

- How the pruned tree will perform on the following test examples:

predictions

- [0.25, 1.7] → 38 vs 0 => [1, 0]
- [0.65, 1.7] → 4 vs 3 => [0.57, 0.43]
- [0.8, 1.7] → 4 vs 3 => [0.57, 0.43]
- [0.4, 2.1] → 38 vs 0 => [1, 0]
- [0.8, 2.2] → 0 vs 34 => [0, 1]



f1 > 0.56

yes

no

5 vs 37

f2 > 1.87

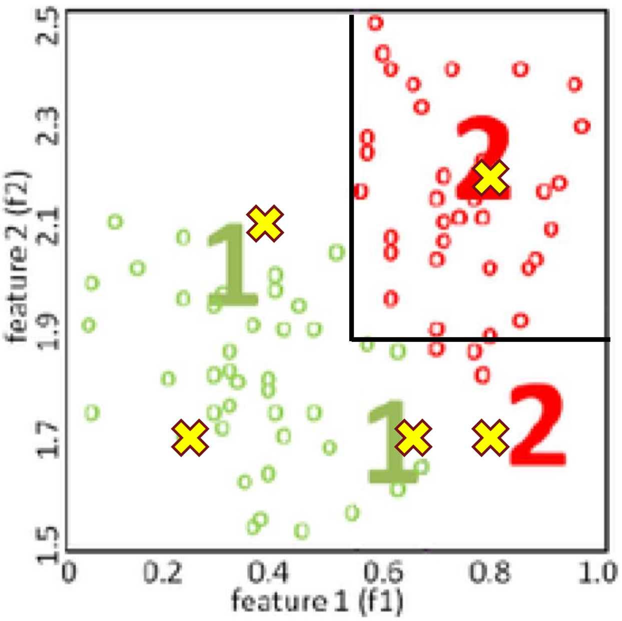
38 vs 0

yes

no

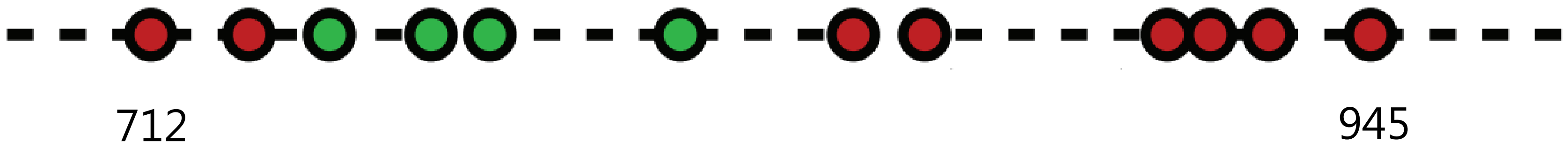
0 vs 34

4 vs 3



Selection of the threshold value

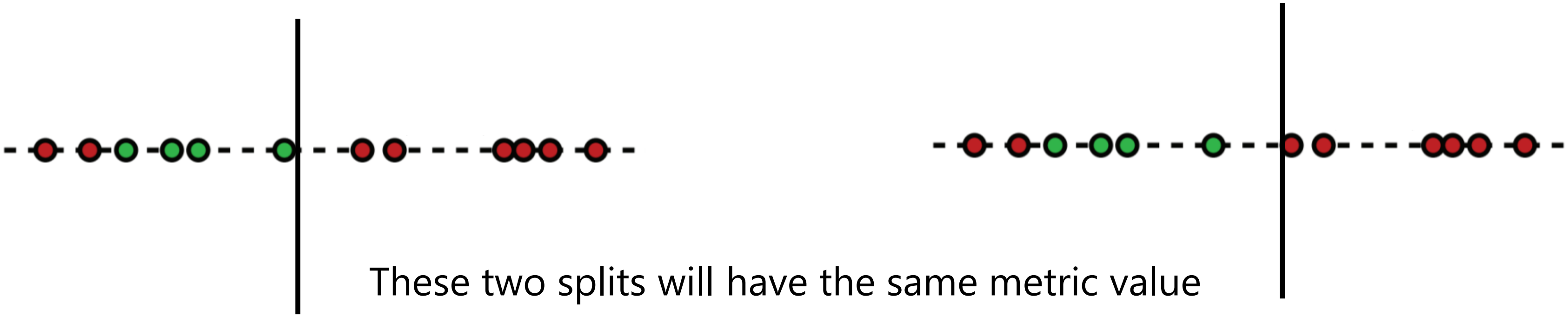
- What kind of intuitive rules we can establish during selection of the optimal threshold?



- Do not look outside interval of feature values [712, 945]

- The split quality metrics assign binary values for each sample.

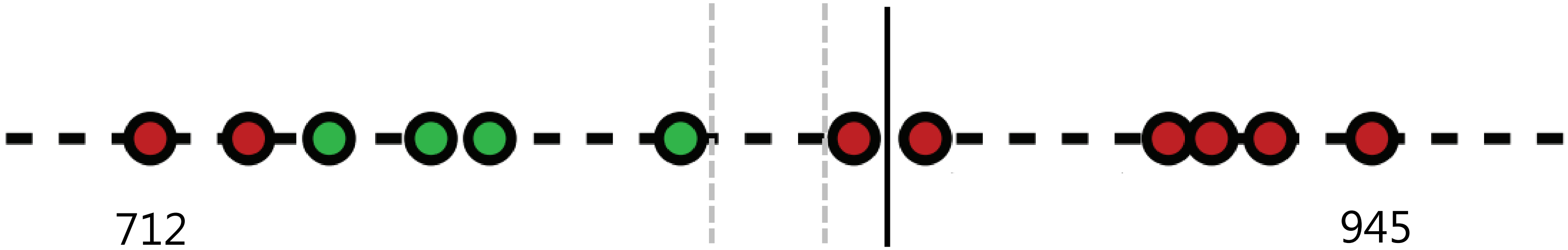
$$\text{Entropy} = -p_+ \log_2 p_+ - p_- \log_2 p_- \qquad \text{Gini} = 1 - \sum_{i=0,1} \left(\frac{|T_i|}{|T|}\right)^2$$



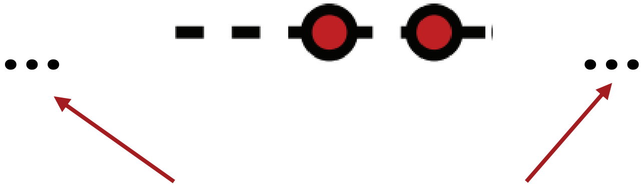
These two splits will have the same metric value

