Data Mining Methods: Tree Induction Algorithms

Weeks 5-6

ITC 6000/6335

From Correlation to Supervised Segmentation

- Supervised segmentation
- Find important variables
- Models, Induction, and Prediction
- Definition of prediction
- Descriptive modeling
- Supervised learning
- Instance – feature vector, row in a dataset
- Attributes
- Dataset, worksheet, table all mean the same
- Dependent and independent variables
- Induction – training and labeled data

Induction in the Data Mining context

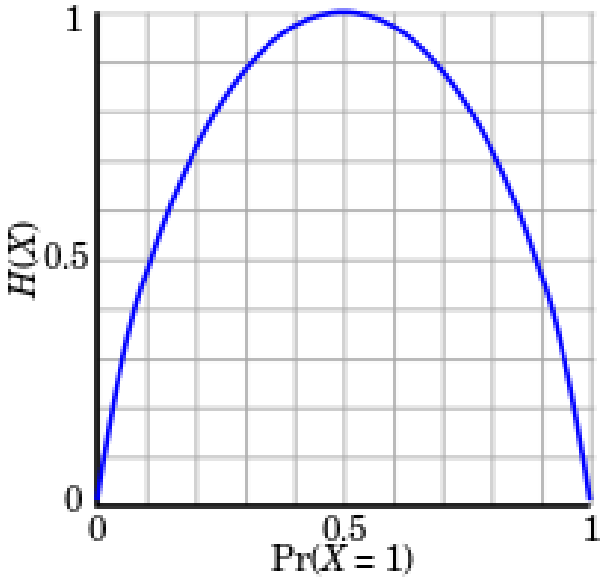
- “Induction” means learning general rules from specific facts or examples you are given.
- In the data mining context, the specific facts or examples are your points in the data set that you have. That is, the data represent all the specific facts or examples about the world we know.
- Learning general rules means taking millions and millions of data points and replacing them with a model that can represent these points, and hopefully future ones too!

Supervised Segmentation

- Segment population into subgroups
- Determine segments
- Multivariate supervised segmentation
- Selecting informative attributes
  - Attributes
    - Head-shape: square or circular
    - Body shape: rectangular or oval
    - Body color: gray or white
  - Target Variable
    - Write-off: Yes or No
- Complications:
  1. Not perfect split
  2. Best split
  3. Not always binary
  4. Numeric values

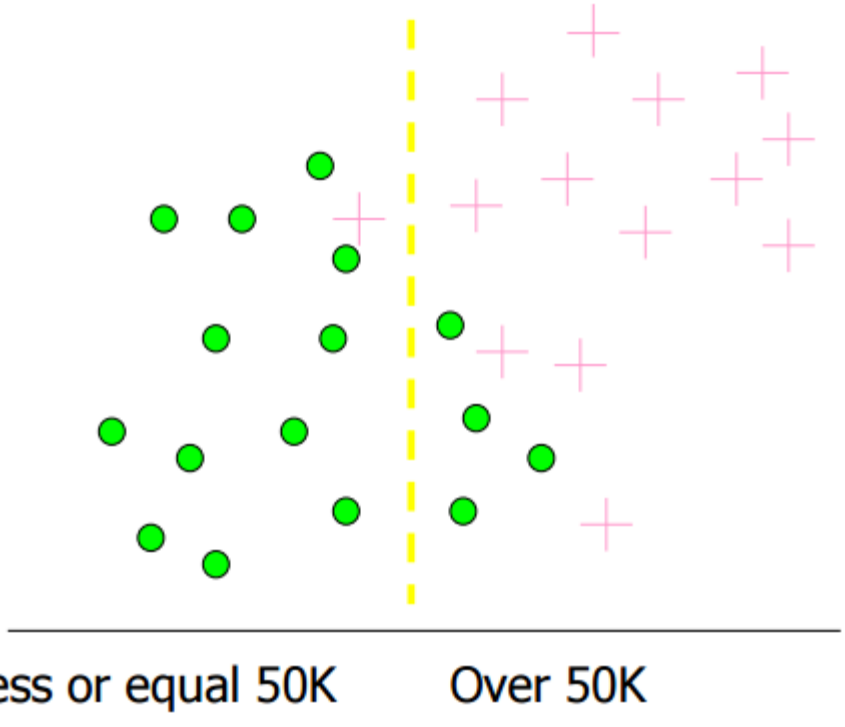
Selecting Informative Attributes

- Purity Measure
- Information gain
- Entropy: $entropy = -p_1 \log(p_1) - p_2 \log(p_2) - \ldots.$

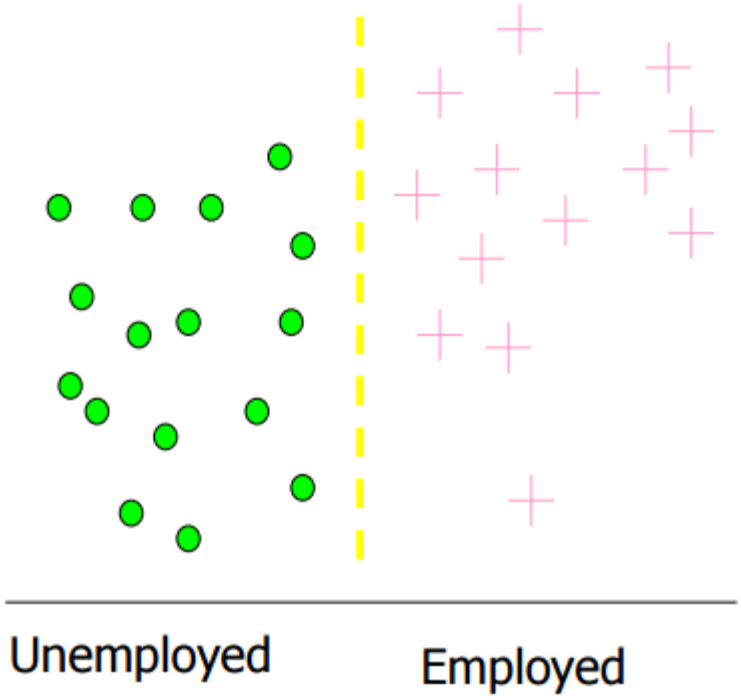


Which test is more informative?

