UVA CS 6316: Machine Learning

Lecture 18: Decision Tree / Bagging

Dr. Yanjun Qi

University of Virginia
Department of Computer Science

Course Content Plan ➔ Six major sections of this course

- ❑ ~~Regression (supervised)~~ ⬅ Y is a continuous
- ❑ Classification (supervised) ⬅ Y is a discrete
- ❑ Unsupervised models ⬅ NO Y
- ❑ Learning theory ⬅ About f()
- ❑ Graphical models ⬅ About interactions among X1,… Xp
- ❑ Reinforcement Learning ⬅ Learn program to Interact with its environment

Three major sections for classification

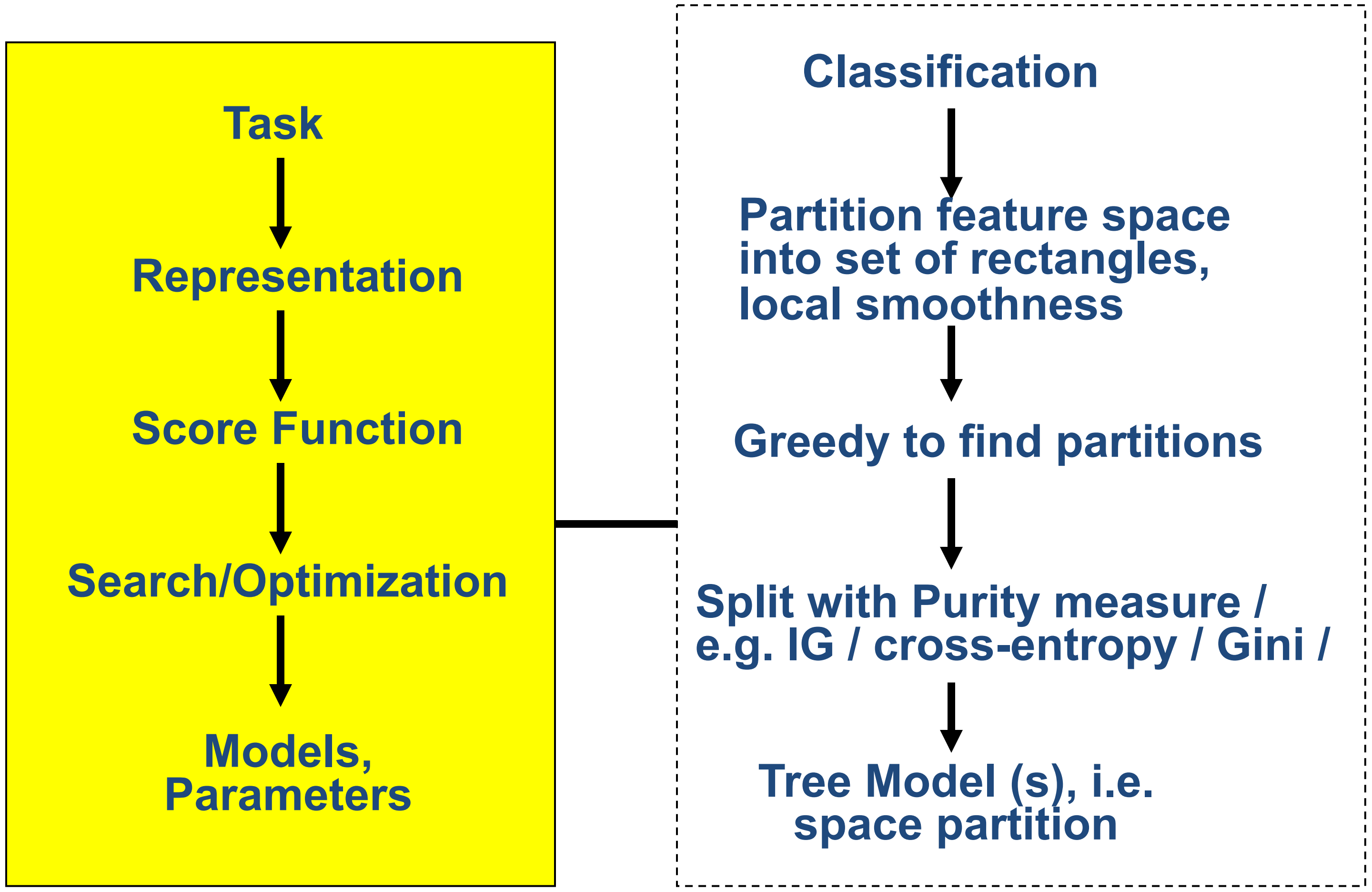
- We can divide the large variety of classification approaches into roughly three major types

1. Discriminative
   directly estimate a decision rule/boundary
   e.g.~~, support vector machine~~, decision tree, ~~logistic regression,~~
   e.g~~. neural networks (NN), deep NN~~

2. Generative:
   build a generative statistical model
   e.g., ~~Bayesian networks, Naïve Bayes classifier~~

3~~. Instance based classifiers~~
   - Use observation directly (no models)
   ~~- e.g. K nearest neighbors~~

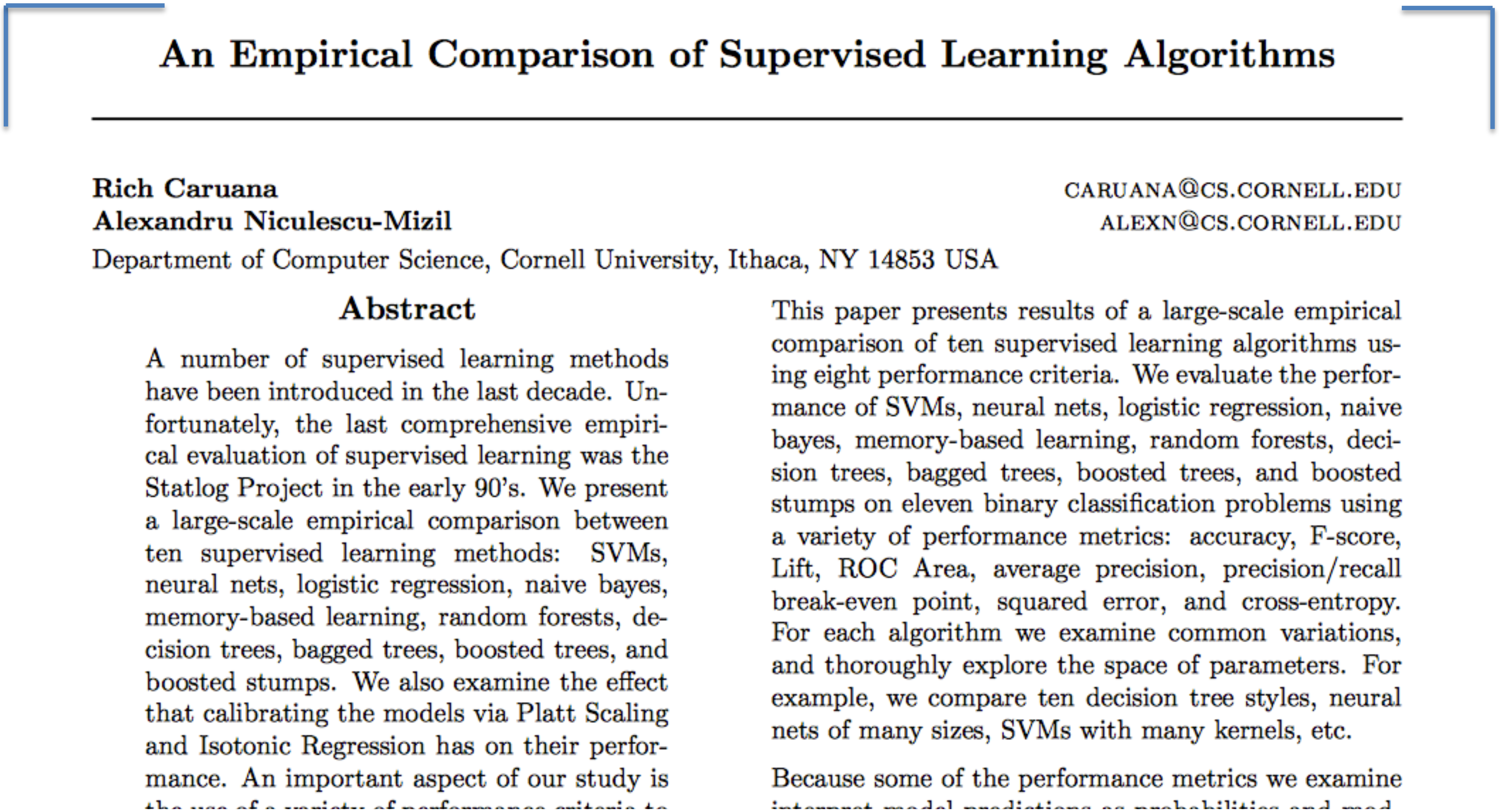
Decision Tree / Random Forest


Task
Representation
Score Function
Search/Optimization
Models, Parameters
Classification
Partition feature space into set of rectangles, local smoothness
Greedy to find partitions
Split with Purity measure / e.g. IG / cross-entropy / Gini /
Tree Model (s), i.e. space partition

Today

- Decision Tree (DT):
  - Tree representation
- Brief information theory
- Learning decision trees
- Bagging
- Random forests: Ensemble of DT
- More about ensemble

A study comparing Classifiers



An Empirical Comparison of Supervised Learning Algorithms

Rich Caruana CARUANA@CS.CORNELL.EDU
Alexandru Niculescu-Mizil ALEXN@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

A number of supervised learning methods have been introduced in the last decade. Unfortunately, the last comprehensive empirical evaluation of supervised learning was the Statlog Project in the early 90's. We present a large-scale empirical comparison between ten supervised learning methods: SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps. We also examine the effect that calibrating the models via Platt Scaling and Isotonic Regression has on their performance. An important aspect of our study is

This paper presents results of a large-scale empirical comparison of ten supervised learning algorithms using eight performance criteria. We evaluate the performance of SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps on eleven binary classification problems using a variety of performance metrics: accuracy, F-score, Lift, ROC Area, average precision, precision/recall break-even point, squared error, and cross-entropy. For each algorithm we examine common variations, and thoroughly explore the space of parameters. For example, we compare ten decision tree styles, neural nets of many sizes, SVMs with many kernels, etc.

Because some of the performance metrics we examine

Proceedings of the 23rd International Conference on Machine Learning (ICML `06).