Selection of the threshold value

- Only average points between neighboring samples need to be considered



- Only check different-class neighbors



There is at least one green point.
Moving the threshold ever right or left will definitely improve separation

Selection of the threshold value

Algorithm:

- Sort input samples according to a feature of interest $\mathbf{X}$:

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

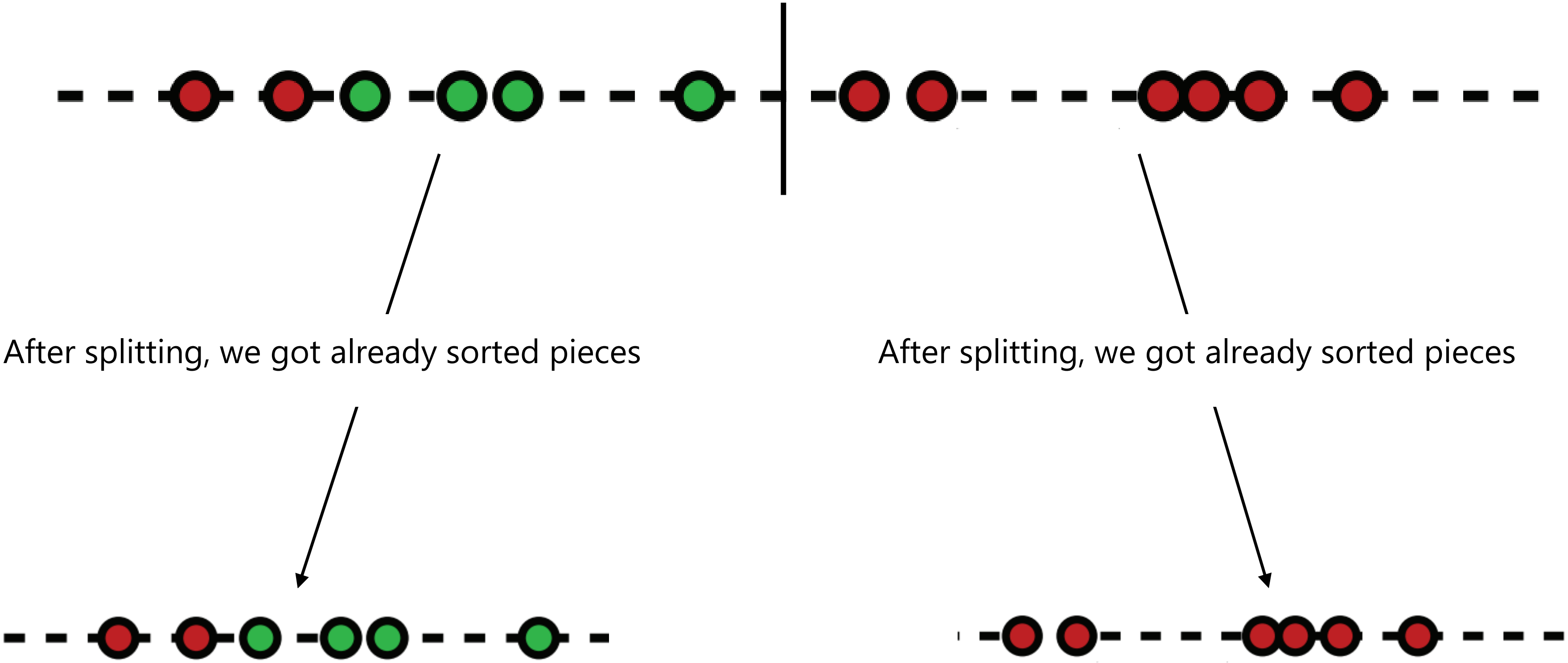
- Consider the following split points:

$$x_i + \frac{(x_{i+1} - x_i)}{2}: \quad y_{i+1} \neq x_i$$

- Find the optimal split point by computing splitting metrics

Selection of the threshold value

- Sorting is very expensive step, but it needs to be done only once at the beginning for every feature:



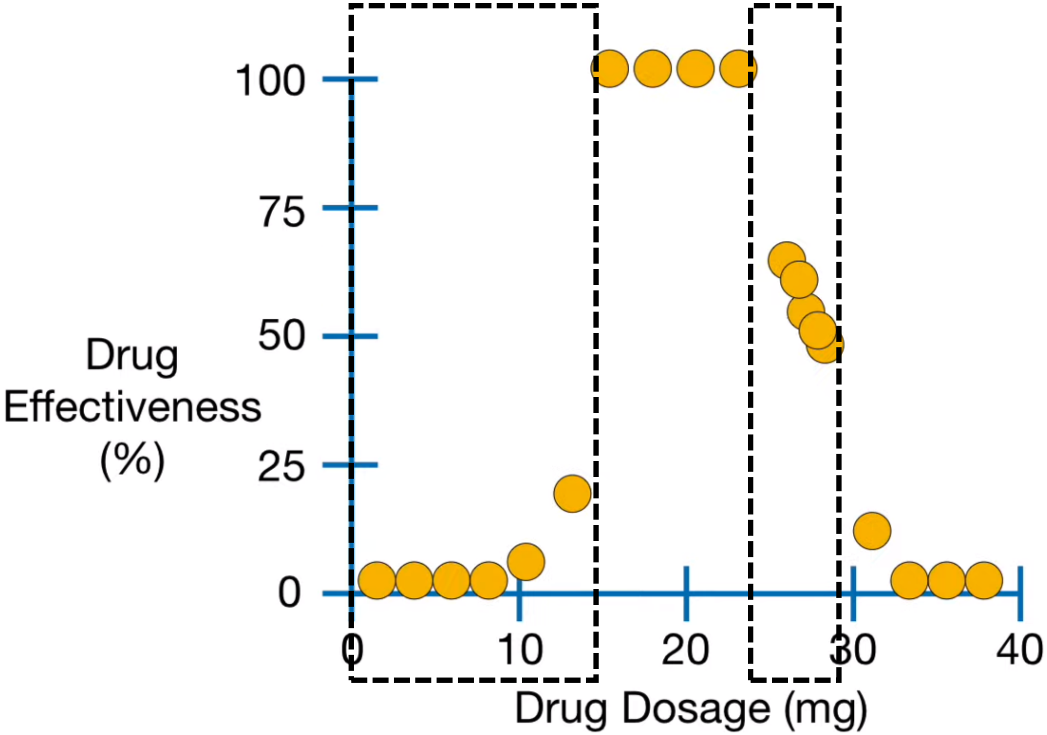
After splitting, we got already sorted pieces

