Data Science

Deriving Knowledge from Data at Scale

Wee Hyong Tok, Ph.D.

October 22nd, 2018

Common Types of Modeling Tasks

- Classification: categorization into types
- Scoring: predicting or estimating a value
- Ranking: Ordering items by affinities
- Clustering: grouping similar items
- Relations/correlations: determine potential causes of effects
- Characterization: report generation

Refresher on Supervised vs. Unsupervised Learning

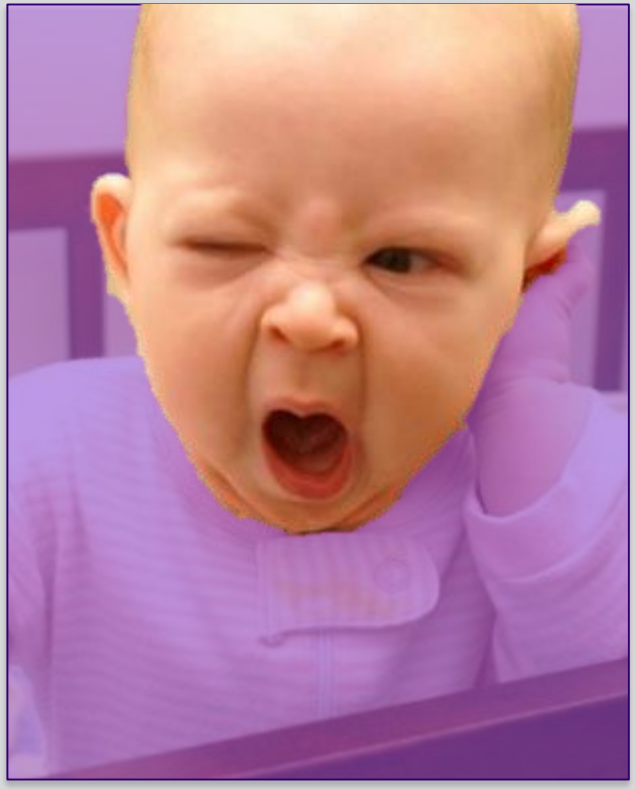
Supervised Learning

- **Definition**: Uses ‘labeled’ training data
- Each set of input features (could be single value or a vector) is accompanied by the “correct” classification or “signal”.

Supervised Learning Examples

- **Fraud Detection**: when we have examples in the data where fraud = True/False is known
- **Patient Readmission**: when we know which patients were readmitted to the hospital
- **Recommendation Systems**: when we know which items were presented to customers resulted in added to the cart or purchased

Supervised Learning for Face Recognition



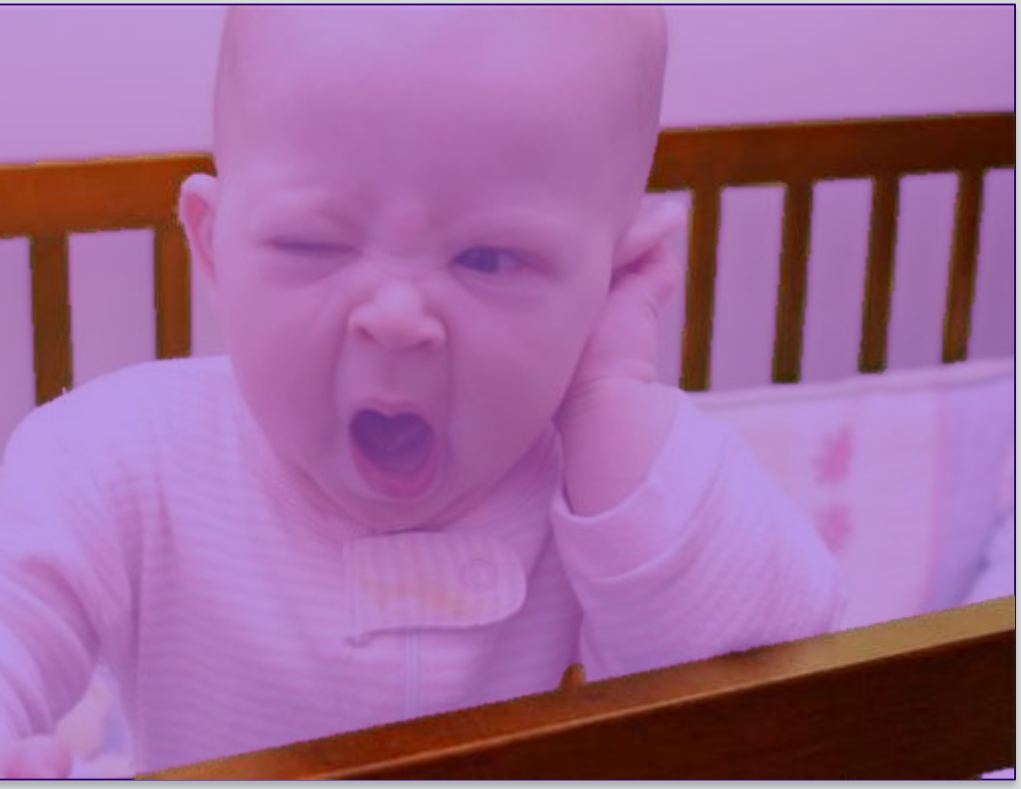
For example: a set of pixels in an image that represent a face. The non-face pixels are “False” and the face pixels are “True”

All of the characteristics of each pixel in this image are represented as input vector; the *Face* or *Not-Face* value is the “signal”

Unsupervised Learning

- **Definition**: unlabeled data is used to create a model *inferred* by the input vector; no correct classification is present.

Unsupervised Learning in Image Recognition



Left to it's own devices, an image recognition system might first look for sharp edges.

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Decision Trees

Classification or Categorization

- What is a decision tree?
- When to use a decision tree (and when not to)
- How to create and tune them

Decision Trees: Definition and Popularity

- Decision trees are a commonly used type of supervised learning for *classification*
- Decision Trees enable automation of grouping “like” things using mathematical relationships between those classes and the input variables
- Decision trees are easy to interpret, understand, and explain
- They are “white box” in that each decision point (branch in the tree) clearly indicates not only the decision, but the path leading to each branch

Types of Decision Trees

- CART – Most commonly used
  - **Classification Trees**: where the target variable is categorical and the tree is used to identify the "class" of a target variable
  - **Regression Trees**: where target variable is continuous and the terminal nodes of the tree contain the predicted output variable values

NOTE: Both of these types can take categorical and continuous input variables

What about these?

Covered in upcoming lectures; in brief

- ID3 – The earliest decision tree algorithm (circa 1975)
- C4.5 and 5 – a faster “boosted” successor to ID3 that uses an ensemble (model on top of a model) to improve accuracy
- Gradient Boosted Trees – sequentially adding predictors to an ensemble each one correcting the predecessor
- Random Forest – trains on a random subset of features and then averages out their predictions

Which is best?

It’s best to try several techniques and see which best suits the domain and problem space. Don’t limit yourself to one.

