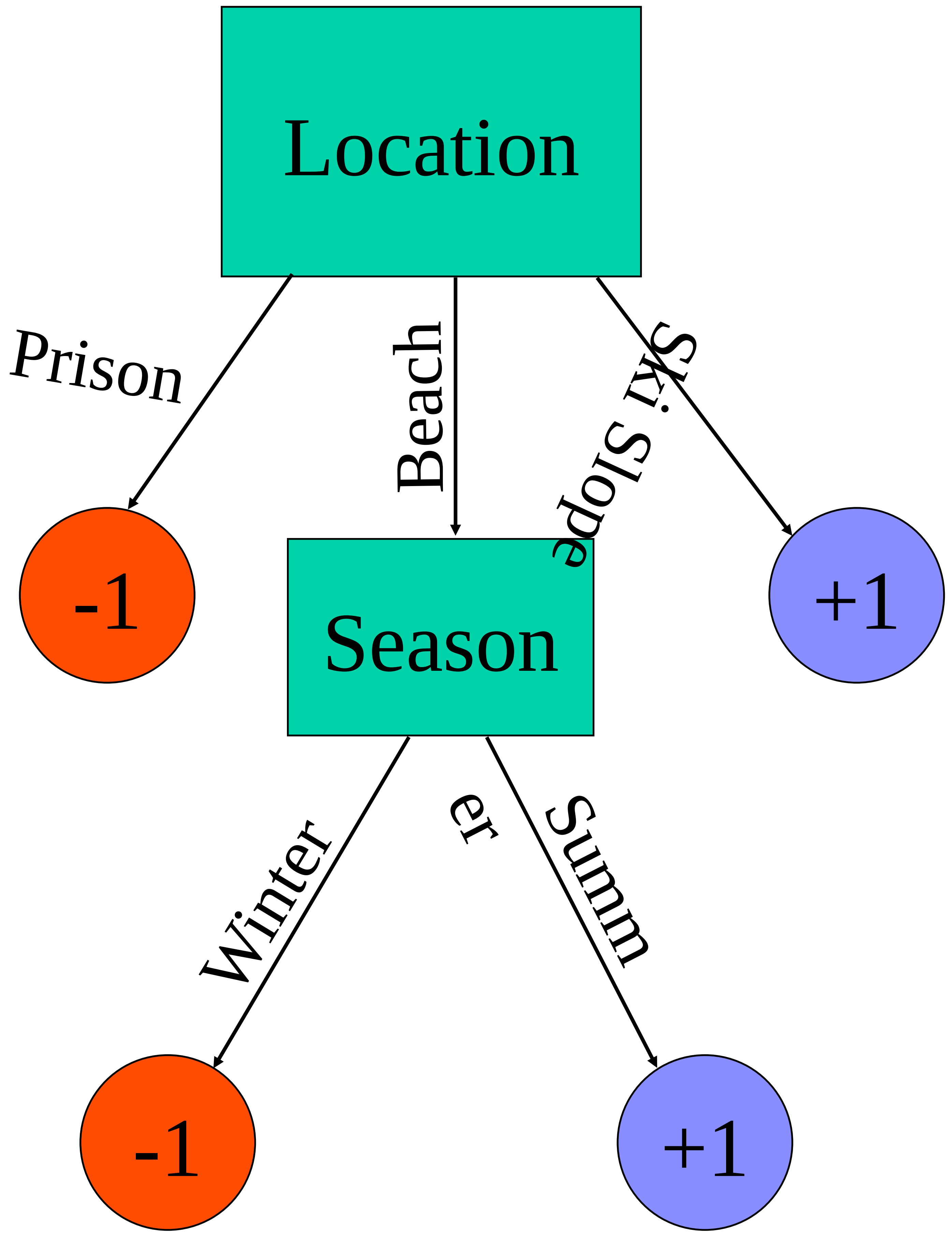
Decision trees

Decision Trees / Discrete Features

Where and when do you have fun?