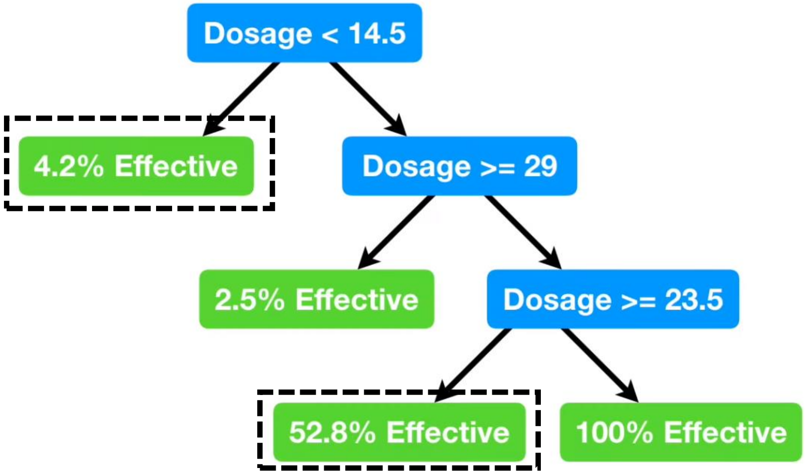
After splitting, we got already sorted pieces

Regression tree

Instead of classification, decision trees can do regression. Two challenges:

- Computing prediction:
  - Average values of the leaf
  - Some simple model of the values on the final leaf





Picture from Josh Starmer

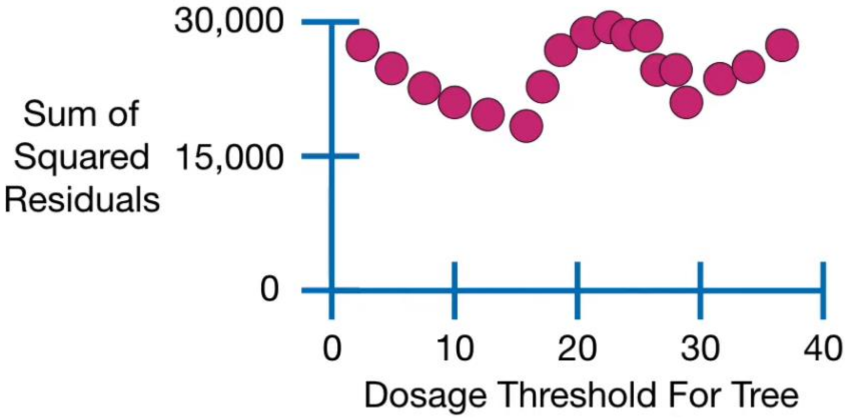
Splitting rule: reduction of variance

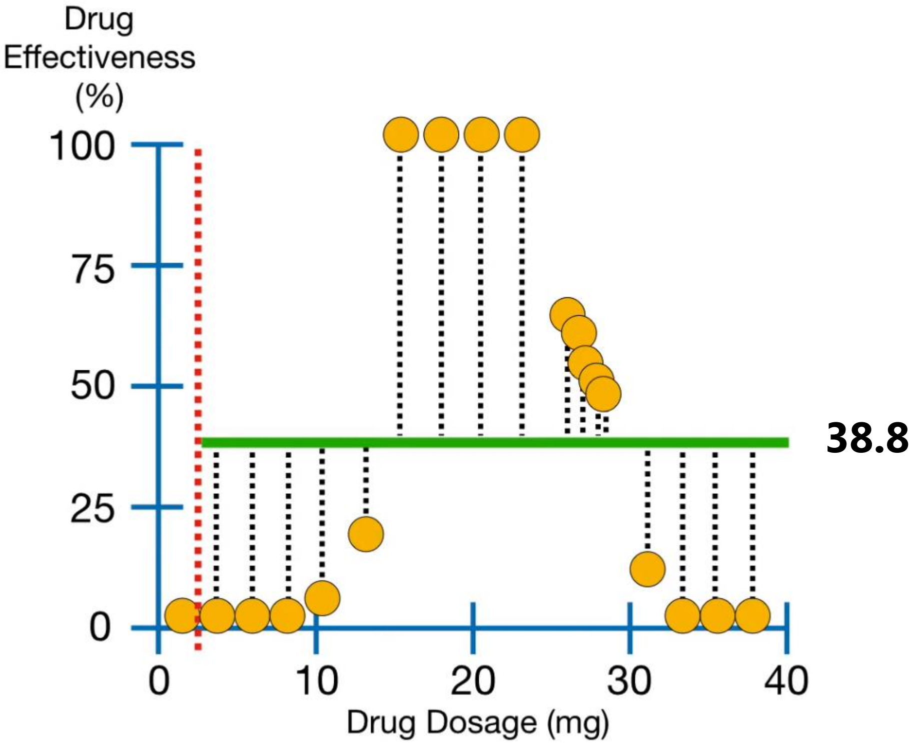
- Splitting rule:

$$F(T) = \sum_{x \in T} (x - \bar{x})^2$$

$$(0 - 0)^2 + (0 - 38.8)^2 + (0 - 38.8)^2 + (0 - 38.8)^2$$
$$+ (5 - 38.8)^2 + (20 - 38.8)^2 + (100 - 38.8)^2$$
$$+ (100 - 38.8)^2 + \dots + (0 - 38.8)^2$$

=27465.5





How would you quantify the quality of this split?