Vocabulary

Root Node: Top of the tree—represents the entire dataset

Splitting (arc/edge): Process of dividing a node into two or more sub-nodes

Decision Node: A test on a single attribute that results in a split

Leaf/Terminal Node: Nodes that are not split

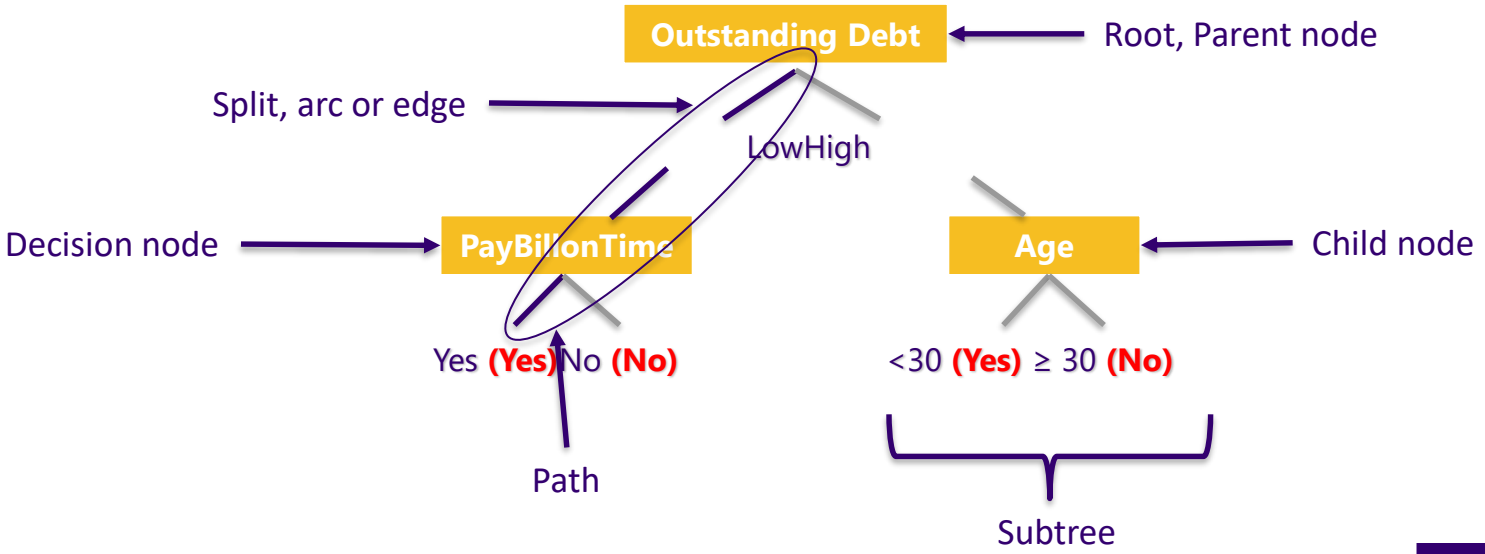
Pruning: When we remove sub-nodes of a decision node, this process is called pruning (ant. Splitting)

Branch/Sub-Tree: A section of entire tree is called branch or sub-tree

Parent/Child Nodes: A node, which is divided into sub-nodes is called a parent node; sub-nodes are the children of parent nodes

Path: a disjunction test that traverses the tree from root to a decision node

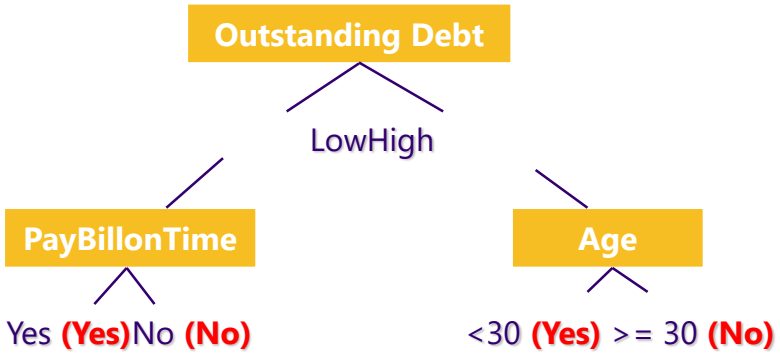
Decision trees classify instances by sorting them from the root of the tree to some leaf nodes (which provides the **classification** of the instance)

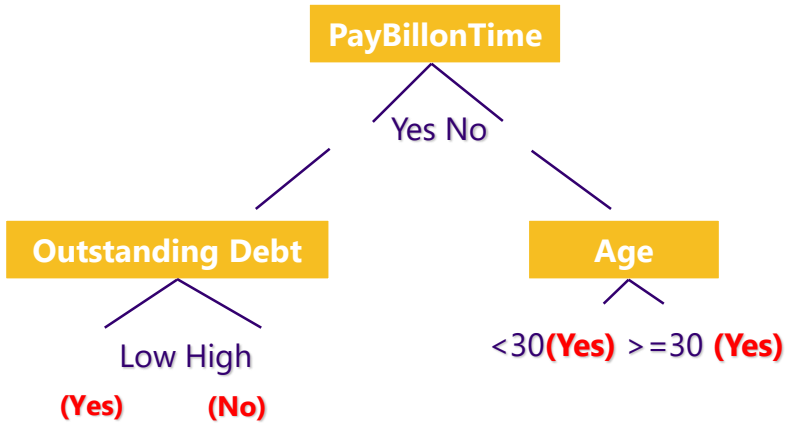


How do we split?

Which attribute should be used first?

- Outstanding Debt?
- Pay bill on time?
- Age?





Determining the Split Attribute

- Each **attribute** (feature) is assigned a score
- The attribute which maximizes the score during each iteration is chosen as the **Split** attribute
- Different methods to compute the score. E.g.,
  - Gini Impurity (node homogeneity)
  - Chi-Square (statistical significance)
  - Information Gain (entropy)

Entropy

Binary Classification - Given a collection S, with (p) and failure (q) example of a target

$$\boldsymbol{Entropy(S) = -p \log_2 p \; - q \log_2 q}$$

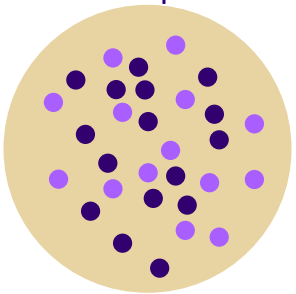
The formula: "–" (minus) percentage of success * (times) the $\log_2$ of the percentage of success subtracted from the percentage of failure * the $\log_2$ of the percentage of failure.

Log allows us to calculate based on two "states" at a time

Understanding Entropy

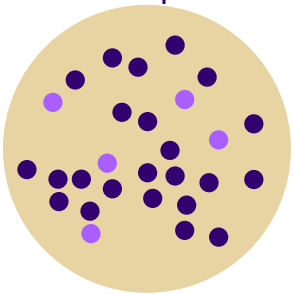
Consider the splits based on the three nodes below:

Group A



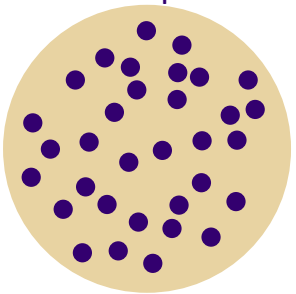
Entropy = 1
-15/30*log(15/30) – 15/30*log(15/30)

Group B



Entropy = 0.211632
-25/30*log(25/30) – 5/30*log(5/30)

Group C



Entropy = 0

Entropy is the degree of disorganization. If the sample is completely homogeneous, then the entropy is zero and if the sample is an equally divided (50% – 50%), it has entropy of one.