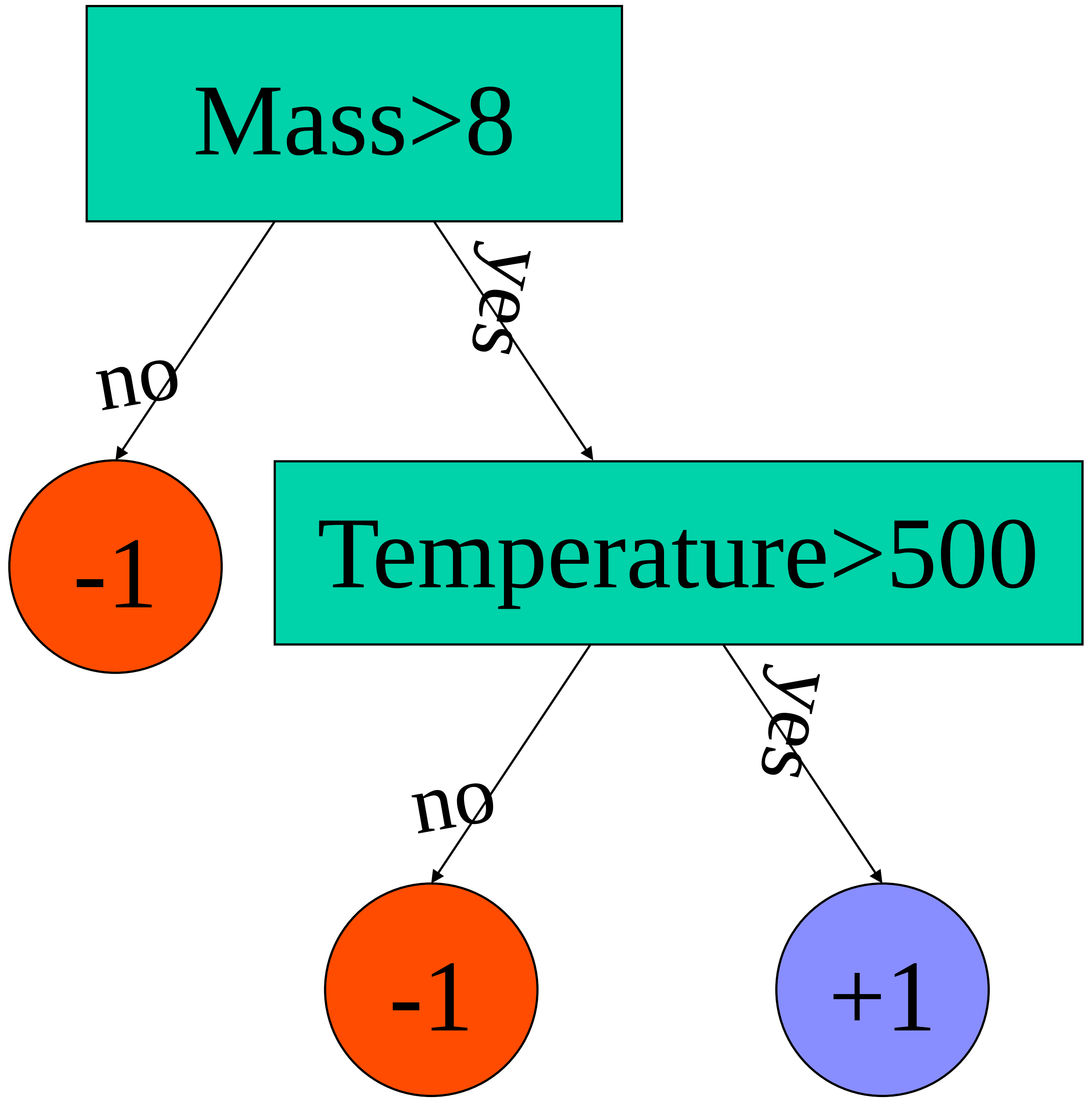
Season	Location	Fun?
summer	prison	-1
summer	beach	+1
Winter	ski-slope	+1
Winter	beach	-1



