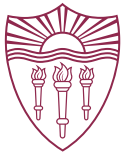
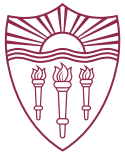


DECISION TREES

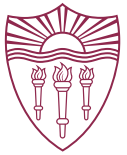
Kristina Lerman
USC Information Sciences Institute
DSCI 552 – Spring 2021
March 22, 2021

Topics this week



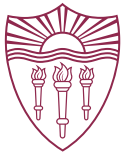
- Decision trees
  - Classification trees
  - Regression trees
- Ensemble methods
  - Boosting and bagging
  - Random forest

Why decision trees?



- Popular method, especially for classification
- Learn quickly, classify new data quickly
- Non-parameteric
- Non-linear
- Highly accurate
- Interpretable
  - Set of IF-THEN rules
  - Can be validated by human experts

Decision Trees

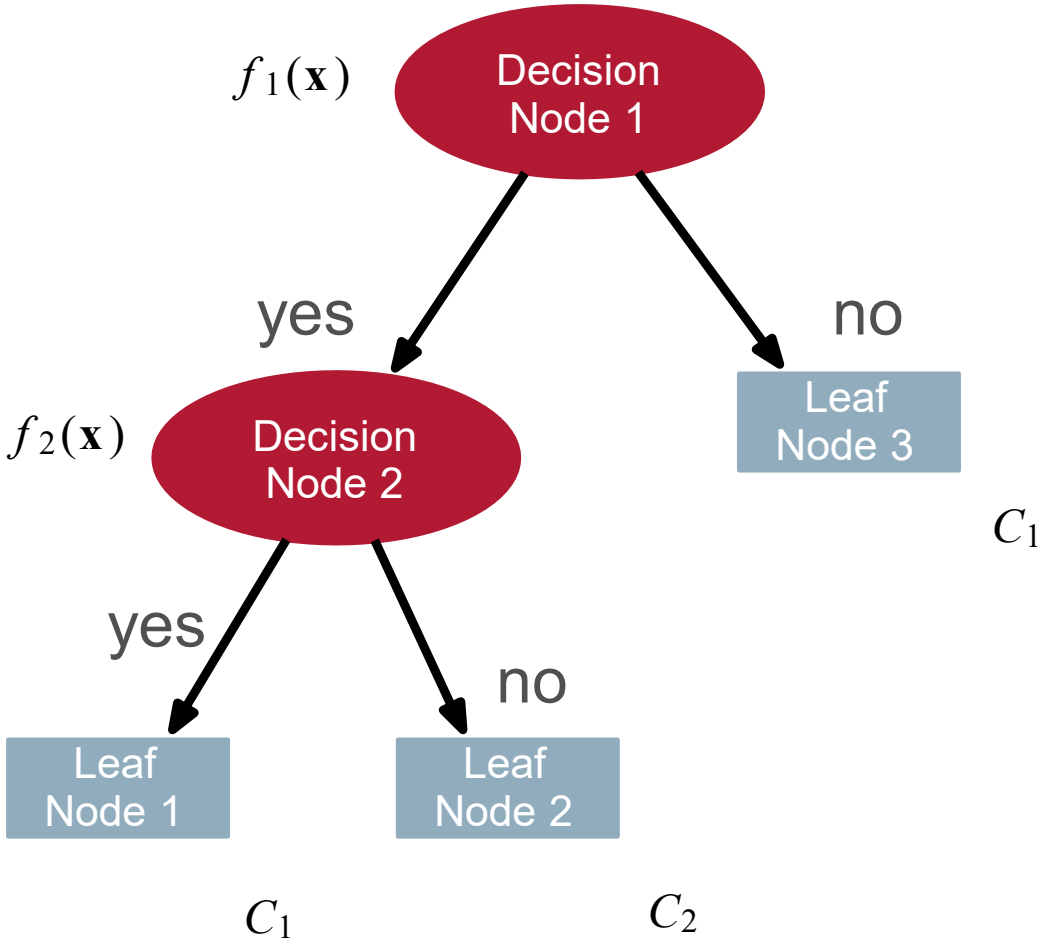


- A *decision tree* is a hierarchical data structure following the "*divide and conquer*" strategy
- Nonparametric ("no predefined parameters") method that is used for:
  - Classification
  - Regression
- Tree based algorithms involve *stratifying* or *segmenting* the *predictor space* into several simple *regions*.
- A set of splitting rules also called as *decisions* govern how this stratification is made.

- These splits of the feature space can be represented in a tree structure. Hence, the name – *decision trees.*

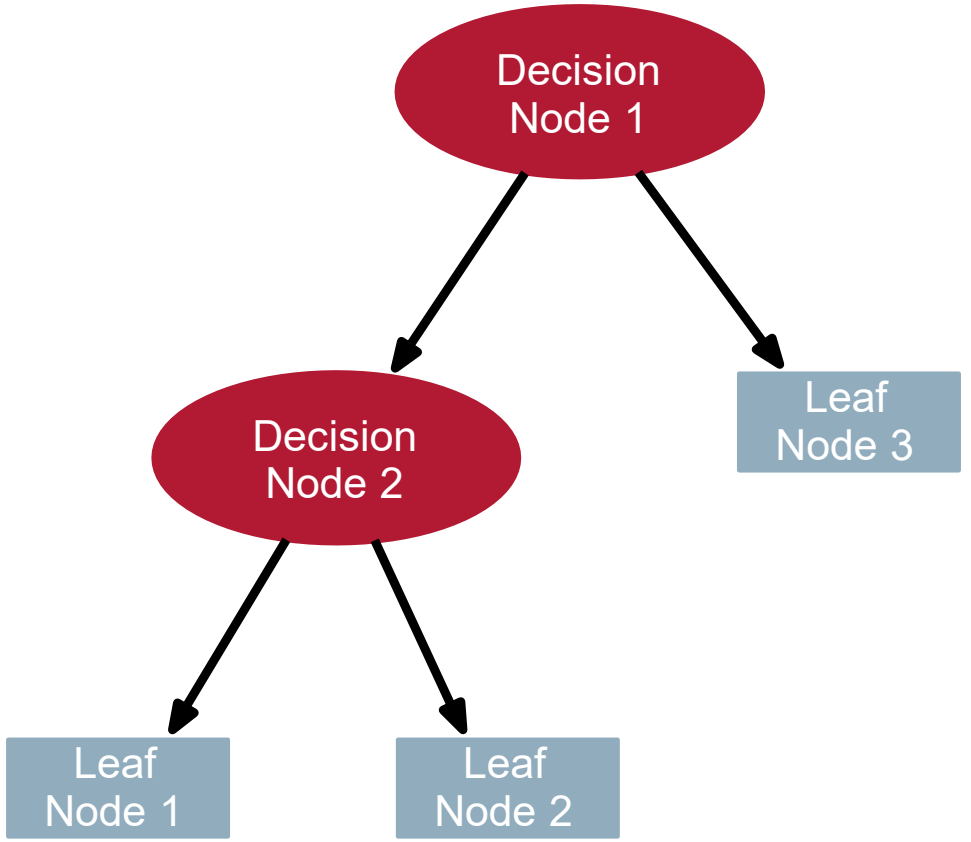
Decision Tree Structure

- A decision tree consists of:
  - Decision nodes
  - Leaf nodes ← data resides here
- Each decision node *m* implements a test function $f_m(\boldsymbol{x})$
- Depending on the test function outcome a branch in the tree is taken.
- This is applied recursively until a *leaf node* is reached.
- The value in the leaf node defines the output.

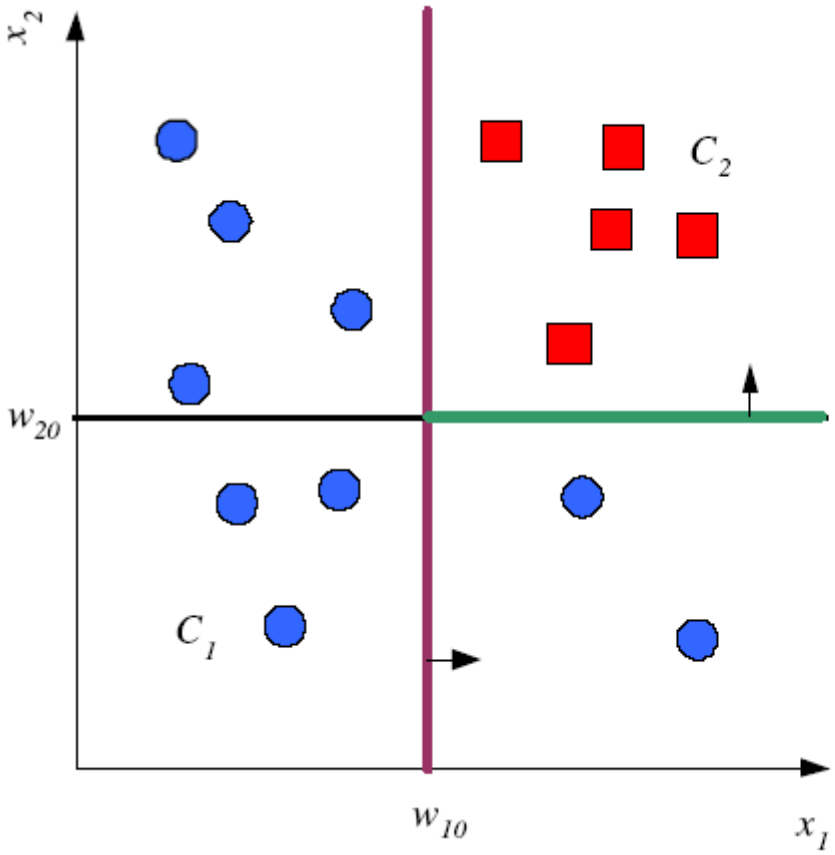


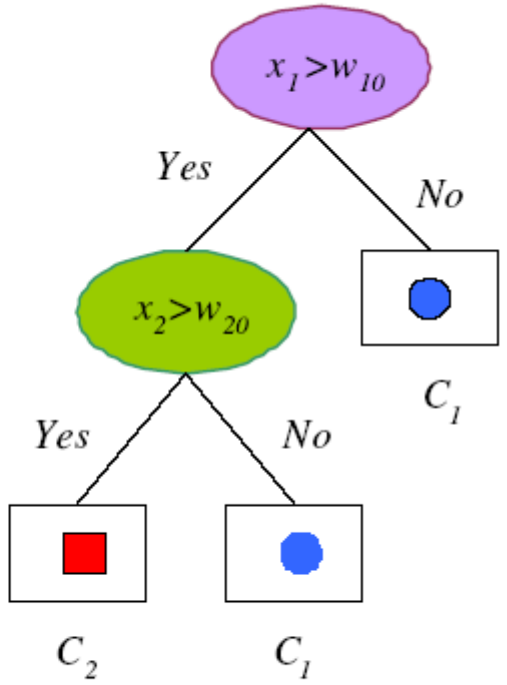
Decision Tree Characteristics

- It is a nonparametric method, as we do not know a priori how the tree will look like in the end. We allow it to grow branches and leaves depending on the complexity of the data.
- In addition, no assumptions about the class densities of data are assumed.