Splitting rule: categorical variables

If a feature is not numerical but categorical:

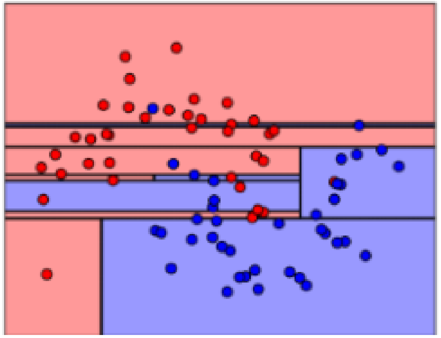
- Binary – {male/female}, {had history of disease/ no history}
- Non-binary – utilized {drug A/drug B/drug C}

For non-binary categorical features:

- Check binary splits in the form drug A vs. drug not A

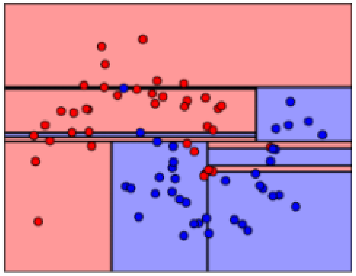
Random Forests

- Single tree cannot generalize well on complex data
- Idea is to generate many weaker trees instead of one full tree

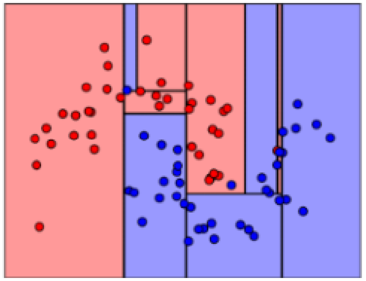


- Introduce randomness during training of each tree:
  - Train each tree on a random subset of training samples S_r
  - During split generation, only consider a random subsample of features $d \ll D$

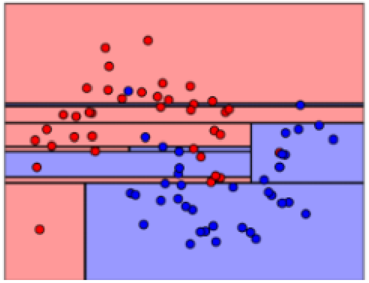
tree 1



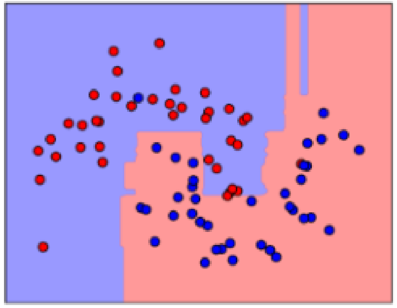
tree 2



tree 3

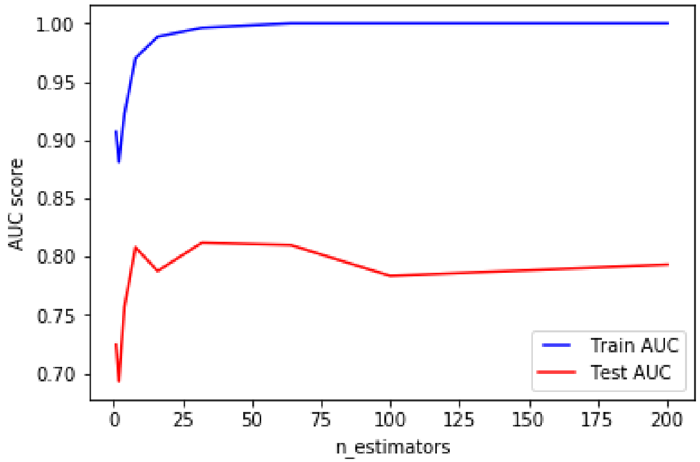


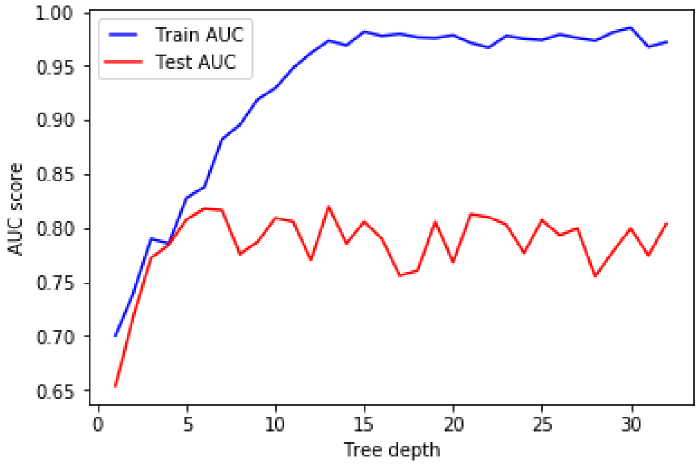
random forest



Random forests

- Tree depth vs number of trees





- How important are ratios $|S_r|/|S|$ and $|d|/|D|$?
  - $|d| = \sqrt{|D|}$
  - $|d| = |D|$
  - $|d| = 1$

https://csiu.github.io/blog/update/2017/05/30/day95.html

Random forests: kernels

- We previously observed axis-aligned kernels:

  $h(\mathbf{x}, i, \tau) = [\tau_1 \geq x_i \geq \tau_2]$ e.g. $[\infty \geq x_i \geq 15.5]$

- Linear kernel:

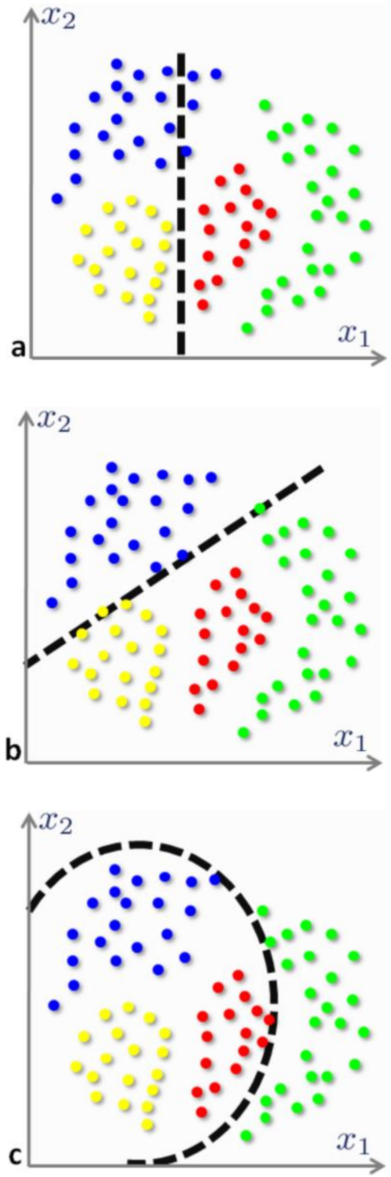
  $$h(\mathbf{x}, \phi, \boldsymbol{\psi}, \tau) = [\tau_1 \geq \phi(\mathbf{x})^T \cdot \boldsymbol{\psi} \geq \tau_2]$$

  $$\text{e.g. } \phi(\mathbf{x}) = [x_\mathrm{i}, x_\mathrm{j}, 1]; \qquad \boldsymbol{\psi} = [\psi_1, \psi_2, \psi_3]$$