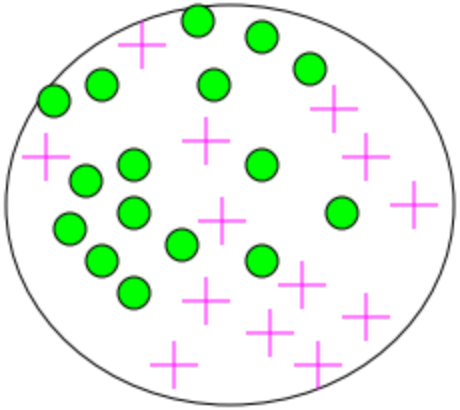
Split over whether Balance exceeds 50K



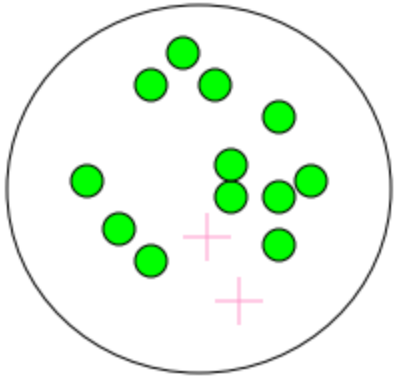
Split over whether applicant is employed



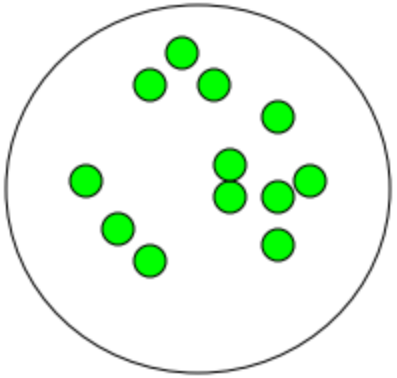
Very impure group



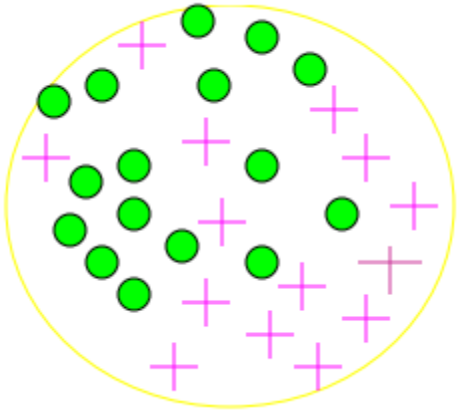
Less impure



Minimum impurity



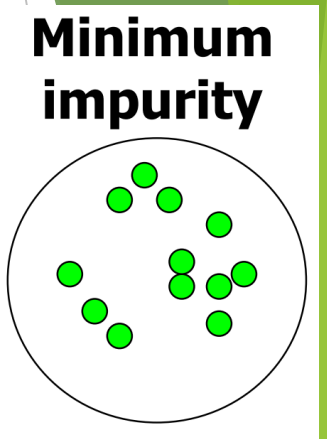
Entropy



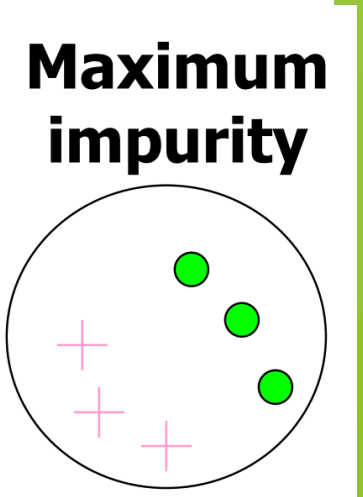
- Entropy is a common way to measure impurity
  - $entropy = \sum_i - p_i log_2 p_i$ where p_i is the probability of class I
  - 16/30 are green circles and 14/30 are pink crosses
  - Log (16/30) = -.9
  - Log (14/30) = -1.1
  - Entropy = -(16/30)(-.9) – (14/30)(-1.1) = .99

2 Class Cases

- What is the entropy of a group in which all examples belong to the same class?
  - Entropy = $-1 \log_2 1 = 0$
  - Not a good training set for learning



- What is the entropy of a group with 50% in either class?
  - Entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$
  - A good training set for learning



Write-Off Example

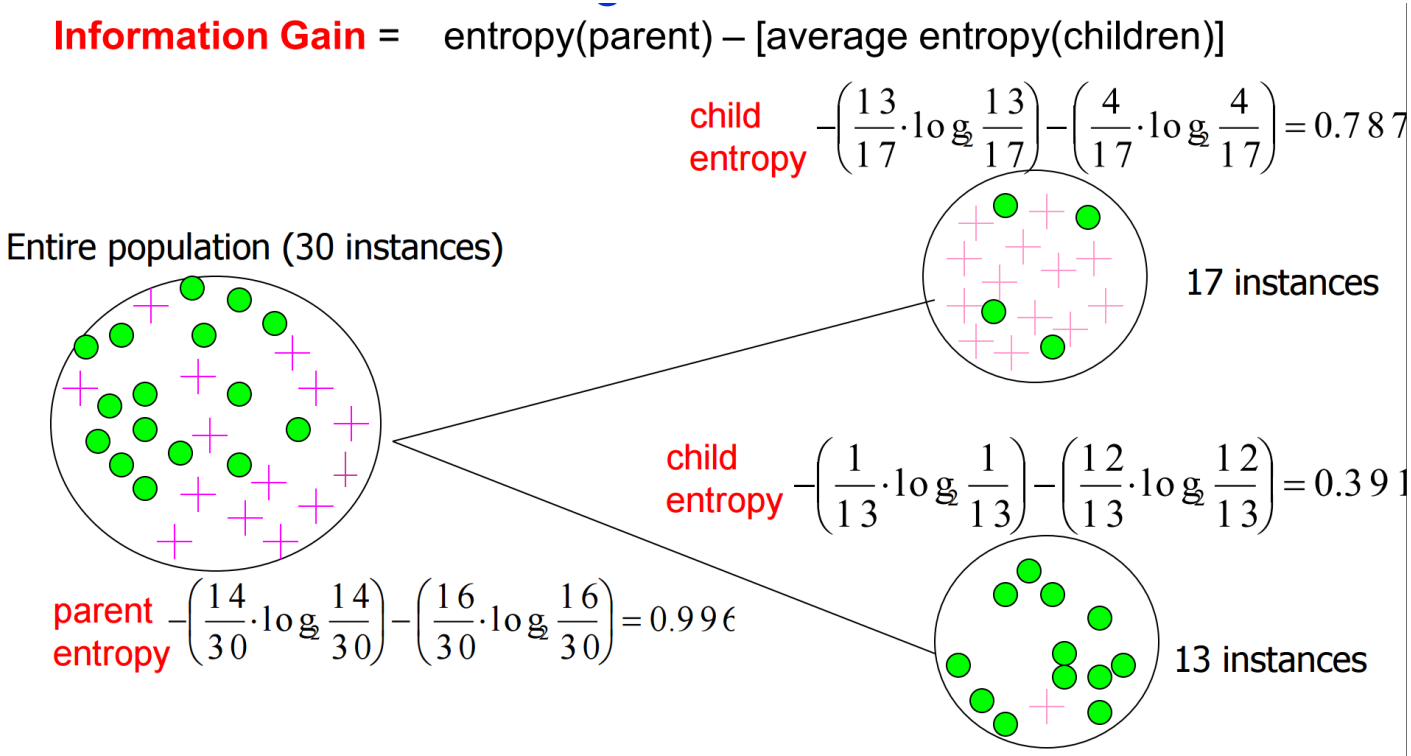
- Consider a set of S of 10 people with 7 of the non-write-off class and 3 of the write-off class
- p(non write-off)=7/10=0.7
- p(write-off)=3/10=0.3