Decision Tree Example

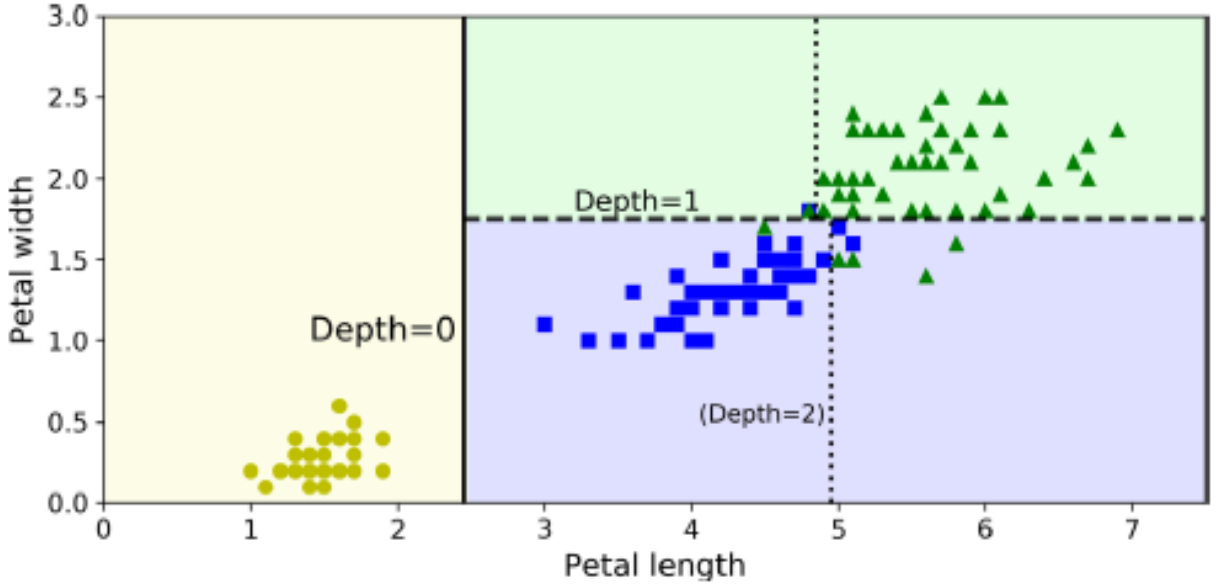


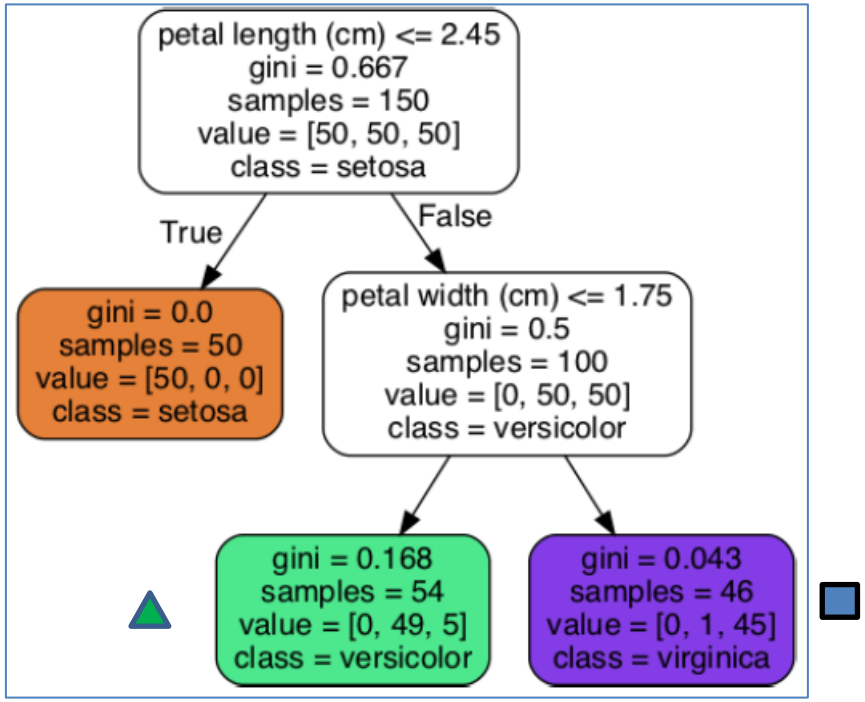


http://www.cmpe.boun.edu.tr/~ethem/i2ml2e/index.html

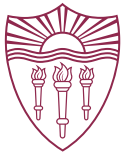
UCI Iris Data

Iris Versicolor **Iris Setosa** **Iris Virginica**



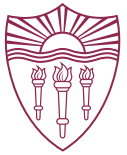


Decision Tree Facts



- Each $f_m(\mathbf{x})$ defines a discriminant in the d-dimensional input space that divides the space into smaller subregions. With ever smaller regions from the root down to the leaves.
- Each decision $f_m(\mathbf{x})$ is a simple function. Each tree is a complex decision broken down into many simple functions (divide and conquer)
- Each leaf node defines a local region with a label:
  - In classification the label is a class label
  - In regression the label is a value
- If there are b regions in the data then in the best case the correct region can be found in log2(b) decisions.
- Another bonus is the interpretability of the tree. Each decision constitutes a simple if-then rule and the complex decision of the tree can be written as a simple set of rules.

Tree Induction



- Tree induction is the process of learning a tree from a given input set.
- There are many possible trees that split the training data perfectly with no error.
- With Occam's Razor in mind we try to find the simplest and smallest tree (i.e. with the least number of nodes).
- This process is NP-complete and we are forced to use local search processes that provide us with reasonable trees in reasonable time.
- Tree learning algorithms are greedy and at each step we look for the ***best split*** of the training data into two or n many ways. Recursively we continue to split the input until we do not need to split anymore (i.e. leaf node).
- What is the ***best split?***

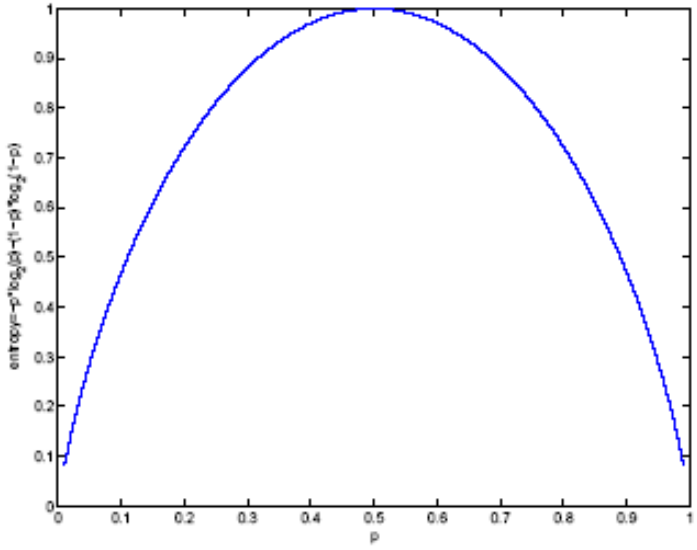
Classification Trees (ID3,CART,C4.5)

- For node m, N_m instances reach m, N^i_m belong to C_i

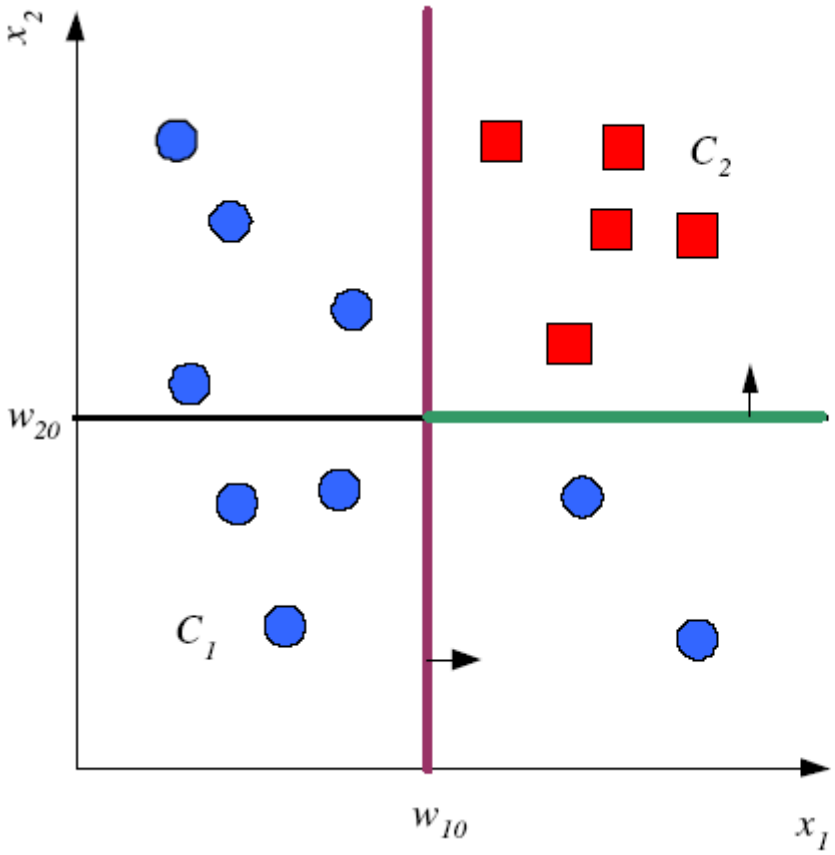
$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p^i_m = \frac{N^i_m}{N_m}$$

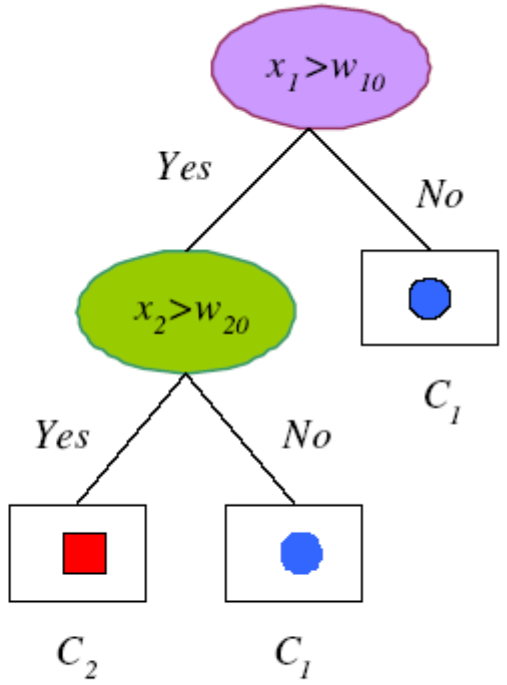
- Node m is pure if p^i_m is 0 or 1
- Measure of impurity is entropy

$$\mathcal{I}_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m$$



Which node is pure?





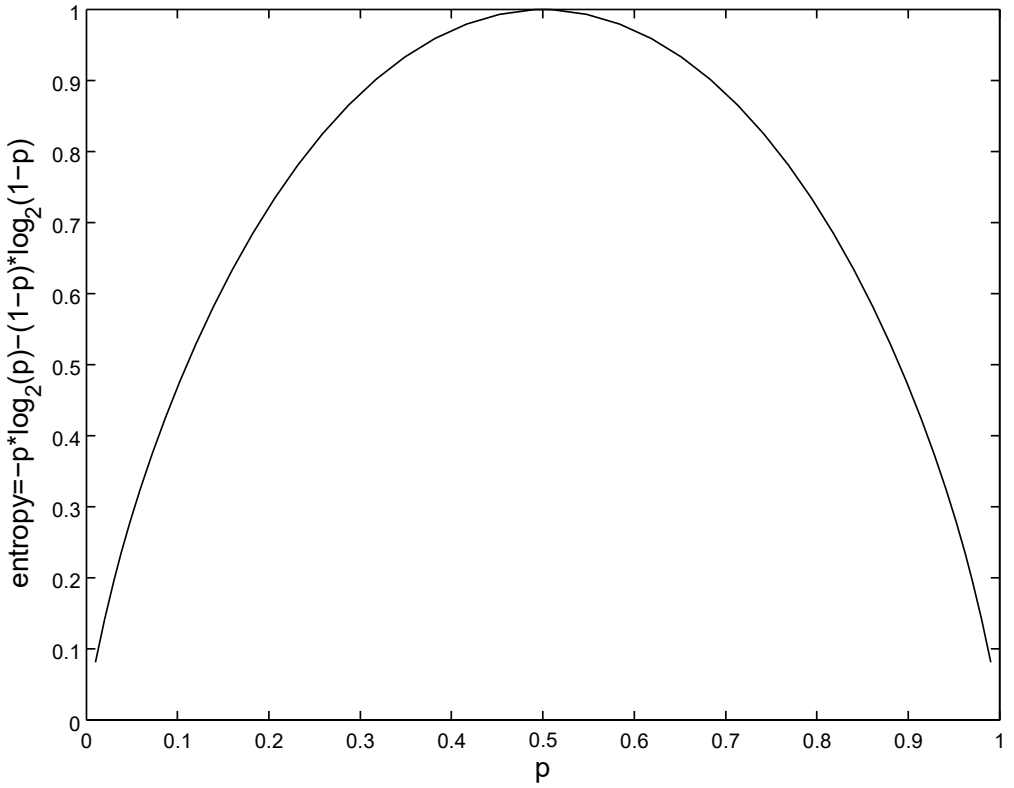
Impurity measure

One possible measure of impurity is to measure entropy:

$$I_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i$$

In information theory entropy specifies the minimum number of bits needed to encode the class code of an instance.

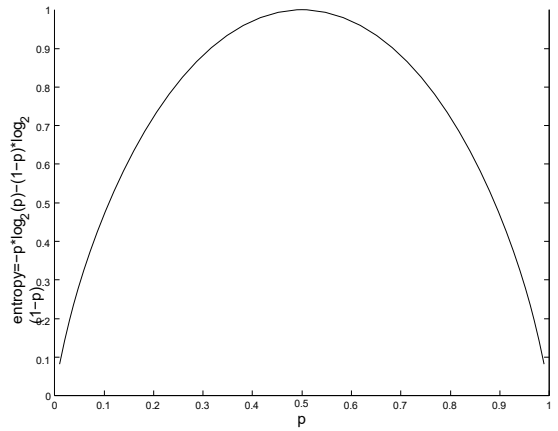
If only one class is possible no information needs to be encoded to know which class it is. Entropy is maximal for $p^1 = p^2 = 0.5$.



http://www.cmpe.boun.edu.tr/~ethem/i2ml2e/index.html

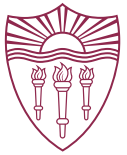
Other Impurity measures

- Let's define: $p^1 = p$ and $p^2 = 1 - p$
- An impurity measure must fulfill:
  1. $\phi(1/2, 1/2) \geq \phi(p, 1-p)$, for any $p \in [0, 1]$
  2. $\phi(0, 1) = \phi(1, 0) = 0$
  3. $\phi(p, 1-p)$ is increasing for $p \in [0, 1/2]$ and decreasing for $p \in [1/2, 1]$



http://www.cmpe.boun.edu.tr/~ethem/i2ml2e/index.html

Other Impurity measures



Let's define: $p^1 = p$ and $p^2 = 1 - p$

1. Entropy: $\phi(p, 1- p) = - p \log_2 p - (1- p) \log_2 (1- p)$
2. Gini-Index: $\phi(p, 1 - p) = 2p(1 - p)$
3. Misclassification Error: $\phi(p, 1 - p) = 1 - \max(p, 1 - p)$

All of these measures estimate the homogeneity of a sample or its inverse the heterogeneity.