- Non-linear kernel:

  $$h(\mathbf{x}, \phi, \boldsymbol{\psi}, \tau) = [\tau_1 \geq \phi(\mathbf{x})^T \cdot \boldsymbol{\psi} \cdot \phi(\mathbf{x}) \geq \tau_2]$$

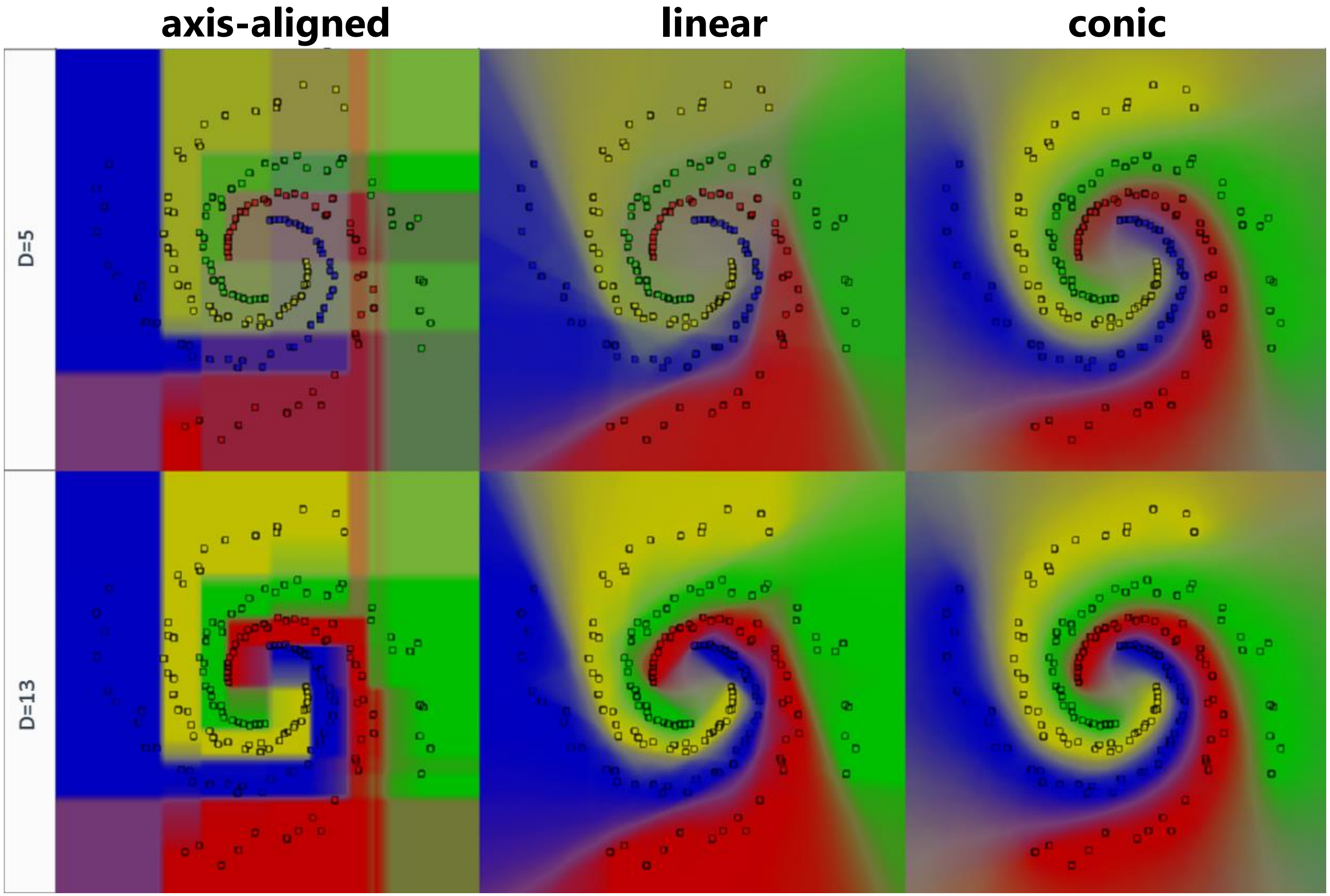
  $$\text{e.g. } \phi(\mathbf{x}) = [x_\mathrm{i}, x_\mathrm{j}, 1]; \qquad \boldsymbol{\psi} = \mathbb{R}^{3\mathrm{x}3}$$



https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/decisionForests_MSR_TR_2011_114.pdf

Random forests: kernels

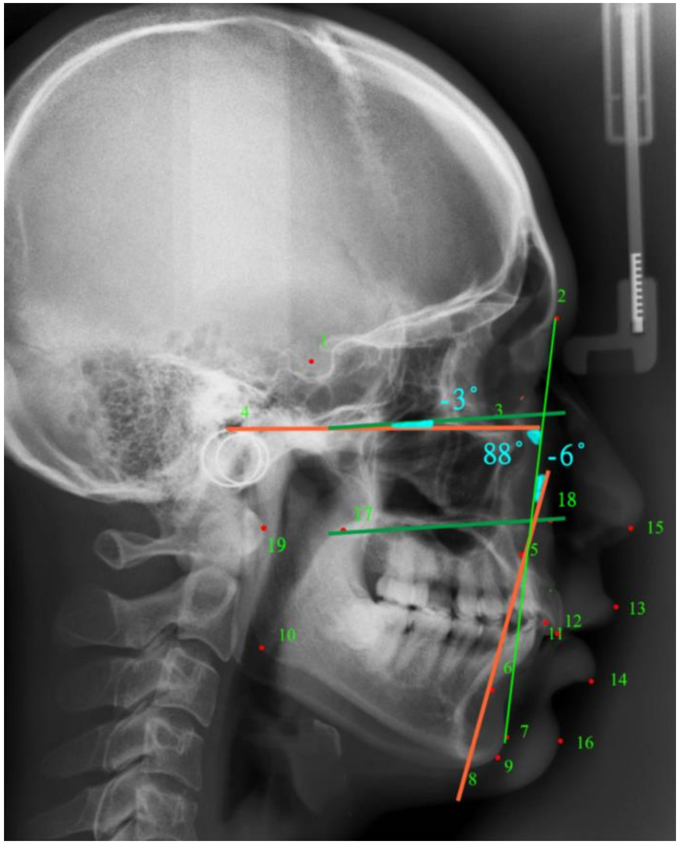
- Kernels are computationally expansive, the number of different combinations of $i, j, \ldots$ in $\phi(\mathbf{x}) = [x_\mathrm{i}, x_\mathrm{j}, 1]$ grows exponentially