Let’s build a theoretical decision tree

Example Training Data: Play Tennis

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Start With the Entropy of Play vs. Not Play

- *S* is a collection of 14 examples
- 9 are positive examples (**playing**), 5 are negative examples (**not playing**)
  Notation: [9+, 5-]
- Entropy(*S*)
  $= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$
  $= 0.940$

Information Gain

Information Gain – is the expected reduction in entropy if attribute was chosen as the splitting attribute

= Entropy_before - Entropy_after

= 0.940 - Entropy_after_split

Four Attributes to Consider

Outlook, Temperature, Humidity and Wind

TEMPERATURE	Play = Yes	Play = No	
Hot	2/4	2/4	4/14
Mild	4/6	2/6	6/14
Cool	3/4	1/4	4/14

OUTLOOK	Play = Yes	Play = No	
Sunny	2/5	3/5	5/14
Overcast	4/4	0	4/14
Rain	3/5	2/5	5/14

HUMIDITY	Play = Yes	Play = No	
High	3/7	4/7	7/14
Normal	6/7	1/7	7/14

WIND	Play = Yes	Play = No	
Strong	3/9	3/5	6/14
Weak	6/9	2/5	8/14

Choosing the Best Split Variable

- Humidity – high

  $= \left(\frac{3}{7}\log_2\frac{3}{7}\right) - \left(\frac{4}{7}\log_2\frac{4}{7}\right)$

  $= .985$

- Humidity – normal

  $= \left(\frac{6}{7}\log_2\frac{6}{7}\right) - \left(\frac{1}{7}\log_2\frac{1}{7}\right)$

  $= .592$

- Gain = S, Humidity [+7, -7]

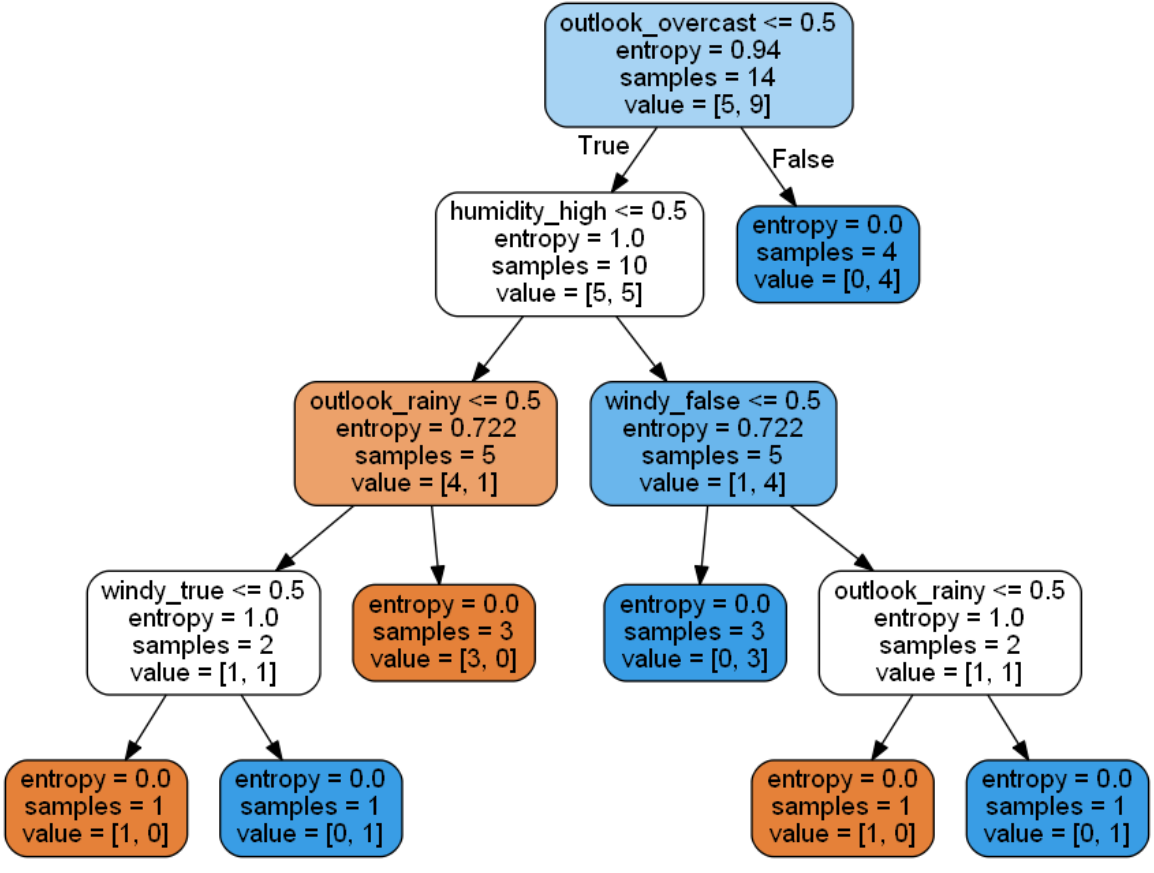
  $= .940 - \frac{7}{14}(.985) - \frac{7}{14}(.592)$