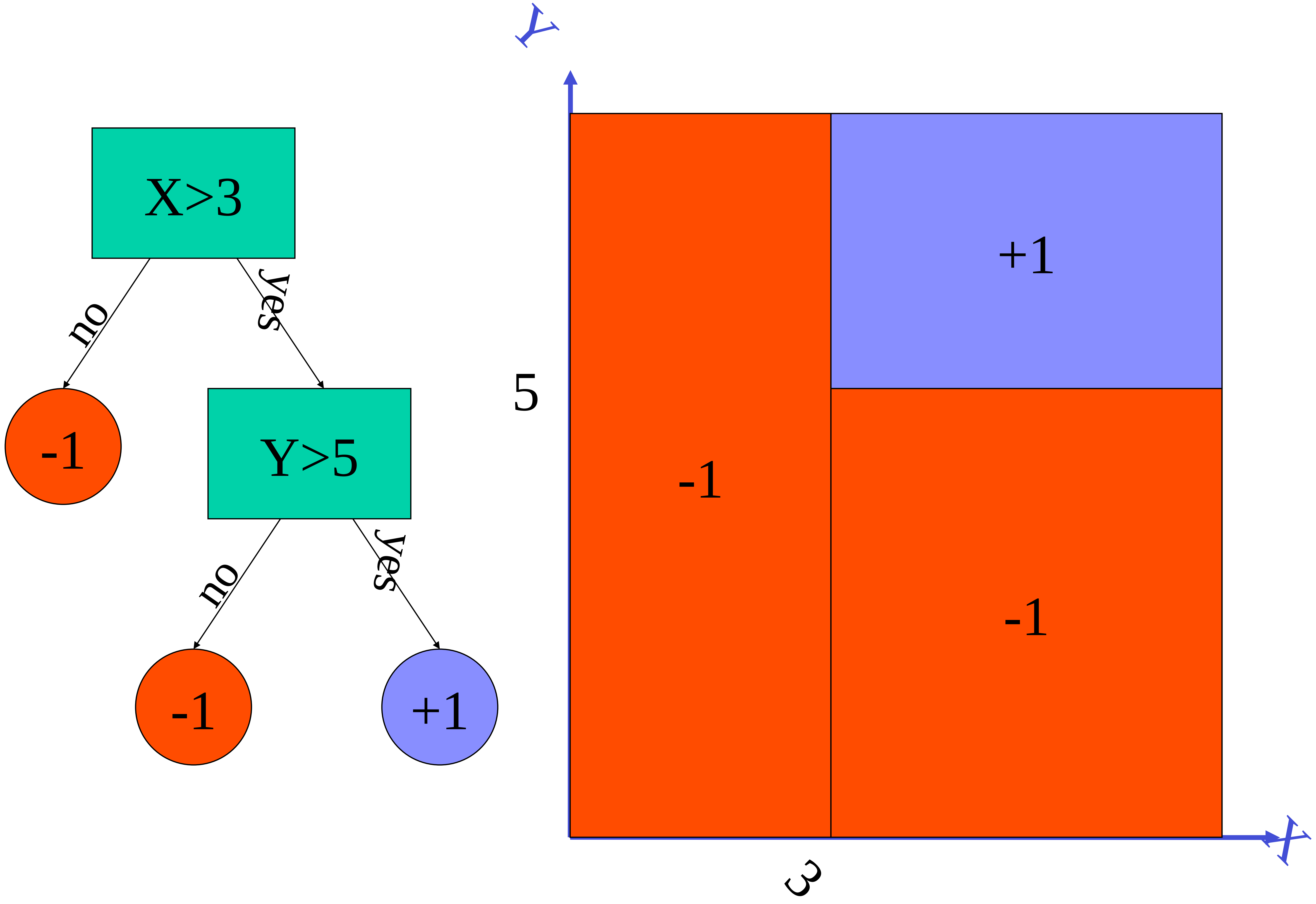
Decision Trees / Continuous Features

Will bomb explode?

Mass	Temperature	explosion
1	100	-1
3.4	945	-1
10	32	-1
11.5	1202	+1



Decision Trees



Decision trees

- Popular because very flexible and easy to interpret.

- Learning a decision tree = finding a tree with small error on the training set.

  1. Start with the root node.

  2. At each step split one of the leaves

  3. Repeat until a termination criterion.

Which node to split?

- We want the children to be more “pure” than the parent.
- Example:
  - Parent node is 50%+, 50%-.
  - Child nodes are (90%+,10%-),(10%+,90%-)
- How can we quantify improvement in purity?