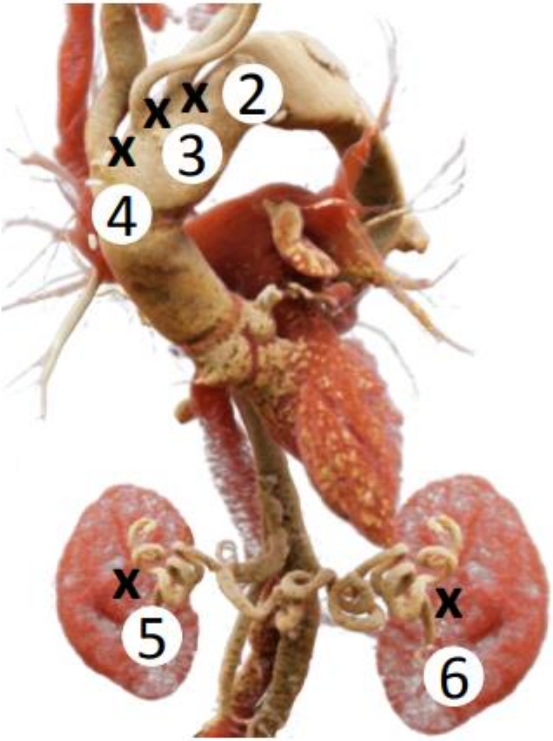


https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/decisionForests_MSR_TR_2011_114.pdf