  $= .151$

HUMIDITY	Play = Yes	Play = No	*S*, Humid
High	3/7	4/7	7/14
Normal	6/7	1/7	7/14

Rinse and repeat for every variable and state

Next Split – based on the previous split…



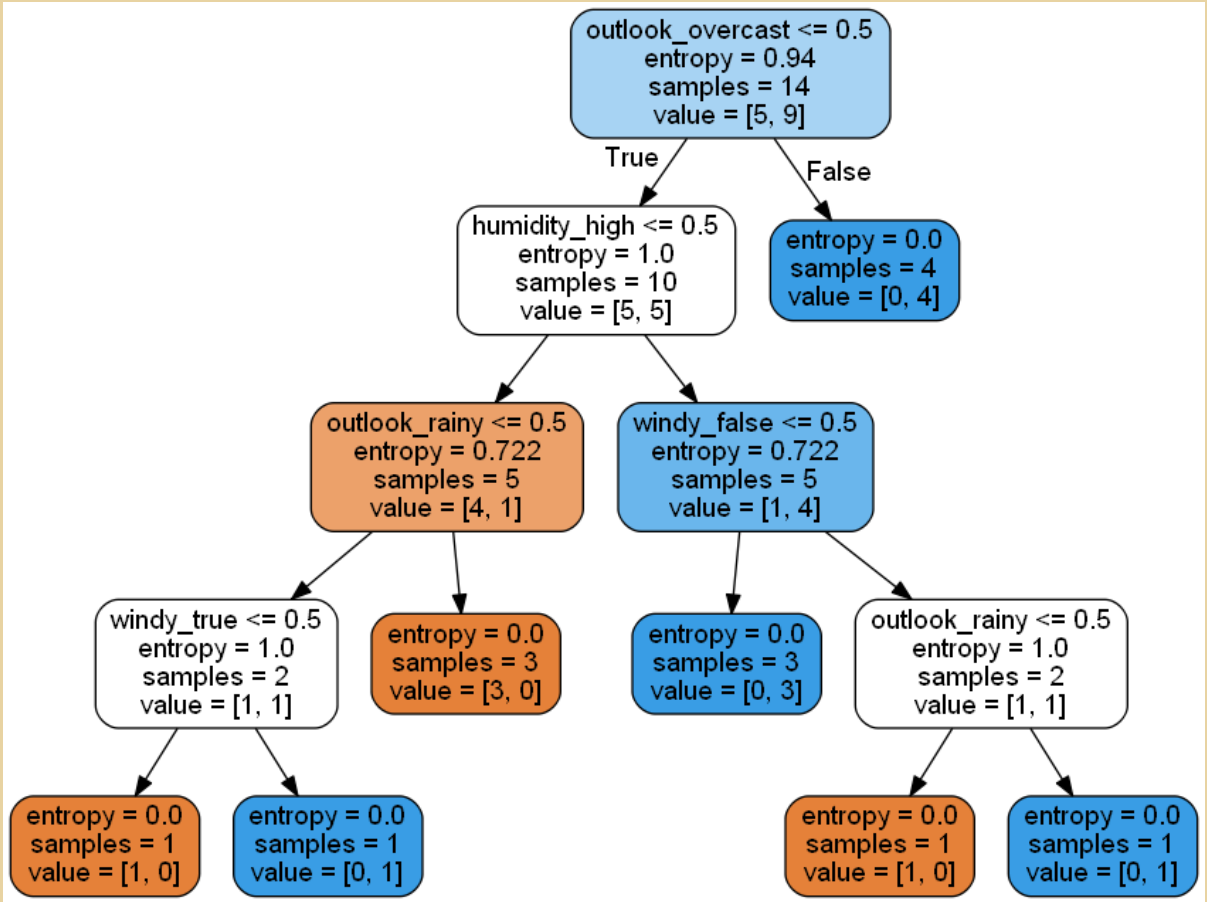
Outlook = Overcast

Humidity = High

Outlook = rainy/Windy = false

Windy = true/Outlook = rainy

So of the input variables – which had the lowest predictive value?

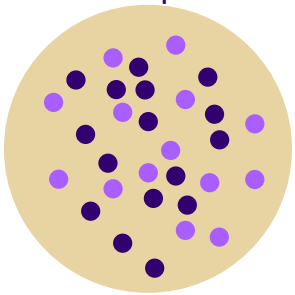


TEMP	Play = Yes	Play = No	
Hot	2/4	2/4	4/14
Mild	4/6	2/6	6/14
Cool	3/4	1/4	4/14

Understanding Gini Impurity (Default)

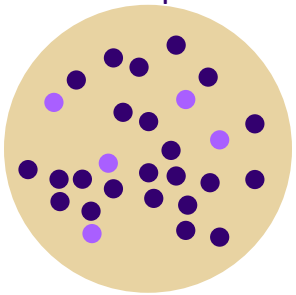
Consider the splits based on the three nodes below:

Group A



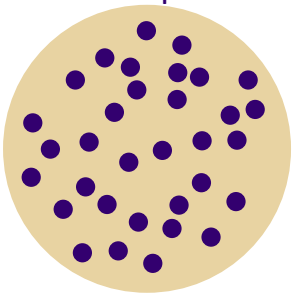
Gini = 1 – 15/50 or 50%

Group B



Gini = 1 – 5/30 or 17%

Group C



Gini = 0

The Gini Coefficient is the degree of node impurity.

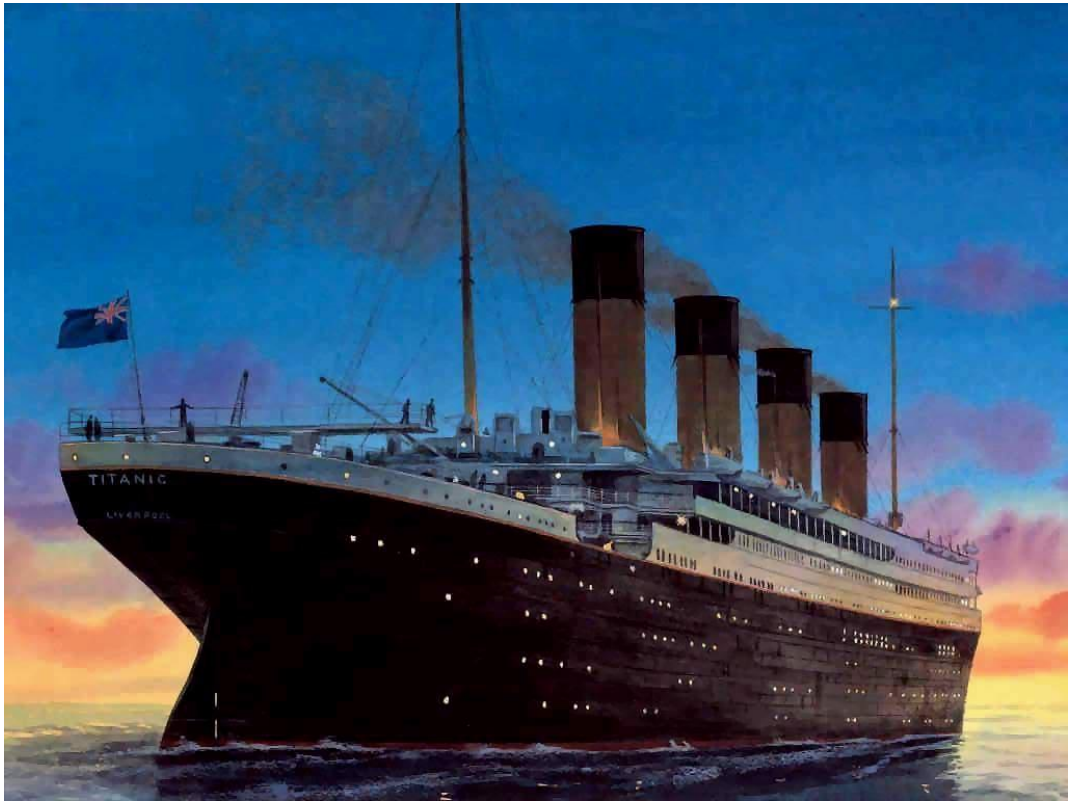
$$G_i = 1 - \sum_{k=1}^{n} {P_{i,k}}^2$$