All of these measures can be generalized to K > 2 classes.

Each of these measures is a valid estimation of impurity and no significant differences are observed.

Best Split

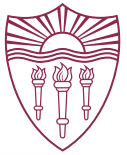
- If node *m* is pure, generate a leaf and stop, otherwise split and continue recursively
- Impurity after split: N_{mj} of N_m take branch *j*. N^i_{mj} belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}}$$

$$\mathcal{I}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

- Find the variable and split that min impurity (among all variables -- and split positions for numeric variables)

Classification Tree Construction



We calculate the impurity for all split positions and choose the one with minimum entropy.

Tree construction is then recursive and in parallel for all impure branches.

For numeric attributes it makes sense to choose splits between samples of opposing classes to minimize search space.

```
GenerateTree(X)
    If NodeEntropy(X)< θ_I /* equation 9.3 */
      Create leaf labelled by majority class in X
      Return
    i ← SplitAttribute(X)
    For each branch of x_i
      Find X_i falling in branch
      GenerateTree(X_i)

SplitAttribute(X)
    MinEnt← MAX
    For all attributes i = 1,...,d
        If x_i is discrete with n values
          Split X into X_1,...,X_n by x_i
          e ← SplitEntropy(X_1,...,X_n) /* equation 9.8 */
          If e<MinEnt MinEnt ← e; bestf ← i
        Else /* x_i is numeric */
          For all possible splits
              Split X into X_1, X_2 on x_i
              e←SplitEntropy(X_1, X_2)
              If e<MinEnt MinEnt ← e; bestf ← i
    Return bestf
```

Example: Predict if John Plays Tennis

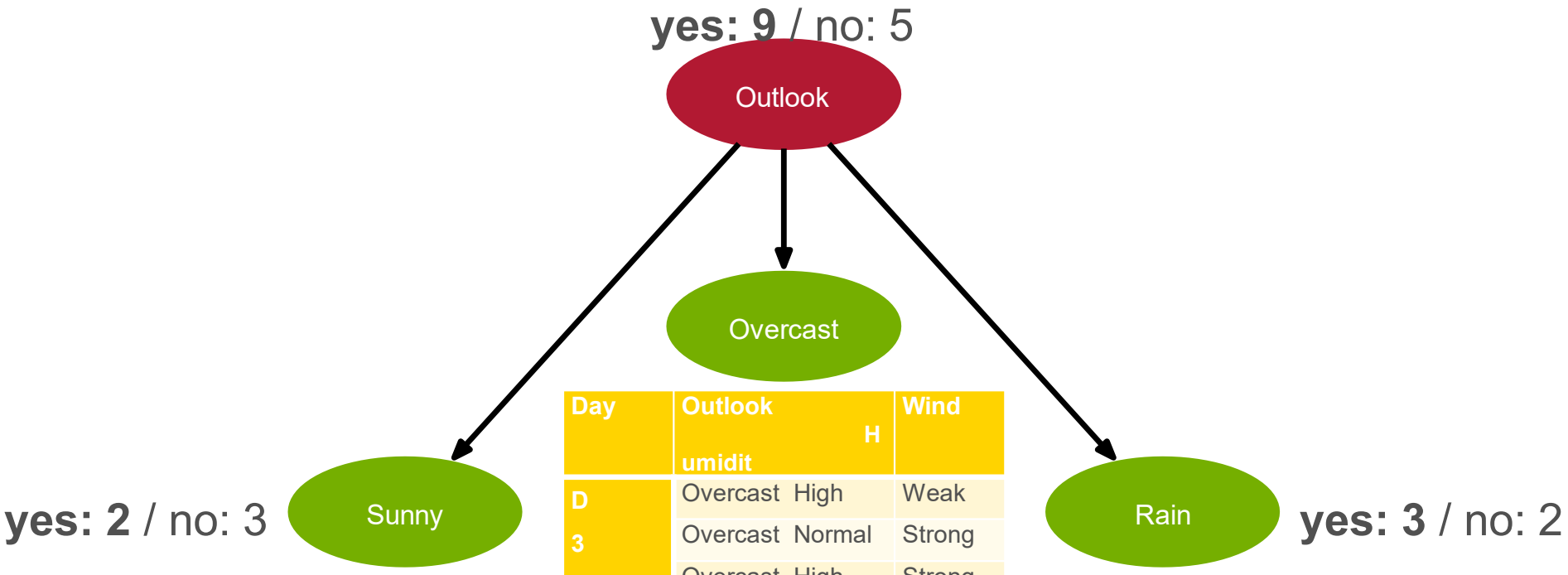
What is John's pattern?

When does he play?

Let's build a decision tree…

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No
New Data:				
D15	Rain	High	Weak	??????

Example: Predict if John Plays Tennis

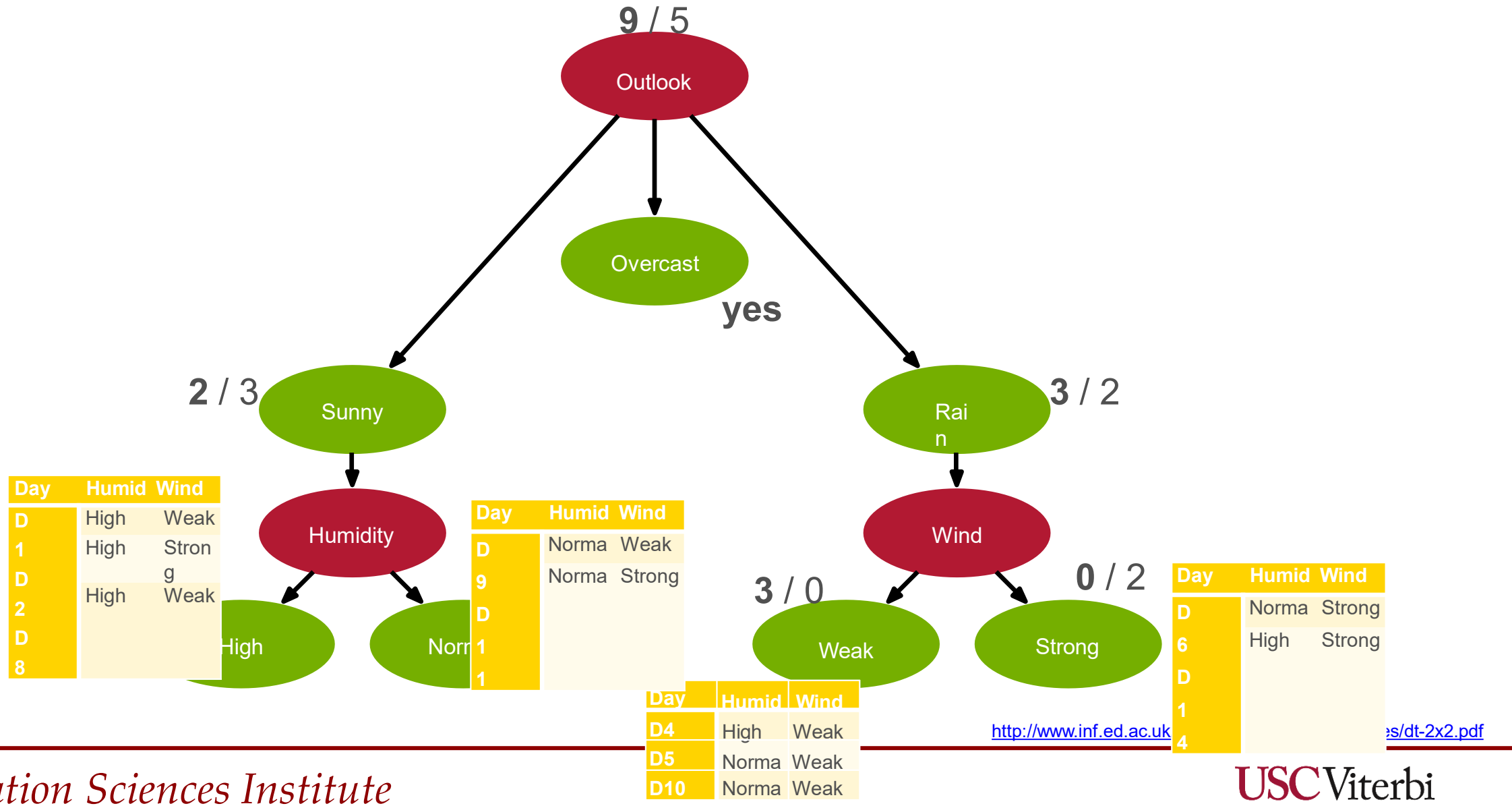


Day	Outlook	Humidit	Wind
D1	Sunny	High	Weak
D2	Sunny	High	Strong
D8	Sunny	High	Weak
D9	Sunny	Normal	Weak
D11	Sunny	Normal	Strong

Day	Outlook	Humidit	Wind
D3	Overcast	High	Weak
D7	Overcast	Normal	Strong
D12	Overcast	High	Strong
D13	Overcast	Normal	Weak

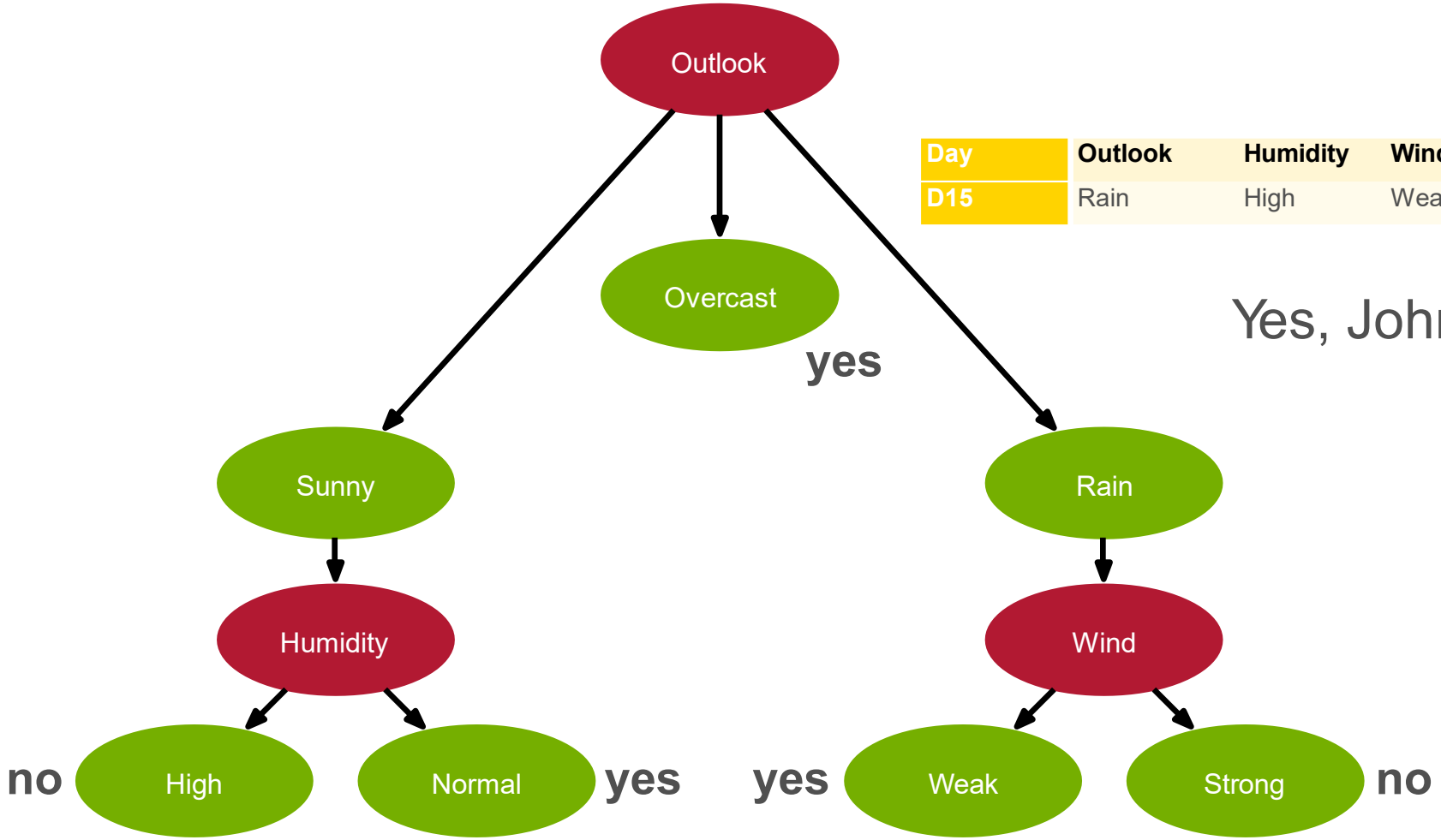
Day	Outlook	Humidit	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D	Rain	High	Strong

Example: Predict if John Plays Tennis



http://www.inf.ed.ac.uk/.../dt-2x2.pdf

Example: Predict if John Plays Tennis



Day	Outlook	Humidity	Wind	Play
D15	Rain	High	Weak	??????