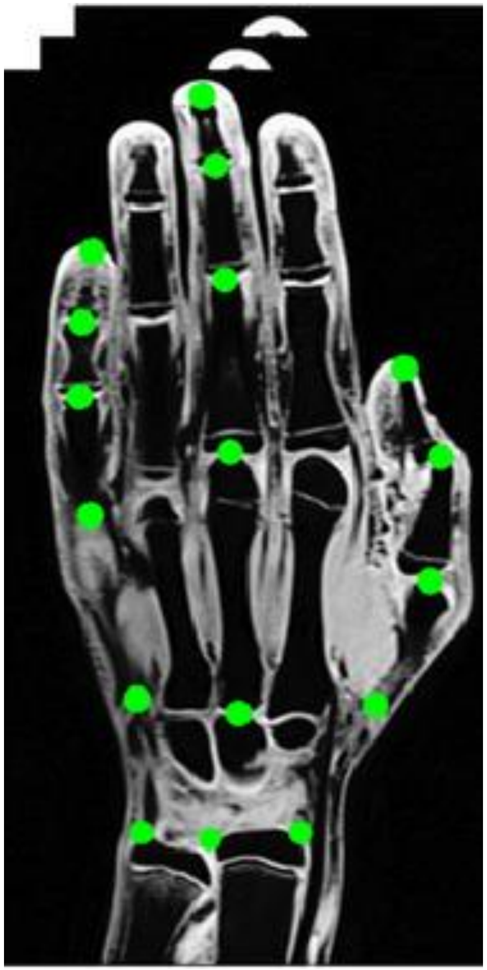
Random forests example: landmarking

- Landmark detection in medical imaging



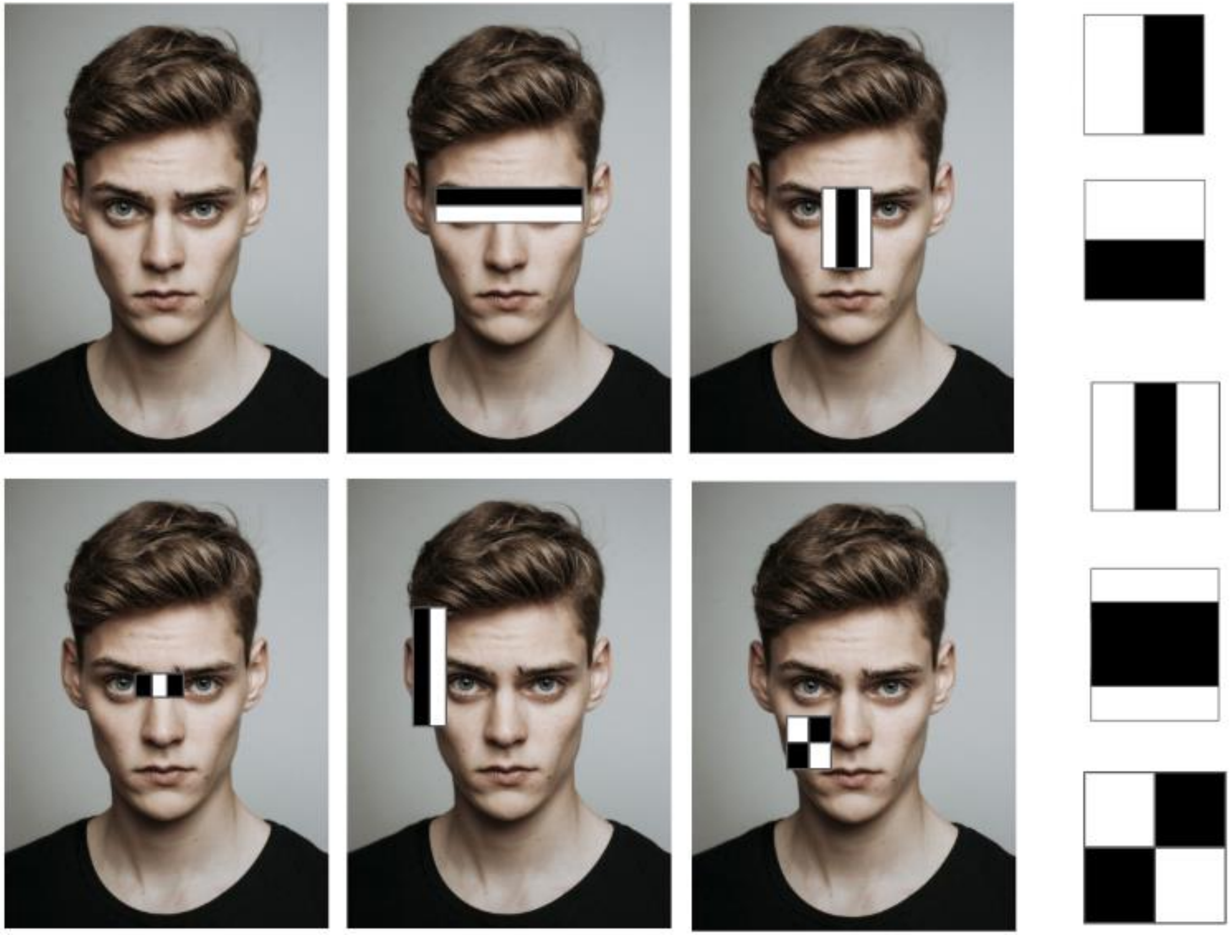


* from Siemens research 2018



Random forests example: features

- Single pixel intensity
- Box feature (or Haar features)

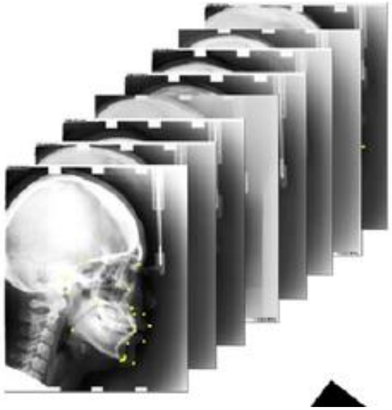


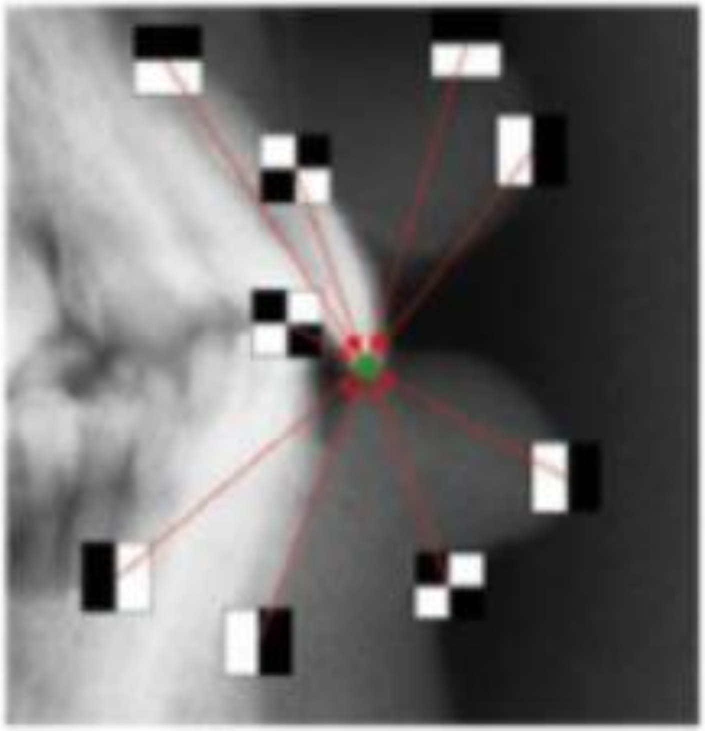
https://towardsdatascience.com/computer-vision-for-beginners-part-3-79de62dbeef7

Random forests example: features

- How to find out which features are important to detect an object?

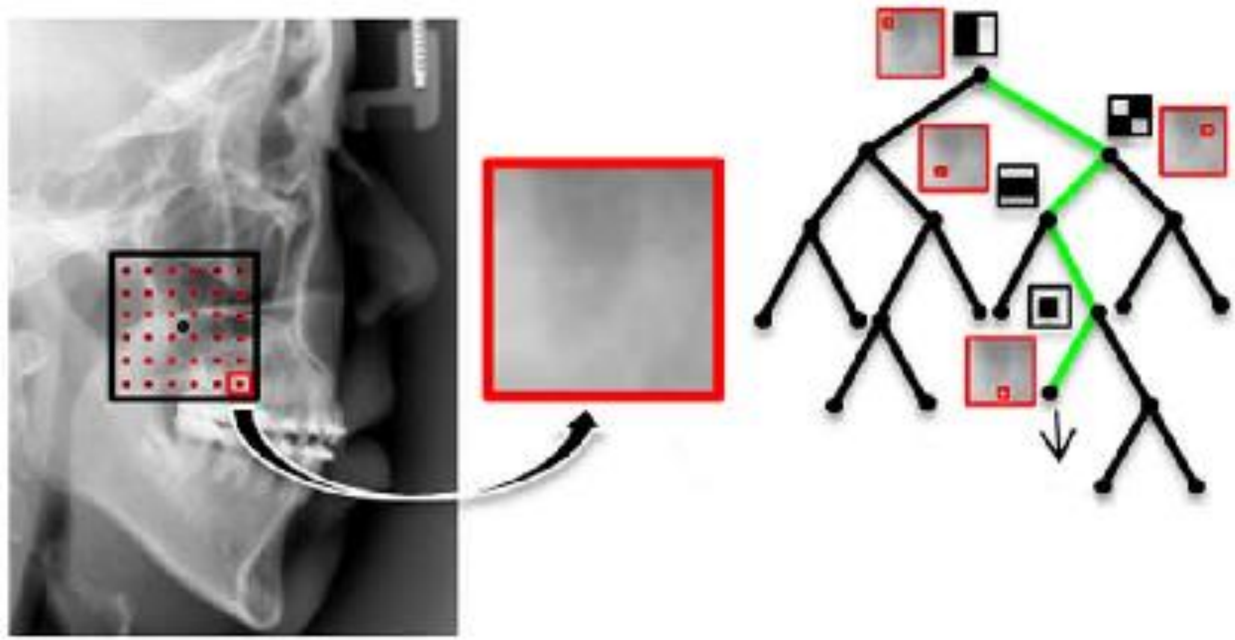
- Collect many images with the object
- Extract all features around the object examples in the images to form positive samples
- Extract all features around random pixels* in the image to form negative samples
- Generate random forest that will recognize important features and create rules for the object recognition