- Entropy(S) = - [0.7 x log (0.7) + 0.3 x log (0.3)]
  = - [0.7 x -0.51 + 0.3 x - 1.74]
  = 0.88

- Informative attributes: Information Gain (IG)
  - $IG\ (parent, children) = entropy(parent) - [p(c_1)\ x\ entropy(c_1) + p(c_2)\ x\ entropy(c_2) + \ \ldots]$

Calculating Information Gain

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



(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

$Information\ Gain = 0.996 - 0.615 = 0.38$ for this split

Variance

- Variance – a natural measure of impurity for numeric measures
- Weighted average variance reduction

Mushroom Example

- Find the most informative attribute
- Target variable: edible
- Find the single best attribute/variable
- *Entropy(parent)*

Supervised Segmentation with Tree-Structured Models

- Multivariate supervised segmentation
- Classification/Decision Tree
- Using the tree
- Tree induction

Visualizing Segmentations

- Scatterplot
- Decision lines and hyperplanes
  - In 2D the line will be either horizontal or vertical. In higher dimensions, since each node of a classification tree tests one variable it may be thought of as "fixing" that dimension of a decision boundary; therefore for a problem of *n* variables, each node of a classification tree imposes an *(n-1)*-dimensional "hyperplane" decision boundary on the instance space.
- Trees as sets of rules
  - IF (Balance<50K) AND (Age<50) THEN Class=Write-off
- Probability Estimation
  - Class purity/entropy
  - n/(n+m)
  - Laplace correction $p(c) = \frac{n+1}{n+m+2}$ where *n* is the number of examples in the leaf belonging to class *c*, and *m* is the number of examples not belonging to class *c*.

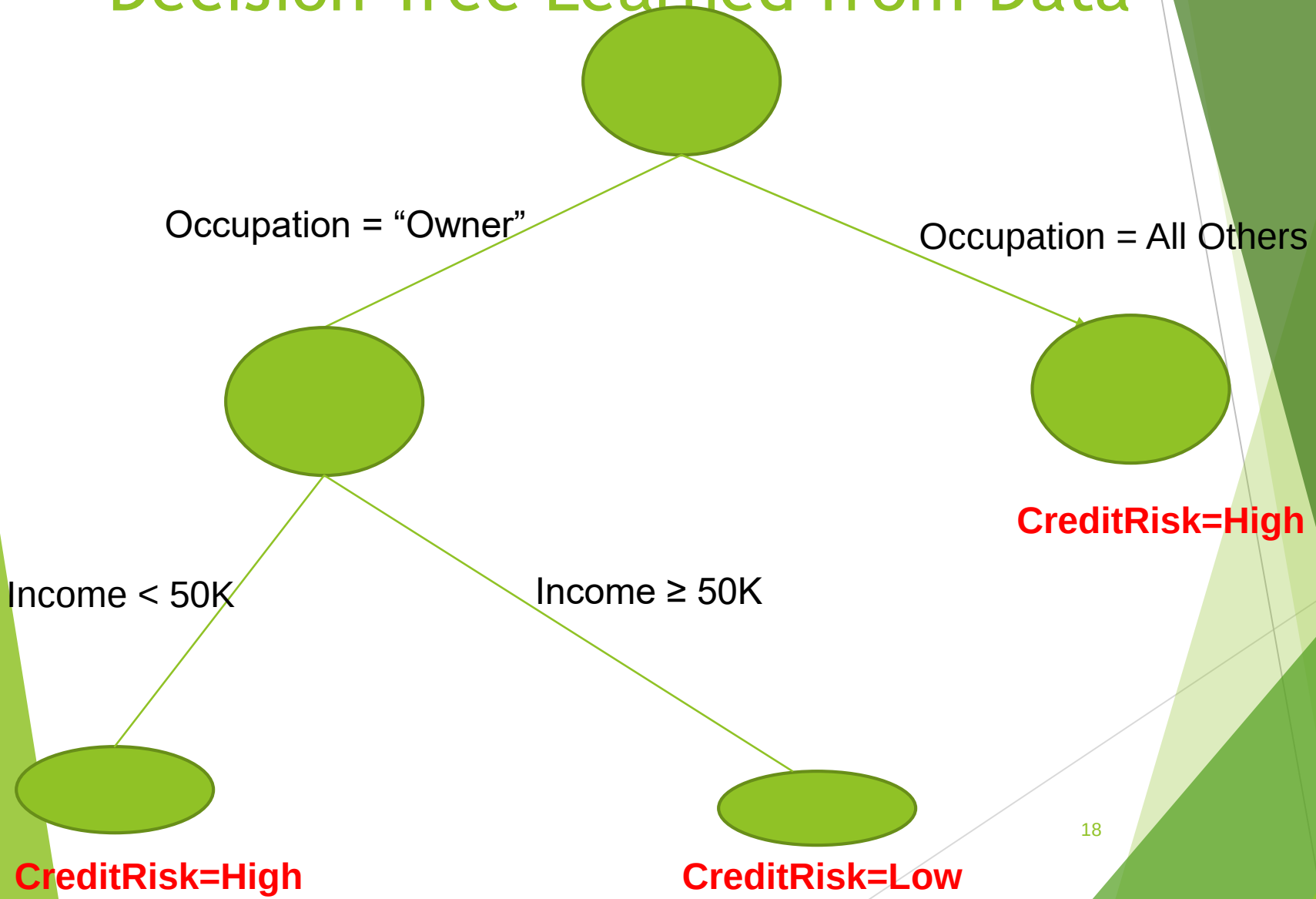
Addressing the Churn Problem with Tree Induction

- In the cell phone example we want to use tree induction to predict which new customers are going to churn
- How good are each of these variables individually?
- House value, number of leftover minutes, number of long calls per month have a higher information gain than the rest.
- Information Gain formula $IG = entropy(parent) - [p(c_1)xentropy(c_1) + p(c_2)xentropy(c_2) + \cdots]$
- Using the IG formula we can rank each feature by how good it is independently