A study comparing Classifiers ➔ 11 binary classification problems / 8 metrics

Top 8 Models

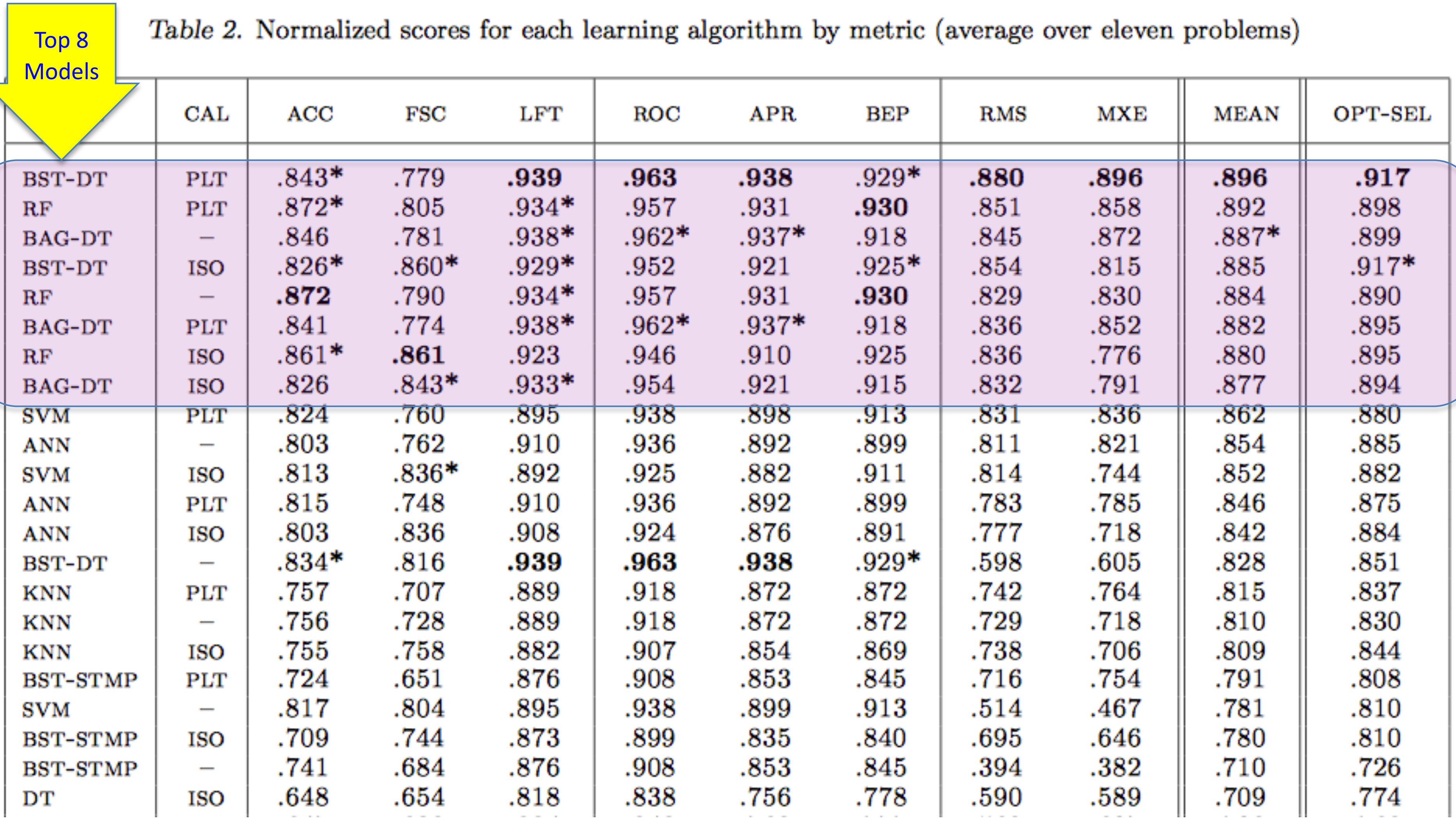
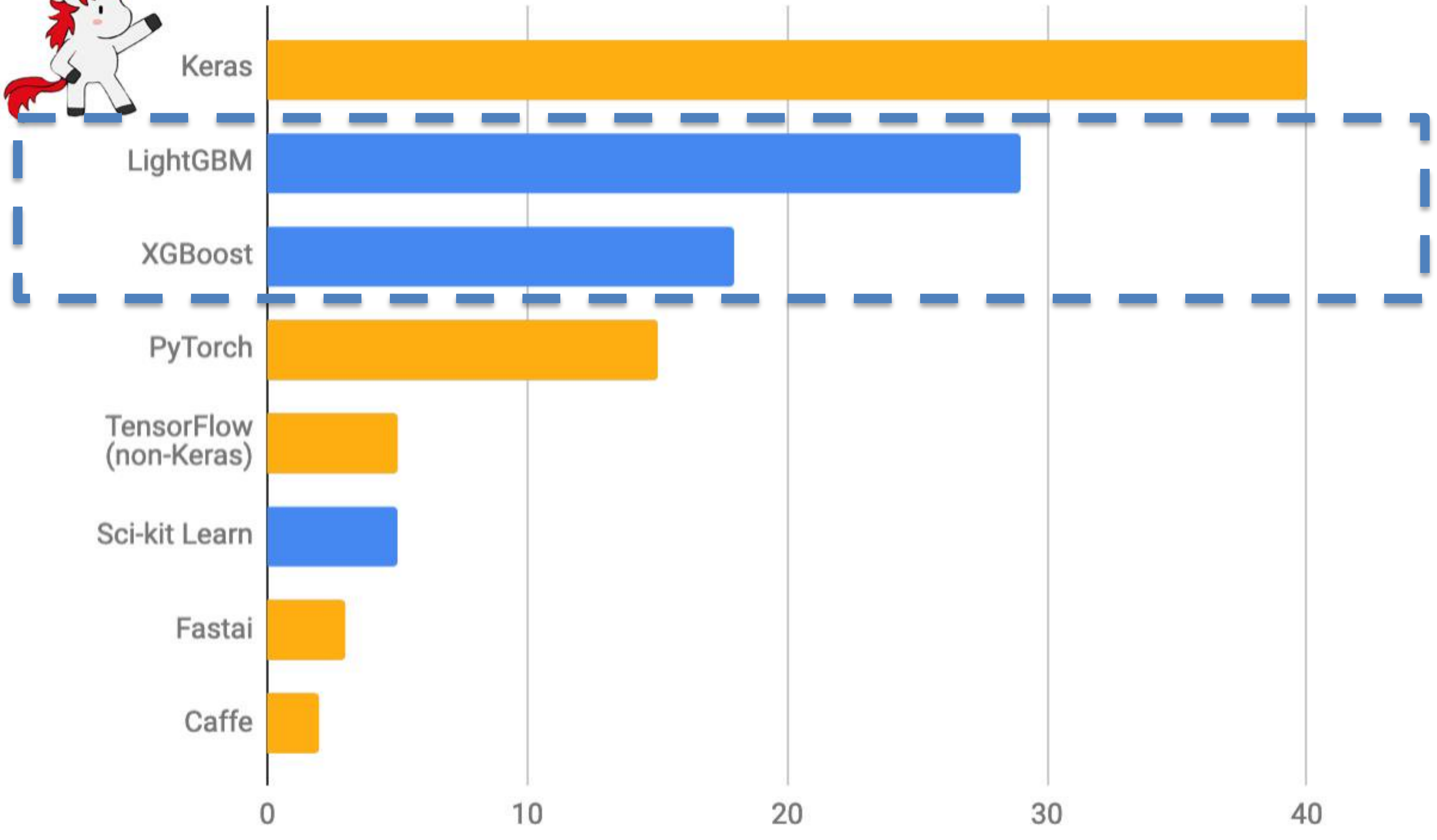


Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	–	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	–	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	–	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	–	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	–	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	–	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	–	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774

Primary ML software tool used by **top-5 teams** on Kaggle in each competition (n=120)



🟧 Deep 🟦 Classic

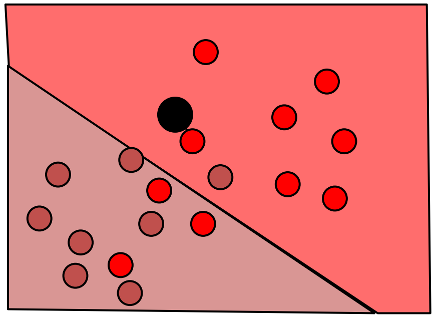
From: François Chollet 2019

Readability Hierarchy

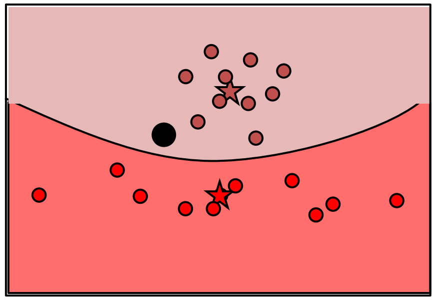
Readable

Decision Trees: Classifies based on a series of one-variable decisions.

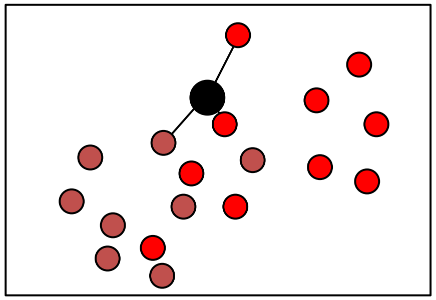
Linear Classifier: Weight vector w tells us how important each variable is for classification and in which direction it points.



Quadratic Classifier: Linear weights work as in linear classifier, with additional information coming from all products of variables.



k Nearest Neighbors: Classifies using the complete training set, no information about the nature of the class difference



Example

$P(P|O,T,H,W)$

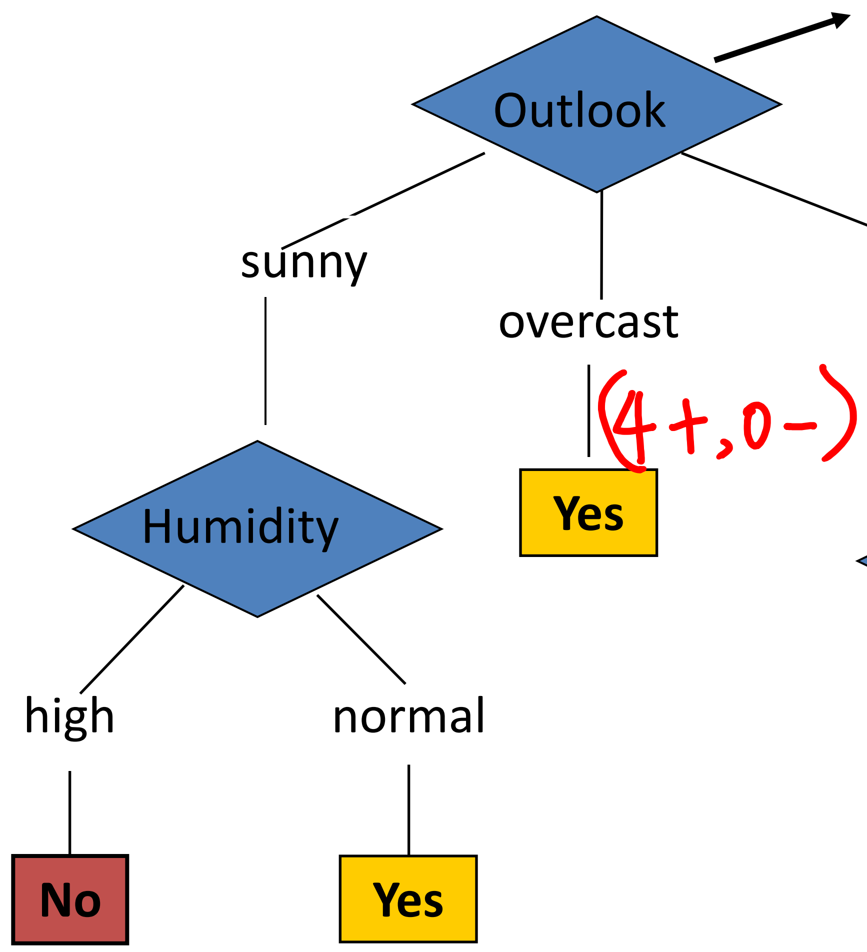
NBC

- Example: Play Tennis

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes ←
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes ←
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes ←
D13	Overcast	Hot	Normal	Weak	Yes ←
D14	Rain	Mild	High	Strong	No