Gini vs. Entropy

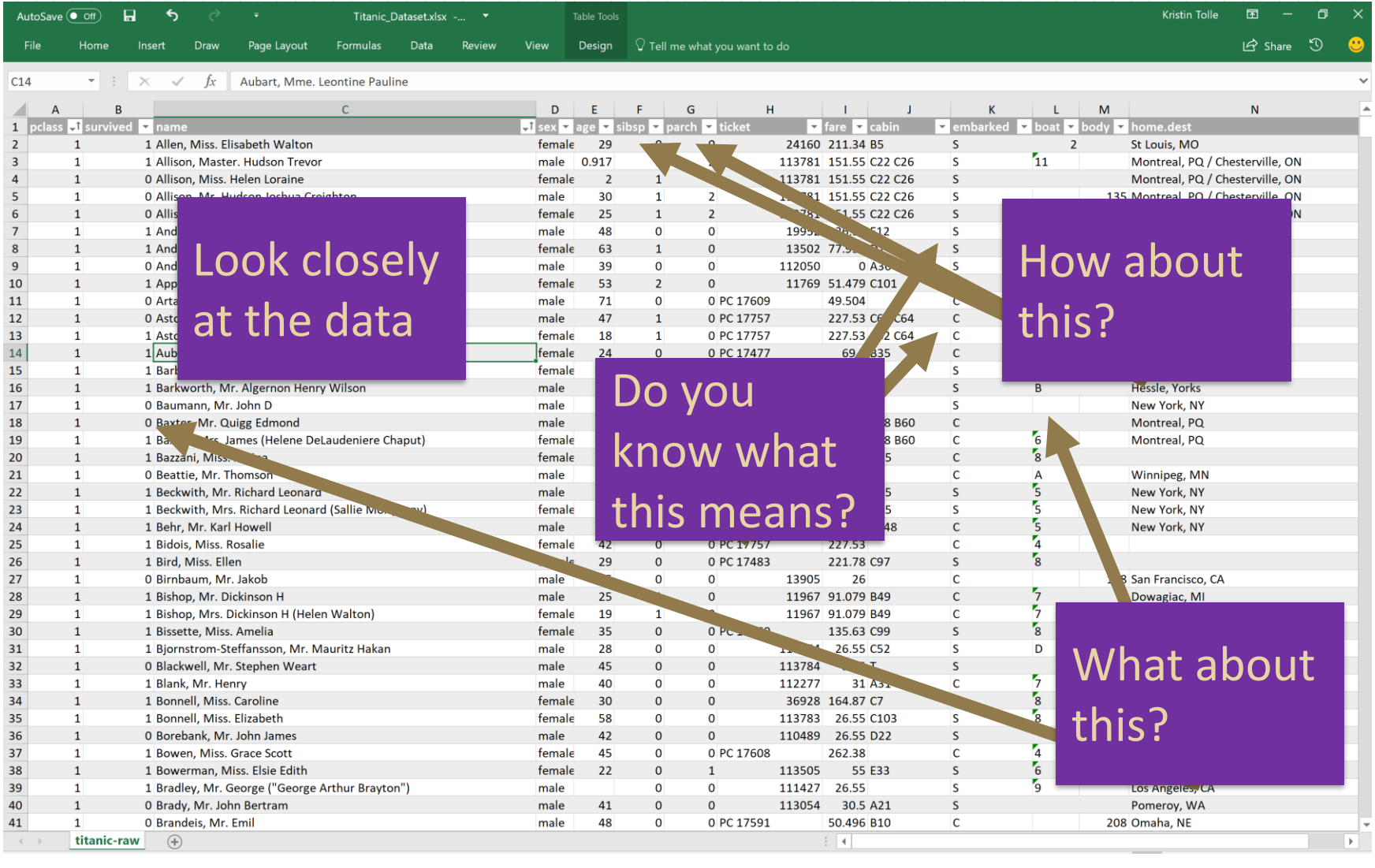
- Usually create similar trees
- When the differ:
  - Gini tends to be faster (greedier) and isolates the most frequently occurring class in one side of the tree
  - Entropy tends to be more *balanced*
- It is worth testing BOTH

In-class Lab 1:

Example: Tragedy of the Titanic



Women and children first?


Look closely at the data
How about this?
Do you know what this means?
What about this?

Start with a Data Definition

Feature	Data Description	Variable Type
survival	Survived (0 = no; 1 = yes)	Dependent variable
pclass	Ticket class 1 = 1st, 2 = 2nd, 3 = 3rd	Independent variable
name	Passenger's name as given	Relevance?
sex	Gender: Male or Female	Independent variable
age	Age	Independent variable
sibsp	Count of siblings and/or spouse aboard	Independent variable
parch	Count of parents or children aboard	Independent variable
ticket	Ticket number	Relevance?
fare	Passenger Fare	Independent variable
cabin	Cabin number	Relevance?
embarked	Port of departure (c = Cherbourg, q=Queenstown, s=Southhampton	Relevance?
boat	Rescue boat number	Relevance?
home	Home city of the passenger and ultimate (assumed) destination	Relevance?
…		

Choose wisely: More variables in the model can negatively affect compute time and potentially accuracy

First Steps… Look for Problems and Predictors

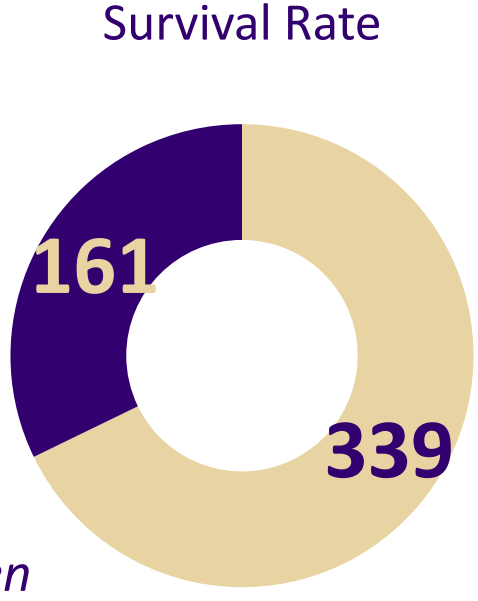
	pclass	survived	name	sex	age	sibsp	parch
261	1	1	Seward, Mr. Frederic Kimber	male	34	0	0
262	1	1	Shutes, Miss. Elizabeth W	female	40	0	0
263	1	1	Silverthorne, Mr. Spencer Victor	male	35	0	0
264	1	0	Silvey, Mr. William Baird	male	50	1	0
265	1	1	Silvey, Mrs. William Baird (Alice Munger)	female	39	1	0
266	1	1	Simonius-Blumer, Col. Oberst Alfons	male	56	0	0
267	1	1	Sloper, Mr. William Thompson	male	28	0	0
268	1	0	Smart, Mr. John Montgomery	male	56	0	0
269	1	0	Smith, Mr. James Clinch	male	56	0	0
270	1	0	Smith, Mr. Lucien Philip	male	24	1	0
271	1	0	Smith, Mr. Richard William	male		0	0
272	1	1	Smith, Mrs. Lucien Philip (Mary Eloise Hughes)	female	18	1	0
273	1	1	Snyder, Mr. John Pillsbury	male	24	1	0
274	1	1	Snyder, Mrs. John Pillsbury (Nelle Stevenson)	female	23	1	0
275	1	1	Spedden, Master. Robert Douglas	male	6	0	2
276	1	1	Spedden, Mr. Frederic Oakley	male	45	1	1
277	1	1	Spedden, Mrs. Frederic Oakley (Margaretta Corning Stone)	female	40	1	1
278	1	0	Spencer, Mr. William Augustus	male	57	1	0
279	1	1	Spencer, Mrs. William Augustus (Marie Eugenie)	female		1	0
280	1	1	Stahelin-Maeglin, Dr. Max	male	32	0	0
281	1	0	Stead, Mr. William Thomas	male	62	0	0
282	1	1	Stengel, Mr. Charles Emil Henry	male	54	1	0
283	1	1	Stengel, Mrs. Charles Emil Henry (Annie May Morris)	female	43	1	0
284	1	1	Stephenson, Mrs. Walter Bertram (Martha Eustis)	female	52	1	0
285	1	0	Stewart, Mr. Albert A	male		0	0
286	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62	0	0
287	1	0	Straus, Mr. Isidor	male	67	1	0
288	1	0	Straus, Mrs. Isidor (Rosalie Ida Blun)	female	63	1	0
289	1	0	Sutton, Mr. Frederick	male	61	0	0
290	1	1	Swift, Mrs. Frederick Joel (Margaret Welles Barron)	female	48	0	0
291	1	1	Taussig, Miss. Ruth	female	18	0	2
292	1	0	Taussig, Mr. Emil	male	52	1	1
293	1	1	Taussig, Mrs. Emil (Tillie Mandelbaum)	female	39	1	1

Look for Missing Values

e.g., Age…

Make sure you understand column heading

Survival by Gender (Passengers)



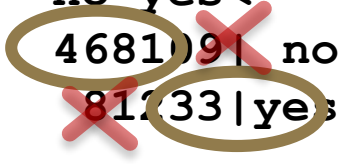
N = 1324
12% of men
26% of women

More Women Survived

IF sex='female' THEN survive=yes
ELSE IF sex='male' THEN survive = no

```
confusion matrix

no yes<-- classified as
468 109 | no
 81 233 | yes
```



(468 + 233) / (468+109+81+233) = 79% correct (and 21% incorrect)

Not bad!!