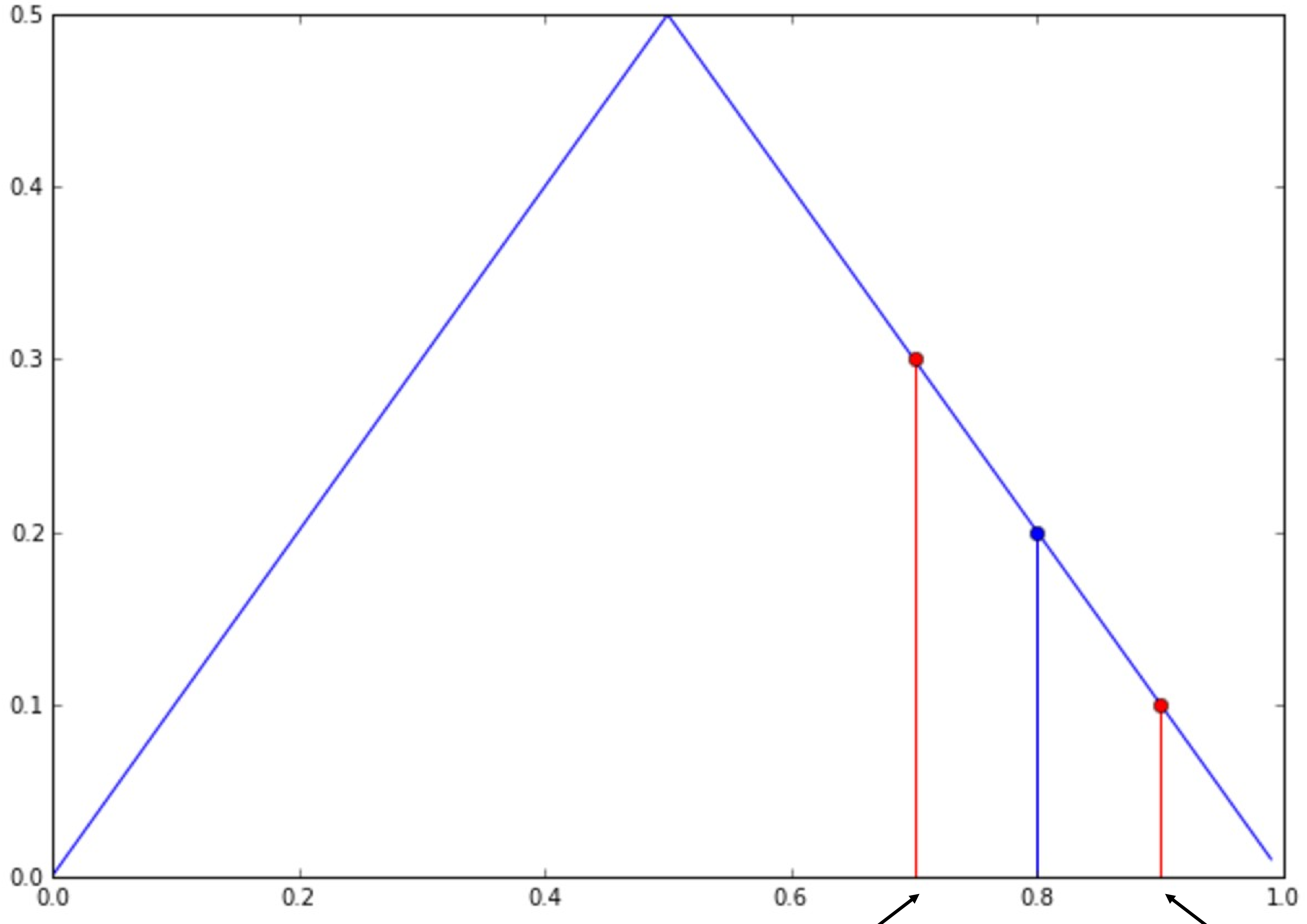
Naive approach: minimize training error

- **A good case**
- Parent node is 55%+, 45%-.
  - Predict + always: Error = 45%
- Child nodes are (90%+,10%-),(10%+,90%-)
  - Predict + on left, - on right: Error = 10%
- Clear Improvement

Naive approach: minimize training error

- **A bad case**
- Parent node is 80%+, 20%-.
  - Predict + always: Error = 20%
- Child nodes are (90%+,10%-),(70%+,30%-)
  - Predict + always: Error = 20%
- No improvement in training error!

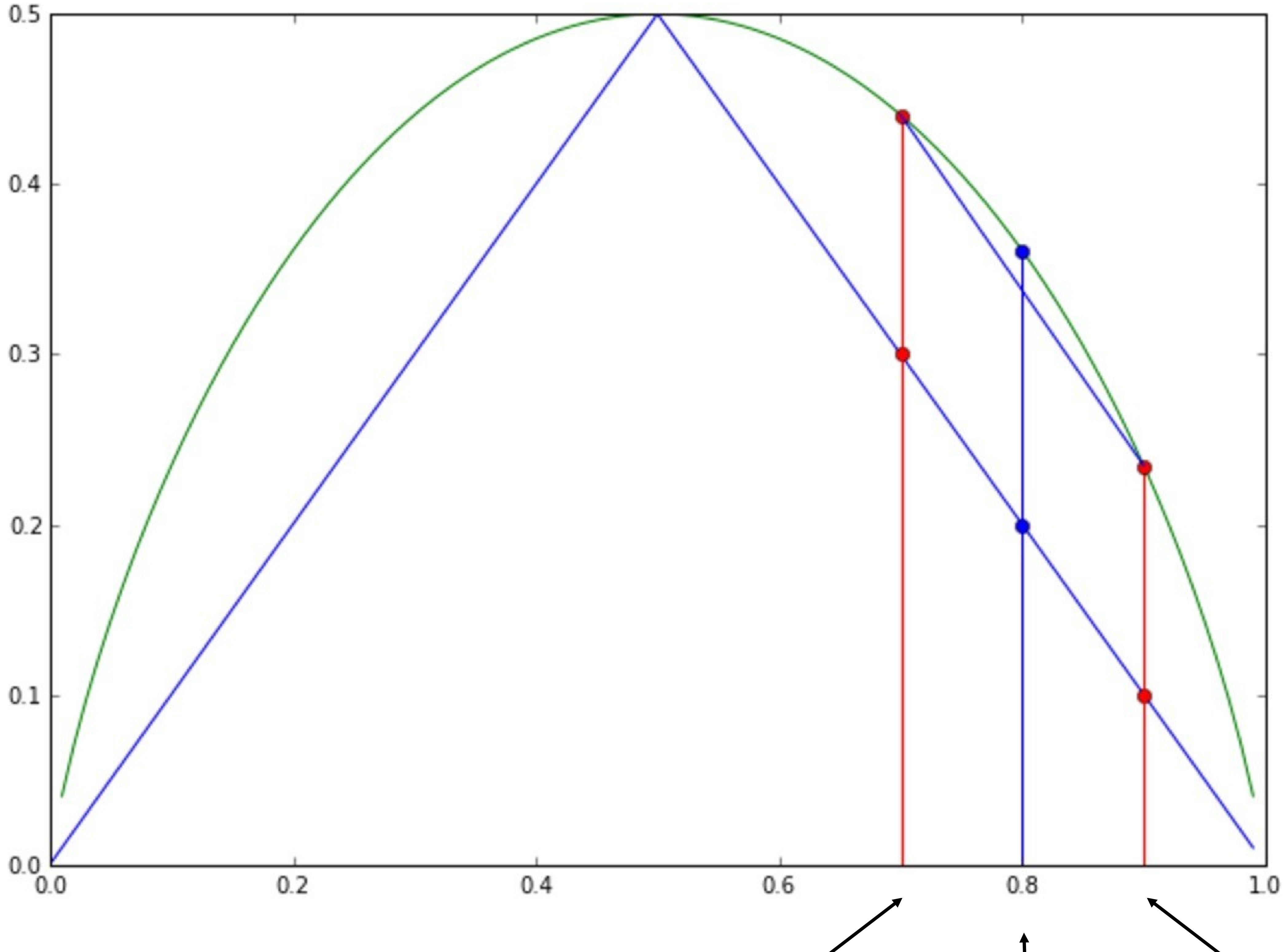
The problem with classification error (pictorially)