Anatomy of a decision tree

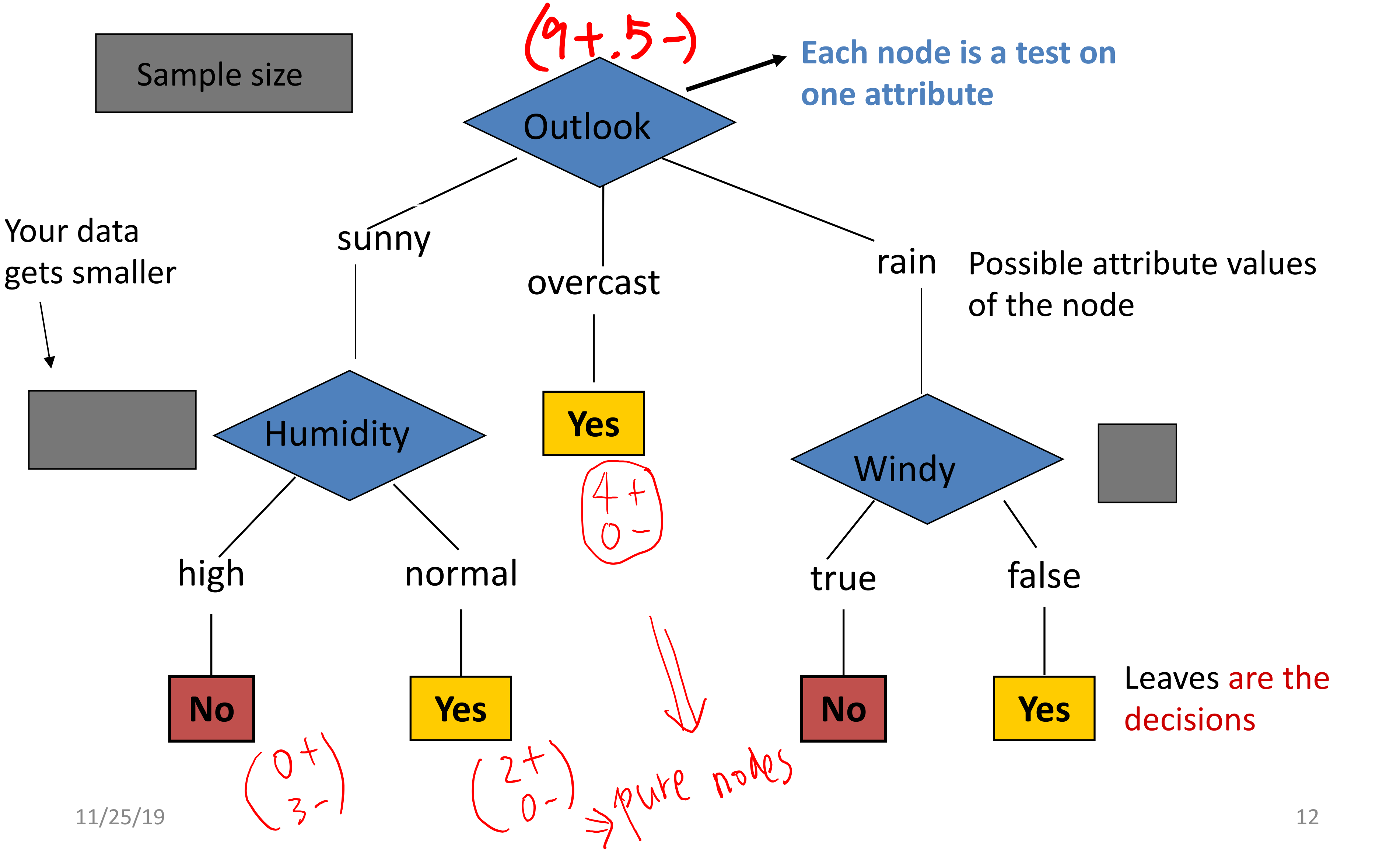


Each node is a test on one feature/attribute

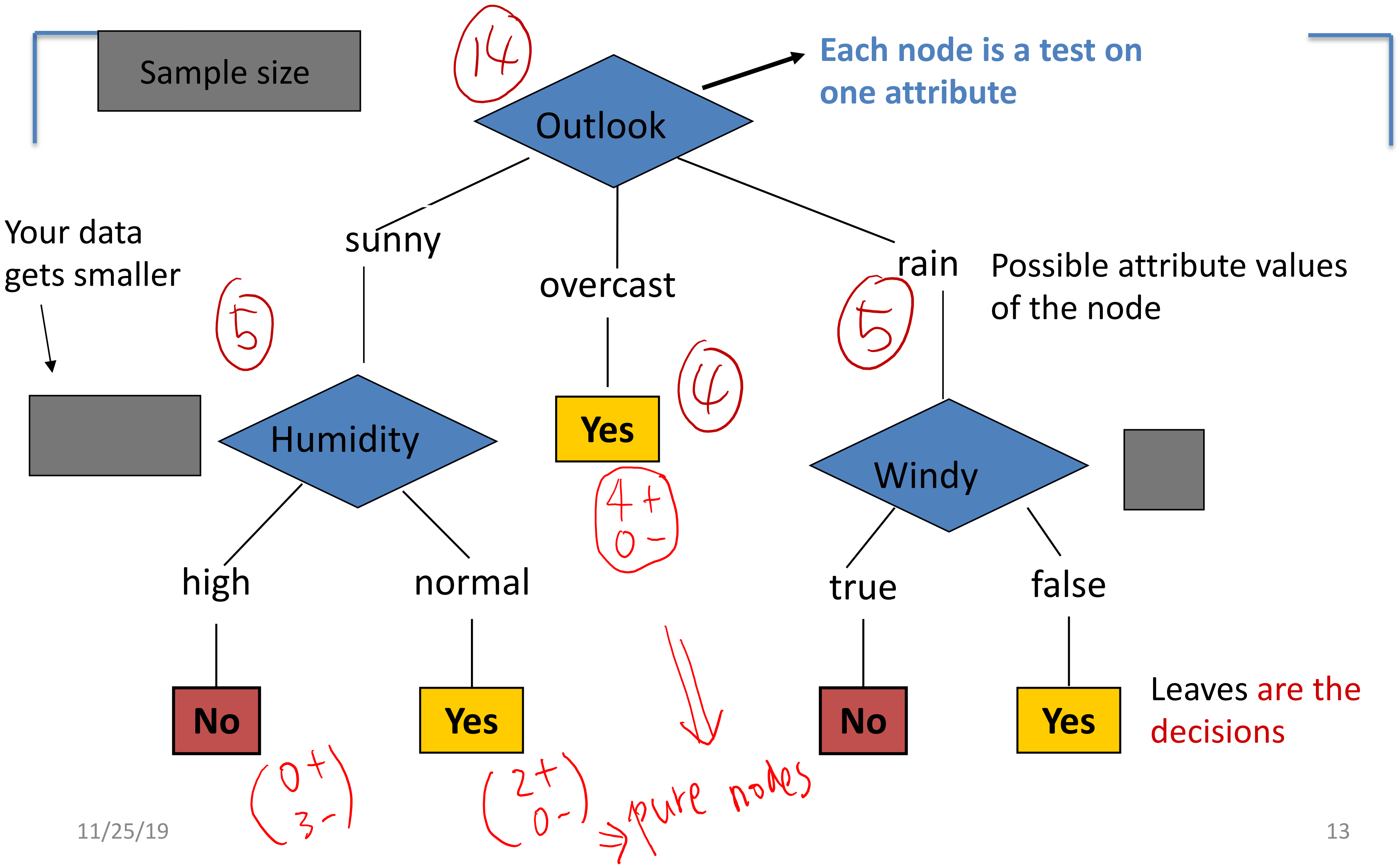
Possible attribute values of the node

Leaves are the decisions

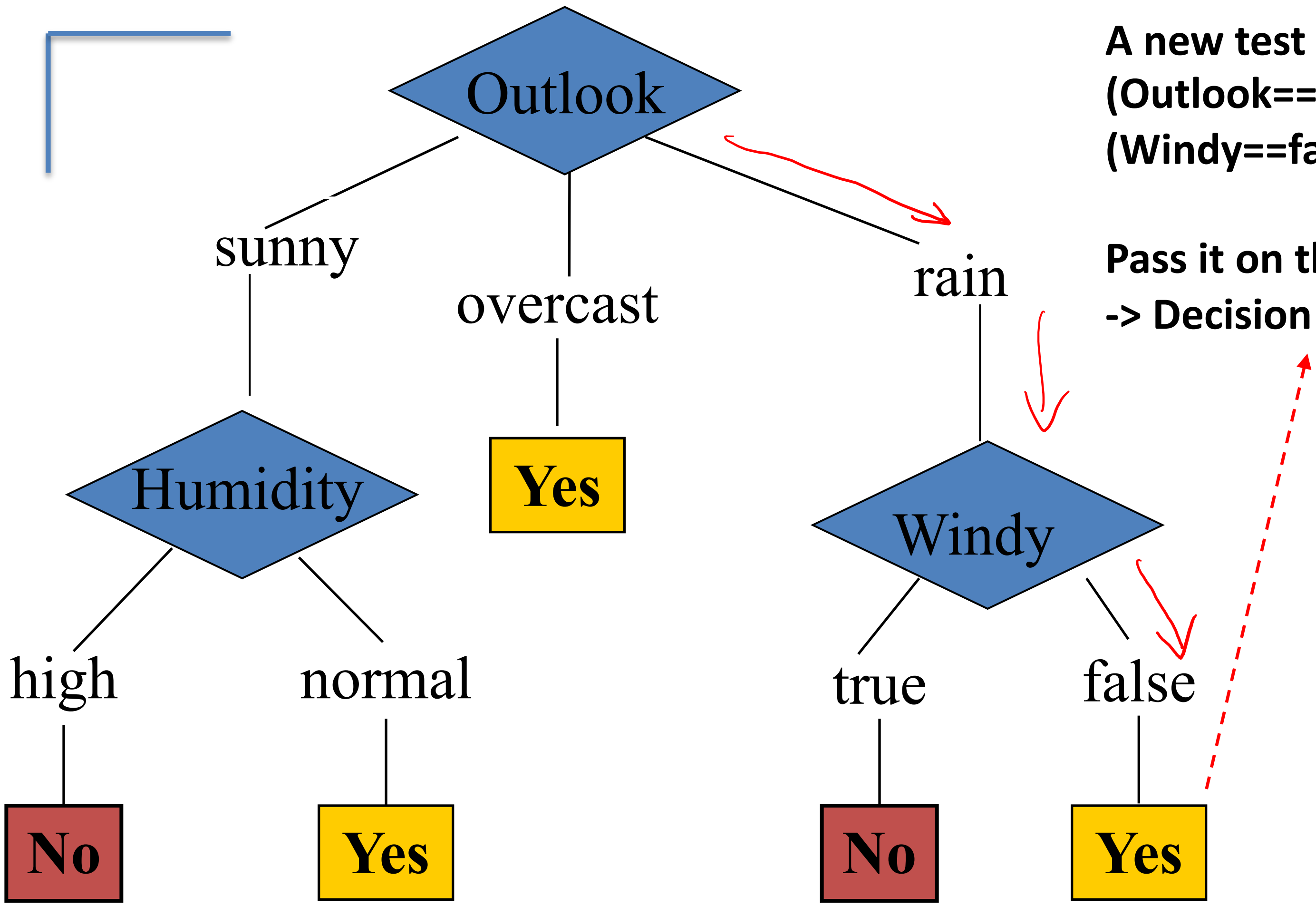
Anatomy of a decision tree



Anatomy of a decision tree



Apply Model to Test Data: To ‘play tennis’ or not.

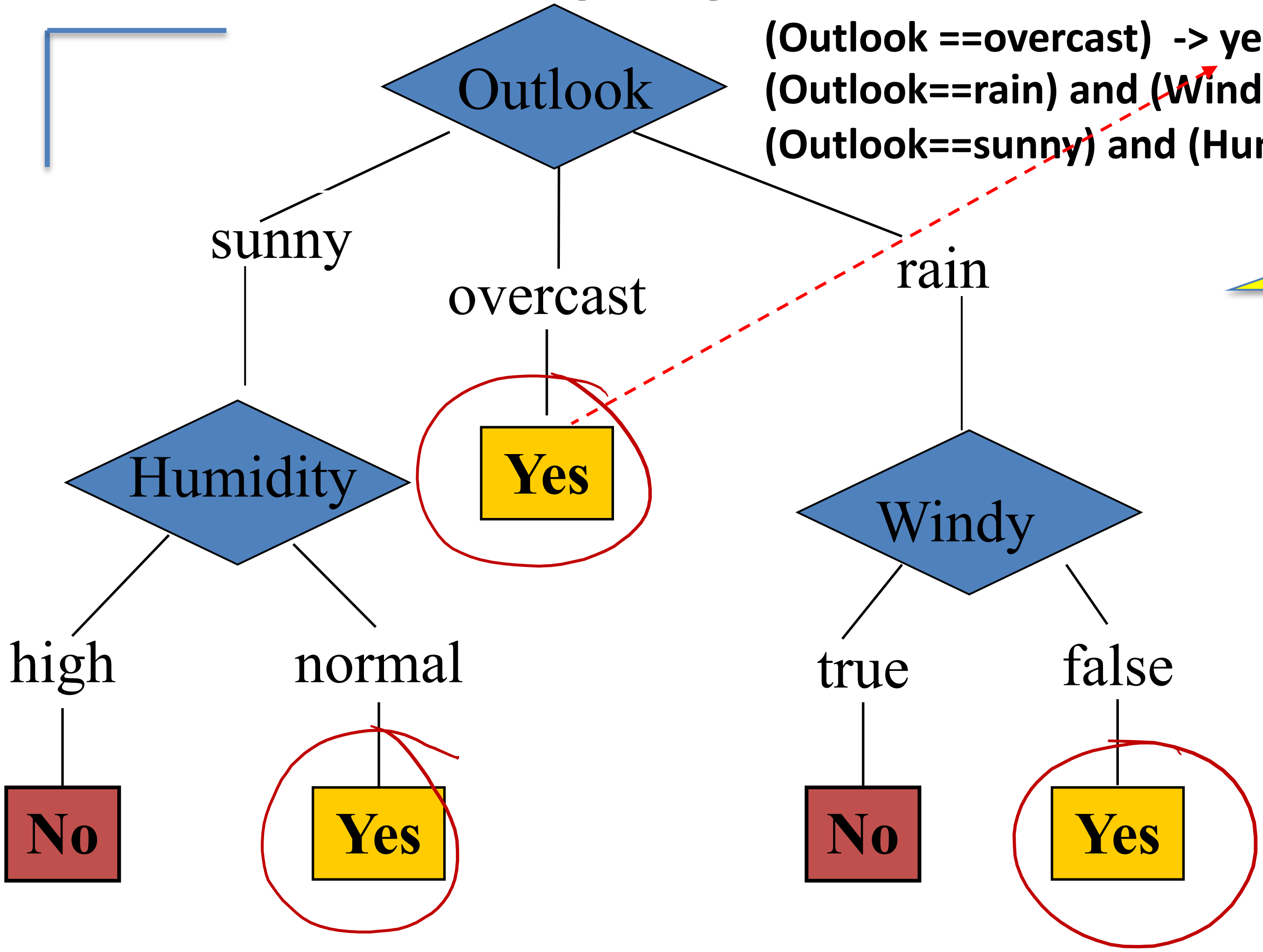


A new test example: (Outlook==rain) and (Windy==false)

Pass it on the tree -> Decision is yes.

Apply Model to Test Data:

To ‘play tennis’ or not.



(Outlook ==overcast) -> yes
(Outlook==rain) and (Windy==false) ->yes
(Outlook==sunny) and (Humidity=normal) ->yes

Three cases of "YES"

Decision trees (on Discrete)

- Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.

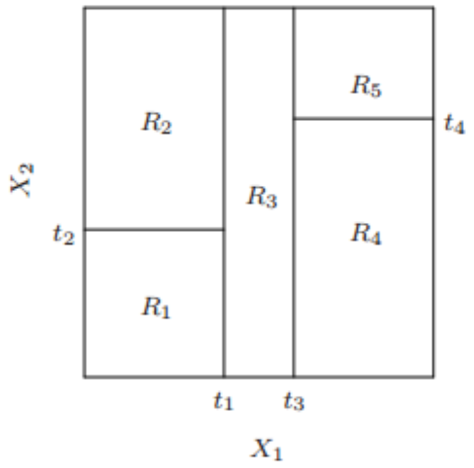
```
(Outlook ==overcast)
 OR
((Outlook==rain) and (Windy==false))
 OR
((Outlook==sunny) and (Humidity=normal))
 => yes play tennis
```

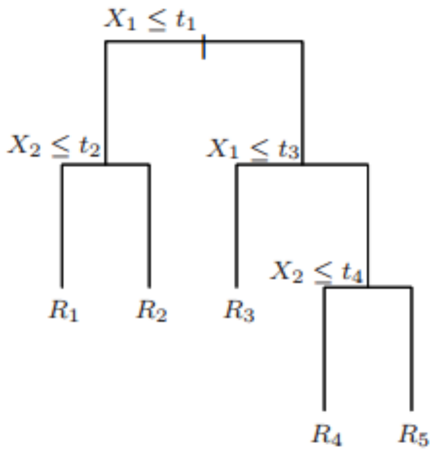
Decision trees (on Continuous)

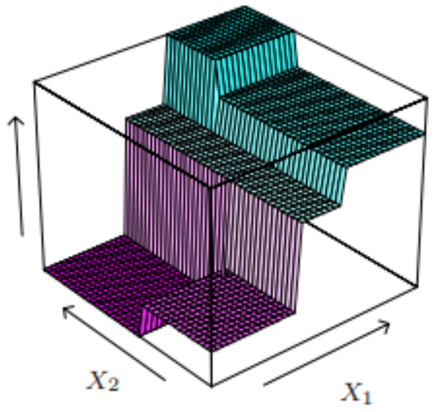
From ESL book Ch9 :

Classification **a**nd **R**egression **T**rees (CART)