An Example

AGE	INCOME	STATE	OCCUPATION	CREDIT RISK
45	150K	MA	Technology	Low
55	45K	FL	Real Estate	High
25	15K	NV	Student	High
….	….	….	….	…..
….	(MILLIONS	OF ROWS OF	CUSTOMER DATA)	…..
….	….	….	….	…..
72	180K	CA	Owner	Low

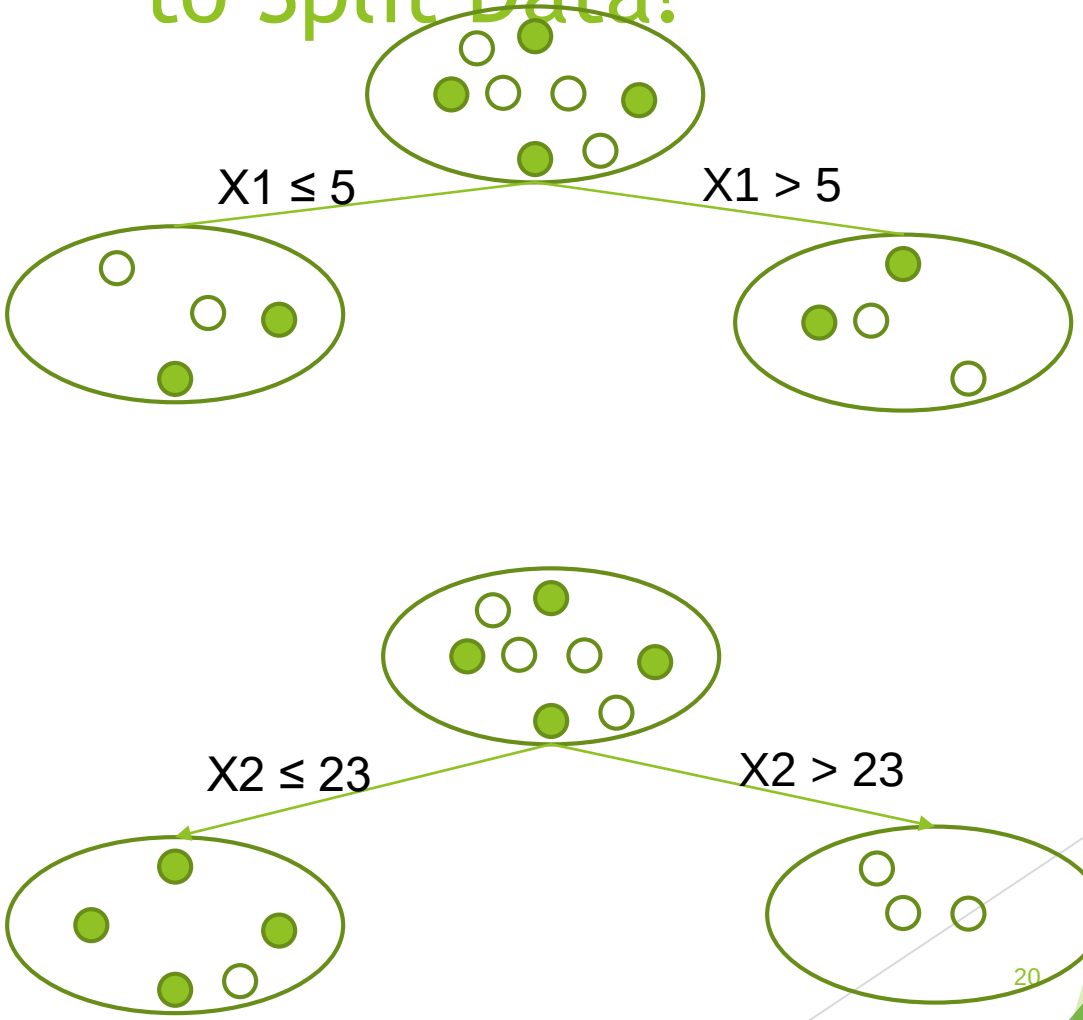
Decision Tree Learned from Data



Building & Using Decision Tree

- How was the tree built from the data?
  - We'll look at examples and generalize from them
- Once built, how do you use the tree for a new data set where you need to make predictions?
  - Easy! Imagine "dropping" new points from the top of the tree and see where it ends up…
  - A reason for the popularity of decision trees as a data mining method
- How do you evaluate the goodness of a decision tree
  - On test data (out of sample) where you know the actual answers
  - You can compute different evaluation metrics
    - Error rate or classification accuracy (binary predictions)
    - MAE or MSE for continuous values
    - Lift
    - Cost-sensitive measures

Main Problem in Tree Induction: How to Split Data?



Build a decision tree that cleanly separates "hi" and "lo" risk categories

Age	Risk
25	Hi
40	Lo
43	Lo
29	Hi
36	Hi
47	Lo
49	Lo
62	Lo

Age	Risk
25	Hi
40	Lo
43	Lo
29	Hi
36	Hi
47	Lo
49	Lo
62	Lo

Income = high

Age	Risk
25	Hi
29	Hi
36	Hi

Income = low

Age	Risk
40	Lo
43	Lo
47	Lo
49	Lo
62	Lo

Age	Risk
25	Hi
40	Lo
43	Lo
29	Hi
36	Hi
47	Lo
49	Lo
62	Lo

Age ≤ 36

Age > 36

Age	Risk
25	Hi
29	Hi
36	Hi

Age	Risk
40	Lo
43	Lo
47	Lo
49	Lo
62	Lo

Age	Risk
25	Hi
40	Lo
43	Lo
29	Hi
36	Hi
47	Lo
49	Lo
62	Lo

Age < 40

Age ≥ 40

Age	Risk
25	Hi
29	Hi
36	Hi

Age	Risk
40	Lo
43	Lo
47	Lo
49	Lo
62	Lo

Age	Risk
25	Hi
40	Lo
43	Lo
29	Hi
36	Hi
47	Lo
49	Lo
62	Lo

Age ≤ 38

Age > 38

Age	Risk
25	Hi
29	Hi
36	**Hi**

Age	Risk
40	**Lo**
43	Lo
47	Lo
49	Lo
62	Lo

Build a decision tree that cleanly separates “hi” and “lo” risk categories

Income	Risk
10K	Hi
20K	Lo
30K	Hi
40K	Hi
50k	Lo
60K	Lo
70K	Lo
80K	Lo

Income	Risk
10K	Hi
20K	Lo
30K	Hi
40K	Hi
50k	Lo
60K	Lo
70K	Lo
80K	Lo

Income ≤ 45K

Income	Risk
10K	Hi
20K	Lo
30K	Hi
40K	Hi

Income > 45K

Income	Risk
50k	Lo
60K	Lo
70K	Lo
80K	Lo

Income ≤ 25K

Income	Risk
10K	Hi
20K	Lo

Income > 25K

Income	Risk
30K	Hi
40K	Hi

Income ≤ 15K

Income	Risk
10K	Hi

Income > 15K

Income	Risk
20K	Lo

(Again ☺) Build a decision tree that cleanly separates “hi” and “lo” risk categories