Yes, John will play.

http://www.inf.ed.ac.uk/teaching/courses/iaml/slides/dt-2x2.pdf

Possible Caveats

Overfitting is a big problem

If all nodes only contain one single item then the generalization for unseen data might be really bad.

It is important to have a simple and small tree.

One option is to introduce stop criteria if entropy is below a certain threshold.

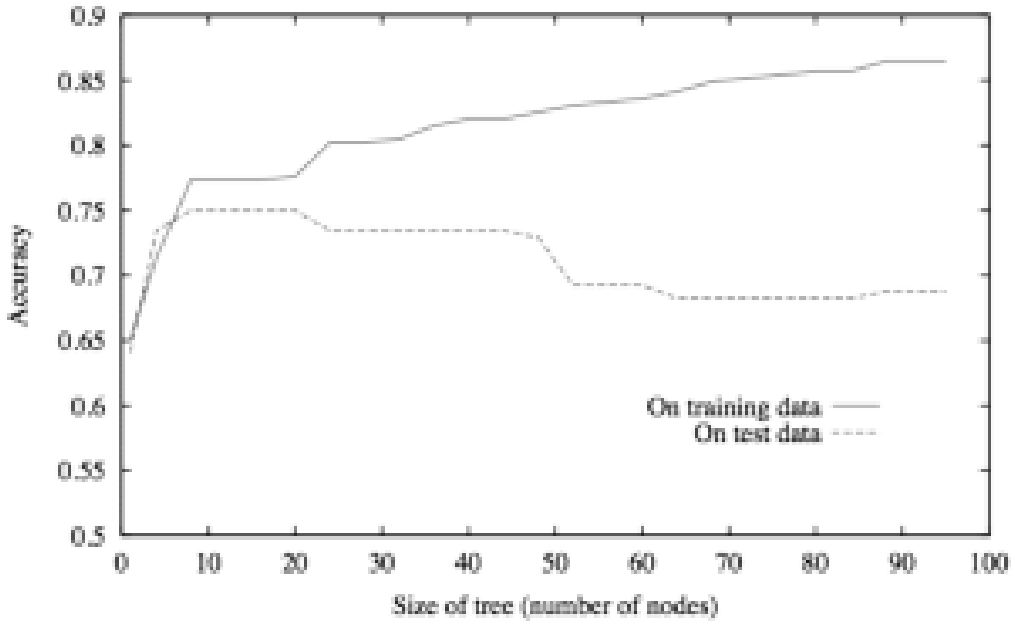
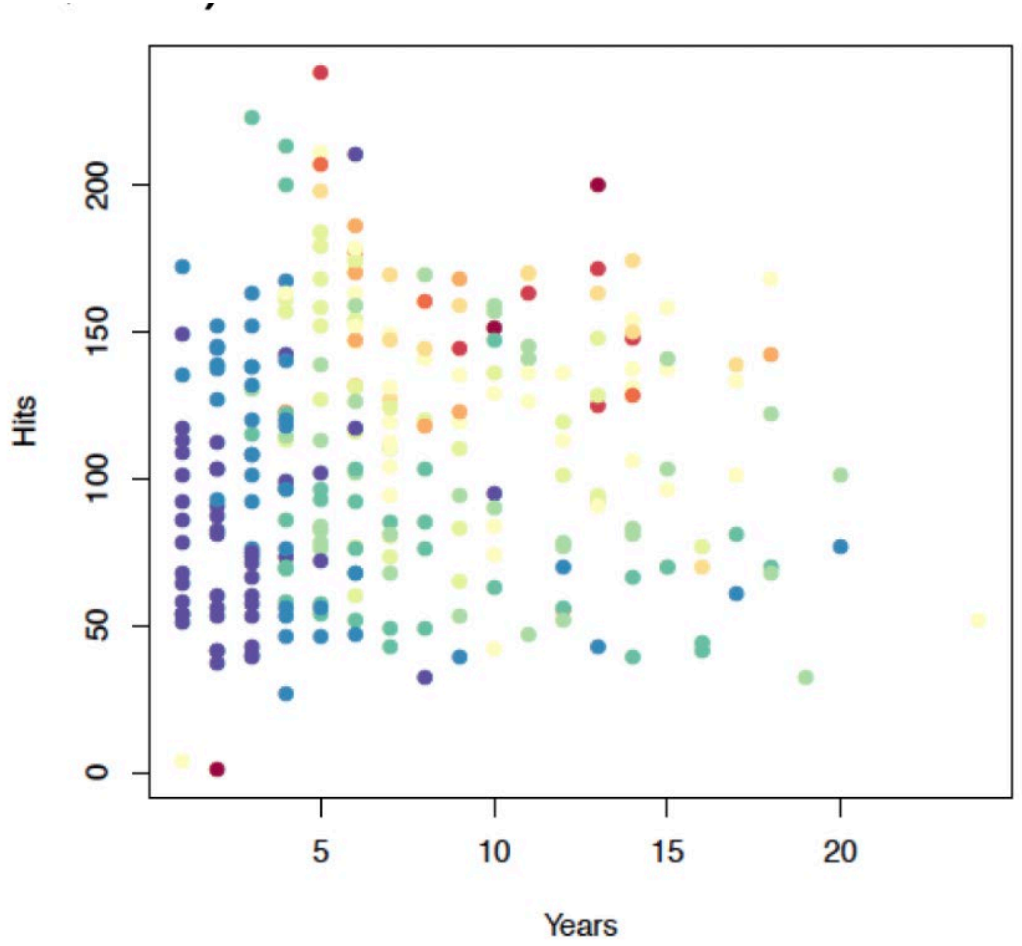


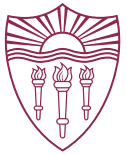
Figure credit: Tom Mitchell, 1997

Regression Trees – An Intuition

Experience vs #hits conditioned on salary

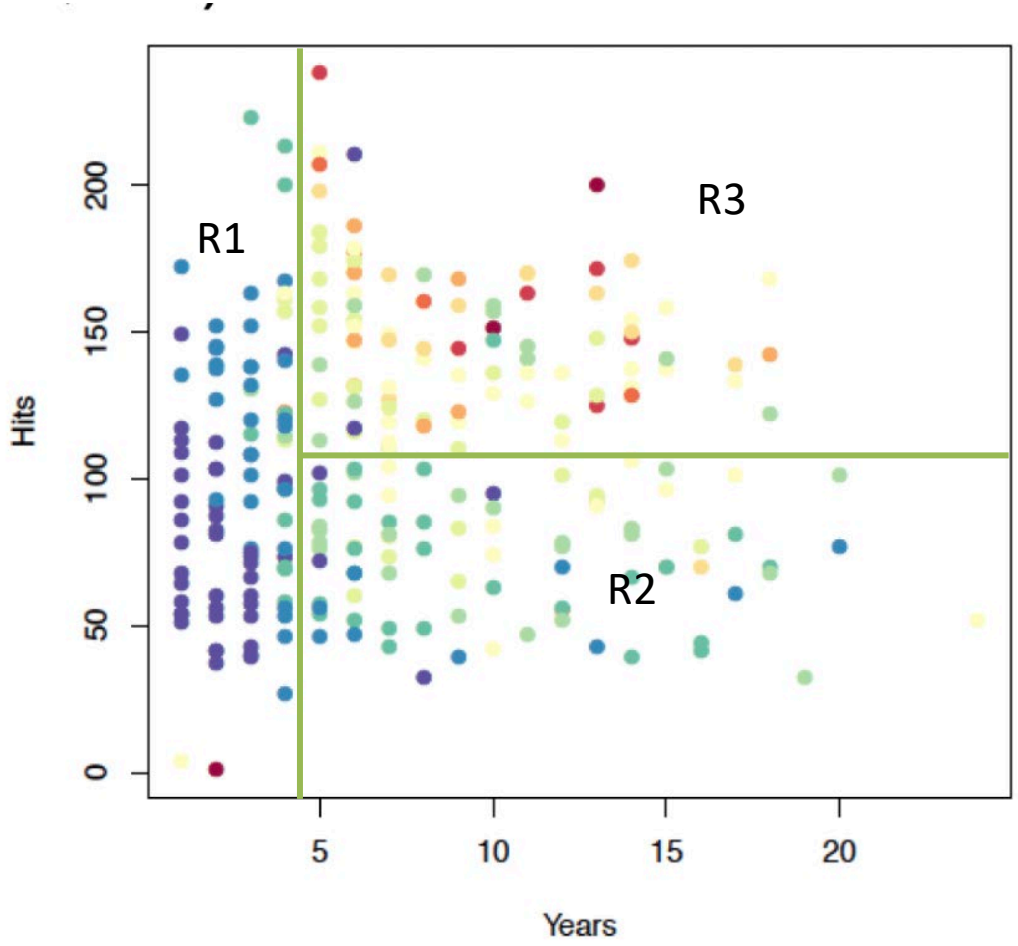


Interpreting the previous slide

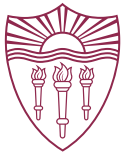


- Eyeballing the dataset, we can perceive that the salary of baseball players largely depends on their years of experience.
- Players with more than 5 years of experience tend to be earning more.
- The number of hits doesn't seem to be a defining predictor of salary for players with less experience.

What do you think ?



Regression Trees – Tree Building



- Building decision trees for a regression task uses the following process:

1. Divide the predictor space $X_1, X_2, \ldots, X_P$ into J *"non-overlapping rectangular"* regions - $R_1, R_2, \ldots, R_J$
2. In training, we compute the mean outcome of observations in the region. This value represents the region. In testing, when an instance falls into a particular region, it is assigned the mean outcome value identified during training.

- The million-dollar question:

How does one split the predictor space into J non-overlapping rectangular regions ?

Regression Trees – Evaluation

- Firstly, let us define a criteria for assessing the performance of our model.
- Akin to linear regression, we make use of Residual Sum of Squares as our performance metric. The goal is to minimize the following function:

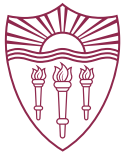
$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

- y_i defines the actual [illegible] servation *i*. $y\hat{}_{R_J}$ is the mean outcome of observations in region R_J during training.

A roadblock ?

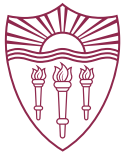
- How do we find the best way to *split* the predictor space using RSS ?
- There can be infinitely many ways to define these *J* rectangular regions.
- In such scenarios, we always resort to greedy approximations.
- Here, we make use of a *top-down greedy* approach known as *recursive binary splitting.*

Recursive Binary Splitting



- This approach is top down because we start off at the top of the tree (at which point all observations belong to a single region) and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.
- It is greedy because at each step we choose the best split at that step irrespective of its future implications.