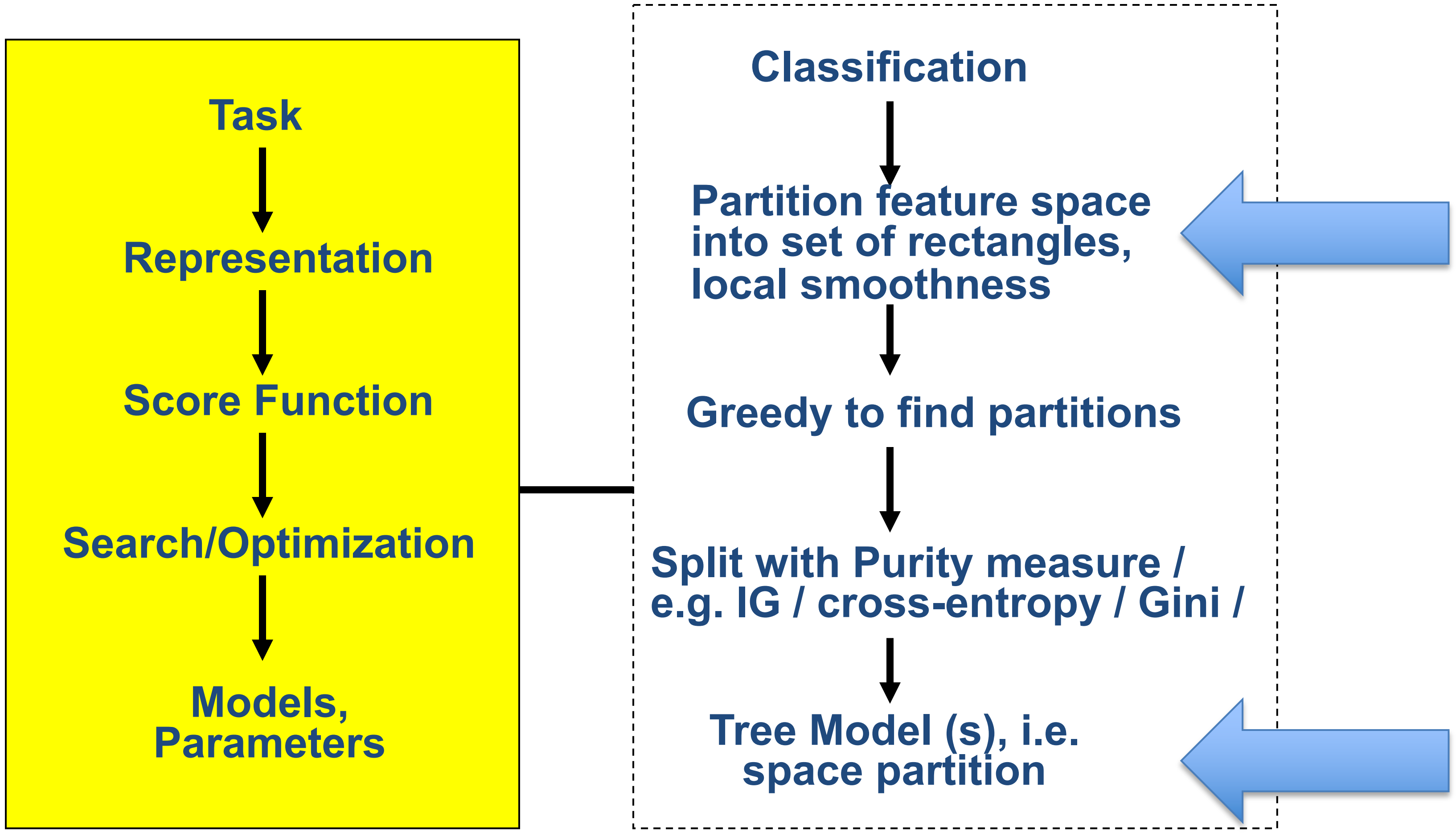
- **Partition feature space into set of rectangles**
- Fit simple model in each partition







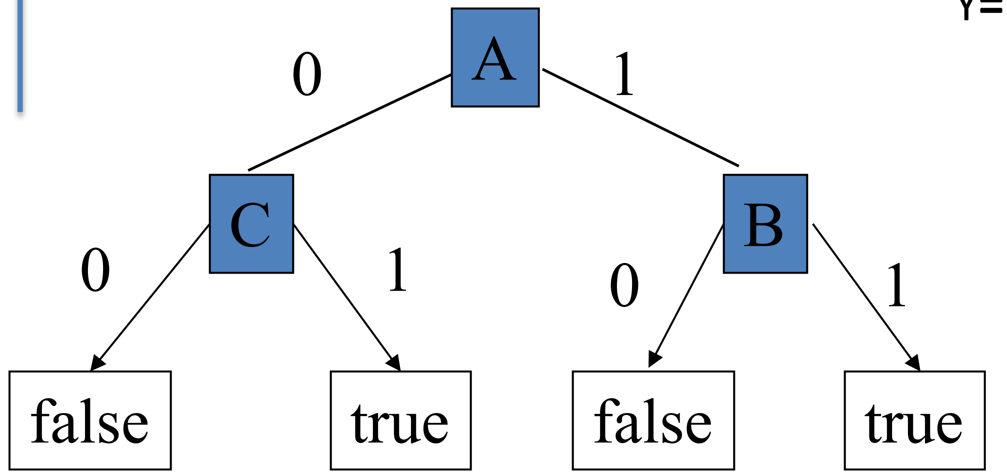
Decision Tree / Random Forest


Task
Representation
Score Function
Search/Optimization
Models, Parameters
Classification
Partition feature space into set of rectangles, local smoothness
Greedy to find partitions
Split with Purity measure / e.g. IG / cross-entropy / Gini /
Tree Model (s), i.e. space partition

Today

- Decision Tree (DT):
  - Tree representation
- Brief information theory
- Learning decision trees
- Bagging
- Random forests: Ensemble of DT
- More about ensemble

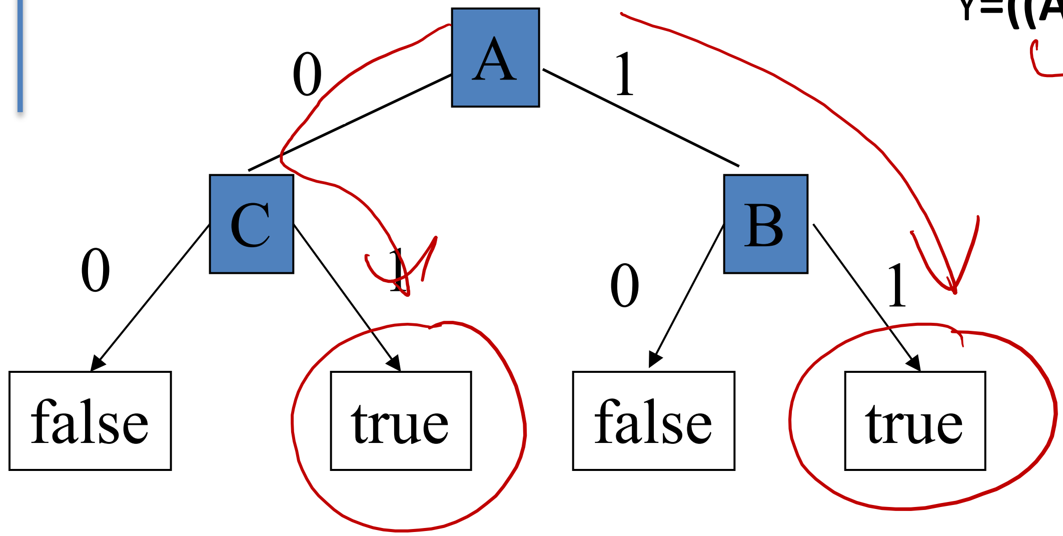
Challenge in Tree Representation



Y=((A and B) or ((not A) and C))

Challenge in Tree Representation

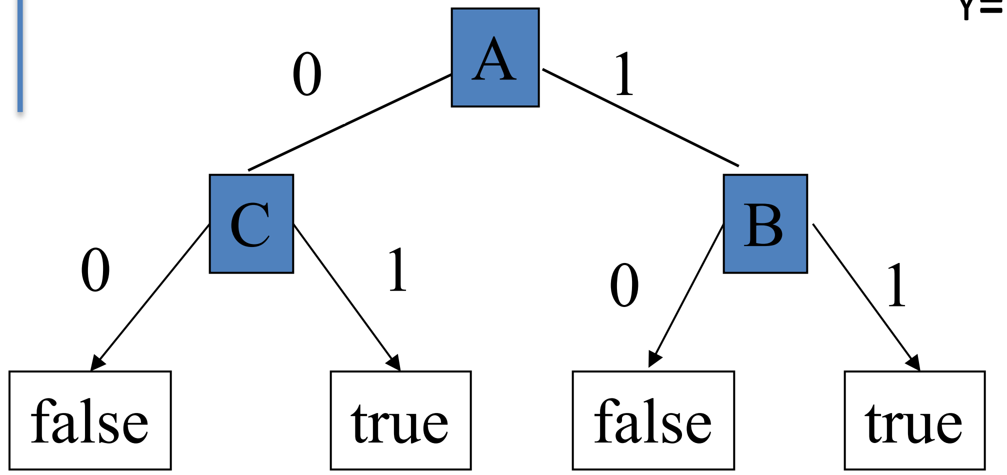
$$Y=(\underbrace{(A \text{ and } B)}\ \text{or}\ \underbrace{((\text{not } A) \text{ and } C)})$$

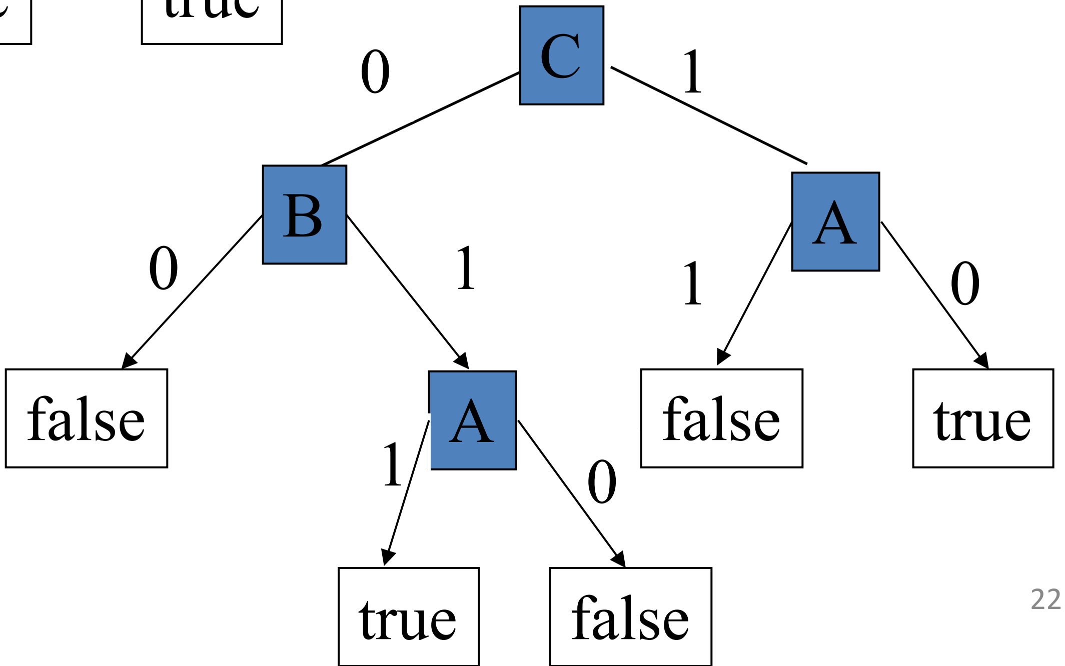


Same concept / different representation

Y=((A and B) or ((not A) and C))

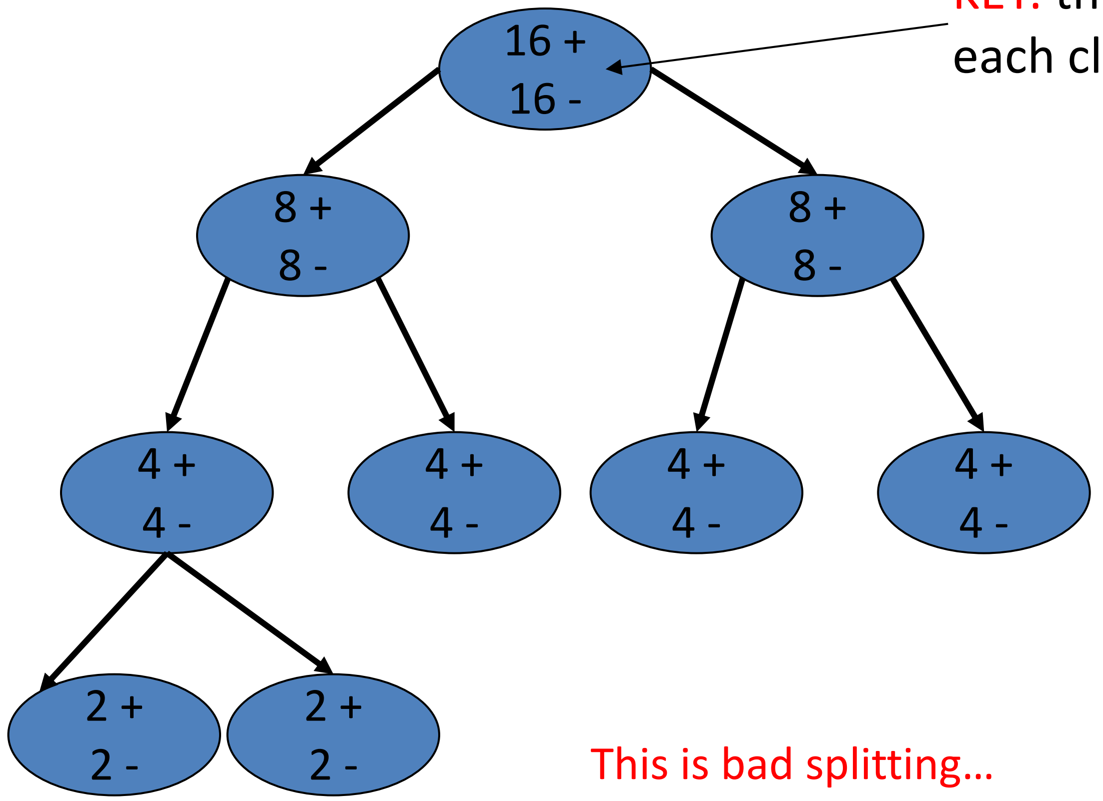
not unique





Which attribute to select for splitting?