Income	Age	Risk
30K	55	Hi
40K	60	Hi
50K	65	Hi
70K	55	Lo
80k	20	Hi
90K	60	Lo
100K	22	Hi

Income	Age	Risk
30K	55	Hi
40K	60	Hi
50K	65	Hi
70K	55	Lo
80k	20	Hi
90K	60	Lo
100K	22	Hi

Inc ≤ 60K

Income	Age	Risk
30K	55	Hi
40K	60	Hi
50K	65	Hi

Inc > 60K

Income	Age	Risk
70K	55	Lo
80k	20	Hi
90K	60	Lo
100K	22	Hi

Age ≤ 38

Income	Age	Risk
80k	20	Hi
100K	22	Hi

Age > 38

Income	Age	Risk
70K	55	Lo
90K	60	Lo

Terminology
"Node"

Income	Age	Risk
30K	55	Hi
40K	60	Hi
50K	65	Hi
70K	55	Lo
80k	20	Hi
90K	60	Lo
100K	22	Hi

Inc ≤ 60K

Income	Age	Risk
30K	55	Hi
40K	60	Hi
50K	65	Hi

Inc > 60K

Income	Age	Risk
70K	55	Lo
80k	20	Hi
90K	60	Lo
100K	22	Hi

Age ≤ 38

Income	Age	Risk
80k	20	Hi
100K	22	Hi

Age > 38

Income	Age	Risk
70K	55	Lo
90K	60	Lo

Income	Age	Risk
30K	55	Hi
40K	60	Hi
50K	65	Hi
70K	55	Lo
80k	20	Hi
90K	60	Lo
100K	22	Hi

Terminology
"Leaf Node"

Inc ≤ 60K

Income	Age	Risk
30K	55	Hi
40K	60	Hi
50K	65	Hi

Inc > 60K

Income	Age	Risk
70K	55	Lo
80k	20	Hi
90K	60	Lo
100K	22	Hi

Age ≤ 38

Income	Age	Risk
80k	20	Hi
100K	22	Hi

Age > 38

Income	Age	Risk
70K	55	Lo
90K	60	Lo

A Thought Exercise to bring out the Computational You

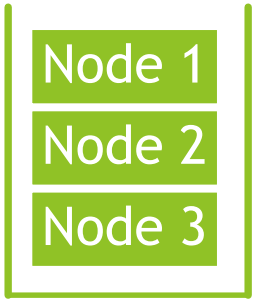
- How did you make each decision to split a node?
- What do you think happens when the number of variables to be considered is very large?
- When would you stop building the tree?
- Once you can precisely lay out these steps, you have designed an “algorithm” for learning trees.
  - Learning decision trees from data is called “tree induction”

The "Recursion" in "Recursive Partitioning"

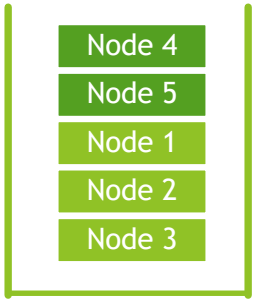
- The coolest idea in tree induction algorithms
- All you need is a systematic method for taking a given dataset and splitting it once.
- The idea of recursion means, all we do is pretend that each smaller dataset now is the actual dataset
  - Simply re-apply the same systematic rule again and again…

Digression: A “Stack” as a Data Structure

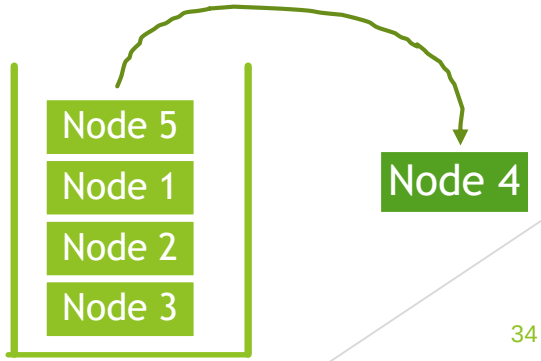
Think of a stack as an open box



Pushing means adding an object or objects to the top of the stack (box)

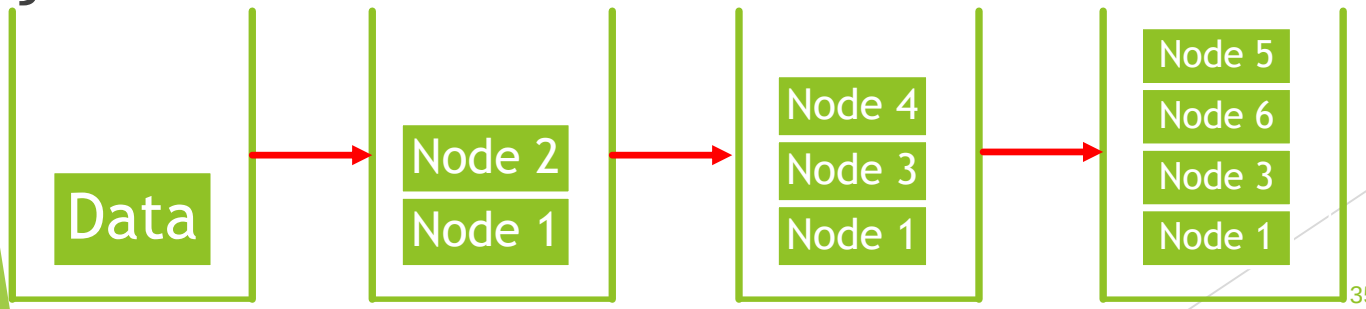


Popping means removing the topmost item in the stack (box)



The Basic Recursive Partitioning Idea

```
node = entire data

push(node)
While stack_not_empty  {
    Node = pop()
    Split(node, right, left)
    push(right), push(left)
}
```



The Basic Recursive Partitioning Idea

```
node = entire data

push(node)

While stack_not_empty  {
    Node = pop()

    Split(node, right, left)

    If not empty (right, left)
        push(right), push(left)

    else mark “node” as a leaf of the tree

}
```

The Basic Recursive Partitioning Idea

```
node = entire data

push(node)

While stack_not_empty  {
        Node = pop()
        Split(node, right, left)
        If not empty (right, left)
                push(right), push(left)
        else mark “node” as a leaf of the tree

}
```

Full tree = set of all leaves

Prune full tree and output a (possibly) smaller tree

Splitting

- At the heart of this algorithm is the concept of “splitting” a node
  - A good split is one that creates “good” nodes as children.
  - So -
- We need a measure of “goodness” of each single node.