Recursive Binary Splitting - Training



- We first select the predictor X_j and the cut point *s* such that splitting the predictor space into the regions{X|Xj<s} and {X|Xj≥s} leads to the greatest possible reduction in RSS.
- In other words, we consider all predictors $X_1,\ldots,X_p$, and all possible values of the cutpoint *s* for each of the predictors, and then choose the predictor and cutpoint such that the resulting tree has the lowest RSS.
- The goal more specifically is to get two regions R_1 and R_2 such that this function is minimized

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Recursive Binary Splitting - Training

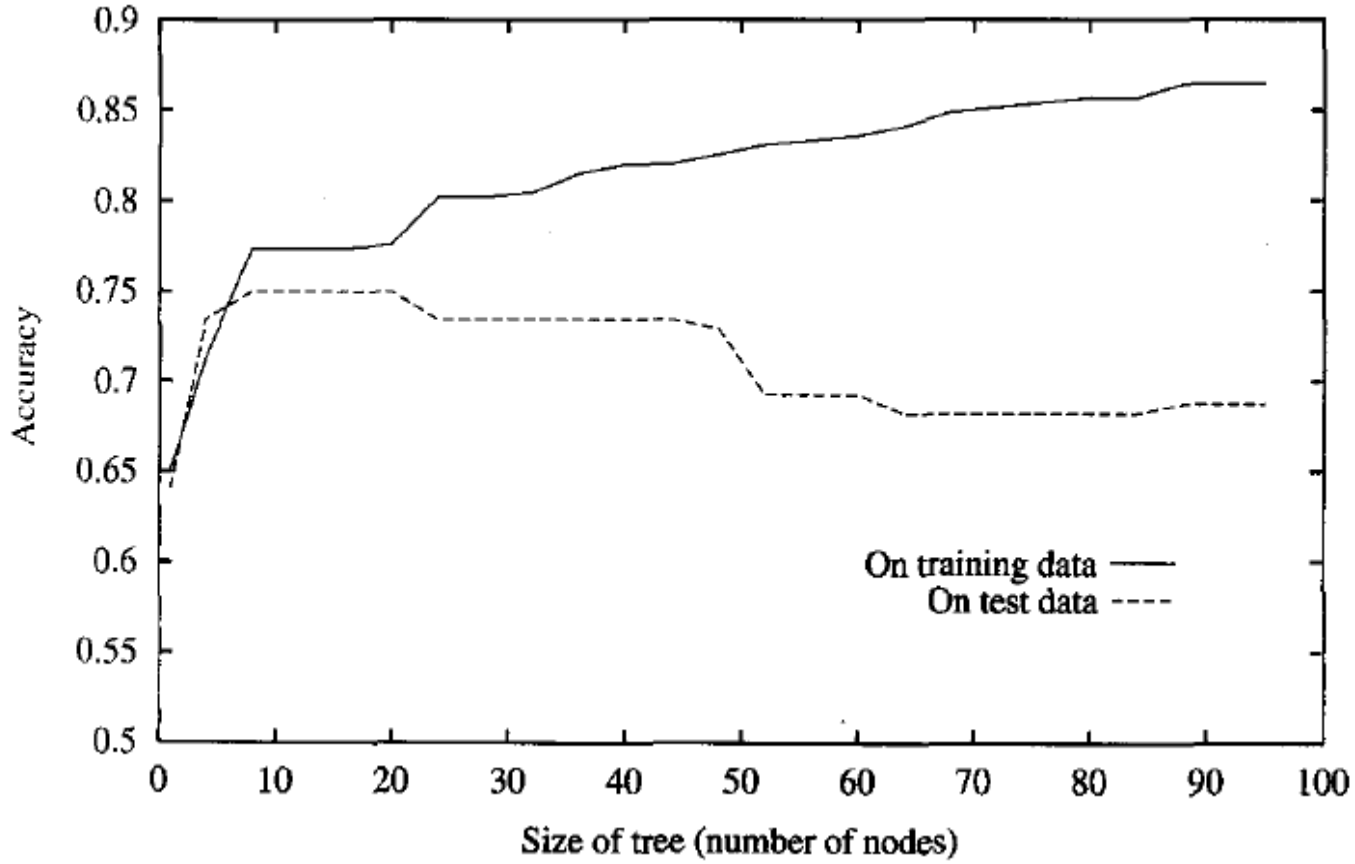
- In the next step of splitting, we consider the best predictor in one of the regions obtained in the previous step and split it in a way that reduces the RSS further. We now have three regions.
- We continue to split in this fashion until a stopping criterion is met, say each region has no more than five observations.

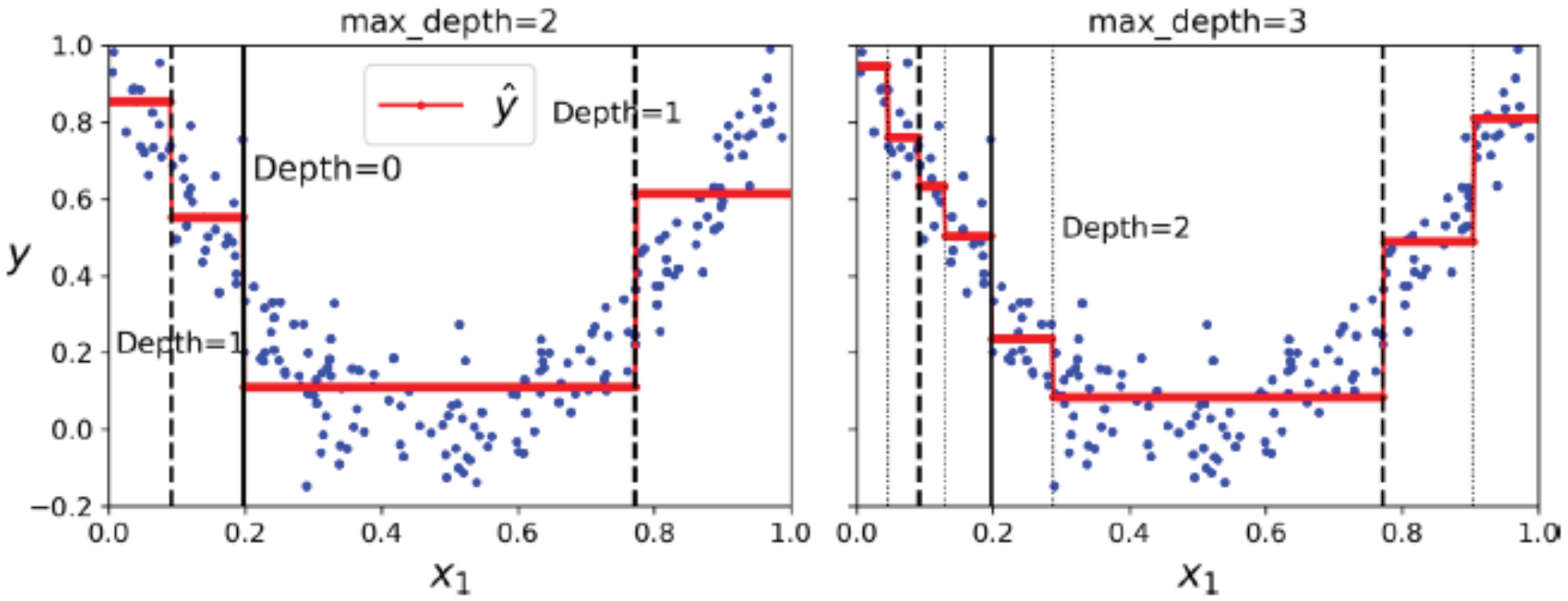
Predicting

- We pass the instances down through the tree we learnt in training.
- At each step, the instance moves either to the left or right depending on its value for a predictor.
- The value for the instance is predicted to be the mean outcome of observations in a region obtained during training.

Problems with Recursive Binary Splitting

- As we grow the tree, the variance of our model increases, and the model begins to incorporate the wigglyness in our training set.
- This can cause the model to overfit.



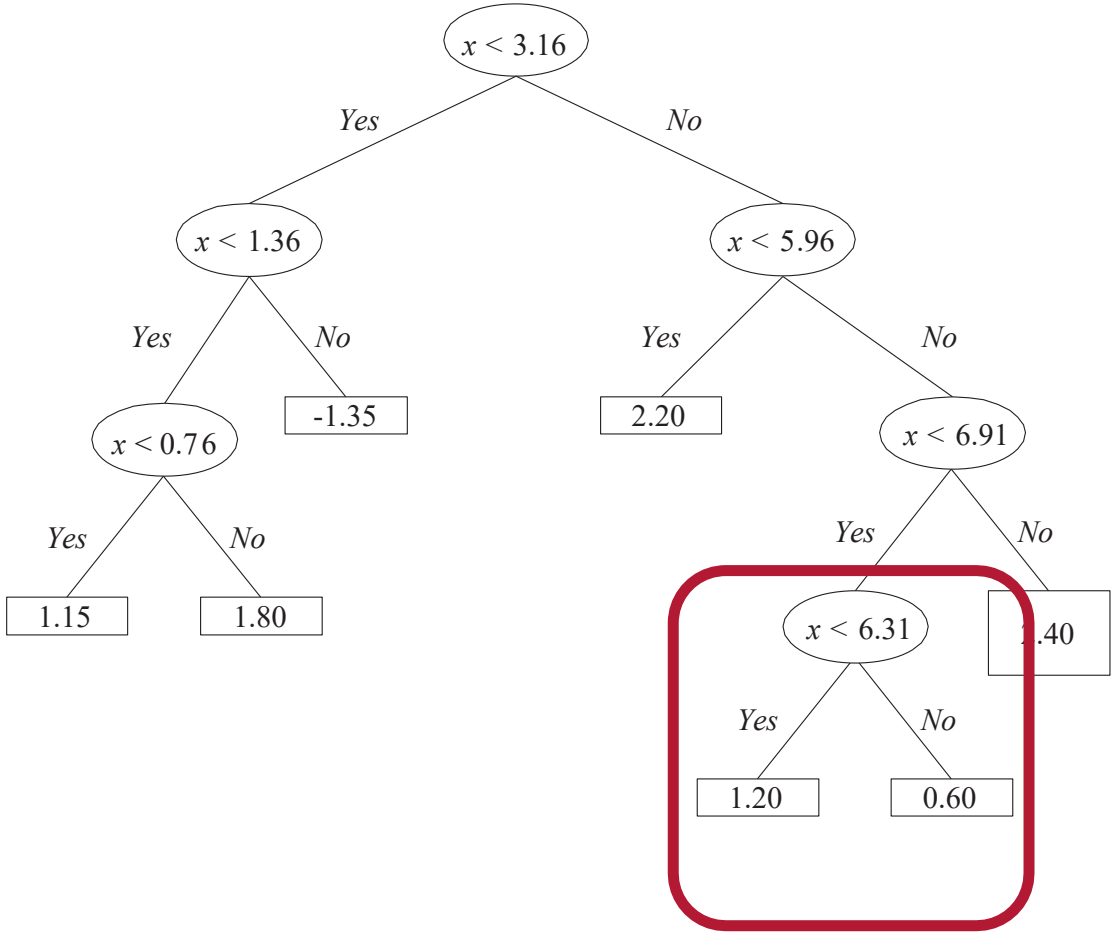

max_depth=2
max_depth=3
$\hat{y}$
Depth=0
Depth=1
Depth=1
Depth=2
y
x_1
x_1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0

Pruning Trees

One method to keep a tree small is to stop splitting after a minimal number of observations in a node is reached.

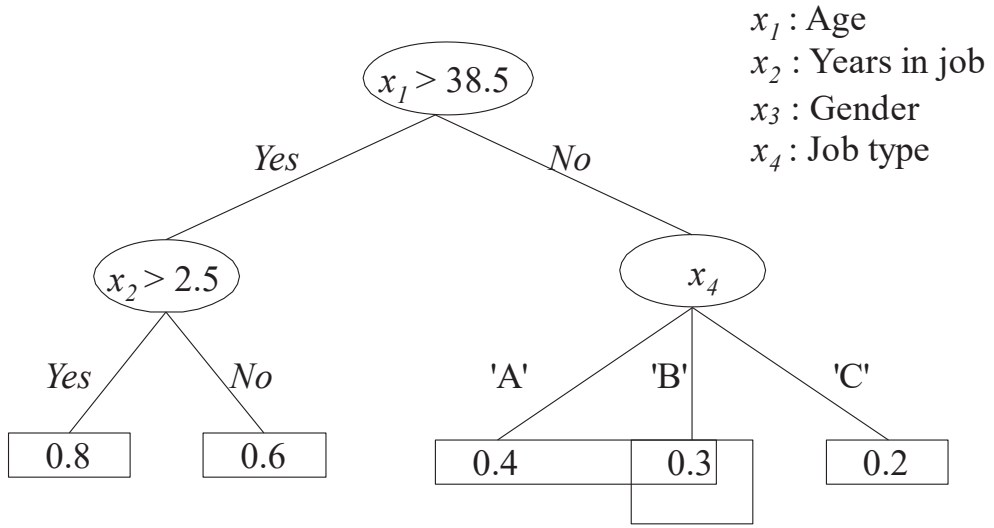
Some pruning techniques involve backtracking after a tree is fully grown:

Pruning set: we hold out a subset of the training data to prune the tree; a tree is shrunk in size if it's child nodes perform as good as the parent on the prune set.