KEY: the distribution of each class (not attribute)



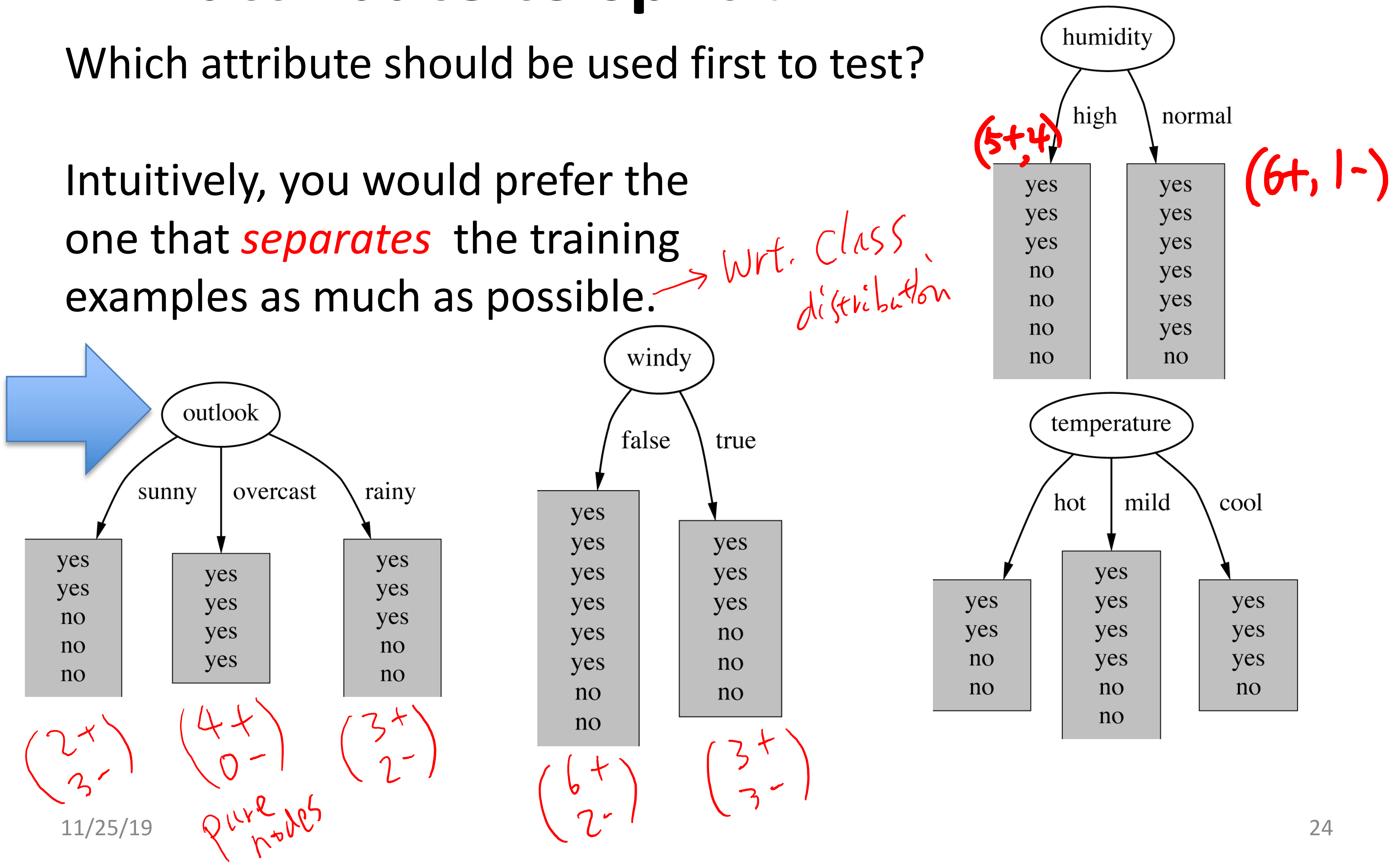
Can not make decisions at unpure nodes

This is bad splitting…

How do we choose which attribute to split ?

Which attribute should be used first to test?

Intuitively, you would prefer the one that *separates* the training examples as much as possible. → Wrt. class distribution

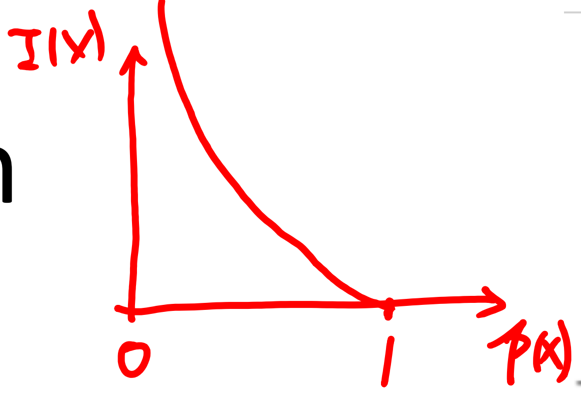


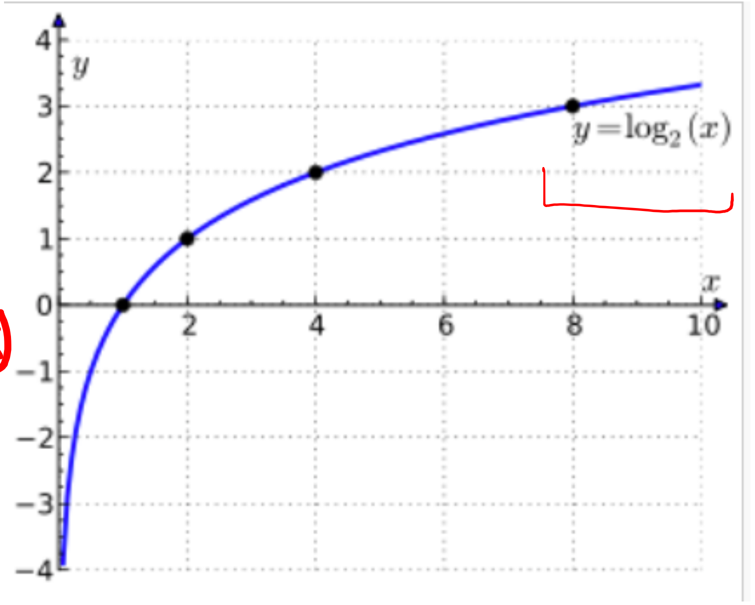
Information gain is one criteria to decide on which attribute for splitting

- Imagine:
  - 1. Someone is about to tell you your own name
  - 2. You are about to observe the outcome of a dice roll
  - 2. You are about to observe the outcome of a coin flip
  - 3. You are about to observe the outcome of a biased coin flip

- Each situation has a different *amount of uncertainty* as to what outcome you will observe.

Information





- Information:

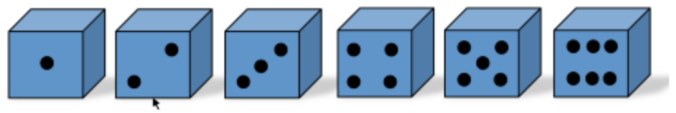
➔ Reduction in uncertainty (amount of surprise in the outcome)

$$I(X) = \log_2 \frac{1}{p(x)} = -\log_2 p(x)$$

If the probability of this event happening is small and it happens, the information is large.

- Observing the outcome of a coin flip is head $\longrightarrow$ $I = -\log_2 1/2 = 1$

- Observe the outcome of a dice is 6 $\longrightarrow$ $I = -\log_2 1/6 = 2.58$



Entropy

- The *expected amount of information* when observing the output of a random variable X

$$H(X) = E(I(X)) = \sum_i p(x_i) I(x_i) = -\sum_i p(x_i)\log_2 p(x_i)$$

If the X can have 8 outcomes and all are equally likely

$$H(X) == -\sum_i 1/8 \log_2 1/8 = 3$$

Entropy

- If there are *k* possible outcomes

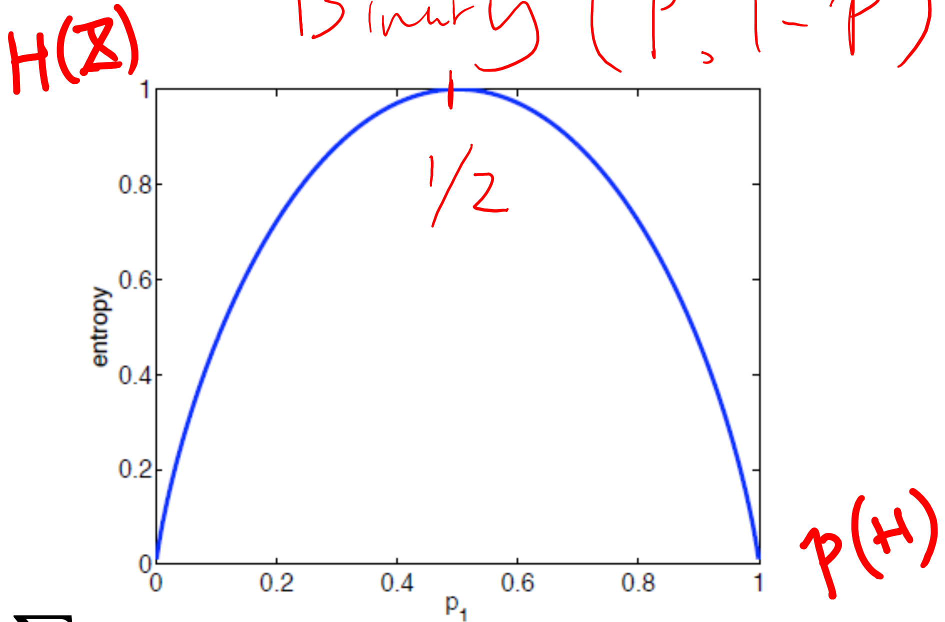
$$H(X) \le \log_2 k$$

Yes, No

Binary $(p, 1-p)$

- Equality holds when all outcomes are equally likely

- The more the probability distribution that deviates from uniformity, the lower the entropy



$$H(X) = E(I(X)) = \sum_i p(x_i) I(x_i) = -\sum_i p(x_i) \log_2 p(x_i)$$

e.g. for a random binary variable

Entropy

- If there are k possible outcomes

$$H(X) \leq \log_2 \underline{\underline{k}}$$

[# unique values of discrete X]

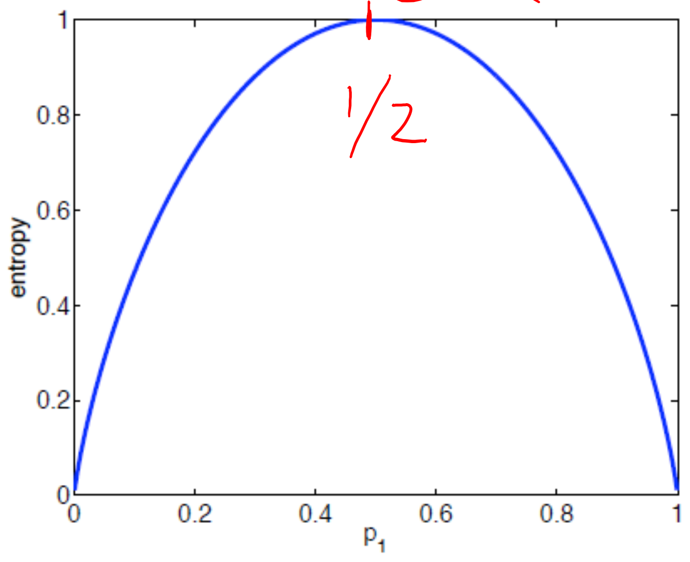
Yes, No

X is Binary (p, 1-p)

- Equality holds when all outcomes are equally likely

- The more the probability distribution that deviates from uniformity, the lower the entropy

↓ the purer



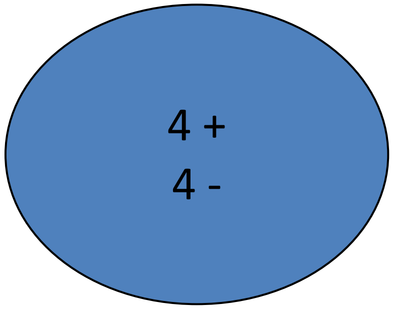
e.g. for a random binary variable

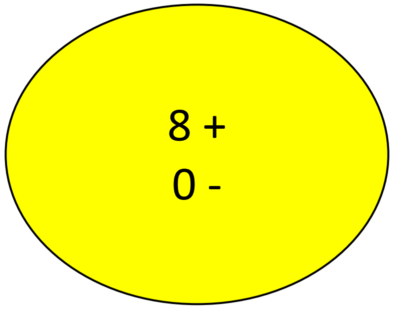
Entropy Lower ➔ better purity

- Entropy measures the purity

$P_{Yes} = p = 4/8$

$P_{No} = 1 - p = 4/8$





$p = 8/8 = 1 = P_{Yes}$

$1 - p = 0 = P_{No}$

The distribution is less uniform
Entropy is lower
The node is purer

Information gain

- IG(X,Y)=H(Y)-H(Y|X)

Reduction in uncertainty of Y by knowing a feature variable X

Information gain:

= (information before split) – (information after split)

= entropy(parent) – [average entropy(children)]

Fixed – the lower, the better (children nodes are purer) = For IG, the higher, the better

Information gain

- IG(X,Y)=H(Y)-H(Y|X)

$\Rightarrow$ pick X_i BY $\underset{X_i}{\operatorname{argmax}} \left\{ \begin{array}{l} IG(X_1, Y) \\ IG(X_2, Y) \\ IG(X_m, Y) \end{array} \right.$

Reduction in uncertainty of Y by knowing a feature variable X

Information gain:

= (information before split) – (information after split)

= entropy(parent) – [average entropy(children)]

$H(Y)$ Fixed – $H(Y|X_i)$ the lower, the better (children nodes are purer) = For IG, the higher, the better