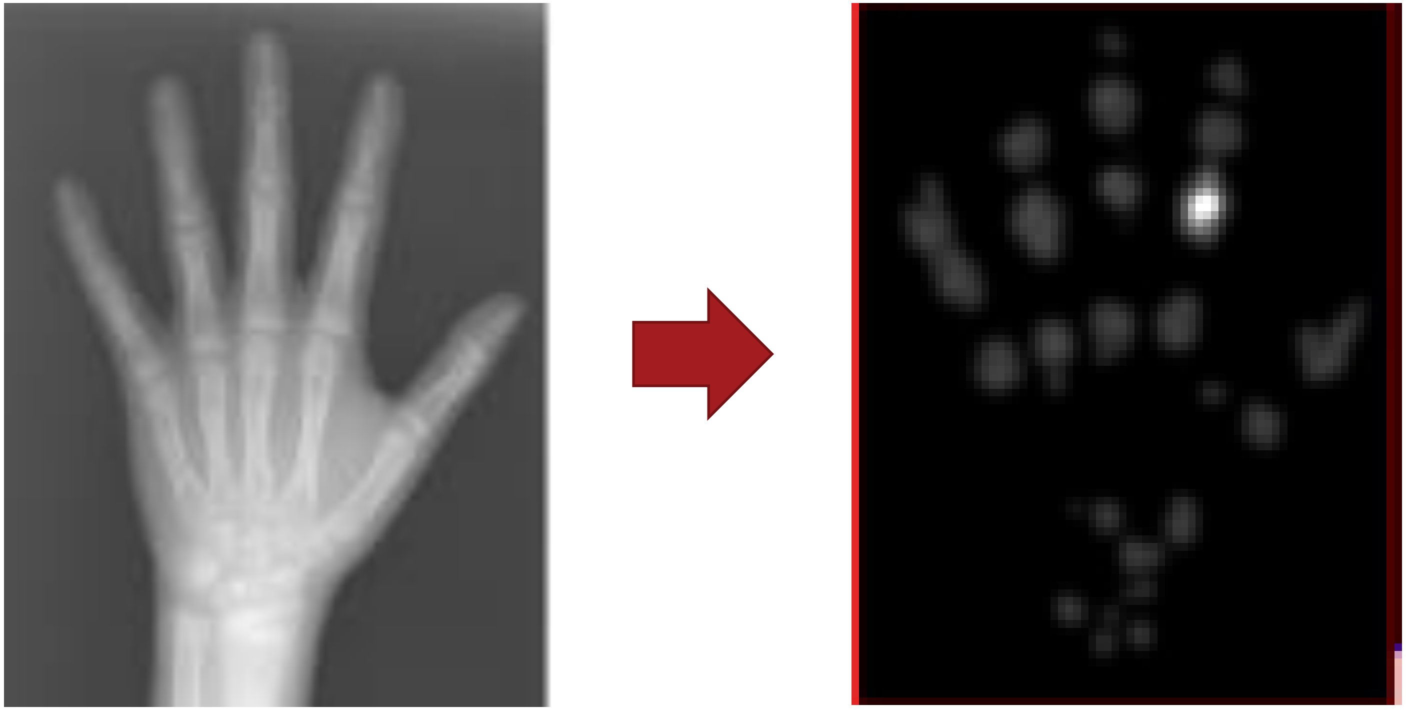
Random forests example: features

- For a new image:
  - Extract features for all pixels, and classify these pixels using the random forest
  - Find the pixels with the highest probability to be the landmark predicted by the random forest



Random forests example: features

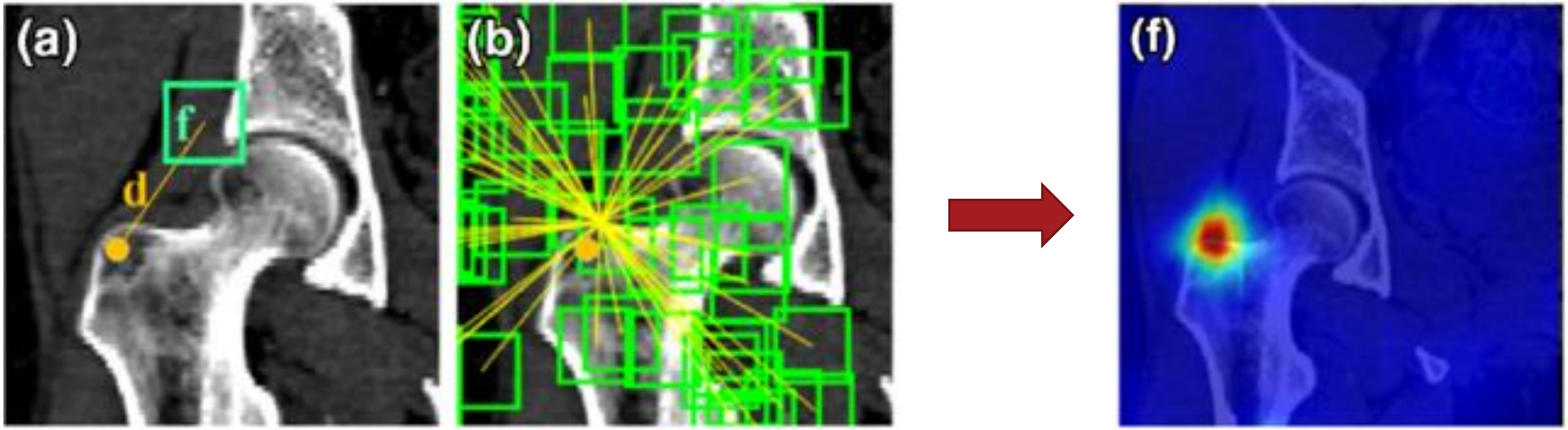
- For a new image:
  - Extract features for all pixels, and classify these pixels using the random forest
  - Find the pixels with the highest probability to be the landmark predicted by the random forest



Random forests example: features

- What type of the random forest was used for landmark detection example?
    - Classification / Regression

- Can we reformulate it as a regression?
    - Hint: if classification forest predicts pixels to be non-landmark, we just throw this pixel away. Can we still extract something useful from it?



Predict the displacement from pixel f to landmark d

from Guoyan Zheng

Questions?