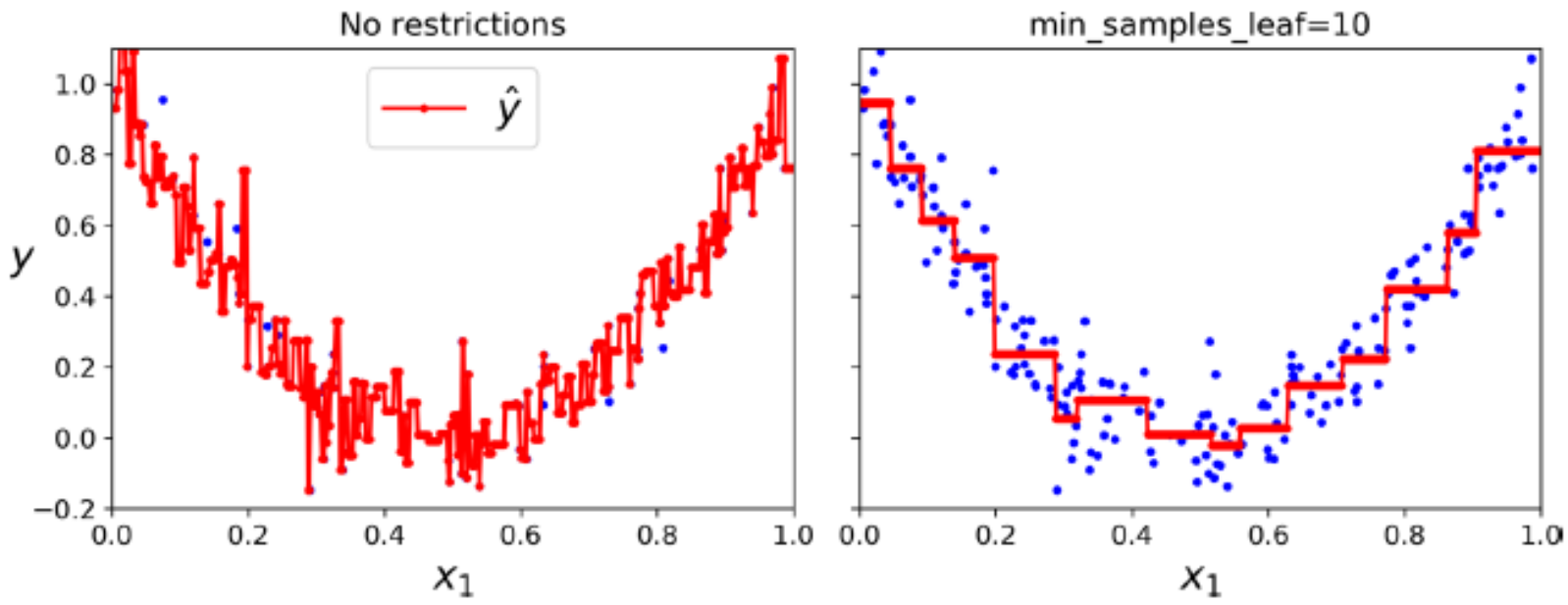
Rule extraction from Trees

- What rules can be extracted from this hypothetical tree?
- Trees are ideal for *knowledge extraction* as they can be easily understood.
- *Rule support* is calculated by the amount of data covered by the rule.
- In case of classification trees one can combine rules with OR to merge multiple leaves:
- IF ($x < w_{10}$) OR (($x_1 > w_{10}$) AND ($x_2 > w_{20}$)) THEN C_1

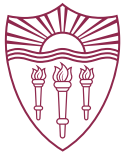


R1: IF (age>38.5) AND (years-in-job>2.5) THEN $y = 0.8$
R2: IF (age>38.5) AND (years-in-jobδ 2.5) THEN $y = 0.6$
R3: IF (age< 38.5) AND (job-type='A') THEN $y = 0.4$
R4: IF (age< 38.5) AND (job-type='B') THEN $y = 0.3$
R5: IF (age< 38.5) AND (job-type='C') THEN $y = 0.2$

Regularizing decision tree with min-samples



Multivariate Trees



Univariate trees only utilize one dimension in each decision node.

This can be generalized to multiple dimensions in multivariate trees.

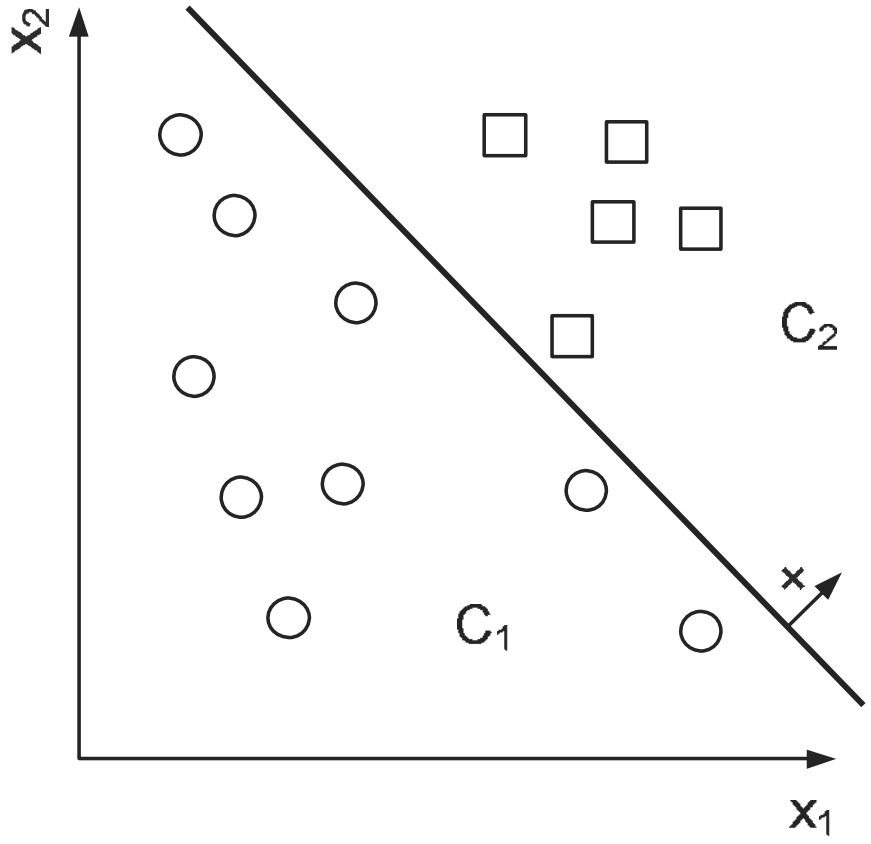
A multivariate linear decision function could for example be: $f_m(x) : w^T_m x + w_{m0}$

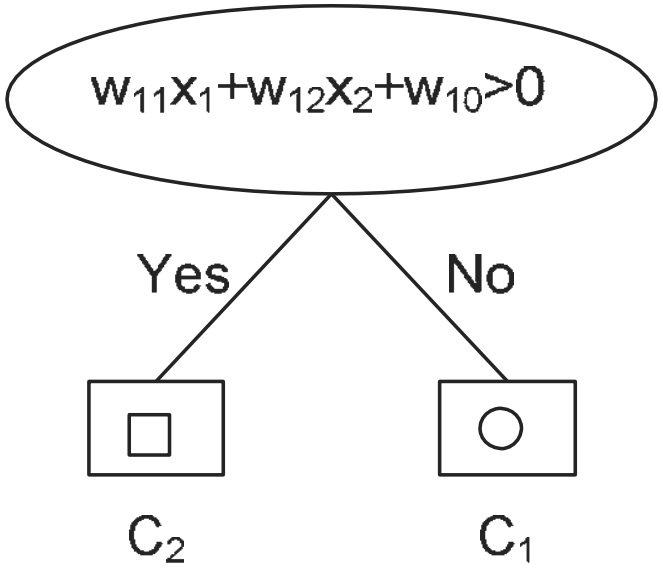
- $f_m(x)$ in this case is a weighted sum with a bias.
- In a high dimensional space many possible $f_m(x)$ are possible and an exhaustive search is not feasible anymore.
- More complex decision functions are possible:

Quadratic decision function: $f_m(x) : x^T W_m x + w^T_m x + w_{m0}$

Others: Neural network, spheres, etc.

Multivariate Trees



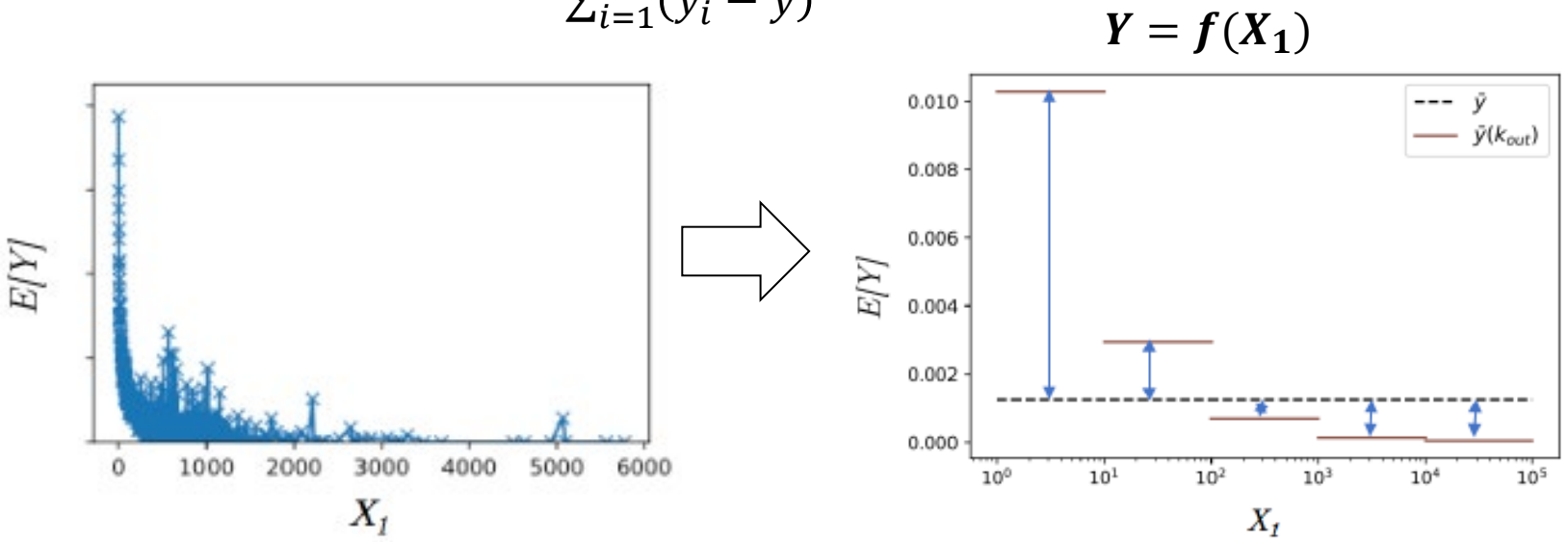


Exploratory Data Analysis with Decision Trees

Fennell, P. G., Zuo, Z., & Lerman, K. (2019). Predicting and explaining behavioral data with structured feature space decomposition. *EPJ Data Science*, *8*(1), 23.

- Structured Sum of Squares Decomposition (S3D)
  - Successively picks features for binning data
  - Creates a non-linear model of data

$$R^2 = \frac{\sum_{g=1}^{G} N_g \left(\bar{y}_g - \hat{y}\right)^2}{\sum_{i=1}^{N} (y_i - \hat{y})^2}$$

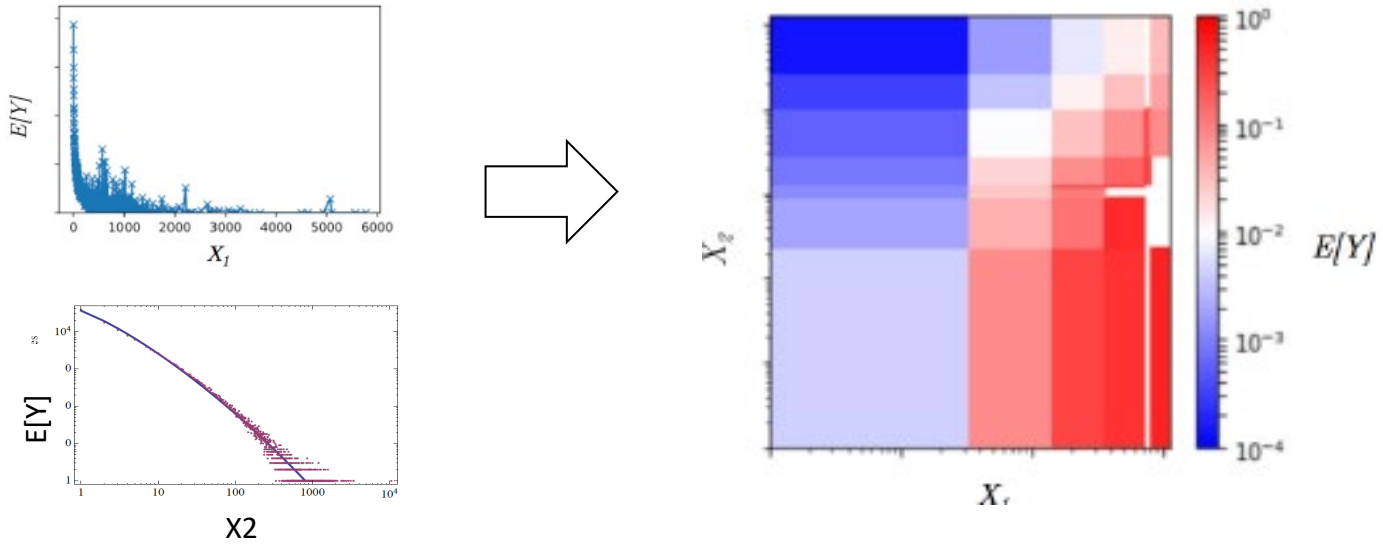


Structured Sum of Squares Decomposition (S3D)

https://github.com/peterfennell/S3D

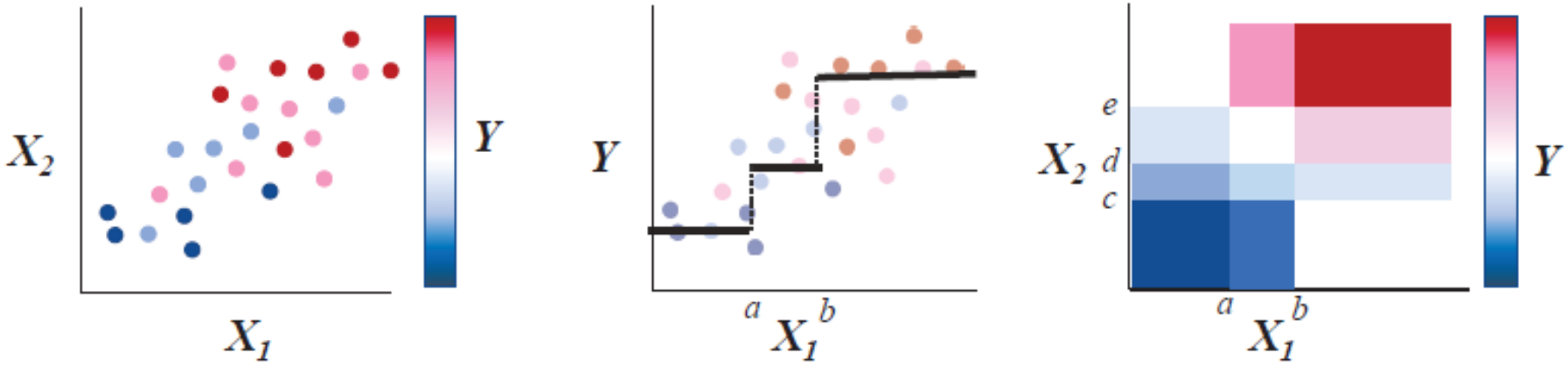
- Successively picks features for binning data
- Creates a non-linear model of data useful for visualization and prediction