$$P(+1 \mid A) = 0.7,\ P(+1) = 0.8,\ P(+1 \mid B) = 0.9$$

Fixing the problem

instead of $\text{err}(p) = \min(p, 1-p)$ use $\dfrac{H(p)}{2} = -\dfrac{1}{2}\big(p\log_2 p + (1-p)\log_2(1-p)\big)$



$P(+1 \mid A) = 0.7,\ P(+1) = 0.8,\ P(+1 \mid B) = 0.9$

Any strictly concave function can be used

$H(P) =$

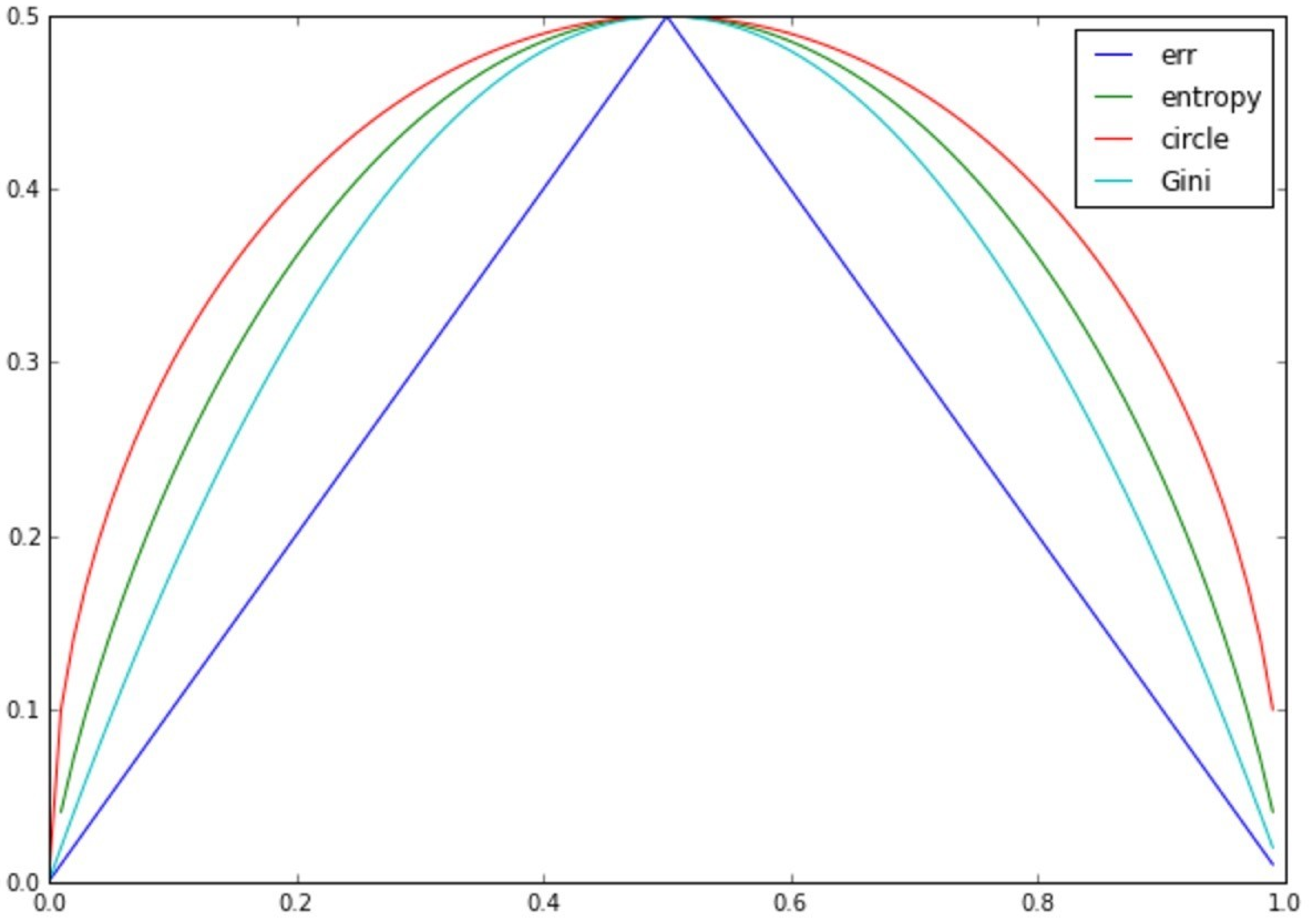
$p \log p + (1-p)\log(1-p)$

$Circle(p) =$

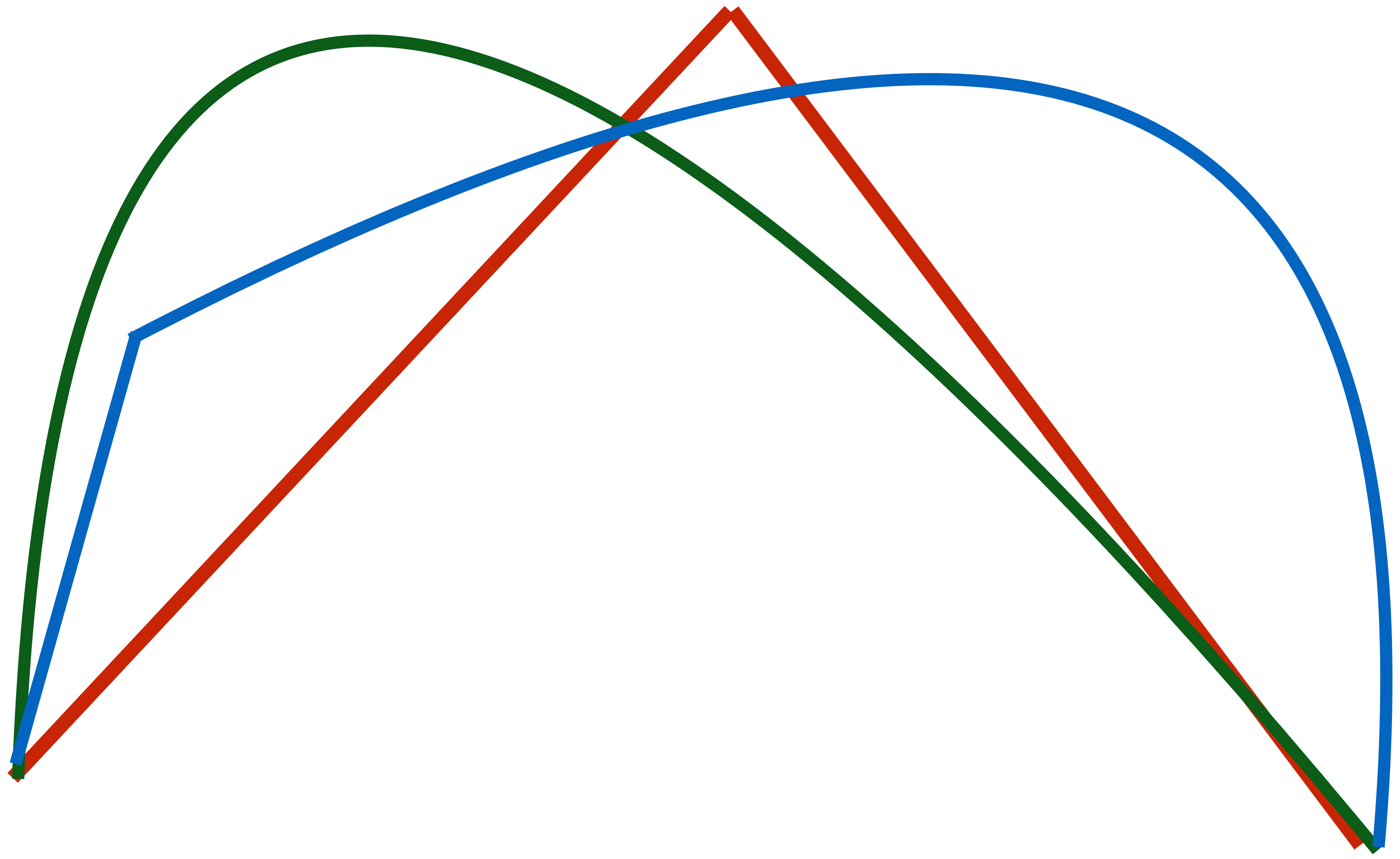
$\sqrt{1/4 - (p - 1/2)^2}$

$Gini(p) =$

$p(1-p)$



Which of the following is strictly concave?



Decision tree learning algorithm

- Learning a decision tree = finding a tree with small error on the training set.

1. Start with the root node.

2. At each step split one of the leaves

3. Repeat until a termination criterion.

4. Next, How do we search for a splitting rule.

The splitting step

- Given: current tree.
- For each leaf and each feature,
  - find all possible splitting rules (finite because data is finite).
  - compute reduction in entropy
- find leaf X feature X split rule the minimizes entropy.
- Add selected rule to split selected leaf.