Conditional entropy

$$H(Y) = -\sum_i p(y_i)\log_2 p(y_i)$$

$$H(Y \mid \underbrace{X = x_j}) = -\sum_i p(y_i \mid x_j)\log_2 p(y_i \mid x_j)$$

$$H(Y \mid X) = \underbrace{\sum_j p(x_j)}H(Y \mid X = x_j)$$

$$= -\sum_j p(x_j)\sum_i p(y_i \mid x_j)\log_2 p(y_i \mid x_j)$$

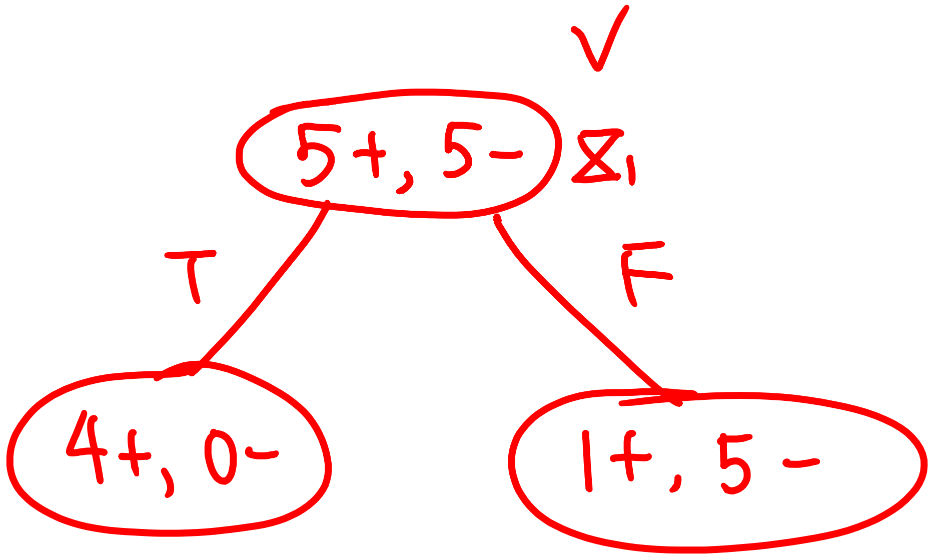
Example

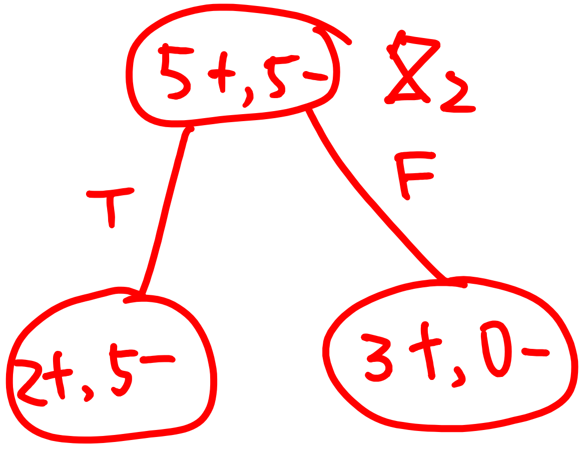
Attributes Labels

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

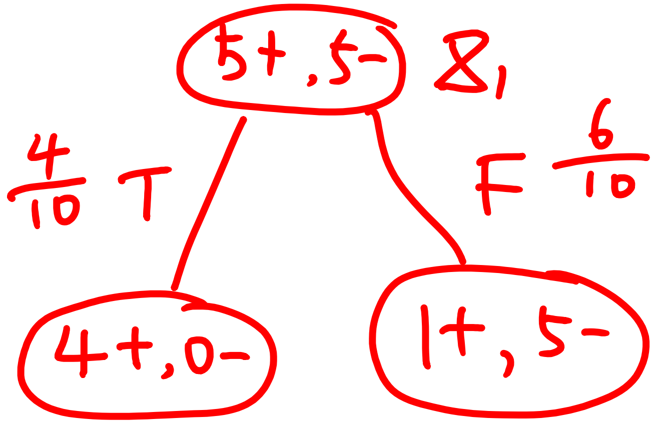
Which one do we choose

X1 or X2?





X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1



$$H(Y \mid X_1 = T) = -\Big\{ p(Y=+ \mid X_1=T) \log p(Y=+ \mid X_1=T)$$

$$\boxed{4+, 0-} \Rightarrow \quad + p(Y=- \mid X_1=T) \log p(Y=- \mid X_1=T) \Big\}$$

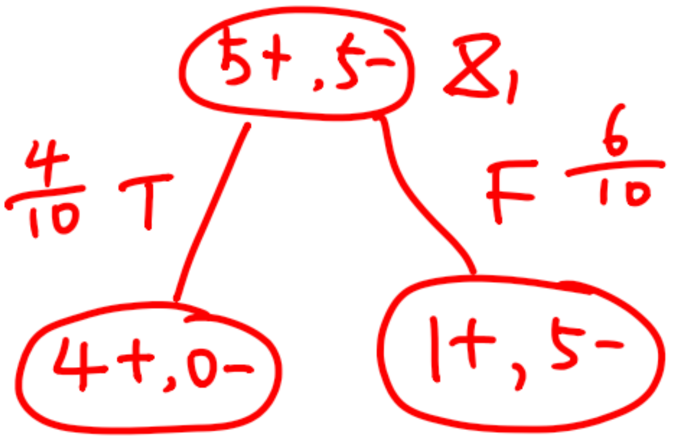
$$= 0$$

$$H(Y|X_1=T) = \boxed{\begin{matrix}4+\\0-\end{matrix}} \Rightarrow -\left(P(+)\log P(+) + P(-)\log(P(-))\right)$$

$$= -(1\lg_2 1 + 0\log 0) = 0$$

$$H(Y|X_1=F) = \boxed{\begin{matrix}1+\\5-\end{matrix}} \Rightarrow -\left(P(+)\log P(+) + P(-)\log P(-)\right)$$

$$= -\left(\frac{1}{6}\log\frac{1}{6} + \frac{5}{6}\log\frac{5}{6}\right)$$



$$H(Y|X_1) = \frac{4}{10}H(Y|X_1=T) + \frac{6}{10}H(Y|X_1=F)$$

Example

Attributes Labels

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Which one do we choose

X1 or X2?

IG(X1,Y) = H(Y) – H(Y|X1)

H(Y) = - (5/10) log(5/10) -5/10log(5/10) = 1

H(Y|X1) = P(X1=T)H(Y|X1=T) + P(X1=F) H(Y|X1=F)

= 4/10 (1log 1 + 0 log 0) +6/10 (5/6log 5/6 +1/6log1/6)

= 0.39

Information gain (X1,Y)= 1-0.39=0.61

Which one do we choose?

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Information gain (X1,Y)= 0.61 $= H(Y) - \underbrace{H(Y|X_1)}$ $\Rightarrow$ Smaller, purer

Information gain (X2,Y)= 0.12 $= H(Y) - \underbrace{H(Y|X_2)}$

$\Downarrow$ IG larger

Better

Pick the variable which provides the most information gain about Y ➔ Pick X1

➔ **Then recursively choose next Xi on branches**

Which one do we choose?

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Split by X1

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

One branch

The other branch

Information gain (X1,Y)= 0.61 $= H(Y) - \underbrace{H(Y|X_1)} \Rightarrow$ smaller, purer

Information gain (X2,Y)= 0.12 $= H(Y) - \underbrace{H(Y|X_2)}$

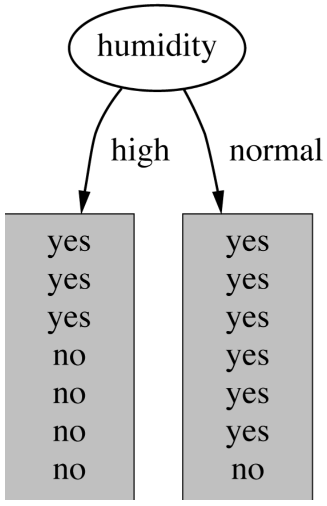
$\Downarrow$ IG larger

Better

Pick the variable which provides the most information gain about Y ➔ Pick X1

➔ **Then recursively choose next Xi on branches**

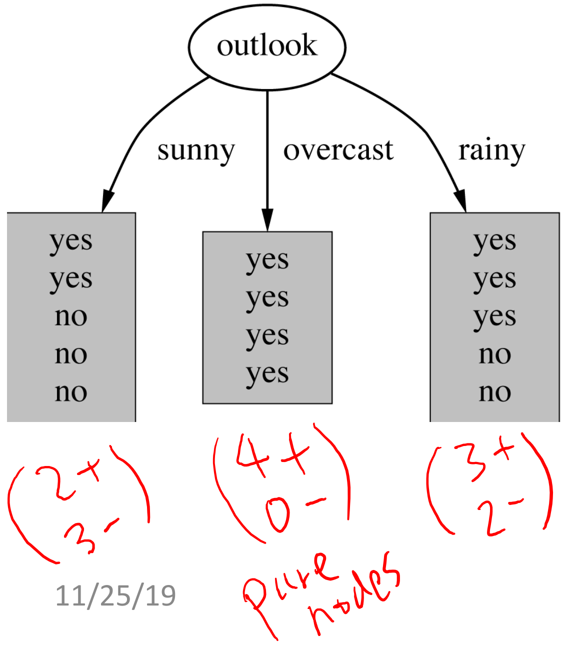
Intuitively, you would prefer the one that *separates* the training examples as much as possible.

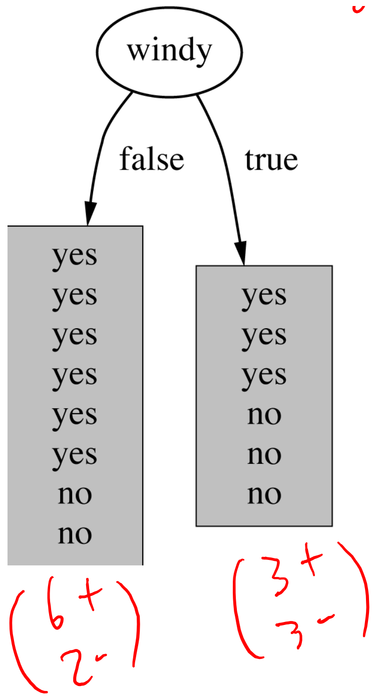


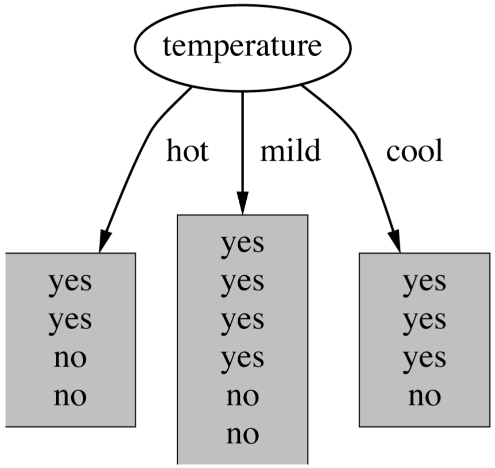
➔ **Then recursively choose next Xi on each of the branches,**

➔ **To compare, e.g.,**

IG(humidity, y|Outlook ==sunny)
IG(windy, y|Outlook ==sunny)
IG(windy, y|Outlook ==rainy)



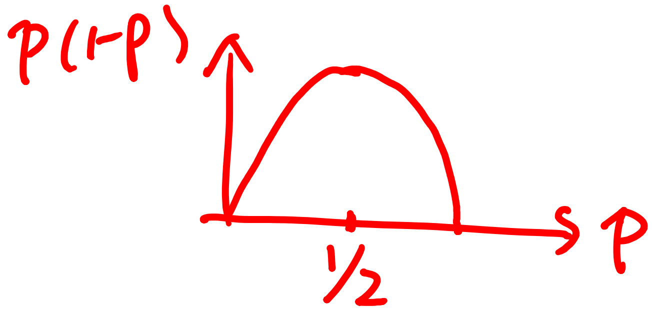




Decision Trees

$H(X) \le \log_2 K$

- **Caveats:** The number of possible values influences the information gain.
  - The more possible values, the higher the gain (the more likely it is to form small, but pure partitions)



- **Other Purity (diversity) measures**
  - Information Gain
  - Gini (population impurity) $\sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk})$
    - where p_{mk} is proportion of class k at node m
  - Chi-square Test

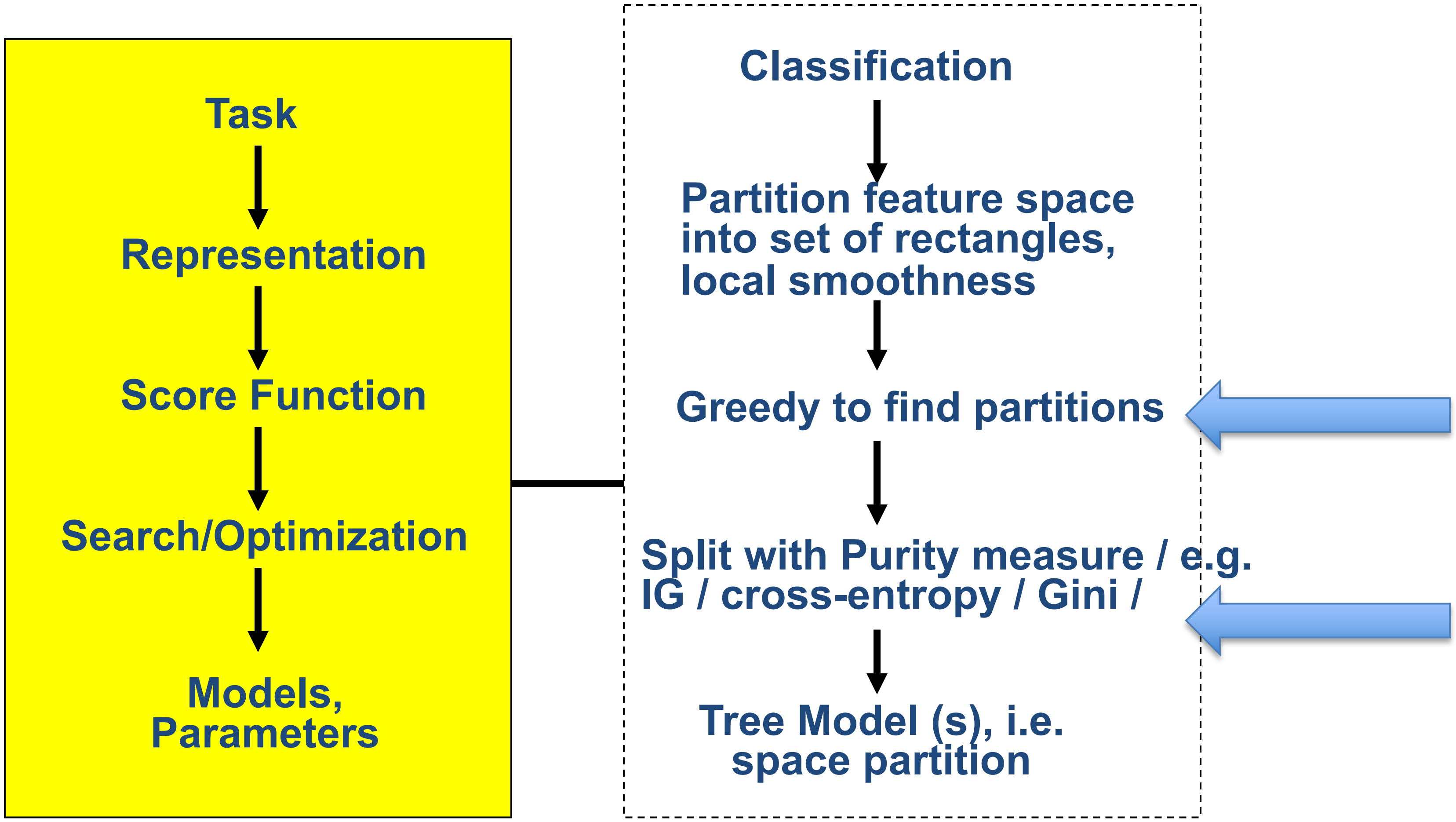
Overfitting

- You can perfectly fit DT to any training data
- Instability of Trees
  - High variance (small changes in training set will result in changes of tree model)
  - Hierarchical structure ➔ Error in top split propagates down
- Two approaches:
  - 1. Stop growing the tree when further splitting the data does not yield an improvement
  - 2. Grow a full tree, then prune the tree, by eliminating nodes.