- How about the accuracy within a node, is that a good idea? Let’s say 80% of data points in a node belong to one class, high risk, can we use that measure, 80%, as a measure of how good that node is?
  - Maybe, but does not generalize well to multiple classes. Say we’re predicting color (red, green, blue, yellow). The accuracy measure which works well in the binary case does not easily generalize.

Evaluating a node: Option 1

- Compute probability that two items chosen at random from this node are the same.
  - Sample with replacement
- Example:
  - A node with 5 zeros and 5 ones
    - Probability of choosing a zero * probability that the next item chosen will also be zero PLUS
    - Probability of choosing a one * probability that the next one chosen will also be a one
      - 0.5*0.5 +_0.5 * 0.5 = 0.5 (not very good)
  - Exercise: Compute this for nodes with (6, 4), (9, 1) and (10, 0) splits
- Scores are between 0 (bad) and 1 (great)

Using this to evaluate a split

- Total score for any split that sends P_L records to the left child and P_R records in the right child is:
  - P_L * score (left-node) + P_R * score (right-node)
- Example:
  - out of 100 records at the root node of a tree
  - split on age < 50; 20 records to the left and 80 records to the right say
  - Hypothetical score for this split
    - 0.2 * score (left node) + 0.8 * score (right node)
- How to pick the best split at each node in the tree?

Option 2: Gini Index

- Assume p_1, p_2, …p_k are the probabilities in your node for a classification problem with *k* classes

- Gini score =

$$\sum_{j=1}^{K} p_j(1 - p_j) = 1 - \sum_{j=1}^{K} p_j^2$$

Option 3: Evaluating a node: Entropy

- Entropy is a measure of disorder
  - a term originated in information theory
  - A measure of how many bits you will need to describe the state of a system
- For a given node, entropy is
  - Σ -p $\log_2$p, summed over all individual probabilities of each class
- Example:
  - If you had a binary outcome, and there were 3 zeros and 7 ones in a "node", the entropy is:
    - -0.3 * log(0.3) – 0.7*log(0.7)
- Entropy values range from zero (great) to higher values (bad).

Evaluating splits when the target variable is numeric

- What if risk was a number from 0-1 instead of just hi/low?
  - Trees can still be built, but you need a different measure to evaluate a node and a split again
- How to evaluate a node?
- Using “variance” as the criterion
- Split score = P_L * variance(left-node) + P_R * variance (right-node)
- Choose a split to have the smallest split score

Revisiting the Pseudo code: Splitting process at each stage

Split(node, left, right) works as follows

Consider all the input variables one at a time

- For each continuous variable consider all binary splits of the form var < value
- For each categorical variable consider all binary splits of the form
  - color = {blue} and color = {green, yellow, orange}
  - Some binary splits can take all possible subsets into account and split like
    - Color = {blue, green} and color = {yellow, orange}

Choose the single best split among all these options

The Basic Recursive Partitioning Idea

```
node = entire data

push(node)

While stack_not_empty  {
    Node = pop()
    Split(node, right, left)
    If not empty (right, left)
        push(right), push(left)
    else mark "node" as a leaf of the tree

}
```

When to stop splitting a node
i.e. return "empty" children?

Stopping rules

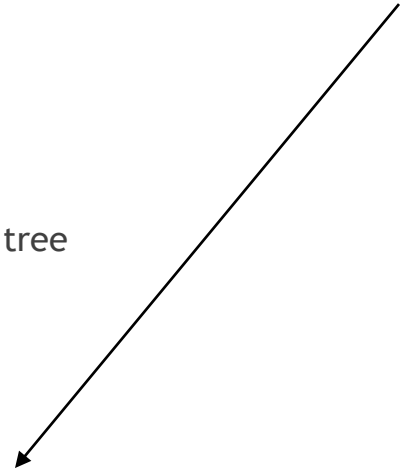
- Node size is too small
- No additional split results in any improvement
- 100% accuracy reached already

The Basic Recursive Partitioning Idea

```
node = entire data

push(node)

While stack_not_empty  {
        Node = pop()
        Split(node, right, left)
        If not empty (right, left)
                push(right), push(left)
        else mark “node” as a leaf of the tree

}

Full tree = set of all leaves

Prune full tree and output a (possibly) smaller tree
```

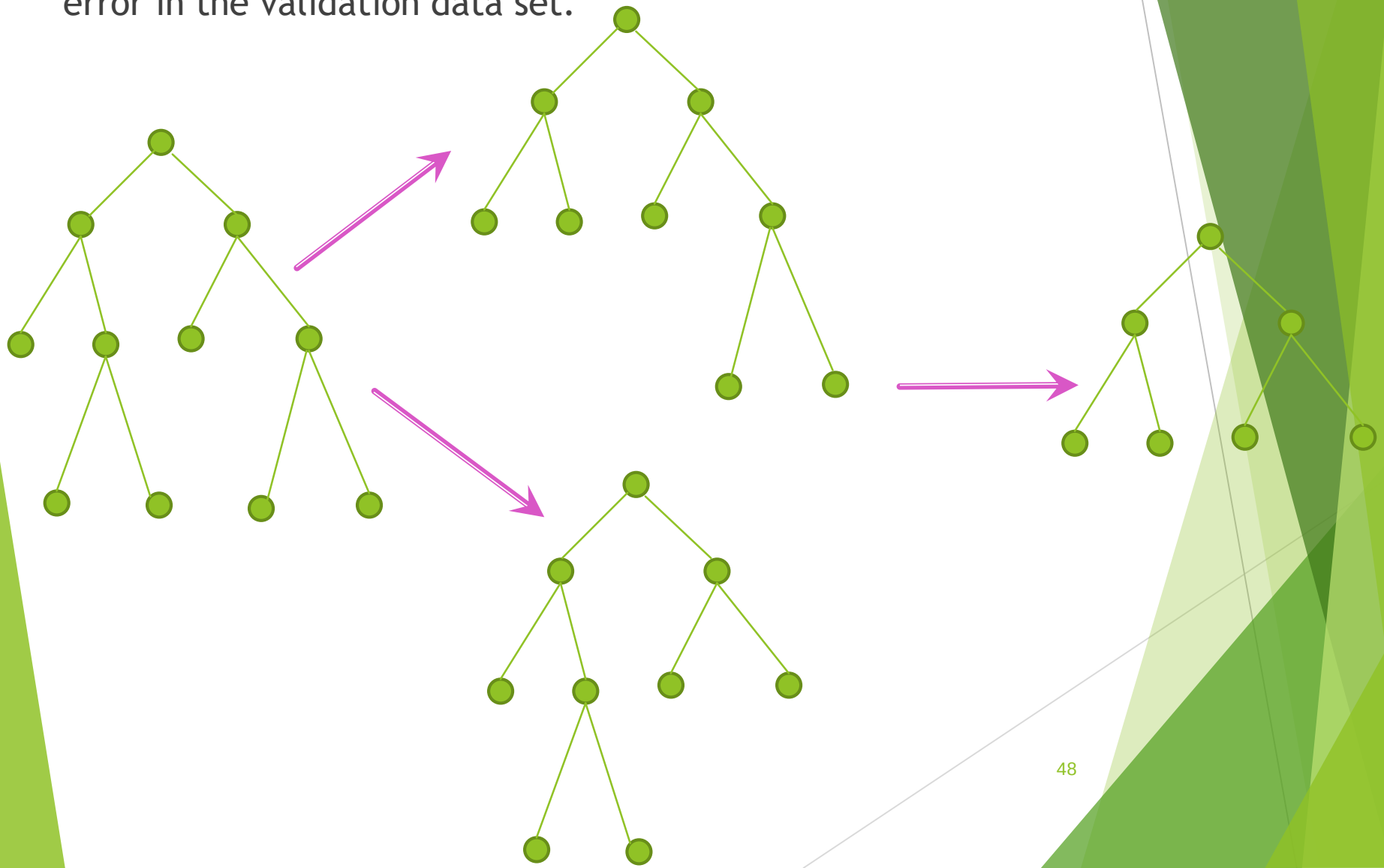
What you have now is one large tree that needs to be pruned