Survival Rate by Gender and Class

Regardless of class, more women survived;

However…

From this view it seems that class mattered more than gender

IF pclass=‘1’ THEN survive=yes
ELSE IF pclass=‘2’ THEN survive=yes
ELSE IF pclass=‘3’ THEN survive=no

```
confusion matrix

no yes<-- classified as
372 119| no
177 223| yes
```

(372 + 223) / (372+119+223+177) = 67% correct (and 33% incorrect)

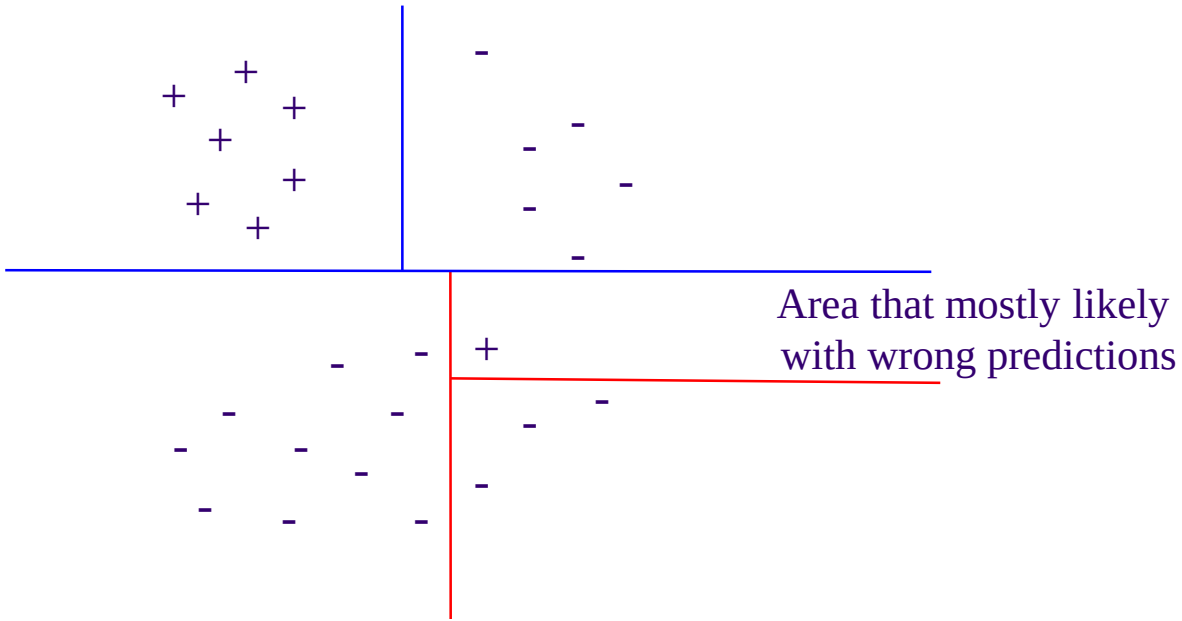
A little worse

In-class Lab 2

Decision Trees and Overfitting

Reasons for Overfitting:

A small number of instances are associated with leaf nodes: In this case it is possible that for coincidental regularities to occur that are unrelatedto the actual borders

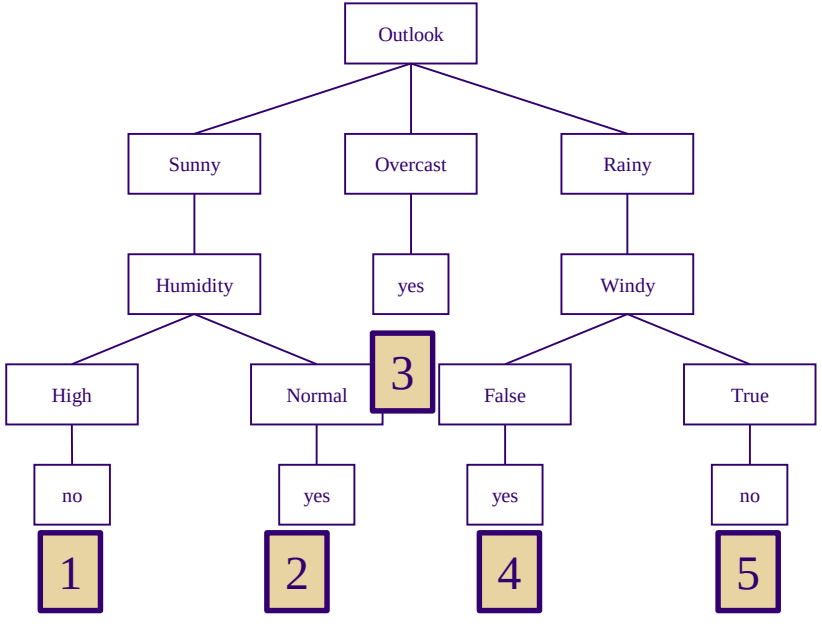


Approaches to Avoid Overfitting:

- **Pre-pruning:** stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- **Post-pruning:** Allow the tree to overfit the data, and then post-prune the tree.

Pre-Pruning:

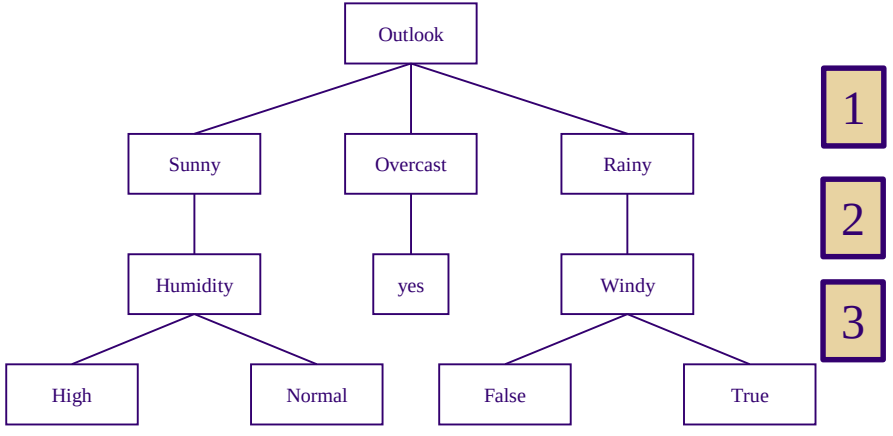
- It is challenging to determine when to stop growing the tree
- One thing to try is limited the number of leaf nodes get less than m training instances.
- max_leaf_nodes= 5



max_leaf_nodes : int or None, optional (default=None)

Pre-Pruning:

- You can also limit the depth (number of decision levels) of the tree
- E. g., max_depth = *3*



max_depth : int or None, optional (default=None)

Reduced-Error Pruning (Sub-tree replacement)

- A **validation set** is a set of instances used to evaluate the utility of nodes in decision trees. The validation set has to be chosen so that it is will not suffer from same errors or fluctuations as the set used for decision-tree training.
- Usually before pruning the training data is split *randomly* into a growing set and a validation set.