Enumerating splitting rules

- If feature has a fixed, small, number of values. then either:
  - Split on all values (Location is beach/prison/ski-slope)
  - or Split on equality to one value (location = beach)
- If feature is continuous (temperature) then either:
  - Sort records by feature value and search for best split.
  - or split on parcentiles: 1%,2%,….,99%

Splitting on percentiles

- Suppose data is in an RDD with 100 million examples.
- sorting according to each feature value is very expensive.
- Instead: use Sample(false,0.00001).collect() to get a sample of about 10,000 examples.
- sort the sample (small, sort in head node).
- pick examples at location 100,200,… as boundaries. Call those feature values T1,T2,T3,… ,T99
- Broadcast boundaries to all partitions.
- Each partition computes it’s contribution to $P(+1 \mid T_i \leq f \leq T_{i+1})$

Summary

- The splitting criteria is a strictly concave function that bounds the training error.
- The search for a splitting rule tests all possible splits of current tree.
- When the feature is continuous and the data large - split on percentiles instead of splitting on each example.
- Next, Performance on the test set and reducing over-fitting .

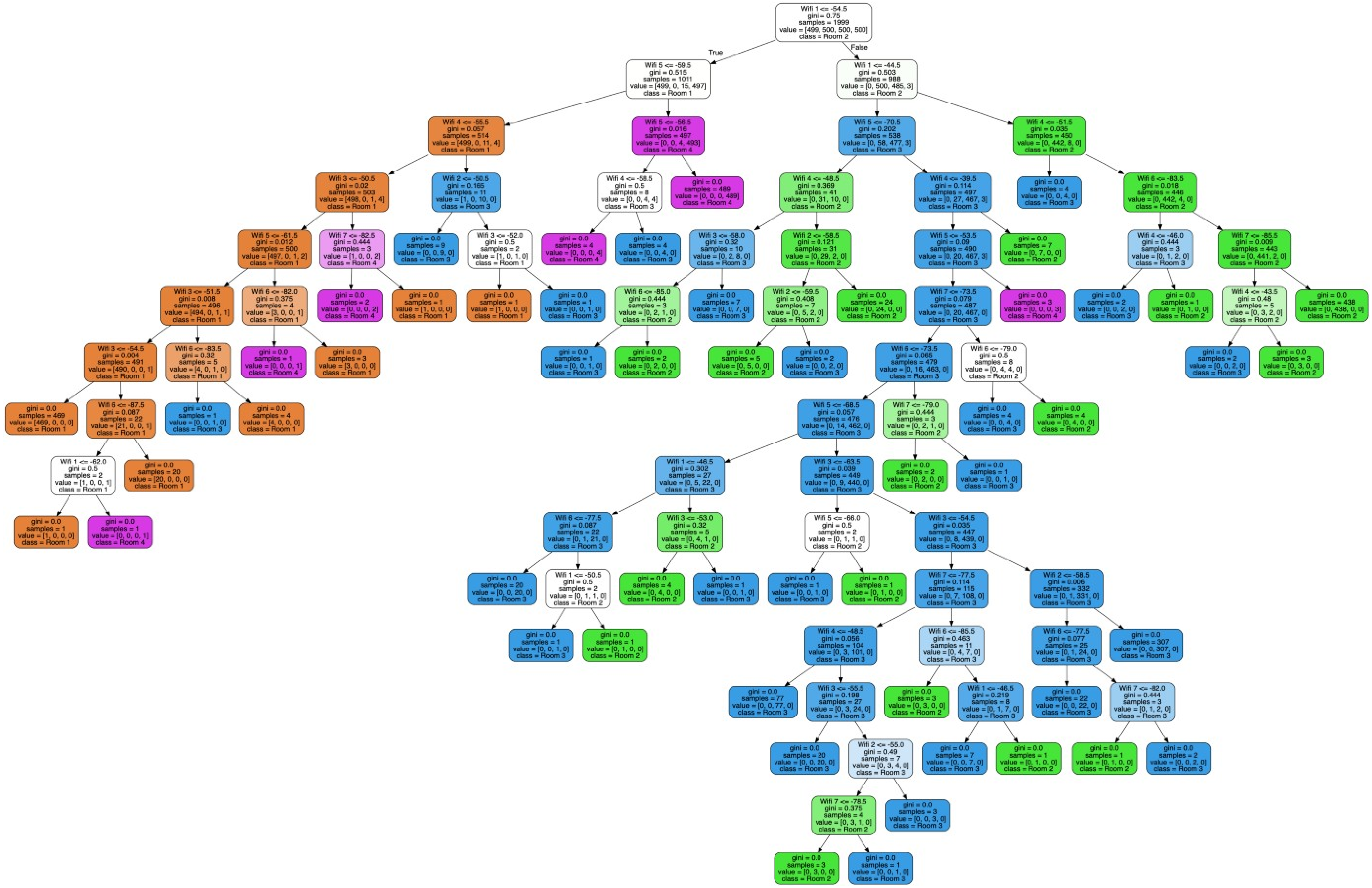
Trees are unstable

- Trees are very flexible.
- A “fully grown” tree is one where all leaves are “pure”, i.e. each leaf contains all + labeled examples or all - labeled examples.
- A fully grown tree has training error zero.
- If the tree is large and the data is limited, the test error of the tree is likely to be high = the tree overfits the data.
- Statisticians say that trees are “high variance” or “unstable”