WHY PRUNE?

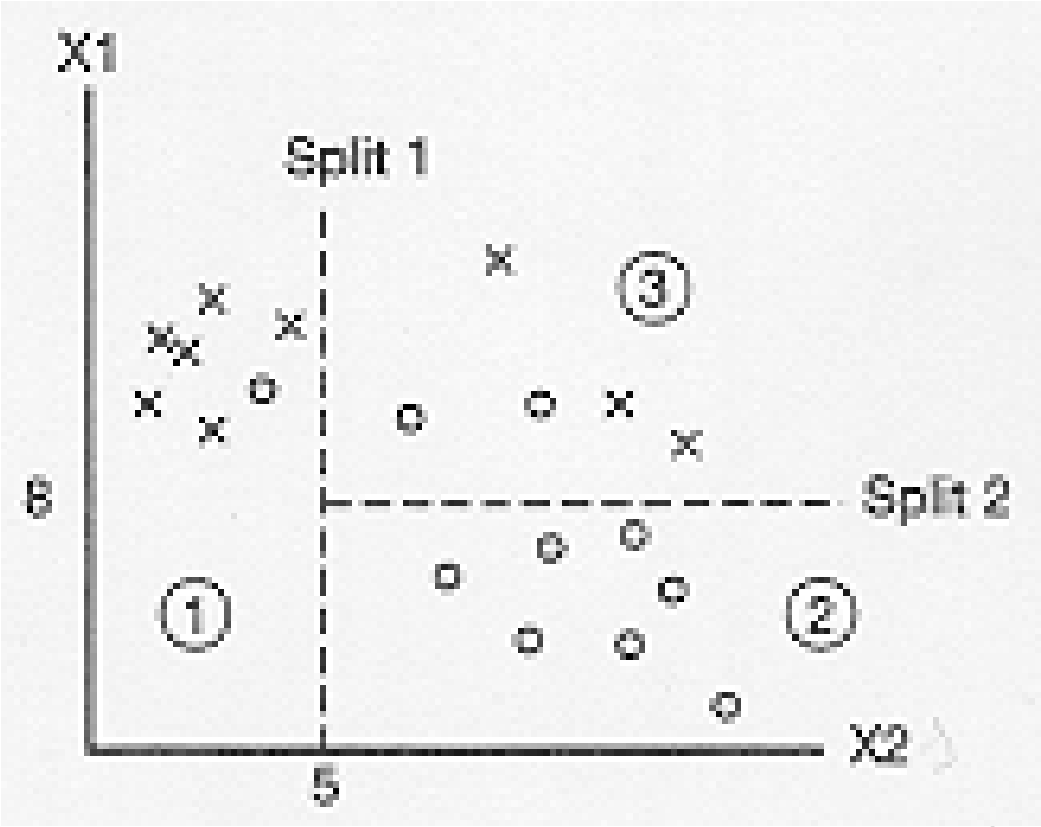


Pruning the Large Tree:

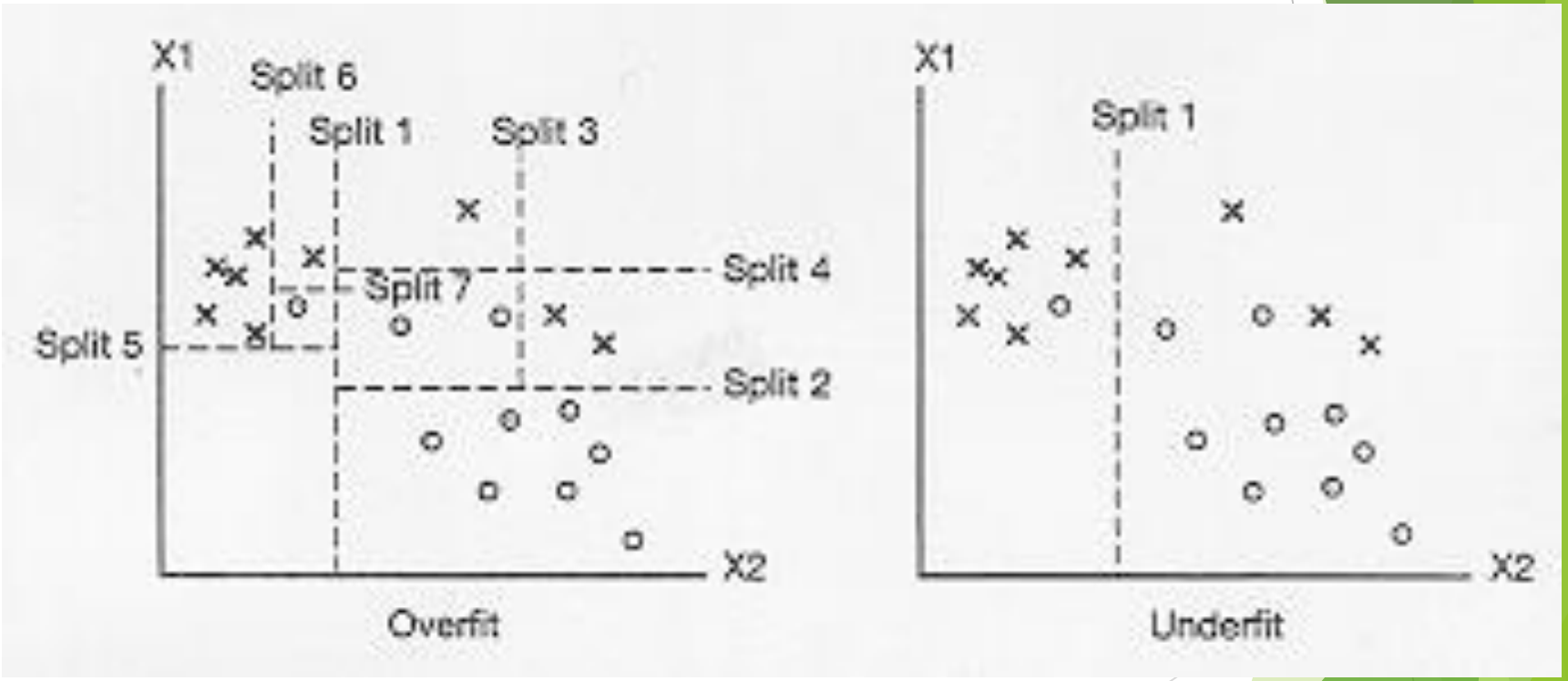
- Create multiple candidate sub-trees and pick the one that has the best error in the validation data set.