Create a Validation Set

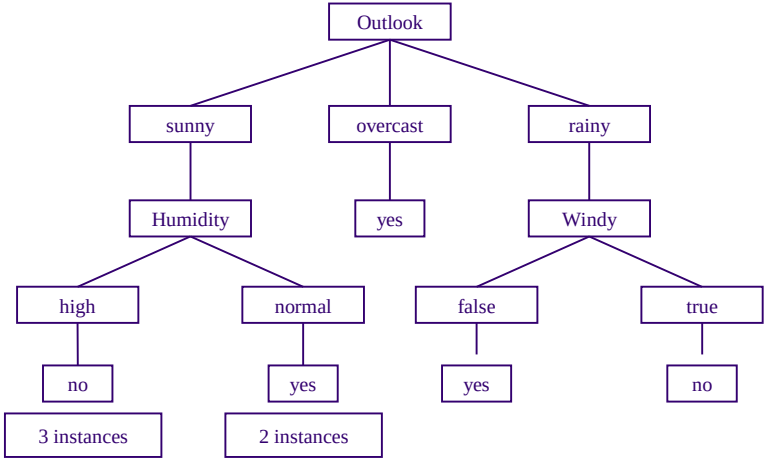
```
train, validate, test = np.split(df.sample(frac=1), /
   [int(.6*len(df)), int(.8*len(df))])
```

Gives us:

- 60% - train set
- 20% - validation set
- 20% - test set

Reduced-Error Pruning

- Split data into growing and validation sets.
- Pruning a decision node *d* consists of:
- removing the subtree rooted at *d*.
- making *d* a leaf node.
- assigning *d* the most common classification of the training instances associated with *d*.



Accuracy of the tree on the validation set is 90%.

Reduced-Error Pruning

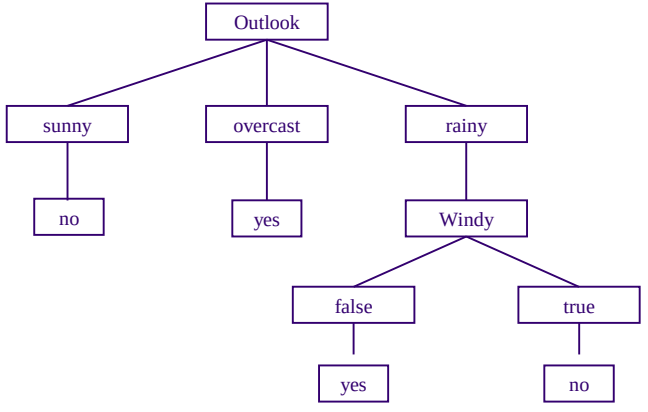
Split data into growing and validation sets.

Pruning a decision node *d* consists of:

1. removing the subtree rooted at *d*.
2. making *d*a leaf node.
3. assigning *d* the most common classification of the training instances associated with *d*.

Do until further pruning is harmful:

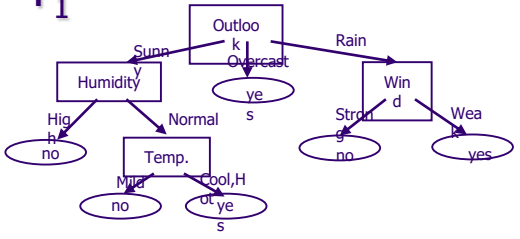
1. Evaluate impact on validation set of pruning each possible node (plus those below it).
2. Greedily remove the one that most improves validation set accuracy.



Accuracy of the tree on the validation set is 92.4%.

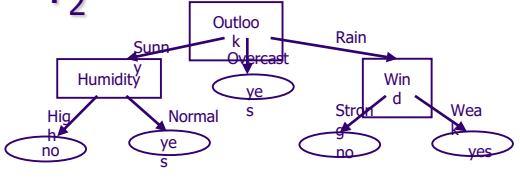
Reduced-Error Pruning – When do you stop?