Pruning Trees

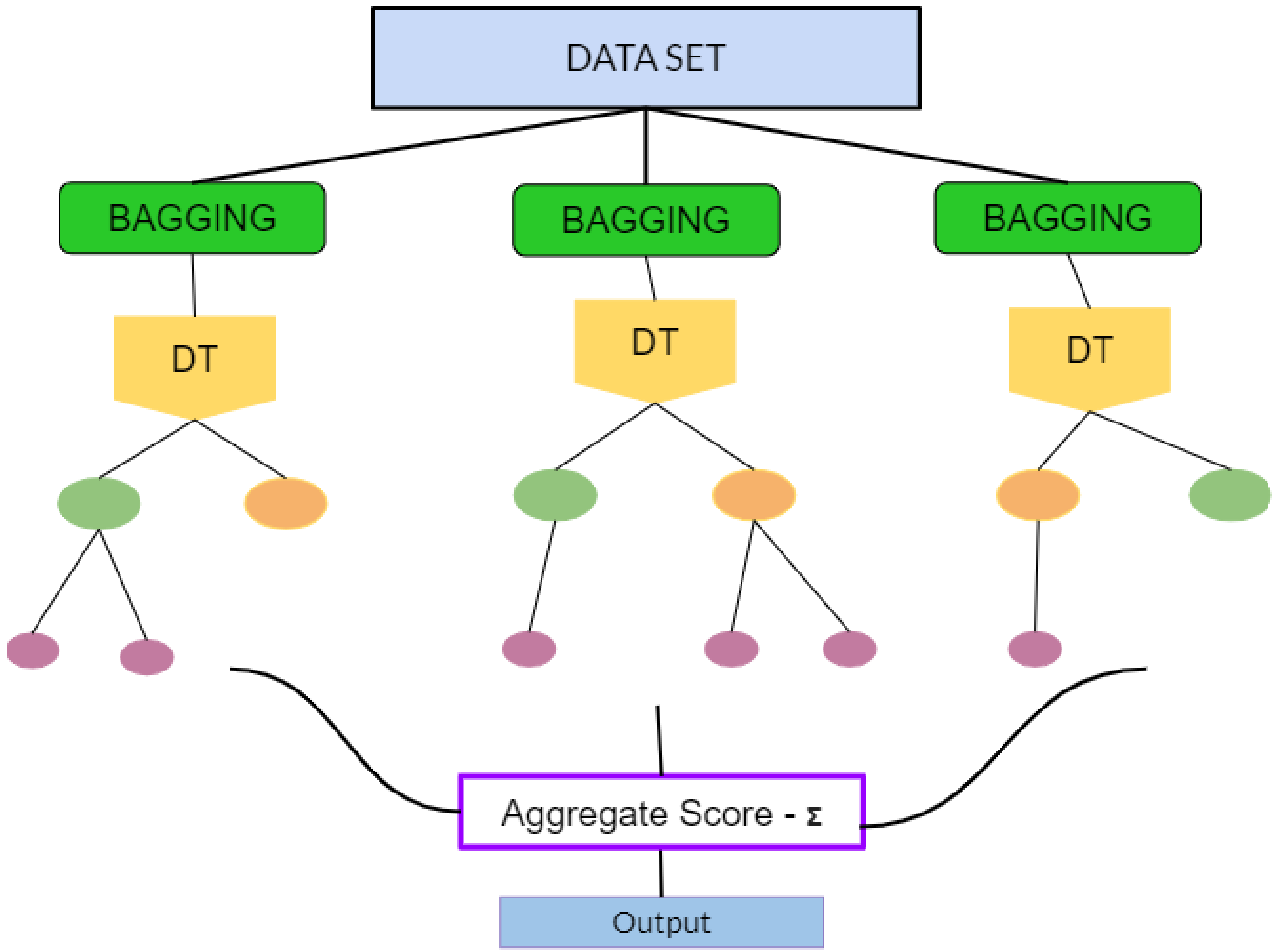
- One way to make trees more stable is to prune.
- Start by building a tree down to training error of zero.
- Iteratively remove leaves until the training error is

$$\epsilon = 1\%$$



Bagging Trees

- Bagging, invented by Leo Breiman in the 90s, is a different way to reduce the variance of trees.
- Instead of pruning the tree, we generate many trees, using randomly selected subsets of the training data.
- We predict using the majority vote over the trees.
- Related to RANDOM FORESTS and to Boosted Trees/XGBoost. Will describe next.



Summary

- Decision trees are a simple and intuitive representation.
- Fully grown trees over-fit the data.
- Two ways to reduce over-fitting:
  - Pruning: remove leaves to make the tree smaller.
  - Bagging, Boosting: take the majority vote over many trees.

Are methods based on decision tree still relevant at the age of Deep Neural Networks?

Kaggle Survey of most popular learning algorithms (2020)