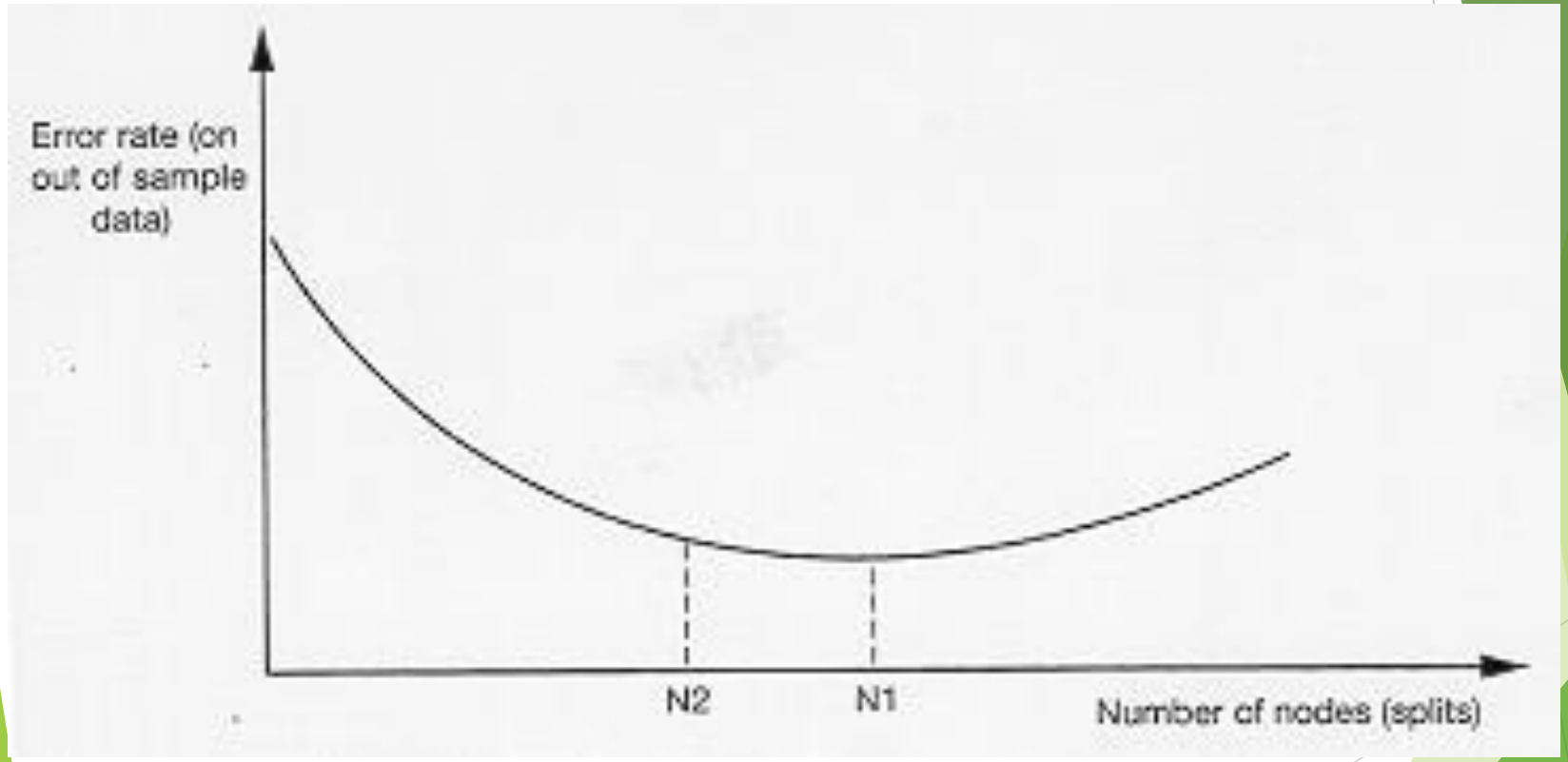
Thinking of splits as Drawing Lines to cut the data



Oversplitting and Undersplitting: Overfitting and Underfitting



Typical error rate on validation data as size of tree grows



Questions to Ponder

- What kind of data will decision trees do well in?
  - What kinds of data might trees do poorly in?
- What are some pros and cons of decision trees?
- Problems with the data: Can these methods be tweaked to handle:
  - Missing values in the data
  - Outliers -- How sensitive is recursive partitioning to outliers?
- Will transformations such as log(price) help? What types of transformations are not likely to be useful?
- How can we combine the benefits of recursive partitioning with abilities to build strong local models?

Decision Tree Advantages

1. Easy to understand and often very accurate
2. Learns non-linear relationships
3. Map nicely to a set of business rules
4. Make no prior assumptions about the data
5. Able to process both numerical and categorical data

Decision Tree Disadvantages

1. Decision tree algorithms can be unstable

   Different trees on slightly different data

   How to exploit this problem?

2. Trees created can be very deep and complex

3. Ability to model global linear relationships

Summary

- What decision trees are
- How tree induction algorithms work
- Concept of recursion
- Concept of the goodness of a node and goodness of a split
- Concept of overfitting and how pruning is used along with the validation data set to pick the final “best” tree