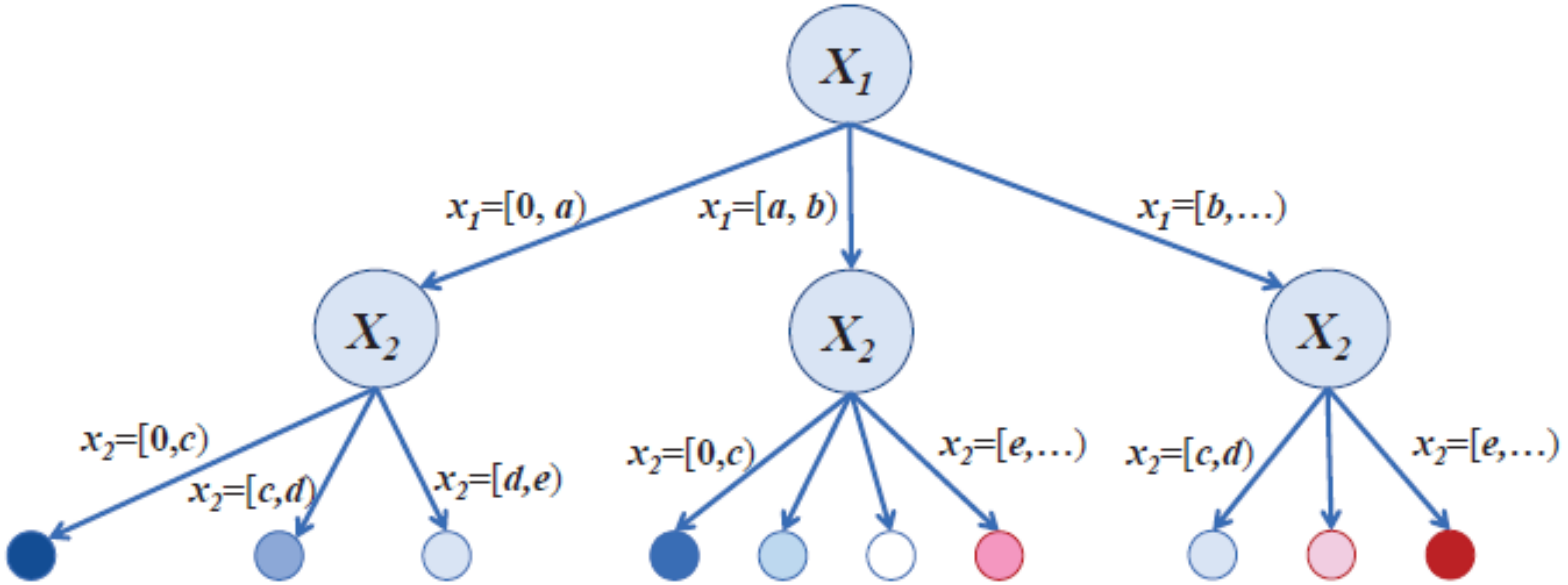
$$R^2 = \frac{\sum_{g=1}^{G} N_g \left(\bar{y}_g - \hat{y}\right)^2}{\sum_{i=1}^{N} (y_i - \hat{y})^2} \qquad \boldsymbol{Y = f(X_1, X_2)}$$



Color gives the average value of outcome in each bin.

← Nonlinear approximation of data





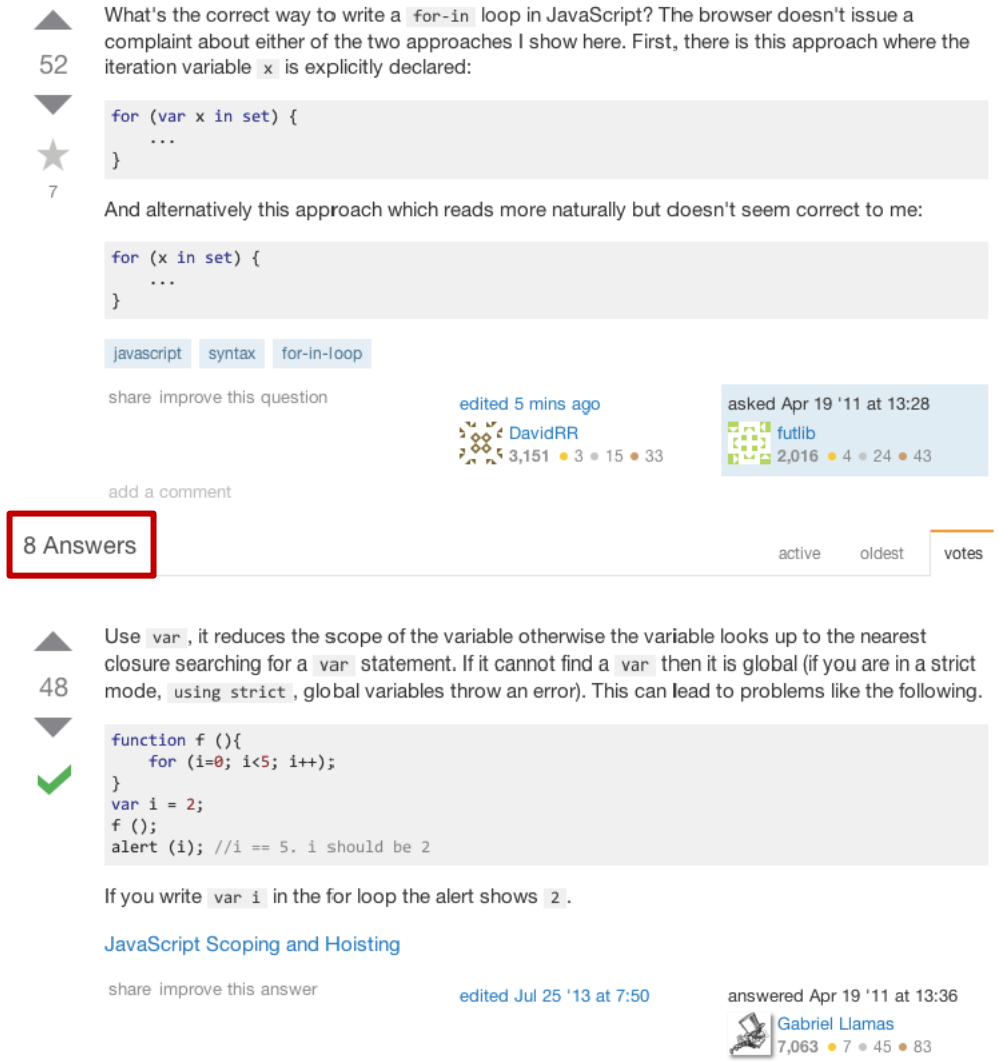
https://github.com/peterfennell/S3D

Anatomy of Stack Overflow

Question

Number of answers to the question →

Answers



What's the correct way to write a `for-in` loop in JavaScript? The browser doesn't issue a complaint about either of the two approaches I show here. First, there is this approach where the iteration variable `x` is explicitly declared:

```
for (var x in set) {
    ...
}
```

And alternatively this approach which reads more naturally but doesn't seem correct to me:

```
for (x in set) {
    ...
}
```

javascript syntax for-in-loop

share improve this question

edited 5 mins ago
DavidRR
3,151 • 3 • 15 • 33

asked Apr 19 '11 at 13:28
futlib
2,016 • 4 • 24 • 43

add a comment

8 Answers

active oldest votes

Use `var`, it reduces the scope of the variable otherwise the variable looks up to the nearest closure searching for a `var` statement. If it cannot find a `var` then it is global (if you are in a strict mode, `using strict`, global variables throw an error). This can lead to problems like the following.

```
function f (){
    for (i=0; i<5; i++);
}
var i = 2;
f ();
alert (i); //i == 5. i should be 2
```

If you write `var i` in the for loop the alert shows `2`.

JavaScript Scoping and Hoisting

share improve this answer

edited Jul 25 '13 at 7:50

answered Apr 19 '11 at 13:36
Gabriel Llamas
7,063 • 7 • 45 • 83

Answer features

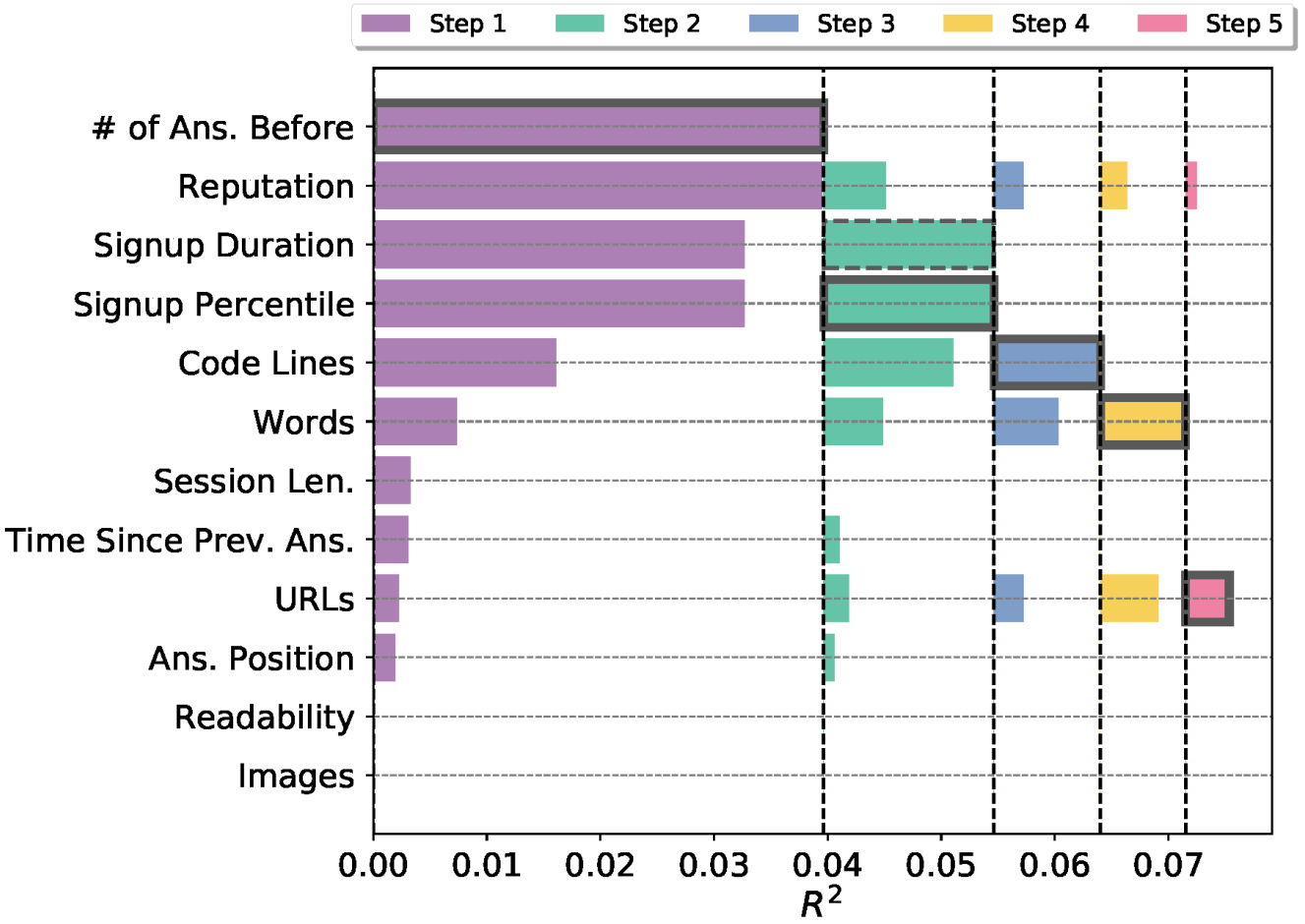
- votes/score
- accepted?
- web page order
- chrono order