Summary: Decision trees

- Non-linear classifier / regression
- Easy to use
- Easy to interpret
- Susceptible to overfitting but can be avoided.

Decision Tree / Random Forest


Task
Representation
Score Function
Search/Optimization
Models, Parameters
Classification
Partition feature space into set of rectangles, local smoothness
Greedy to find partitions
Split with Purity measure / e.g. IG / cross-entropy / Gini /
Tree Model (s), i.e. space partition

Today

- Decision Tree (DT):
  - Tree representation
- Brief information theory
- Learning decision trees
- Bagging
- Random forests: Ensemble of DT
- More about ensemble

Bagging

- Bagging or *bootstrap aggregation*
  - a technique for reducing the variance of an estimated prediction function.

- For instance, for classification, a *committee* of trees
  - Each tree casts a vote for the predicted class.

Bootstrap

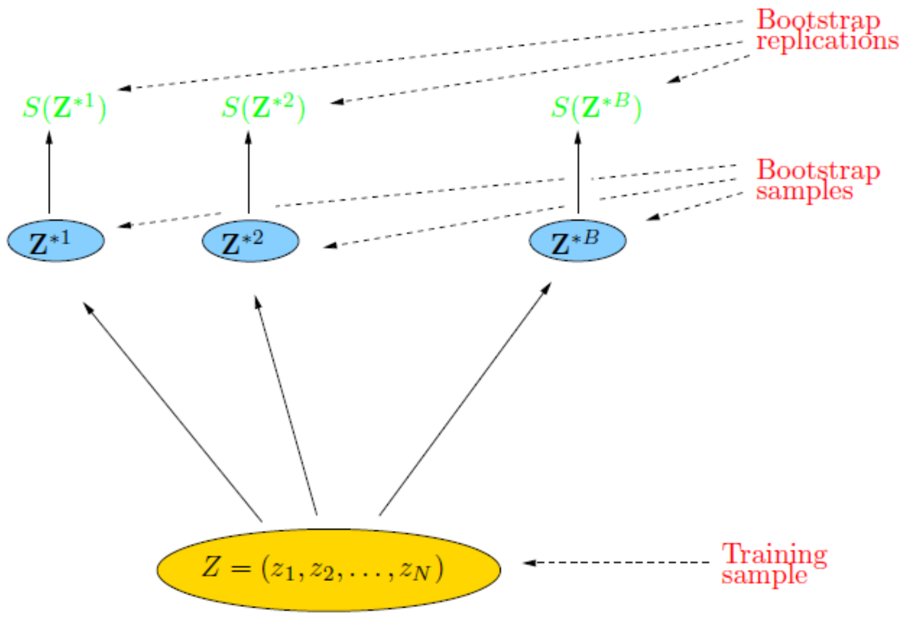
The basic idea:

randomly draw datasets *with replacement (i.e. allows duplicates)* from the training data, each samples *the same size as the original training set*

Bootstrap

The basic idea:

randomly draw datasets *with replacement (i.e. allows duplicates)* from the training data, each samples *the same size as the original training set*



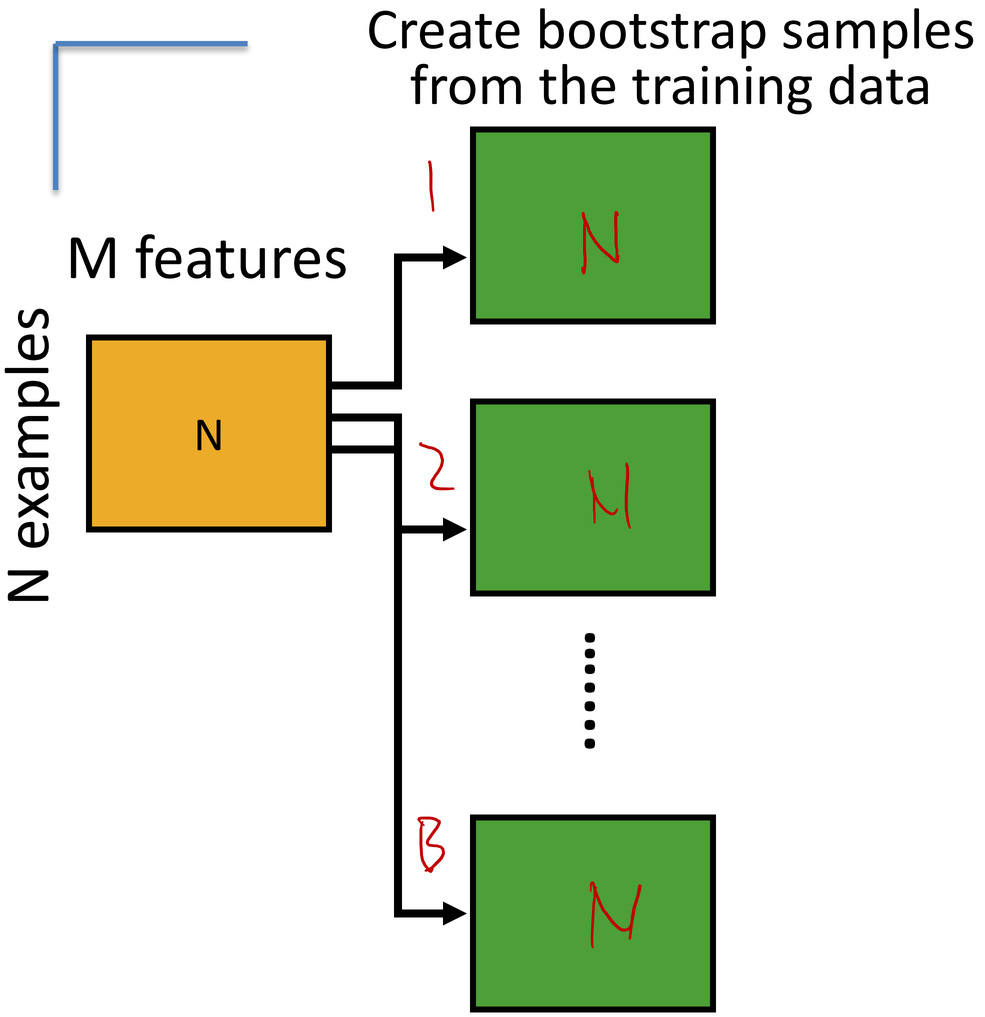
$$\widehat{\mathrm{Var}}[S(\mathbf{Z})] = \frac{1}{B-1}\sum_{b=1}^{B}(S(\mathbf{Z}^{*b}) - \bar{S}^*)^2,$$

With vs Without Replacement

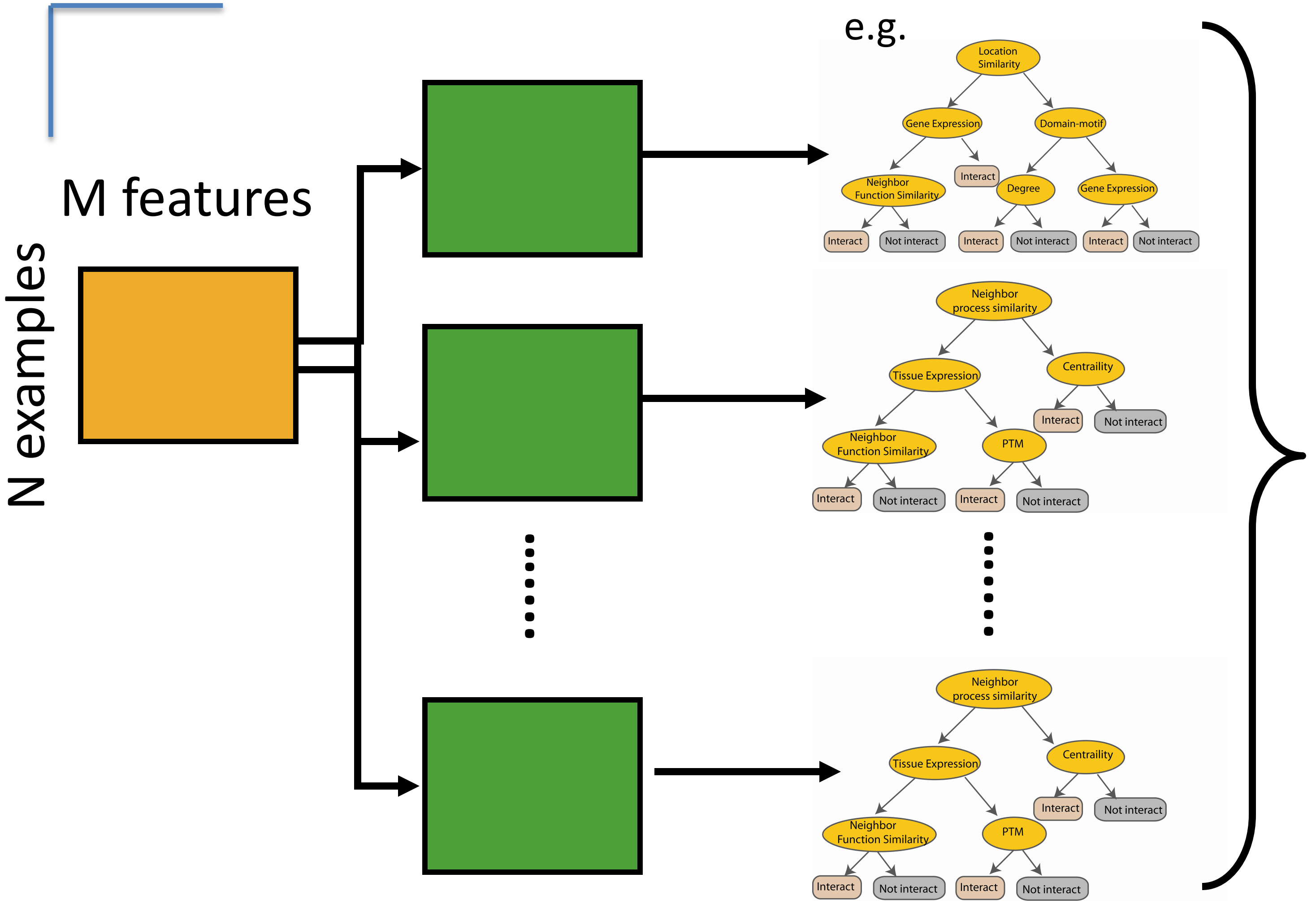
- Bootstrap with replacement can keep the sampling size the same as the original size for every repeated sampling. The sampled data groups are independent on each other.

- Bootstrap without replacement cannot keep the sampling size the same as the original size for every repeated sampling. The sampled data groups are dependent on each other.

Bagging


Create bootstrap samples from the training data
M features
N examples
N
1
N
2
N
B
N

Bagging of DT **Classifiers**



Take the majority vote

i.e. Refit the model to each bootstrap dataset, and then examine the behavior over the B replications.

Peculiarities of Bagging

- Model Instability is good when bagging
  - The more variable (unstable) the basic model is, the more improvement can potentially be obtained
  - Low-Variability methods (e.g. SVM, LDA) improve less than High-Variability methods (e.g. decision trees)

Can understand the bagging effect in terms of a consensus of independent ***weak leaners*** and ***wisdom of crowds***

Bagging : an example with simulated data

N = 30 training samples,

two classes and p = 5 features,

Each feature N(0, 1) distribution and pairwise correlation .95

Response Y generated according to:

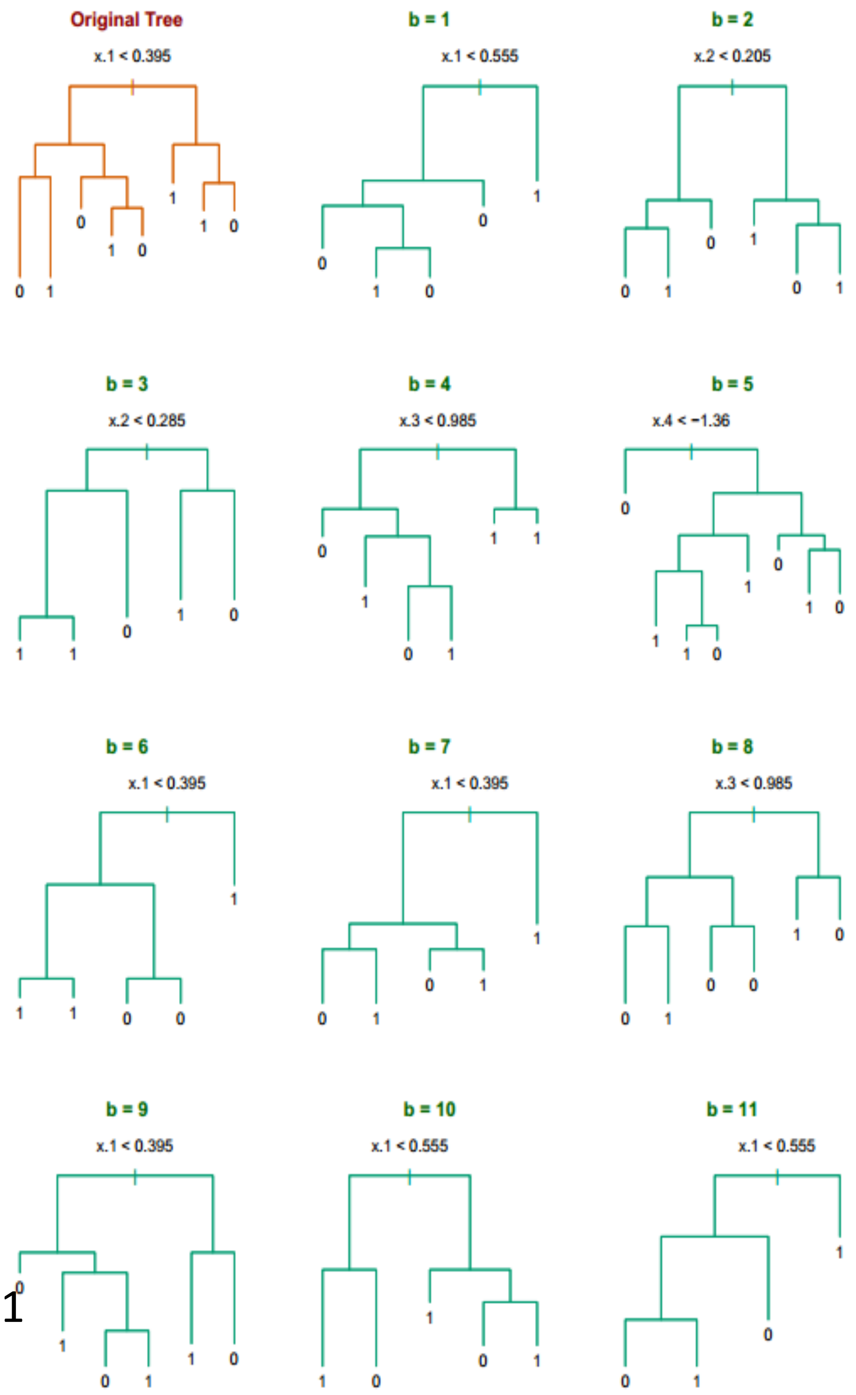
$$\Pr(Y = 1|x_1 \leq 0.5) = 0.2 \qquad \Pr(Y = 1|x_1 > 0.5) = 0.8$$

Test sample size of 2000

Fit classification trees to training set and bootstrap samples

B = 200

ESL book / Example 8.7.1



Notice the bootstrap trees are different than the original tree

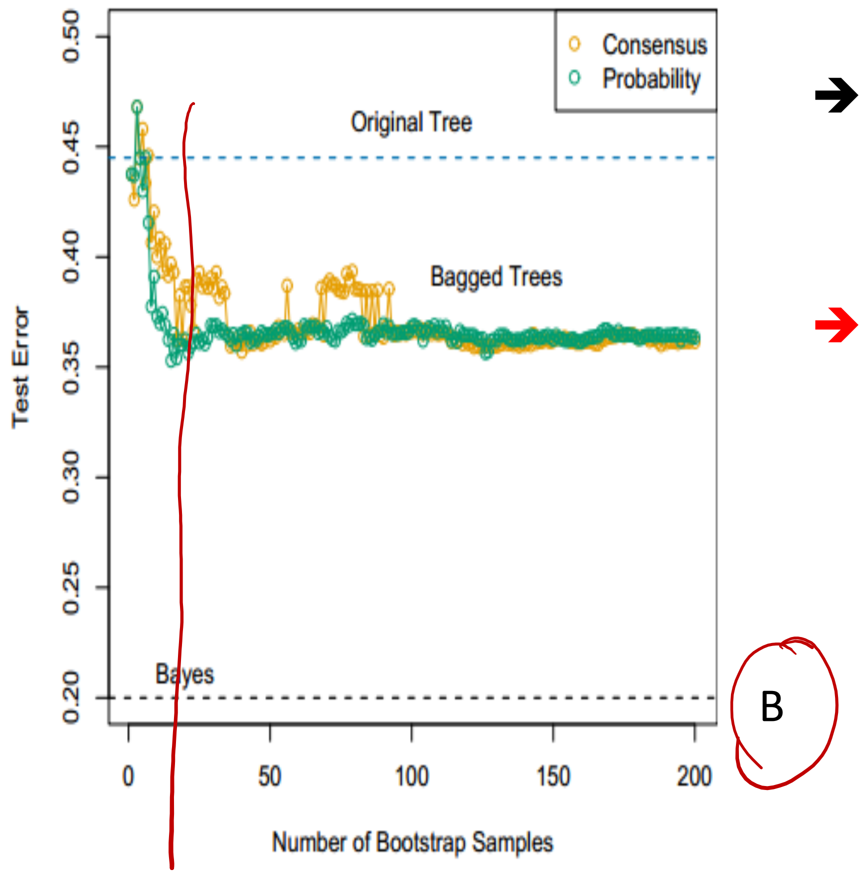
Five features highly correlated with each other

➔ No clear difference with picking up which feature to split

➔ Small changes in the training set will result in different tree

➔ But these trees are actually quite similar for classification

ESL book / Example 8.7.1



- ➔ For B>30, more trees do not improve the bagging results
- ➔ Since the trees correlate highly to each other and give similar classifications

Consensus: Majority vote

Probability: Average distribution at terminal nodes

ESL book / Example 8.7.1

Bagging

- Slightly increases model space
  - Cannot help where greater enlargement of space is needed

- Bagged trees are correlated
  - Use random forest to reduce correlation between trees

Bagged Decision Tree

Task

↓

Representation

↓

Score Function

↓

Search/Optimization

↓

Models, Parameters

Classification / Regression

↓

multiple (almost) full decision trees / bootstrap samples

↓

Split with Purity measure / e.g. IG / cross-entropy / Gini /

↓

Greedy like Decision Tree (e.g. GINI)

↓

Multiple Tree Model (s), i.e. space partition

References

- ❑ Prof. Tan, Steinbach, Kumar's "Introduction to Data Mining" slide
- ❑ ESLbook : Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: Springer, 2009.
- ❑ Dr. Oznur Tastan's slides about RF and DT
- ❑ Dr. Camilo Fosco's slides

Some Extra Slides

Tree-building algorithms

ID3: Iterative Dichotomiser 3. Developed in the 80s by Ross Quinlan.

- Uses the top-down induction approach described previously.
- Works with the Information Gain (IG) metric.
- At each step, algorithm chooses feature to split on and calculates IG for each possible split along that feature.
- Greedy algorithm.

From Dr. Camilo Fosco

Tree-building algorithms

C4.5: Successor of ID3, also developed by Quinlan ('93). Main improvements over I3D:

- Works with both continuous and discrete features, while ID3 only works with discrete values.
- Handles missing values by using **fractional cases** (penalizes splits that have multiple missing values during training, fractionally assigns the datapoint to all possible outcomes).
- Reduces overfitting by pruning, a bottom-up tree reduction technique.
- Accepts weighting of input data.
- Works with multiclass response variables.

From Dr. Camilo Fosco

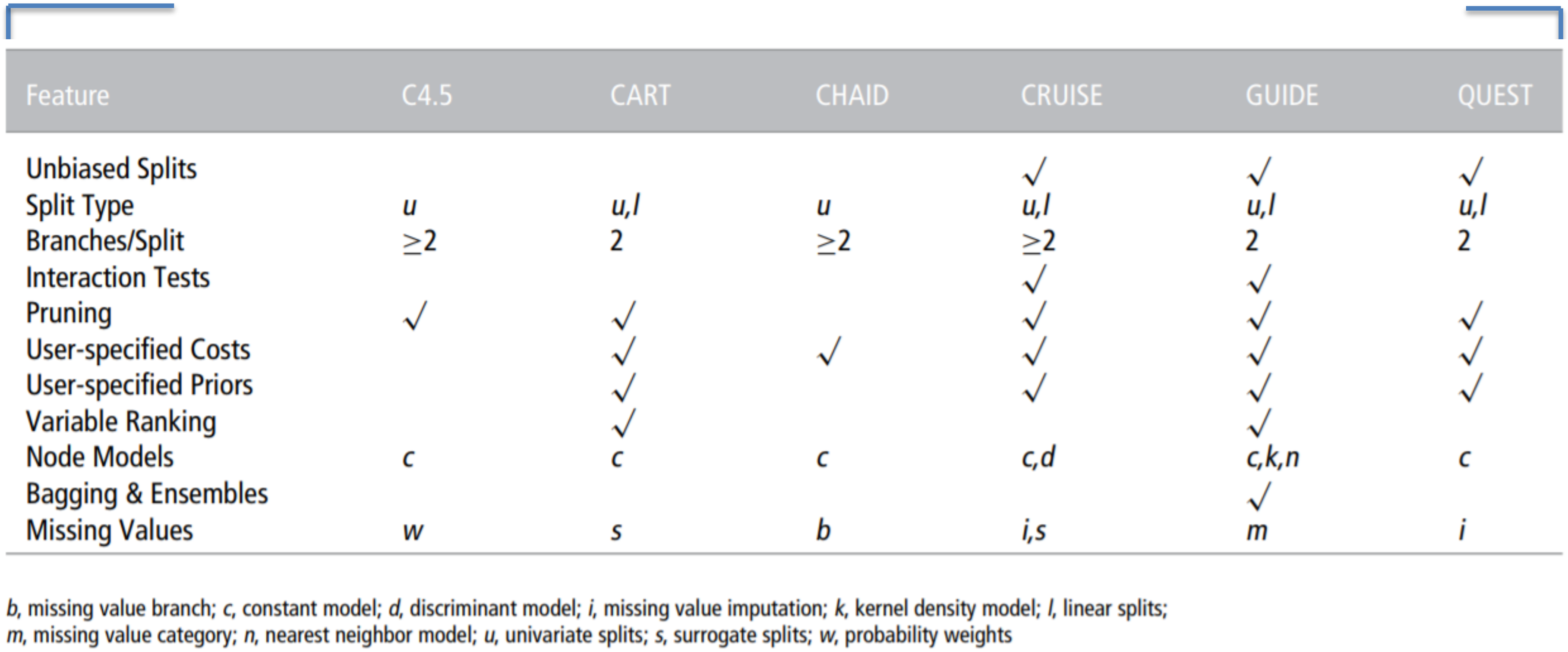
Tree-building algorithms

CART: Most popular tree-builder. Introduced by Breiman et al. in 1984. Usually used with Gini purity metric.

- Main characteristic: **builds binary trees.**
- Can work with discrete, continuous and categorical values.
- Handles missing values by using **surrogate splits**.
- Uses **cost-complexity pruning**.
- Sklearn uses CART for its trees.

From Dr. Camilo Fosco

Many more algorithms…



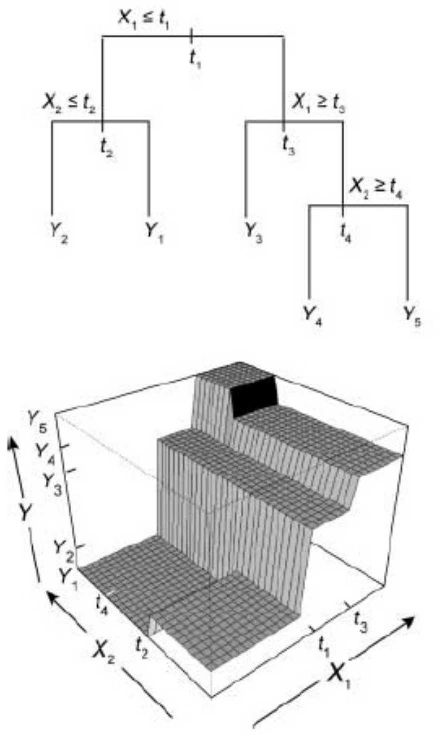
Feature	C4.5	CART	CHAID	CRUISE	GUIDE	QUEST
Unbiased Splits				√	√	√
Split Type	*u*	*u,l*	*u*	*u,l*	*u,l*	*u,l*
Branches/Split	≥2	2	≥2	≥2	2	2
Interaction Tests				√	√	
Pruning	√	√		√	√	√
User-specified Costs		√	√	√	√	√
User-specified Priors		√		√	√	√
Variable Ranking		√			√	
Node Models	*c*	*c*	*c*	*c,d*	*c,k,n*	*c*
Bagging & Ensembles					√	
Missing Values	*w*	*s*	*b*	*i,s*	*m*	*i*

b, missing value branch; *c*, constant model; *d*, discriminant model; *i*, missing value imputation; *k*, kernel density model; *l*, linear splits; *m*, missing value category; *n*, nearest neighbor model; *u*, univariate splits; *s*, surrogate splits; *w*, probability weights

From Dr. Camilo Fosco

Regression trees

- Can be considered a *piecewise constant regression model*.
- Prediction is made by averaging values at given leaf node.
- Two advantages: interpretability and modeling of interactions.
- The model's decisions are easy to track, analyze and to convey to other people.
- Can model complex interactions in a tractable way, as it subdivides the support and calculates averages of responses in that support.



From Dr. Camilo Fosco

Regression trees - Cons

- Two major disadvantages: difficulty to capture simple relationships and instability.
- Trees tend to have high variance. Small change in the data can produce a very different series of splits.
- Any change at an upper level of the tree is propagated down the tree and affects all other splits.
- Large number of splits necessary to accurately capture simple models such as linear and additive relationships.
- Lack of smoothness.

From Dr. Camilo Fosco

Surrogate splits

- When an observation is missing a value for predictor X, it cannot get past a node that splits based on this predictor.
- We need surrogate splits: Mimic of original split in a node, but using another predictor. It is used in replacement of the original split in case a datapoint has missing data.
- To build them, we search for a feature-threshold pair that most closely matches the original split.
- “Association”: measure used to select surrogate splits. Depends on the probabilities of sending cases to a particular node + how the new split is separating observations of each class.

From Dr. Camilo Fosco

Pruning

- Reduces the size of decision trees by removing branches that have little predictive power. This helps reduce overfitting. Two main types:
- **Reduced Error Prunning:** Starting at leaves, replace each node with its most common class. If accuracy reduction is inferior than a given threshold, change is kept.
- **Cost Complexity Pruning:** remove subtree that minimizes the difference of the error of pruning that tree and leaving it as is, normalized by the difference in leaves:

$$\frac{err(T,S) - err(T_0,S)}{|leaves(T)| - |leaves(T_0)|}$$

From Dr. Camilo Fosco