T_1



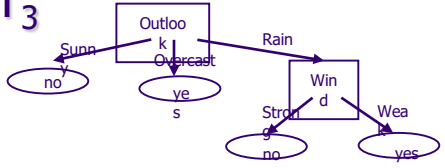
$Error_{GS}$=0%, $Error_{VS}$=10%

T_2



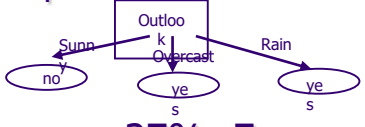
$Error_{GS}$=6%, $Error_{VS}$=8%

T_3



$Error_{GS}$=13%, $Error_{VS}$=15%

T_4

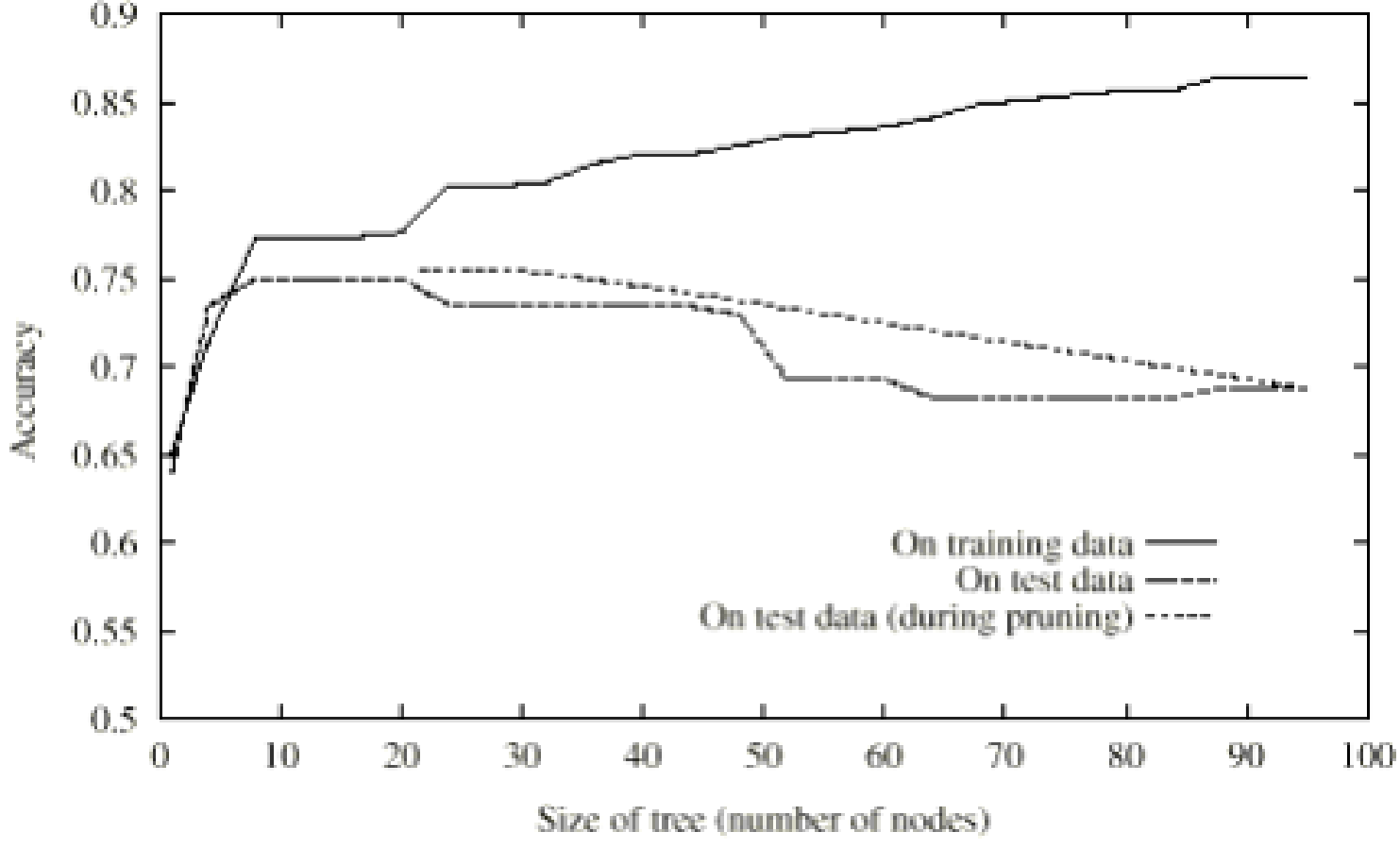


$Error_{GS}$=27%, $Error_{VS}$=25%

T_5

$Error_{GS}$=33%, $Error_{VS}$=35%

Reduced Error Pruning Example

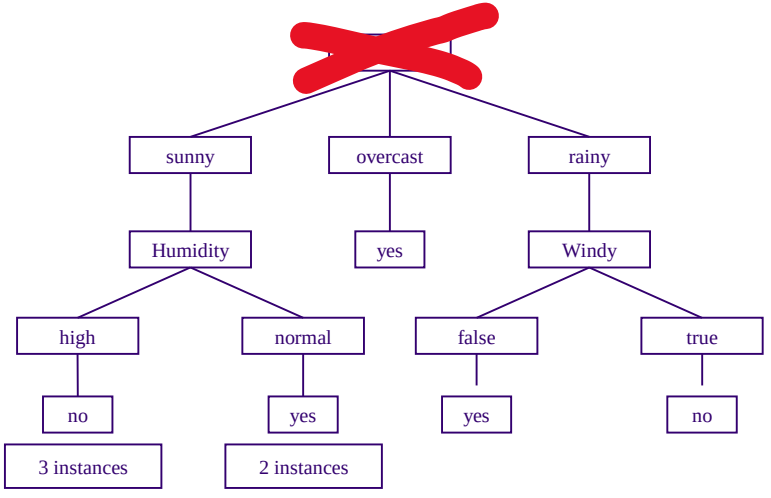


Reduced-Error Pruning – Sub-tree Raising

Split data into growing and validation sets.

Raising a sub-tree with root *d* consists of:

1. removing the sub-tree rooted atthe parent of *d*.
2. place *d* at the place of its parent.
3. Sort the training instancesassociated with the parent of *d* using the sub-tree with root ***d*** .



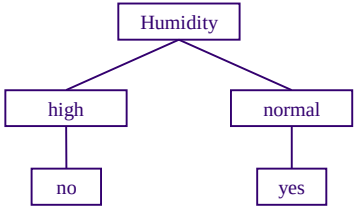
Accuracy of the tree on the validation set is 90%.

Reduced-Error Pruning – Sub-tree Raising

Split data into growing and validation sets.

Raising a sub-tree with root *d* consists of:

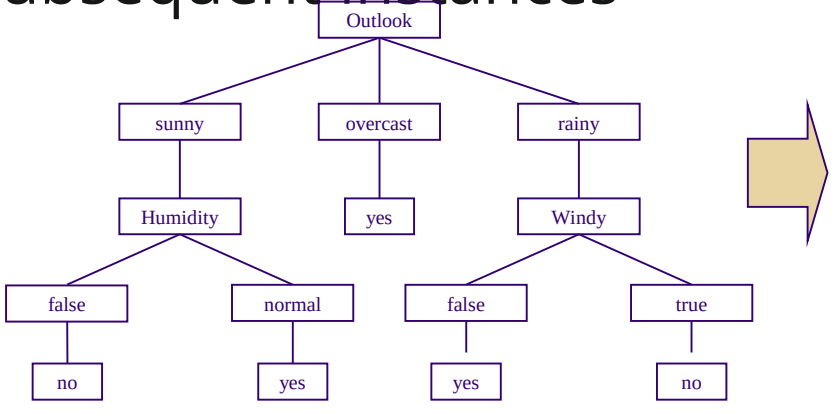
1. removing the sub-tree rooted atthe parent of *d*.
2. place *d* at the place of its parent.
3. Sort the training instancesassociated with the parent of *d* using the sub-tree with root ***d*** .



Accuracy of the tree on the validation set is 73%. So, No!

Rule Post Pruning

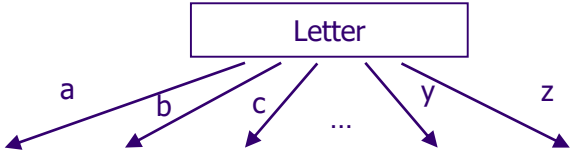
- Convert tree to equivalent set of rules
- Prune each rule independently of others
- Sort final rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances



IF (Outlook = Sunny) & (Humidity = High)
THEN PlayTennis = No
IF (Outlook = Sunny) & (Humidity = Normal)
THEN PlayTennis = Yes
……

Data Problems for Decision Trees

Variables with Many Values



Problem:

- Not good splits: they fragment the data too quickly, leaving insufficient data at the next level
- The reduction of impurity of such test is often high (example: split on the object id)

Several solutions:

- Change the splitting criterion to penalize variables with many values and threshold on impurity (min_impurity_split)
- Consider only binary splits (max_leaf_nodes=2)
- only consider splits with multiple values (min_samples_leaf)

Handling Missing Values

- If node n tests variable X_i, assign most common value of X_i among other instances sorted to node n.
- If node n tests variable X_i, assign a probability to each of possible values of X_i. These probabilities are estimated based on the observed frequencies of the values of X_i. These probabilities are used in the information gain measure (via info gain).

Summary

Strengths of Decision Trees

- Generates understandable rules
- Performs classification without much computation
- Handles continuous and categorical variables
- Provides a clear indication of which fields are most important for prediction or classification

Weaknesses of Decision Trees

- Not suitable for prediction of continuous target
- Perform poorly with many class and small data
- Computationally expensive to train
  - At each node, each candidate splitting field must be sorted before its best split can be found
  - In some algorithms, combinations of fields are used and a search must be made for optimal combining weights
  - Pruning algorithms can be expensive since many candidate sub-trees must be formed and compared
- Don’t work well non-rectangular regions.

Model Validation

Model Validation

To estimate generalization error, we need data unseen during training. We split the data as

- Training set (50%)
- Validation set (25%)
- Test (publication) set (25%)

Resample when there is **few data**

Resample from **the minority** when there is **data skew**

Evaluating Models

- Infinite data is best, but…
- N fold Cross validation
  - Create N folds or subsets from the training data (approximately equally distributed with approximately the same number of instances).
  - Build N models, each with a different set of N-1 folds, and evaluate each model on the remaining fold
  - Error estimate is average error over all N models