- num words
- word share
- hyperlinks
- readability
- age

- answerer reputation
- tenure

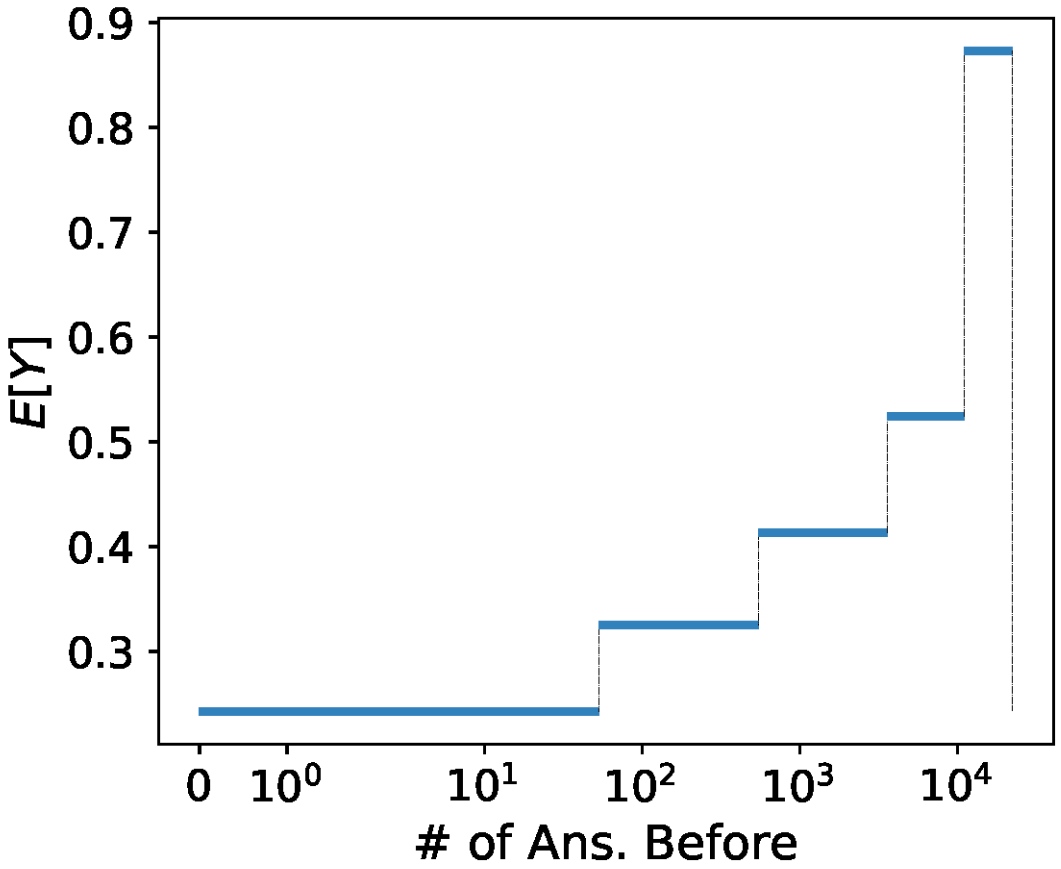
S3D illustration on StackOverflow data

Identifying important features explaining whether user's answer is accepted as best answer to the question (Y=binary outcome)



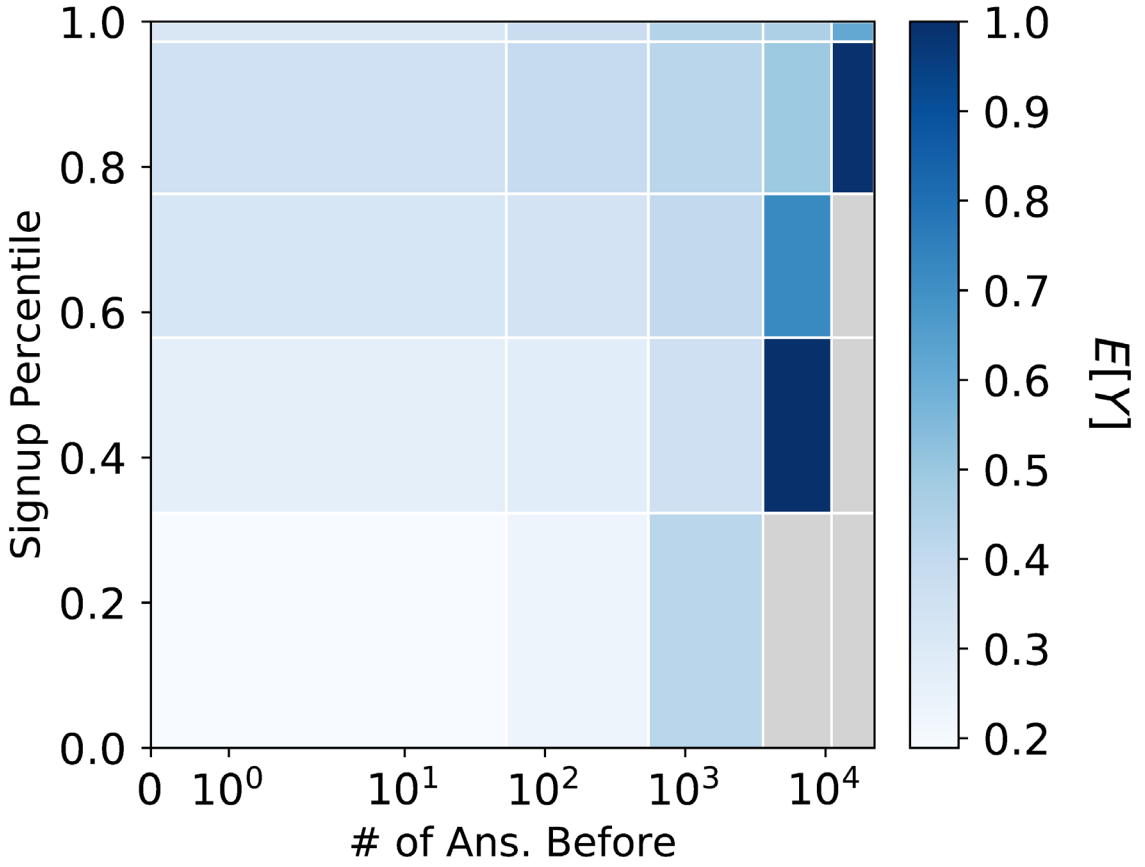
Visualizing data: 1st Important Feature

***# Answers Before**:* more experienced users (who had answered more questions on StackOverflow) are more likely to have their new answers accepted

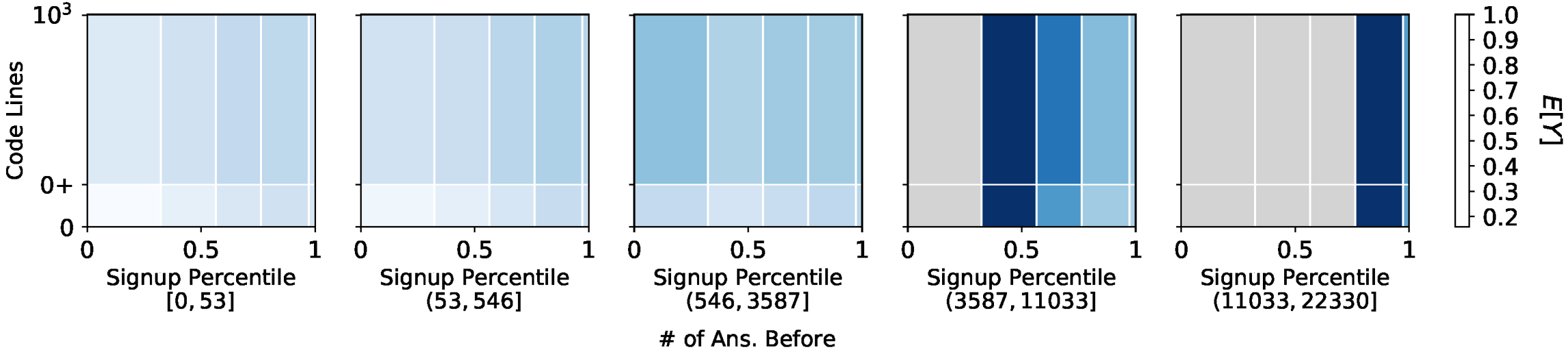


2nd Important Feature

Signup percentile: rank of user's tenure on StackOverflow. Older users who had written the same number of answers are less likely to have a new answer accepted as best answer.

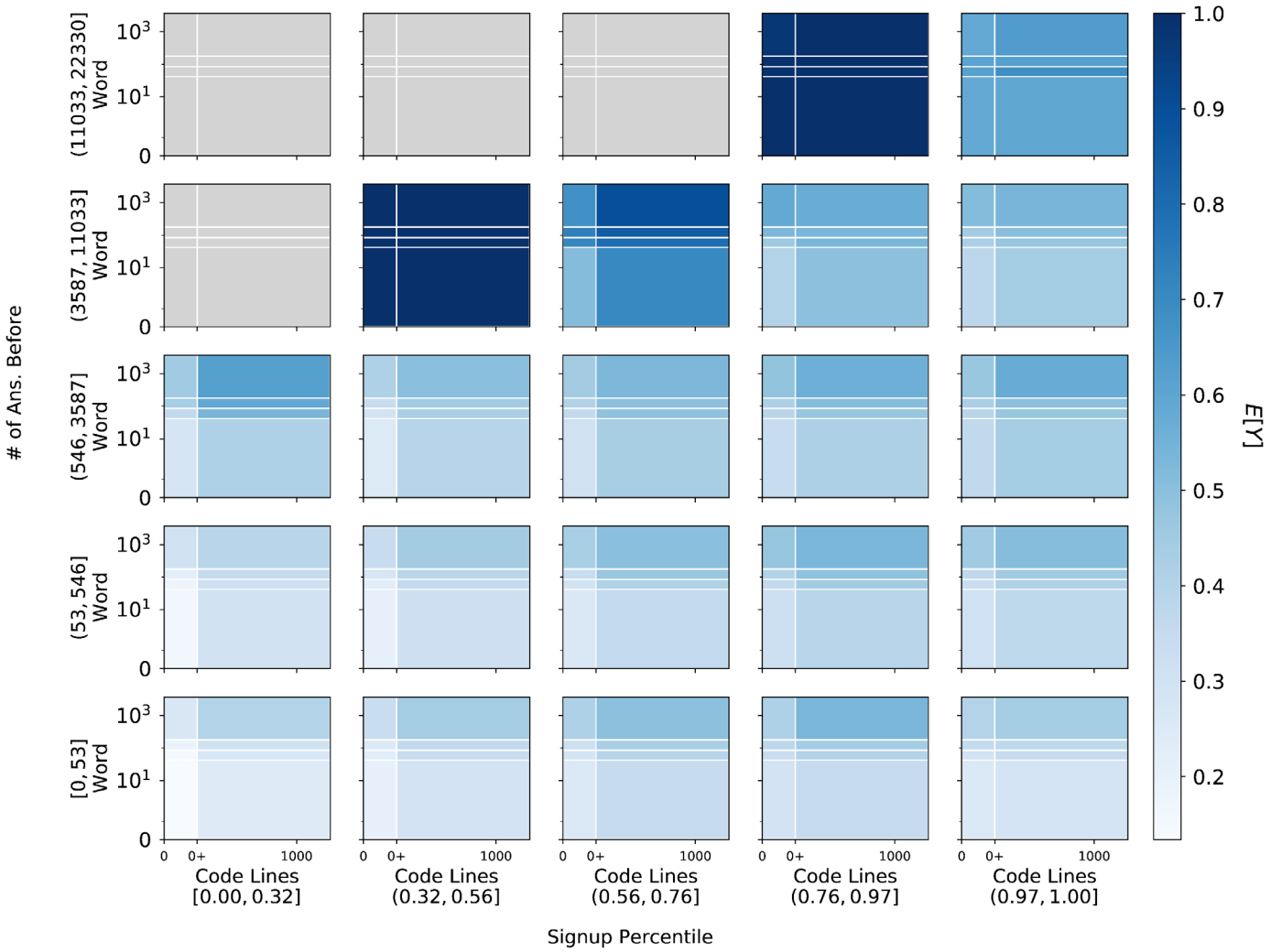


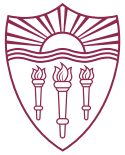
3rd important feature



Code lines: otherwise similar users who include more code in their answer are somewhat more likely to have the answers accepted as best answers.

4th important feature





- Questions?

- Virtual office hour
- https://usc.zoom.us/j/95136500603?pwd=VEJhblhWK25lT2N3RC9FNWk3eTJKQT09